Herramienta de análisis del uso y abuso de los permisos en aplicaciones Android

Tool for analysis the use and abuse of permissions in Android based applications.

UNIVERSIDAD
COMPLUTENSE
MADRID

Andrés Ramiro Ramiro
Álvaro Casado Molinero
Alejandro Ortiz Pintado

Trabajo de fin de grado
Grado en Ingeniería de Computadores / Grado en Ingeniería Informática

Facultad de Informática
Universidad Complutense de Madrid

Madrid, junio de 2021

Directores:

Inmaculada Pardines Lence
Marcos Sánchez-Élez Martín

Agradecimientos

Queremos agradecer a los directores de nuestro trabajo Inmaculada Pardines Lence y Marcos Sánchez-Élez Martín todo el apoyo, ayuda y tiempo que nos han dedicado durante el desarrollo de este trabajo.

También queremos agradecer a todos los familiares y amigos que nos han ayudado durante el periodo de pruebas y entrenamiento de AnPerMis, en especial queremos agradecer la ayuda de Marta Álvarez García, Paula Lobo Rodríguez, Nicolás Benito Barrionuevo, Ángel Ramiro Ramiro y Pedro Ramiro Ramiro, gracias a ellos hoy AnPerMis es mucho mejor.

Por último, queremos dedicar este trabajo a nuestras familias por todo el apoyo que nos han proporcionado durante su desarrollo y durante toda nuestra etapa educativa, por soportarnos y sobre todo por ayudarnos a no rendirnos y seguir siempre adelante.

¡GRACIAS!

Resumen

La preocupación por el uso de los datos personales que terceros recopilan sobre los usuarios de aplicaciones móviles ha aumentado considerablemente en los últimos años. La frase "Si el producto es gratis es porque el producto eres tú" ha calado en la sociedad y cada vez se tiene más cuidado de a quién y cómo cedemos nuestros datos personales. Mientras que hasta hace no tanto se le permitía a multitud de aplicaciones el acceso total a información sensible, cada día somos más conscientes de las amenazas que esto puede suponer y estamos actuando en consecuencia. No es raro encontrarse con aplicaciones, que alertan de las posibles amenazas en los dispositivos donde se instalan, o artículos que exponen buenas prácticas a la hora de configurar los permisos de las aplicaciones instaladas.

Las aplicaciones se suelen diseñar de forma que soliciten permiso para acceder a más elementos de los que realmente necesitan para su correcto funcionamiento. Es por eso que el trabajo aquí presentado se ocupa de la creación de una herramienta que detecte los posibles abusos por parte de aplicaciones al solicitar permisos. La herramienta desarrollada analiza los permisos que Android define como peligrosos solicitados por cada una de las aplicaciones instaladas en el dispositivo, contrastándolos con los permisos solicitados por las aplicaciones de la misma categoría. A este análisis se le une la aplicación de un filtro, basado en un análisis ético de los permisos realizado por los autores de este trabajo. Como resultado se obtiene una puntuación cualitativa que indicará si la aplicación está utilizando los permisos de forma abusiva.

De esta manera no solo hemos creado una herramienta completamente funcional para el análisis de posibles abusos por parte de las aplicaciones, sino que la misma herramienta puede ayudarnos a ser conscientes de los permisos que otorgamos a las aplicaciones para su funcionamiento y si esto pudiera significar un abuso. Porque, como veremos en las conclusiones de nuestro trabajo, la mayoría de las aplicaciones, según nuestro criterio, solicitan más permisos de los que realmente son necesarios, logrando tener acceso a gran cantidad de información privada sobre los usuarios que no es necesaria para su correcto funcionamiento.

Palabras clave

Permisos Android, Permisos peligrosos, Análisis de permisos, Análisis de aplicaciones, Uso abusivo de permisos, Categorías Google Play Store, Google Play Store, Aplicación Android, Android Studio, Java, Room, Sockets SSL.

Abstract

The concern about the use of personal data that third parties compile from mobile application users has considerably increased in the past few years. The saying "If the product is free, you are the product" has entered in people's minds and more often people care more about who and how we give our personal data. Not so long ago we granted a multitude of applications access to our personal information, every day we make ourselves more aware and we are acting in consequence. Is not unusual to find apps that alert of possible threats on the devices that they are installed, or articles that teach good practices when configuring application permissions.

The apps are often designed to ask permission for more elements that they really need. That is the reason why this project took care in the design of a tool to detect the possible misuse of applications requesting permissions. The tool analyzes the permissions, defined by Android as dangerous, requested by the applications on the device, matching the with the requested permissions for every app in the same category. To this study is added a filter, based on an ethical analysis carried out by the authors of this project. As a result, it shows a qualitative score that suggests if the applications are misusing permissions.

Along these lines, we have not only created a complete functional tool for the analysis of permissions misuse, but also this tool can help us to be more conscious about the permissions we granted to the applications. As we will see in the conclusions, most popular applications, according to our criteria, request more permissions than are really needed, managing to have access to a large amount of private information, which is not necessary for its proper functioning.

Keywords

Android permissions, Dangerous permissions, Permission analysis, Application analysis, Permission abuse, Google Play Store categories, Google Play Store, Android Application, Android Studio, Java, Room, SSL sockets.

Índice general

Índice general IV
Índice de figuras VII
Índice de tablas IX
Índice de códigos XIII
1. Introducción 2
1.1. Motivación 2
1.2. Objetivos 3
1.3. Estado del arte 4
1.4. Plan de trabajo 6
1.5. Estructura del documento 7
2. Introduction 9
2.1. Motivation 9
2.2. Objectives 10
2.3. State of the art 10
2.4. Work plan 12
2.5. Document structure 13
3. Aplicaciones, permisos y categorías de aplicaciones Android 15
3.1. Aplicaciones en Android 15
3.2. Permisos en Android 17
3.3. Categorías en Google Play 24
4. Perfilado de aplicaciones 27
4.1. Utilidad de las categorías de Google Play a la hora del perfilado 27
4.2. Criterios de evaluación de aplicaciones 28
4.2.1. Categorías potencialmente maliciosas 29
4.2.2. Criterio común 32
5. Diseño e implementación de la aplicación AnPerMis 37
5.1. Herramientas de desarrollo, testeo y diseño 37
5.2. Aplicación AnPerMis 39
5.2.1. Almacenamiento persistente de información (aplicación) 40
5.2.2. Comunicación con el servidor remoto. 43
5.2.3. Identificación de las aplicaciones instaladas. 48
5.2.4. Extracción de los permisos peligrosos 49
5.2.5. Obtención de la categoría de cada aplicación en Google Play Store .. 49

5.2.6. Almacenamiento de la información de cada aplicación 53
5.2.7. Envío de información de la aplicación al servidor remoto 54
5.2.8. Clasificación de los permisos peligrosos 55
5.2.9. Mostrar los resultados del análisis 57
5.2.10. Revocar permisos abusivos de una aplicación 58
5.2.11. Publicación de comentarios en Google Play Store 59
5.2.12. Modo Offline 60
5.3. Servidor 63
5.3.1. Creación de instancias virtuales en AWS 64
5.3.2. Control y gestión del servidor 69
5.3.3. Almacenamiento persistente de información en el servidor 71
5.3.4. Creación de la tabla de frecuencias 80
5.3.5. Histórico de aplicaciones 80
5.3.6. Caché de aplicaciones 80
5.3.7. Actualización automática de la base de datos 81
5.3.8. Comunicación entre la aplicación cliente AnPerMis y el servidor 82
6. Interfaz de usuario y funcionalidades de la aplicación 86
6.1. Diseño de la interfaz gráfica 86
6.1.1. Recursos de diseño utilizados 87
6.2. Pantallas y funcionalidad 93
6.3. Diagrama de flujo (Figura 37) 101
7. Resultados 102
7.1. Frecuencia de aparición de los permisos peligrosos 102
7.2. Análisis de resultados por categorías 103
8. Trabajo individual 143
8.1. Andrés Ramiro Ramiro 143
8.2. Álvaro Casado Molinero 145
8.3. Alejandro Ortiz Pintado 146
9. Conclusiones, asignaturas utilizadas y trabajo futuro 147
9.1. Conclusiones 147
9.2. Aprendizajes del grado empleados en el desarrollo de este proyecto 149
9.3. Trabajo futuro 149
10. Conclusions and Future Work 151
10.1. Conclusions 151
10.2. Degree subjects used in the development of this project 152
10.3. Future Work 153

BIBLIOGRAFÍA 156

Índice de figuras

Figura 1: Desglose de fases y tareas del trabajo. 7
Figura 1.2: Project phases and tasks. 13
Figura 2: Flujo de trabajo para usar permisos en Android 18
Figura 3: Flujo de trabajo para solicitar y usar permisos en tiempo de ejecución. 18
Figura 4: Pantalla que muestra en Google Play Store los permisos en tiempo de ejecución que puede solicitar una aplicación 20
Figura 5: Categorías en Google Play Store 31
Figura 6: Arquitectura de AnPerMis. 40
Figura 8: Ejemplo de funcionamiento de Sockets 44
Figura 9: Descripción del funcionamiento de una conexión SSL sin autenticación de cliente 45
Figura 10: Señales de advertencia 58
Figura 12: estructura del servidor remoto de AnPerMis 64
Figura 13: Seleccionar Amazon EC2 65
Figura 14: Seleccionar Crear par de claves 65
Figura 15: Crear par de claves, dar nombre y elegir formato 65
Figura 16: Instancias, lanzar instancia. 66
Figura 17: Configuración de la instancia, paso 1. 66
Figura 18: Configuración de la instancia, paso 2. 67
Figura 19: Configuración de la instancia, paso 3. 67
Figura 20: Configuración de la instancia, paso 4. 68
Figura 21: Configuración de la instancia, paso 5. 68
Figura 22: Configuración de la instancia, paso 6. 68
Figura 23: Diagrama de clases del servidor remoto 69
Figura 24: Primer diseño de la interfaz de usuario de AnPerMis 87
Figura 25. Pantalla Splash Screen 94
Figura 26. Pantalla principal de la aplicación 94
Figura 27. Pantalla de información 95
Figura 28. Aviso de aplicación potencialmente mal clasificada 96
Figura 29. Aviso de cambio de categoría 96
Figura 30. Aviso de cambio de número de permisos peligrosos 97
Figura 31. Pantalla información Aplicación 98
Figura 33. Pantalla Información Permisos 98
Figura 34. Pantalla revocar permisos 99

Figura 35. Pantalla Recomendación tras revocar permisos..... 99
Figura 36. Pantalla Comentar 100
Figura 38. Resultados del análisis de dos apps específicas de la categoría arte y diseño en AnPerMis..... 104
Figura 39. Resultados del análisis de dos apps específicas de la categoría empresa en AnPerMis 108
Figura 40. Resultados del análisis de dos apps específicas de la categoría comunicación en AnPerMis..... 112
Figura 41. Resultados del análisis de dos apps específicas de la categoría entretenimiento en AnPerMis 115
Figura 42. Resultados del análisis de dos apps específicas de la categoría bibliotecas y demos en AnPerMis..... 122
Figura 43. Resultados del análisis de dos apps específicas de la categoría estilo de vida en AnPerMis..... 124
Figura 44. Resultados del análisis de dos apps específicas de la categoría noticias y revistas en AnPerMis 128
Figura 45. Resultados del análisis de dos apps específicas de la categoría herramientas en AnPerMis..... 134
Figura 46. Resultados del análisis de dos apps específicas de la categoría social en AnPerMis 136
Figura 47. Resultados del análisis de dos apps específicas de la categoría herramientas en AnPerMis..... 138

Índice de tablas

Tabla 1: Valores de los campos de la tabla de frecuencias 72
Tabla 2: Valores de los campos de la tabla de frecuencias 72
Tabla 3: Valores de los campos de la tabla de criterios 73
Tabla 4: Valores de los campos de la tabla de históricos 73
Tabla 5. Resumen del estudio de permisos para las app de la categoría arte y diseño 103
Tabla 6. Estudio de permisos para dos apps específicas de la categoría arte y diseño 104
Tabla 7. Resumen del estudio de permisos para las app de la categoría automoción 105
Tabla 8. Estudio de permisos para dos apps específicas de la categoría automoción 105
Tabla 9. Resumen del estudio de permisos para las app de la categoría belleza... 106
Tabla 10. Estudio de permisos para dos apps específicas de la categoría belleza 106
Tabla 11. Resumen del estudio de permisos para las app de la categoría empresa 107
Tabla 12. Estudio de permisos para dos apps específicas de la categoría empresa. 107
Tabla 13. Resumen del estudio de permisos para las app de la categoría cómics. 108
Tabla 14. Estudio de permisos para dos apps específicas de la categoría cómics. 109
Tabla 15. Resumen del estudio de permisos para las app de la categoría libros y obras de consulta..... 109
Tabla 16. Estudio de permisos para dos apps específicas de la categoría libros y obras de consulta..... 110
Tabla 17. Resumen del estudio de permisos para las app de la categoría comunicación 110
Tabla 18. Estudio de permisos para dos apps específicas de la categoría comunicación 111
Tabla 19. Estudio de permisos para dos apps específicas de la categoría comunicación. 111
Tabla 20. Resumen del estudio de permisos para las app de la categoría citas..... 112
Tabla 21. Estudio de permisos para dos apps específicas de la categoría citas. ... 112
Tabla 22. Resumen del estudio de permisos para las app de la categoría educación 113
Tabla 23. Estudio de permisos para dos apps específicas de la categoría educación 113

Tabla 24. Resumen del estudio de permisos para las app de la categoría entretenimiento 114
Tabla 25. Estudio de permisos para dos apps específicas de la categoría entretenimiento 114
Tabla 26. Estudio de permisos para dos apps específicas de la categoría entretenimiento 115
Tabla 27. Resumen del estudio de permisos para las app de la categoría eventos 116
Tabla 28. Estudio de permisos para dos apps específicas de la categoría eventos 116
Tabla 29. Estudio de permisos para dos apps específicas de la categoría eventos 117
Tabla 31. Resumen del estudio de permisos para las app de la categoría finanzas117
Tabla 32. Estudio de permisos para dos apps específicas de la categoría finanzas 118
Tabla 33. Estudio de permisos para dos apps específicas de la categoría finanzas 118
Tabla 34. Resumen del estudio de permisos para las app de la categoría comer y beber 119
Tabla 35. Estudio de permisos para dos apps específicas de la categoría comer y beber 119
Tabla 36. Resumen del estudio de permisos para las app de la categoría salud y bienestar 120
Tabla 37. Estudio de permisos para dos apps específicas de la categoría salud y bienestar 120
Tabla 38. Resumen del estudio de permisos para las app de la categoría casa y hogar 121
Tabla 39. Estudio de permisos para dos apps específicas de la categoría casa y hogar 121
Tabla 40. Resumen del estudio de permisos para las app de la categoría bibliotecas y demos 121
Tabla 41. Estudio de permisos para dos apps específicas de la categoría bibliotecas y demos 122
Tabla 42. Resumen del estudio de permisos para las app de la categoría estilo de vida 123
Tabla 43. Estudio de permisos para dos apps específicas de la categoría estilo de vida 124
Tabla 44. Resumen del estudio de permisos para las app de la categoría mapas y navegación 125
Tabla 45. Estudio de permisos para dos apps específicas de la categoría mapas y navegación 125
Tabla 46. Resumen del estudio de permisos para las app de la categoría medicina 125

Tabla 47. Estudio de permisos para dos apps específicas de la categoría medicina 126
Tabla 48. Resumen del estudio de permisos para las app de la categoría música y audio 126
Tabla 49. Estudio de permisos para dos apps específicas de la categoría música y audio 127
Tabla 50. Resumen del estudio de permisos para las app de la categoría noticias y revistas 127
Tabla 51. Estudio de permisos para dos apps específicas de la categoría noticias y revistas 128
Tabla 52. Resumen del estudio de permisos para las app de la categoría ser padres 129
Tabla 53. Estudio de permisos para dos apps específicas de la categoría ser padres 129
Tabla 54. Resumen del estudio de permisos para las app de la categoría personalización 129
Tabla 55. Estudio de permisos para dos apps específicas de la categoría personalización 130
Tabla 56. Resumen del estudio de permisos para las app de la categoría fotografía 131
Tabla 57. Estudio de permisos para dos apps específicas de la categoría fotografía 131
Tabla 58. Resumen del estudio de permisos para las app de la categoría productividad..... 131
Tabla 59. Estudio de permisos para dos apps específicas de la categoría productividad..... 132
Tabla 60. Estudio de permisos para dos apps específicas de la categoría productividad..... 132
Tabla 61. Resumen del estudio de permisos para las app de la categoría compras 133
Tabla 62. Estudio de permisos para dos apps específicas de la categoría compras 133
Tabla 63. Resumen del estudio de permisos para las app de la categoría social... 134
Tabla 64. Estudio de permisos para dos apps específicas de la categoría social... 135
Tabla 65. Estudio de permisos para dos apps específicas de la categoría social... 135
Tabla 66. Resumen del estudio de permisos para las app de la categoría deportes 136
Tabla 67. Estudio de permisos para dos apps específicas de la categoría deportes 137

Tabla 68. Resumen del estudio de permisos para las app de la categoría herramientas 137
Tabla 69. Estudio de permisos para dos apps específicas de la categoría herramientas 138
Tabla 70. Resumen del estudio de permisos para las app de la categoría viajes y guías 139
Tabla 71. Estudio de permisos para dos apps específicas de la categoría viajes y guías 139
Tabla 72. Resumen del estudio de permisos para las app de la categoría reproductores de vídeo 140
Tabla 73. Estudio de permisos para dos apps específicas de la categoría reproductores de vídeos 140
Tabla 74. Resumen del estudio de permisos para las app de la categoría tiempo . 141
Tabla 75. Estudio de permisos para dos apps específicas de la categoría tiempo . 141
Tabla 76. Resumen del estudio de permisos para las app de la categoría Juegos 142
Tabla 77. Estudio de permisos para dos apps específicas de la categoría tiempo . 142

Índice de códigos

Código 1. getUpdateKeyStore() 46
Código 2. getTrustManagerFactory() 46
Código 3. getSSLContext() 46
Código 4. getSSLSocketFactory() 47
Código 5. sendMsg() 47
Código 6. getSocketMsg() 48
Código 7. createApplicationList() 48
Código 8. Script en Python para obtener la categoría de una aplicación desde Google Play Store 50
Código 9. getResponse() 51
Código 10. convertIsToString() 52
Código 11. fromResponse() 52
Código 12. searchCategory() 53
Código 13. uploadInfo() 54
Código 14. getUploadMessage() 54
Código 15. updateDangId() 56
Código 16. getDangerousPermissions() 57
Código 17. onClickAjustes() 58
Código 18. onRestart() 59
Código 19. onClickComentar() 59
Código 20. onSearchAppResult() 61
Código 21. testNetwork() 62
Código 22. testIp() 63
Código 23. start() 70
Código 24.1 actualizarBBDD() 70
Código 24.2 sendFreqTable() 70
Código 25. sendTcriterios() 70
Código 26. comprobarApps() 71
Código 27. conectar() 74
Código 28. actualizarTablasDesdeCero() 74
Código 29. init() 74
Código 30. getCriteriosTable() 75
Código 31. getFreqTable() 75

Código 32. sendFreqTable() 76
Código 33. sendTcriterios() 76
Código 34. comprobarApp() 77
Código 35. comprobarMismaApp() 77
Código 36. formatearResultadoSelect() 77
Código 37. subirAppsBBDD() 78
Código 38. queryParaActualizarTablaPermisosappsyHistorico() 79
Código 39. setPermiso() 79
Código 40. setFreqPermiso() 80
Código 41. ActualizarBBDD#run() 81
Código 42 Servidor#run() 83
Código 43. ClienteAceptado#run() 84
Código 44. ClienteAceptado#comprobarApps() 85
Código 45. ApplicationAdapter#onCreateViewHolder() 88
Código 46. ApplicationAdapter#onBindViewHolder() 89
Código 47. ApplicationAdapter#ViewHolder#ViewHolder() 89
Código 48. PermissionAdapter#onCreateViewHolder() 89
Código 49. PermissionAdapter#onBindViewHolder() 90
Código 50. PermissionAdapter#ViewHolder#ViewHolder() 90
Código 51. Creación de CardView de aplicaciones y permisos 91
Código 52. alertDialog() 91
Código 53. Contenido del fichero /res/values/styles.xml 92
Código 54. Contenido del fichero /res/drawable/background_splash.xml 92
Código 55. Definición de la actividad SplashInit. 93
Código 56. Actualización del archivo AndroidManifest.xml 93

1. Introducción

1.1. Motivación

Hoy en día, el uso de los dispositivos móviles se ha extendido a todos los ámbitos de la vida. Mientras que hace un par de décadas la finalidad de estos dispositivos era únicamente realizar llamadas telefónicas y, en el mejor de los casos, tomar fotos; hoy contamos con multitud de funcionalidades concentradas en un mismo instrumento. Los móviles nos ayudan a tomar la ruta más corta en nuestros viajes, navegar con total libertad por internet e incluso compartir nuestras fotos en redes sociales. Podemos descargarnos desde los juegos más completos hasta programas de entrenamiento a medida… ¿Pero a qué precio? [1]

Aunque parezca que todo son facilidades y beneficios, al usar estas aplicaciones, estamos cediendo gran cantidad de información sobre nosotros. Al usar una aplicación móvil, le estamos concediendo una serie de permisos para acceder a distintos recursos de nuestro dispositivo, como nuestra cámara, micrófono o ubicación con los que pueden recopilar información sensible. Por lo general, se tiende a banalizar este tema y no se le presta la atención que de verdad se merece, a pesar de que la recolección y tratamiento de nuestros datos por parte de las empresas desarrolladoras de aplicaciones en muchos casos es abusivo, ya que diseñan sus productos de manera que, además de realizar la función para la que los usuarios los descargan, recopilan todo tipo de datos acerca de ellos para venderlos a terceros. A priori la situación descrita nos puede llevar a pensar en aplicaciones de mensajería o redes sociales; sin embargo, todo lo dicho, valdría para cualquier aplicación que ofrece un servicio de forma gratuita, ya que en este caso el producto que se comercializa no suele ser la aplicación, sino los datos que se recopilan del usuario. Además, con la situación actual de la pandemia del Covid-19, se ha intensificado aún más el uso de los dispositivos móviles y, por tanto, la descarga de este tipo de aplicaciones.

Por todo esto, la divulgación de información acerca del uso abusivo de permisos por parte de las aplicaciones es de vital importancia a la hora de concienciarnos sobre nuestra seguridad y privacidad. La motivación de este trabajo nace precisamente de la intención de educar sobre estos aspectos y ayudar a los usuarios a entender cómo funcionan sus dispositivos y de qué manera pueden estar difundiendo información personal a empresas que ganan dinero a su costa.

1.2. Objetivos

Como hemos comentado anteriormente, el principal objetivo de este trabajo es crear una herramienta para que el usuario conozca, desde un punto de vista ético, cómo son los permisos dados a las aplicaciones instaladas en su teléfono móvil. Para conseguir este objetivo hemos tenido también que alcanzar otros objetivos intermedios que también podrían considerarse objetivos en sí mismos, tales como:

- Estudiar y analizar los permisos peligrosos que utilizan las aplicaciones instaladas por el usuario. Android define los permisos peligrosos como los permisos autorizan el acceso a los datos privados del usuario, es decir, datos restringidos en los que se incluye información que puede ser sensible. En este documento, cuando hablamos de permisos, nos referimos a este tipo de permisos.
- Estudiar y analizar los permisos típicos que utilizan las diferentes categorías de aplicaciones de la Google Play Store.
- Generar una heurística para poder clasificar a las aplicaciones en función de sus permisos y ofrecer esa información al usuario, indicando si algún permiso (o grupo de estos) es abusivo.
- Generar una aplicación que ofrezca toda esta información al usuario de una forma amigable y a la vez conseguir que esta se adapte, de forma transparente al usuario, a los cambios de permisos que se produzcan en aplicaciones y categorías.

Además de estos objetivos, a la hora de desarrollar esta herramienta, nos mueve una motivación de mejora del ecosistema de aplicaciones Android. Por un lado, nuestra intención también es hacer consciente al usuario sobre los permisos otorgados a las aplicaciones conjuntamente con el contexto en el que esa aplicación está inscrita. Por otro lado, consideramos que si la aplicación desarrollada en este trabajo, a futuro, fuera usada de forma masiva influiría sobre los desarrolladores de aplicaciones Android consiguiendo indirectamente que los desarrolladores de aplicaciones tuvieran en cuenta la ética en el diseño y la programación de sus aplicaciones, de tal forma que se velase por la privacidad de los usuarios y tuvieran que exponer de manera clara las razones de la solicitud de los permisos que necesitan sus aplicaciones.

1.3. Estado del arte

La información que suelen dar los desarrolladores de aplicaciones sobre el tratamiento de los datos que recogen sus productos suele ser escasa o difícil de encontrar. Por lo general suelen dar estos datos entre gran cantidad de información de menor importancia para que los pocos usuarios que se interesen por ellos se abrumen al intentar entender toda la terminología legal y cedan en su intento de conocer la finalidad del tratamiento de los datos personales recogidos. Si bien es cierto que hace unos años no se trabajaba para mejorar en este aspecto, últimamente se están haciendo avances para hacer más transparentes estos procedimientos y tanto las nuevas versiones de Android como las actualizaciones de la Google Play Store, contienen cambios en esta dirección.

En los últimos años se ha estado investigando acerca de la eficiencia y seguridad del sistema de permisos. Adrienne Porter Felt et al. en [2] estudian la efectividad del sistema de permisos de las aplicaciones en Android. Analizan los permisos de 856 aplicaciones gratuitas y 100 de pago, contrastando el uso de permisos que hace cada aplicación. El artículo llega a la conclusión de que, no solo el sistema de permisos actual es más seguro para el usuario que su predecesor (sistema en el que se concedían todos los permisos por defecto a las aplicaciones) sino que el número de aplicaciones maliciosas o con más permisos de los necesarios, decrece considerablemente. Aunque también se apunta que, con este sistema, más de un 10% de las aplicaciones solicitan más permisos de los necesarios y sugieren la creación de herramientas para la detección de errores y cambios en la plataforma para reducir la solicitud abusiva de permisos. También comentan que la gran mayoría de aplicaciones solicitan permisos peligrosos para su funcionamiento (93% de las gratuitas y 82% de las de pago).

En otros estudios, como en [3] se investiga sobre el incremento de los problemas de seguridad asociados al sistema de permisos de Android, relacionados con la granularidad de los permisos, los problemas en la administración de los permisos por parte de los desarrolladores y usuarios, los ataques de escalada de privilegios y la insuficiente documentación existente sobre los permisos. En este estudio también se ilustran los recientes avances de seguridad en el sistema Android.

En [4] se investiga el problema de la delegación de los permisos en otras aplicaciones, que es una vulnerabilidad muy popular del sistema de permisos de Android relacionada con la escalada de privilegios y que permite a una aplicación maliciosa adquirir privilegios a través de acciones solicitadas a otra aplicación que dispone de estos privilegios. En este estudio se detectaron un total de 30 aplicaciones procedentes de la tienda oficial de aplicaciones vulnerables a este tipo de ataque, lo que demuestra que esta es una de las vulnerabilidades más comunes en aplicaciones Android.

En Android 6.0 API 23 se incorporan al sistema de permisos los permisos en tiempo de ejecución para proteger la información delicada del usuario, de manera que las

aplicaciones desde esta versión deben solicitar estos permisos potencialmente peligrosos cuando el usuario comience a interactuar con la función que los requiere. Antes de la introducción de estos permisos en tiempo de ejecución todos los permisos se otorgaban de manera automática a las aplicaciones.

En Android 11.0 API 30 [5] se incorporan más opciones para manejar los permisos, de esta manera, el usuario puede tener más libertad a la hora de compartir información o dar acceso a distintas funcionalidades. Esta versión, ofrece cambios en la visibilidad del diálogo de permisos, es decir, si el usuario presiona "Rechazar" en un permiso específico más de una vez no volverá a ver el diálogo de solicitud del permiso, también añade la opción de conceder un permiso a una aplicación temporalmente durante su ejecución (Permisos únicos), de manera que se revoca al finalizar el proceso. Otra característica importante añadida es el restablecimiento automático de los permisos de aplicaciones que no se usan, es decir, si una aplicación no se ha ejecutado durante algunos meses, el sistema revoca automáticamente los permisos sensibles a fin de proteger los datos de los usuarios.

Aunque nuestro trabajo está basado en aplicaciones y sistemas Android, cabe destacar que también se están realizando avances en otros sistemas operativos como iOS. En iOS 14.5, Apple ha implementado un sistema de notificación al usuario con la opción de bloquear la obtención de información privada [6]. Esta opción impide que las aplicaciones tomen datos del usuario para crear perfiles con el objetivo de venderlos a terceros. Con estos perfiles, que se pueden obtener, por ejemplo, haciendo un seguimiento de los hábitos del usuario para inferir qué productos necesita o está buscando, se puede crear publicidad personalizada para cada usuario.

También está proliferando el desarrollo de aplicaciones que analizan el estado del teléfono para ayudar al usuario a comprobar la seguridad de su dispositivo y las aplicaciones instaladas en él. Por ejemplo, "*CONAN mobile*" [7] es una aplicación de INCIBE que, entre otras funcionalidades, permite clasificar las aplicaciones por permisos relevantes o realizar un análisis de las conexiones o accesos a recursos protegidos. Esta aplicación basa su análisis en una clasificación de permisos por categoría de riesgo, es decir, los clasifica como permisos de alto, medio y bajo riesgo. La aplicación AnPerMis, desarrollada en este TFG, también analiza el uso de los permisos por parte de las aplicaciones, pero hace un estudio más exhaustivo de estos permisos.

1.4. Plan de trabajo

La idea inicial de nuestro TFG era la implementación de una aplicación funcional capaz de detectar malware en Android. Sin embargo, en la etapa previa de documentación, al informarnos sobre el funcionamiento de las aplicaciones Android y su sistema de permisos, nos dimos cuenta de que, en gran cantidad de ocasiones, las aplicaciones solicitan muchos más permisos de los que realmente necesitan. Esto nos llamó especialmente la atención y decidimos centrar nuestro trabajo en el estudio del uso que hacen las aplicaciones de los permisos y cómo en muchas ocasiones abusan de ellos y realizan acciones para las que, a priori, no parecen estar diseñadas (lo cual en cierto modo se puede considerar un comportamiento malicioso).

Una vez que tuvimos claro cuál era el objetivo principal de nuestro trabajo, decidimos estructurar su desarrollo en tres fases, una primera fase de investigación, documentación y planificación del proyecto, una segunda fase de desarrollo e implementación de la herramienta de análisis AnPerMis y una última fase de redacción del trabajo y obtención del resultado de los distintos análisis. Además, estas fases se están desglosadas en distintas tareas para mejor organización y reparto del trabajo, tal como muestra la tabla de la Figura 1.

En la primera fase, recopilamos toda la información necesaria para empezar a orientar la herramienta al análisis de los permisos usados por las distintas aplicaciones de la Google Play Store. Nos planteamos los criterios que se deberían seguir para realizar dicho análisis y nos documentamos sobre el funcionamiento de los permisos en Android y sobre las categorías de las aplicaciones en la Google Play Store.

Tras esto, empezamos a desarrollar la herramienta programando una aplicación funcional para Android y configurando un servidor para almacenar y tratar la información generada por la aplicación. De esta manera, no solo disponemos de una aplicación que se puede instalar en gran variedad de dispositivos (es compatible con todas las versiones de Android), sino que contamos con un almacenamiento persistente de la información que se va actualizando con las aplicaciones analizadas y garantiza que la información está siempre actualizada.

Por último analizamos los resultados obtenidos exponiendo conclusiones acerca de los permisos de las aplicaciones más descargadas o mejor valoradas en la Google Play Store y su posible uso abusivo. En esta fase también incorporamos la redacción de la memoria de este trabajo, que incluye un resumen de la información recopilada en la primera fase, la descripción de cada una de las partes de nuestra herramienta y de los conceptos necesarios para entenderla y la presentación de las conclusiones a las que hemos llegado en la realización de este trabajo.

Fase	Tarea	Descripción
Fase 1	T1	Documentación acerca del funcionamiento de aplicaciones y sistema de permisos en Android.
	T2	Documentación sobre el sistema de categorías de Google Play.
	T3	Trazar el método de análisis de las aplicaciones mediante el perfilado por categorías.
	T4	Planear el desarrollo de la herramienta.
Fase 2	T5	Implementación de los criterios de análisis de aplicaciones
	T6	Desarrollo aplicación móvil y servidor
	T7	Implementación de la comunicación cliente-servidor
	T8	Entrenamiento de base de datos para mejorar los resultados del análisis
Fase 3	T9	Obtención de resultados de los análisis de la herramienta
	T10	Redacción del trabajo

Figura 1: Desglose de fases y tareas del trabajo.

1.5. Estructura del documento

La memoria de este trabajo se ha organizado en 10 capítulos, que siguen el mismo orden que se ha seguido en el desarrollo e implementación del proyecto. A continuación, incluimos una breve descripción de dichos capítulos, explicando su contenido y su finalidad.

Los capítulos 1 y 2 los hemos dedicado a la introducción del proyecto y a explicar por qué y cómo lo hemos llevado a cabo. En ellos se detalla la motivación y los objetivos del trabajo, se exponen las razones por las cuales hemos decidido trabajar en este ámbito y se explican los beneficios que esto puede reportar. También incluimos un breve resumen de la situación actual de la temática tratada en este trabajo y explicamos cómo hemos estructurado el proyecto y la redacción de la memoria.

El capítulo 3 incluye el estudio del funcionamiento de las aplicaciones en Android, su sistema de permisos y el sistema de categorías que usa la Google Play Store para clasificar las aplicaciones. Estos conceptos son clave para entender el capítulo 4, en el que se destaca por qué es interesante el perfilado por categorías de la Google Play Store a la hora de realizar el análisis de las aplicaciones y cómo podemos aplicar todo esto en el desarrollo de nuestra aplicación. Explicaremos todos los conceptos necesarios para comprender nuestro análisis de aplicaciones Android, desde cómo están estructuradas estas aplicaciones hasta la clasificación por categorías de la Google

Play Store; detallaremos todos los elementos desarrollados de la herramienta, como las conexiones con el servidor o la aplicación Android; y expondremos resultados y conclusiones tras el análisis de más de 600 aplicaciones descargadas.

En el capítulo 5 se explica en detalle cómo hemos programado nuestra herramienta, describiendo las partes fundamentales de la aplicación Android y del servidor, junto con todas sus funcionalidades relevantes y fases de mejora. Mientras que en el capítulo 6 explicamos la aplicación de una manera más visual.

En el capítulo 7 presentamos los resultados de diversos análisis realizados con la aplicación para mostrar su funcionalidad. En el capítulo 8 se explica el trabajo realizado por cada uno de los miembros del equipo. Los capítulos 9 y 10 exponen las principales conclusiones que hemos obtenido durante el desarrollo del proyecto y las posibles fases de mejora que se podrían aplicar al mismo en el futuro.

2. Introduction

2.1. Motivation

Nowadays, the use of mobile devices is everywhere. While a couple of decades the purpose of these devices was only to make phone calls, at best, take pictures; today we have a multitude of functionalities concentrated in the same instrument. Phones help us taking the shortest route in our trips, browse the internet and even share our pictures on social networks. We can download from the most complete games to customized training programs… But at what price? [1]

Although it seems that everything are facilities and benefits, by using these applications, we are giving a large amount of information about ourselves. By using an app, we are granting a series of permission to access different resources on our device, like the camera, microphone, location… with which they can collect sensitive information. Usually, this topic tends to be trivialized, and it is not paid the attention it really deserves, even though the recollection and treatment of our personal data by application developers in many cases is abusive, since they design their products in a way that, in addition to perform the function for which the users download the application, the collect all kinds of data to sell it to third parties. At first, the described situation may lead us to think about messaging application or social networks; however, after all that has been said, it would be well-founded for any application that offers a free service, since in this case the product that is marketed is not usually the application, but the data collected from the users. Furthermore, with the current situation of the pandemic Covid-19, it has been further intensified the use of mobile devices and thus, the download of such applications.

For all this, publishing information about the misuse of permissions by applications, is of vital importance when it comes to raising awareness about our security and privacy. The motivation for this project is born from the intention of educating on these aspects and helping users to understand how their devices work and how they may be diffusing personal information to companies that earn money at their expense.

2.2. Objectives

As we said before, the main goal of this work is creating a tool for the user to get to know, from an ethical point of view, how are the permissions given to the apps installed on your mobile phone. To reach this goal we had to achieve some intermediate goals that could also be considered goals in itself, such as:

- Study and analyze the permissions that the apps, installed by the user, use.
- Study and analyze the typical permissions that the different categories of apps in the Google Play Store use.
- Generate a heuristic to classify the apps depending on their permissions and offering that information to the user, indicating if any of these permissions (or groups of them) are abusive.
- Generate an app that offers all that information to the user in a friendly way while it adapts, in a transparent way to the user, to the changes of permissions that occur in apps and categories.

Apart from these goals, while developing this tool, we are moved by the motivation of improving the ecosystem of android apps. On one hand, our intention is making the user aware of the permissions given to the apps together with the context where that application is enrolled. On the other hand, we considered if our app, in the future, was used in a massive way it would influence the Android app developers, making them indirectly take in account the ethics of the design and programing of their apps in a way that they defend the privacy of the users and expose in a clear way the reasoning for soliciting the permissions that their apps need.

2.3. State of the art

The information that app developers tend to give about the treatment that the data collected by their products is given, is usually limited or hard to find. In general, they give this information immersed in large quantities of less important knowledge so the few users that are interested by it get overwhelmed trying to understand all the legal terminology and give up in their attempt to understand the purpose of the treatment of the personal data collected. It is true that a few years ago we did not work to improve this aspect, but lately, advances to make more transparent these procedures are being made. The new Android versions and the Google Play Store updates, move in this direction.

In the past years there has been investigations about the efficiency and security of the permit system. Adrienne Porter Felt et al. en [2] studies the effectivity of the system of

permissions in Android apps. They analyze the permissions of 856 free apps and 100 paid apps, comparing the use of permissions that each app does. The article reaches the conclusion that, not only the actual system of permissions is safer to the user that its predecessor (system in which all the permissions were given by defect to the apps) but also the number of malicious apps or those with more permissions than needed, decreases considerably. But also notes that with this system, more than 10% of the apps require more permissions than necessary and suggest the creation of tools to detect errors and changes in the platform to reduce the abusive soliciting of permissions. It also notes that most free apps ask for dangerous permissions for its own functioning (93% free and 82% of the paid ones).

In other studies, like the [3] it analyzes the increase of the security problems associated to the Android permit system, related to the granularity of the permissions, the problems in the permit administration by developers and users, the attacks of privilege escalation and the insufficient documentation that exists about the permissions. In this investigation, it also enlightens the recent advances in security in Android and analyzes the possible solutions to these problems.

In [4] it is investigated the problem of the delegation of permissions in other apps, that is a popular vulnerability of the permit system in Android related to the privilege escalation that allows a malicious app to acquire privileges through actions solicited to other apps that possesses these privileges. In this investigation a total of 30 apps were detected as coming from the official store of apps vulnerable to this type of attack, which shows that is one of the most common vulnerabilities in Android apps.

In Android 6.0 API 23 the permissions of time of execution are incorporated to the permit system to protect the delicate information of the user, in a way that the apps from this version need to request these potentially dangerous permissions when the user starts to interact with the function that requires them. Before the introduction of these permissions in time of execution all the permissions were given automatically to the app.

In Android 11.0 API 30 [5] more options are incorporated to manage the permissions and, this way, the user can have more liberty when sharing information or giving access to the apps to various functionalities. This version offers changes in the visibility of the permissions dialogue, which means that if the user presses “deny” in a specific permit more than once he would no longer see the dialogue of permit solicitation, also adding the option of granting a permit to an app temporarily during its execution (unique permissions), in a way that it is denied once the process ends. Another important characteristic added is the automatic reestablishment of the permissions that the app does not use, meaning that if an app is not executed in several months, the system automatically revokes the sensitive permissions in order to protect the data of the users.

Even if our work is based in Android apps and systems, we should also note that there have been advancements in other operative systems such as iOS. In the latest iOS version (14.5), Apple has implemented a system of notification that avoids the apps

from taking data of the users to create profiles with the goal of selling it to a third party [6]. With these profiles, that could for example be obtained by following the habits of the user to interfere in the products that he needs or is looking for, personalized advertising could be created for each user.

There is also the proliferation of the development of apps that analyze the state of the pone to help the user prove the security of its device and the apps installed on it. For example, "CONAN mobile" [7] it is an app from INCIBE that, amongst other features, allows the classification of apps by relevant permissions or make an analysis of the connections or access to protected resources. This app bases its analysis in a classification of permissions by risk categories, classifying them in permissions with high, medium and low risk. The app AnPerMis, developed in this TFG, also analyzes the use of permissions by the apps, but it makes a more comprehensive study of these permissions.

2.4. Work plan

The initial idea of our Final Degree Project was the implementation of a functional application capable of detecting malware on Android. However, in the previous investigation stage, we learned how their permissions system works, and realized that most of the time, applications request more permissions than they really need. This caught our attention and we decided to focus our work on studying the use that applications make of our permissions and how many times they misuse them and do actions for which they were not designed for (which somehow can be considered as malicious behavior).

Once we were clear about the main objective of our work, we decided to structure its development in three phases, a first phase of investigation, documentation and planning of the project, a second phase of development and implementation of AnPerMis the analysis tool and a last phase or writing the obtained results of the different analyzes. In addition, these phases are broken down into different tasks for better organization and distribution, as shown in the table Figure 1.

On the first phase, we recollect all the needed information to start orienting the tool to the analysis of permissions used by the different applications in Google Play Store. We consider the criteria that should be followed to carry out this analysis and we document ourselves on the operation of permissions in Android and on the application categories in Google Play Store.

After this, we started implementing the tool, we programmed a functional Android application and set up a server for storage and data treatment. We not only have an application that can be installed on a wide variety of devices (as full compatibility with all Android versions), but also, we have a persistent storage that updates itself with every analyzed application and guarantees that the information is always up to date.

Finally, we analyzed the obtained results, and we present the conclusions about the permissions of the most downloaded or best-rated applications on the Google Play Store and their possible abusive use. At this stage we also incorporate the report of this project, which includes a summary of the information collected on the first phase, the description of each part of our tool and the concepts needed to understand it and the conclusions we have reached while putting together this work.

Phase	Task	Description
Phase 1	T1	Documentation about applications and permission system in Android.
	T2	Documentation about Google Play Store category system.
	T3	Design an analysis method for the applications' profiling by Google Play Store category.
	T4	Tool development planning
Phase 2	T5	Application analysis criteria implementation
	T6	Server and Android app development
	T7	Client-server model implementation
	T8	Database training to improve results.
Phase 3	T9	Obtaining results from the analysis of the tool
	T10	Project writing.

Figura 1.2: Project phases and tasks.

2.5. Document structure

The memory of this project has been organized into 10 chapters, which follow the same order that has been followed in the development and implementation of the tool. Now we are including a brief description of these chapters, explaining their content and purpose.

We have dedicated chapters 1 and 2 to introducing the project and explaining why and how we have carried it out, detailing the motivation and objectives of the work, explaining the reasons why we have decided to work in this area and explain the benefits that this project can bring. We also include a summary of the current situation of the subject treated in this project and we explain how we have structured the writing of the report.

Chapter 3 includes the study of applications on Android, its permissions system and the category system that Google Play Store uses to classify applications. These are key

concepts to understand chapter 4, which highlights why profiling by Google Play Store categories is interesting when analyzing applications and how we can apply all this in the development of our application. We will explain all the key concepts to understand our analysis of Android applications, from how these applications are structured to the classification by categories of the Google Play Store; we will detail all the developed elements of the tool, such as the connections with the server or the Android application; and we will present results and conclusions after analyzing more than 600 downloaded applications.

Chapter 5 explains in detail how we have programmed our tool, describing the fundamental parts of the Android application and the server, along with all its relevant functionalities and improvement phases. While chapter 6 explains the application in a visual way.

In Chapter 7 we present the results of various analysis performed with the application to show its functionality. Chapter 8 explains the work done by each of the team members and Chapters 9 and 10 present the main conclusions that we have obtained during the development of the project and the possible improvement phases that could be applied to it in the future.

3. Aplicaciones, permisos y categorías de aplicaciones Android

3.1. Aplicaciones en Android

Las aplicaciones Android [8] se compilan en un paquete Android APK mediante las herramientas de Android SDK, donde se pueden programar en lenguajes Kotlin, Java o C++. Este paquete con sufijo *.apk* cuenta con todos los recursos y datos para que un sistema Android pueda instalar la aplicación.

Android es un sistema Linux multiusuario en el que cada aplicación es un usuario diferente al que se le asigna un ID único y establece los permisos para todos los archivos de manera que solo ese ID puede acceder a ellos. Cada proceso cuenta con una máquina virtual independiente en la que se ejecuta aislado de otras aplicaciones. El sistema operativo Android lanza este proceso cuando se intenta acceder a algún recurso de la aplicación y lo libera cuando se cierra o cuando otras aplicaciones necesitan memoria para ser lanzadas. En ocasiones muy concretas, se puede hacer que dos aplicaciones compartan un ID para acceder a sus recursos. Como explicaremos más adelante, el sistema Android implementa el principio de mínimo privilegio, por el cual se le permite a cada aplicación el acceso solo a los componentes protegidos por el sistema que se necesitan para su ejecución.

Los componentes de la aplicación son los pilares de una aplicación Android, ya que sirven para que los usuarios o el sistema accedan a la aplicación. Cada componente tiene una labor específica y se puede clasificar dentro de los siguientes tipos:

- **Actividades**

  El manual de desarrollo de Android [8] define a una actividad como "el punto de entrada de interacción con el usuario". Y lo ilustra con el ejemplo de una aplicación de correo electrónico, que puede tener una actividad para mostrar la lista de correos nuevos, otra para leer un correo electrónico, otra para guardar un borrador… Son como las funciones de un programa, se pueden consultar (si son públicas) desde otras aplicaciones para aumentar la funcionalidad. Siguiendo con el ejemplo de la aplicación de correo electrónico, de esta manera se puede lanzar la actividad de enviar un correo desde una aplicación de edición de texto para poder enviar el documento que se esté elaborando mediante el correo.

- **Servicios**

  Los servicios [8] son puntos de entrada que permiten mantener en segundo plano la ejecución de una aplicación. Por ejemplo, un servicio podría reproducir en

segundo plano la radio mientras que el usuario está en otra aplicación o puede consultar la ubicación mientras el teléfono está bloqueado. Un servicio se puede vincular con otros componentes como una actividad para interactuar. Los servicios se administran de dos maneras distintas en función de si el usuario es consciente o no de que se están ejecutando. En el ejemplo de la radio, el usuario ha querido que se ejecute en segundo plano, por lo que el sistema no lo puede intervenir. Mientras que, en el caso de la ubicación, el sistema puede administrar el servicio con mayor libertad, ya que el usuario no es consciente de su ejecución y no va a notar cambios en sus actividades.

- **Receptores de emisión**

Los receptores de emisión [8] son entradas a las aplicaciones por las que el sistema es capaz de entregar eventos fuera del flujo habitual de la aplicación, como por ejemplo las alertas de batería baja o de apagado del dispositivo. Las aplicaciones pueden crear eventos como programar notificaciones al usuario y entregárselas a un receptor de emisión para que, cuando sea el momento de lanzar la notificación, la aplicación no tenga por qué estar ejecutándose.

- **Proveedores de contenido**

Los proveedores de contenido[8] son los encargados de administrar el flujo de datos desde una ubicación de almacenamiento persistente. El manual de desarrollo de Android pone como ejemplo la información de contacto del usuario del sistema, una aplicación con los permisos correspondientes puede consultar el proveedor de contenido encargado de estos datos para leerlos o modificarlos.

Otra parte fundamental de una aplicación en Android es el manifiesto [9], que es un archivo XML situado en la raíz de la fuente del proyecto que incluye toda la información relevante de la aplicación para las herramientas de creación de aplicaciones, el sistema operativo o Google Play. El manifiesto contiene:

- El nombre del paquete de la aplicación y su ID. El nombre de la aplicación es un atributo para que, a la hora de compilar en el paquete APK, se puedan ubicar las distintas entidades de código necesarias al definir un espacio de nombres con él. Al compilar el APK, este nombre de aplicación pasa a ser un ID único para distinguirlo del de cualquier otra aplicación.

- La declaración de todos los componentes de la aplicación en estructuras XML.

- La declaración de todos los permisos que usa la aplicación.

- La declaración de los tipos de funciones de software y hardware que requiere la aplicación para su correcto funcionamiento. De esta manera se puede comprobar la compatibilidad de la aplicación con los distintos dispositivos y versiones de Android.

Para poder publicar una aplicación en Google Play, se requiere que el paquete APK esté firmado con un certificado electrónico [125] para que la aplicación se pueda instalar y actualizar en los distintos dispositivos. Además, la guía de desarrollo de Android detalla toda clase de buenas prácticas a la hora de desarrollar, firmar y categorizar las aplicaciones.

3.2. Permisos en Android

El sistema de permisos de las aplicaciones Android constituye la primera línea de defensa frente al malware, protegiendo el acceso a datos restringidos (como la información de contacto del usuario o sus contactos) y a funcionalidades restringidas (como el acceso a la cámara o a la ubicación). De esta manera, los usuarios del dispositivo pueden habilitar el acceso de las aplicaciones solo a aquellas funcionalidades necesarias para su ejecución.

Cabe recalcar que a la hora de desarrollar una aplicación Android se puede acceder a numerosas funcionalidades sin necesidad de requerir permisos, y por ello, la guía de desarrollo [10] ilustra el flujo de trabajo para decidir si es completamente necesario solicitar permiso (Figura 2), o se puede acceder a la funcionalidad o la información protegida por otros medios.

La guía de desarrollo de Android pone mucho énfasis en las buenas prácticas a la hora de solicitar permisos [11] e insta a que se desarrollen las aplicaciones de manera que los usuarios tengan control sobre los datos que comparten con la aplicación [12], que se muestre transparencia sobre los datos tratados por dicha aplicación y que se minimice lo máximo posible la cantidad de información a la que se accede. Además, como hemos visto en el diagrama de la Figura 2 si se llega a la conclusión de que se necesitan permisos peligrosos para la ejecución de una aplicación, la guía detalla los pasos que se deben seguir para declarar los permisos, tal y como se muestra en la Figura 3.

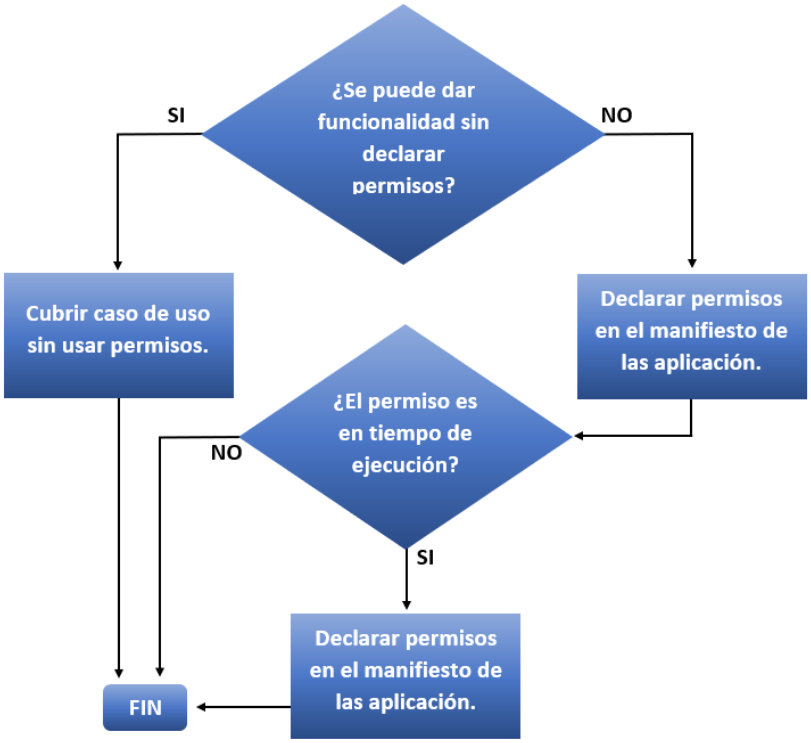


Figura 2: Flujo de trabajo para usar permisos en Android

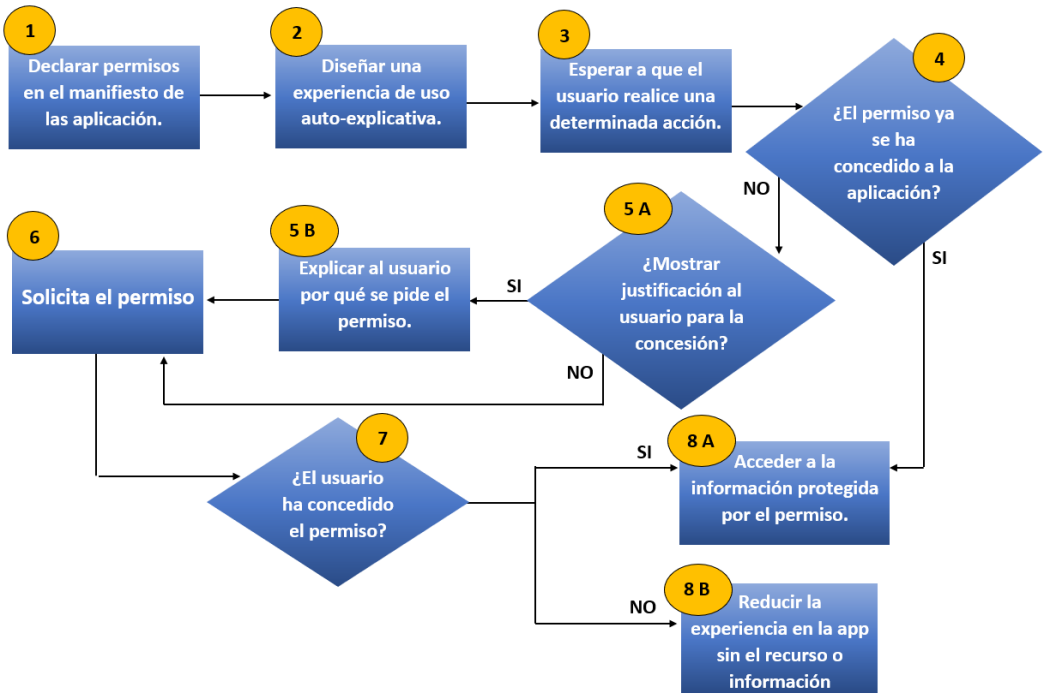


Figura 3: Flujo de trabajo para solicitar y usar permisos en tiempo de ejecución.

Como se puede observar, la guía hace mucho énfasis en que se deben solicitar los permisos mínimos, y en el caso de que estos no sean concedidos, que se diseñe la aplicación para que pueda ser funcional sin necesidad de ellos.

Existen varios tipos de permisos [13]: permisos en el momento de la instalación, permisos en tiempo de ejecución y permisos especiales.

- Los permisos en el momento de la instalación son permisos que se aceptan automáticamente al instalar una aplicación, ya que no suponen ningún riesgo para los datos restringidos del dispositivo ni acceden a funcionalidades protegidas del mismo. Dentro de este tipo de permisos, se distinguen dos subtipos: los permisos normales (son los permisos que se aceptan automáticamente por el mínimo riesgo que suponen para el usuario) y los permisos de firma (son permisos que, si están firmados con el mismo certificado que otra aplicación ya instalada, se detectan como confiables y también se aceptan automáticamente al instalar la aplicación).

- Los permisos en tiempo de ejecución son los permisos definidos como permisos peligrosos, ya que estos sí que acceden a información privada del usuario del dispositivo o permiten que se realicen acciones restringidas que pueden afectar más seriamente a otras aplicaciones o al sistema. Estos permisos requieren de una concesión expresa por parte del usuario cuando se instala la aplicación, y son los permisos que muestran las aplicaciones en Google Play antes de su instalación (ver Figura 4)

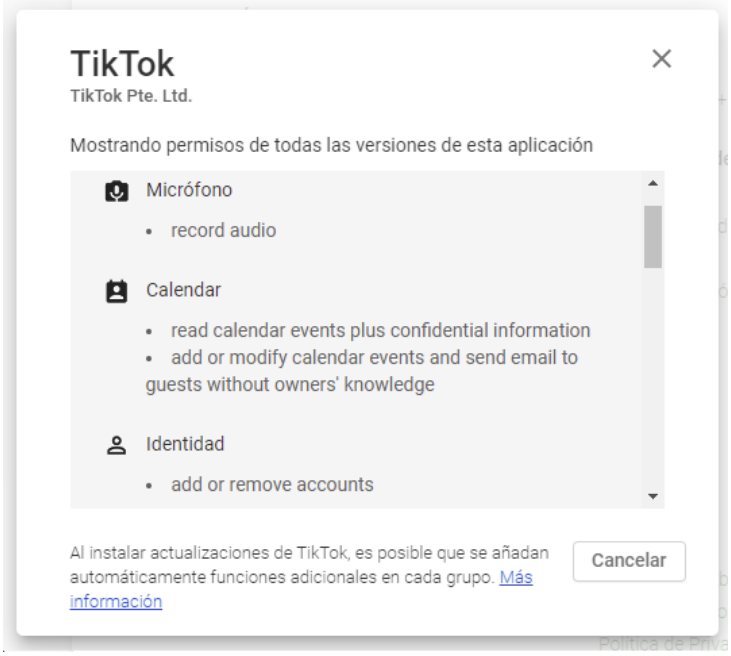


Figura 4: Pantalla que muestra en Google Play Store los permisos en tiempo de ejecución que puede solicitar una aplicación

- Por último, contamos con los permisos especiales, que sirven para que la plataforma y los fabricantes puedan proteger acciones particularmente importantes. Estos permisos están implementados de manera diferente a los anteriores y cuentan con un nivel distinto de protección.

En este análisis nos vamos a centrar únicamente en los permisos en tiempo de ejecución que son los que se consideran como permisos peligrosos, ya que acceden a los datos privados del usuario, es decir, datos restringidos en los que se incluye información que puede ser sensible. Explicaremos los recursos protegidos por cada uno de ellos [14] y haremos una breve descripción de un hipotético escenario en el que un usuario se puede ver comprometido por un mal uso de los distintos permisos:

- ACCEPT_HANDOVER: permite que una aplicación continúe la llamada que ha empezado otra. Esto puede afectar al usuario si la app que continúa la llamada hace uso del plan de llamadas en lugar de los datos móviles, lo cual puede significar un gasto considerable. También puede ser usado para grabar las llamadas sin autorización expresa.

- ACCESS_BACKGROUND_LOCATION: permite a la aplicación acceder a la ubicación estando en segundo plano. No sirve de nada si no se ha concedido previamente el permiso a la ubicación, pero de haberlo concedido, la aplicación puede usarla aun estando cerrada.

- READ_EXTERNAL_STORAGE: permite leer datos de dispositivos de almacenamiento externo. Esto incluye tarjetas SD o incluso un portátil conectado al dispositivo. La aplicación podría acceder a toda la información almacenada en esos dispositivos sin un aviso explícito.

- WRITE_EXTERNAL_STORAGE: permite modificar datos de dispositivos de almacenamiento externo. Si se concede este permiso, se concede también implícitamente el READ_EXTERNAL_STORAGE: Un posible atacante no solo podría acceder a los datos almacenados en estos dispositivos externos, sino que además podría modificarlos, borrarlos o crear datos nuevos.

- ACCESS_COARSE_LOCATION: permite el acceso a la ubicación aproximada del dispositivo basándose en la torre de telecomunicaciones más cercana. Esta

funcionalidad permite a la aplicación conocer en qué zona se encuentra el usuario, identificando la ciudad o incluso el barrio.

- ACCESS_FINE_LOCATION: permite el acceso a la ubicación precisa usando GPS y WiFi. Así como el acceso a la ubicación aproximada puede revelar dónde se encuentra el usuario en un área relativamente amplia, este permiso permite su geolocalización con una precisión de pocos metros. Una aplicación maliciosa podría saber dónde se encuentra en todo momento (incluso si está cerrada si se ha concedido también el permiso ACCESS_BACKGROUND_LOCATION) pudiendo generar patrones de movimiento o infiriendo dónde trabaja el usuario y donde vive analizando cuánto tiempo pasa en estos lugares.

- ACCESS_MEDIA_LOCATION: permite el acceso a la geolocalización del contenido multimedia del usuario. Es decir, si tiene activado que su dispositivo guarde la localización del lugar donde ha tomado las fotos y vídeos de su galería, esta aplicación puede leerla. Un posible atacante podría analizar estas localizaciones y saber con bastante precisión dónde ha estado el usuario.

- READ_PHONE_STATE: permite a la aplicación la lectura (únicamente lectura) del estado del teléfono, información de la red del teléfono, estado de las llamadas en curso o las cuentas registradas en el teléfono.

- ACTIVITY_RECOGNITION: permite a la aplicación reconocer la actividad física. Esto incluye métricas como el cuentapasos, si el usuario está conduciendo, corriendo, en bicicleta... Por separado puede no parecer peligroso, pero si se junta con otro tipo de información como la localización, se podría saber dónde y qué está haciendo el usuario en un determinado momento.

- ADD_VOICEMAIL: permite a la aplicación añadir mensajes al buzón de voz. Un posible atacante podría dejar mensajes en el buzón de voz haciéndose pasar por otra persona o empresa.

- ANSWER_PHONE_CALLS: permite a la aplicación contestar llamadas telefónicas. Una aplicación maliciosa podría contestar las llamadas entrantes al dispositivo haciéndose pasar por el usuario.

- BODY_SENSORS: permite a la aplicación acceder a la información suministrada por sensores que miden la actividad corporal, como las pulsaciones. Como ocurría con el permiso ACTIVITY_RECOGNITION, por sí solo no revela mucha

información, pero juntándolo con otros permisos, se puede saber con bastante detalle qué está haciendo el usuario del dispositivo en cualquier momento.

- CALL_PHONE: permite a la aplicación iniciar llamadas telefónicas sin necesidad de confirmación por parte del usuario. Las posibles aplicaciones son parecidas a las de ACCEPT_HANDOVER, un posible atacante podría llamar a teléfonos de pago constantemente consiguiendo mucho dinero a costa del usuario.

- CAMERA: permite el uso de la cámara para tomar fotos o grabar vídeos. Un posible atacante podría tomar fotos comprometidas del usuario para más tarde extorsionarle con ellas.

- GET_ACCOUNTS: permite el acceso a la lista de cuentas del dispositivo. Una aplicación maliciosa podría saber cuáles son las redes sociales o los correos electrónicos del usuario. Desde Android 6, se pueden manejar las cuentas sin pedir este permiso, anteriormente, para poder manejar estas cuentas había que solicitarlas.

- PROCESS_OUTGOING_CALLS: permite a una aplicación saber con qué número se ha establecido la llamada, pudiendo redireccionarla o incluso cortarla. Las aplicaciones son similares a ACCEPT_HANDOVER y CALL_PHONE.

- READ_CALENDAR: permite a la aplicación leer todos los datos del calendario, incluyendo información sensible sobre los eventos del usuario. Adicionalmente, si se han incluido anotaciones en las entradas del calendario, se pueden leer, lo que permite obtener información de por qué el usuario está en ese lugar o cita u otro tipo de datos. Una vez más, junto con permisos de localización, una aplicación maliciosa puede no solo conocer esta información, sino la ruta que se ha tomado para llegar a esa cita o si realmente se ha acudido a ella.

- WRITE_CALENDAR: permite añadir eventos al calendario del usuario de la aplicación. Un posible uso malicioso de este permiso daría la posibilidad a un atacante de modificar eventos para cambiar su información como números de teléfono, direcciones o directamente borrar eventos para que el usuario no acuda.

- READ_CALL_LOG: permite a la aplicación acceder al registro de llamadas del dispositivo pudiendo crear perfiles de con quién se comunica el usuario y con qué frecuencia.

- WRITE_CALL_LOG: permite añadir elementos al registro de llamadas (aunque no permite consultarlo). A pesar de que no es un permiso muy común, un posible atacante podría añadir llamadas falsas en el registro o borrarlo para extorsionar de alguna manera al usuario.

- READ_CONTACTS: permite a la aplicación leer la agenda de contactos del usuario del dispositivo. Mediante esta funcionalidad, una aplicación maliciosa podría generar una red de contactos para saber con quién se relaciona el usuario.

- WRITE_CONTACTS: permite a la aplicación añadir o sobrescribir los contactos en la agenda del usuario. Un posible ataque sería editar el contacto de un familiar y cambiar su número por uno de pago, o cambiar el número de un contacto comercial por el de algún estafador.

- READ_PHONE_NUMBERS: permite a la aplicación consultar el número o números de teléfono del dispositivo. De esta manera, cualquier aplicación maliciosa podría conocer el número de teléfono del usuario.

- READ_SMS: permite a la aplicación leer los SMS entrantes, leyendo información sensible o interceptando claves de confirmación de cambio de contraseña o de suscripción a algún servicio.

- RECEIVE_SMS: permite a la aplicación la monitorización de los mensajes SMS entrantes.

- RECEIVE_MMS: permite a la aplicación la monitorización de mensajes MMS entrantes, pudiendo acceder a fotos o vídeos recibidos mediante esta vía.

- RECEIVE_WAP_PUSH: permite a la aplicación la recepción de mensajes WAP PUSH. Este tipo de mensajes contienen enlaces a páginas web, lo que permite recibir un enlace de *phishing* o la descarga de algún malware.

- RECORD_AUDIO: permite a la aplicación el uso del micrófono para grabar audio. Un atacante podría escuchar las conversaciones del usuario o sacar información basada en los sonidos de su entorno.

- SEND_SMS: permite a la aplicación el envío de SMS. Este permiso es especialmente peligroso ya que puede suscribir al usuario a servicios de pago por SMS.

- USE_SIP: permite a la aplicación el uso del servicio SIP.

3.3. Categorías en Google Play

La gestión de las aplicaciones por parte de Google Play Store implica que estas deben estar asignadas a una categoría que debería tener relación directa con sus funcionalidades o finalidades, con el objetivo, en un principio de facilitar su búsqueda a los usuarios. (los usuarios pueden ver las aplicaciones más relevantes, más vendidas o mejor valoradas de cada categoría). Además, en este trabajo esta clasificación nos va a permitir realizar una comparación de los permisos requeridos por las aplicaciones en función de su categoría, ya que cabría esperar que todas las aplicaciones que pertenecen a una misma categoría presenten permisos parecidos.

Hay un gran número de categorías (Figura 5), segmentando mucho las aplicaciones de tal forma que casi hay una categoría para cada tipo de aplicación. Google distingue entre distintas categorías de aplicaciones y juegos, y en la ayuda al desarrollador proporciona ejemplos de qué aplicaciones entran en cada categoría [15].

Las categorías de aplicaciones, junto con sus ejemplos, son las siguientes:

- **Arte y diseño**: Cuadernos de bocetos, herramientas de pintura, herramientas de arte y de diseño, libros de colorear.

- **Automoción**: Compra de coches, seguros de coches, comparación de precios de coches, seguridad vial, noticias y opiniones sobre vehículos.

- **Belleza**: Tutoriales de maquillaje, utensilios de maquillaje, peluquería, compra de productos de belleza, simuladores de maquillaje.

- **Libros y obras de consulta**: Lectores de libros, libros de referencia, libros de texto, diccionarios, tesauros, wikis.

- **Negocios**: Lectores y editores de documentos, seguimiento de paquetes, escritorio remoto, administración de correo electrónico, búsqueda de empleo.

- **Cómics**: Personajes de cómics, libros de cómics.

- **Comunicación**: Mensajes, chats/mensajería instantánea, marcadores, libretas de direcciones, navegadores, gestión de llamadas.

- **Citas**: Emparejamiento, noviazgo, mantener una relación, conocer nuevas personas, enamorarse.

- **Educación**: Preparación de exámenes, ayudas para el estudio, vocabulario, juegos educativos, aprendizaje de idiomas.

- **Entretenimiento**: Reproducción de vídeo, películas, programas de TV y contenido de ocio interactivo en streaming.

- **Eventos**: Entradas de conciertos, de eventos deportivos y de cine, reventa de entradas, tarjetas de cumpleaños.

- **Finanzas**: Banca, pagos, buscadores de cajeros, noticias financieras, seguros, impuestos, cartera y comercio, calculadoras de propinas.

- **Comer y beber:**: Recetas, restaurantes, guías gastronómicas, descubrimiento y cata de vinos, recetas de bebidas.

- **Salud y bienestar**: Bienestar personal, seguimiento de ejercicios, dietas y consejos nutricionales, salud y seguridad, etc.

- **Casa y hogar**: Búsqueda de casa y piso, reformas, interiorismo, hipotecas, inmobiliarias.

- **Bibliotecas y demos**: Bibliotecas de software, demostraciones técnicas.

- **Estilo de vida**: Guías de estilo, organización de fiestas y bodas, guías de instrucciones.

- **Mapas y navegación**: Herramientas de navegación, GPS, creación de mapas, herramientas para la circulación, transporte público.

- **Medicina**: Referencias clínicas y farmacológicas, calculadoras, manuales para el personal sanitario, noticias y revistas médicas.

- **Música y audio**: Servicios musicales, radios, reproductores de música.

- **Noticias y revistas**: Periódicos, agregadores de noticias, revistas, blogs.

- **Ser padres**: Embarazo, cuidado de bebés y de niños.

- **Personalización**: Fondos de pantalla, fondos de pantalla animados, pantalla de inicio, pantalla de bloqueo, tonos de llamada.

- **Fotografía**: Cámaras, herramientas de edición fotográfica, aplicaciones para administrar y compartir fotos.

- **Productividad**: Blocs de notas, listas de tareas, teclados, impresión, calendarios, copias de seguridad, calculadoras, herramientas de conversión.

- **Compras**: Compras online, subastas, cupones, comparaciones de precios, listas de la compra, reseñas de productos.

- **Sociedad**: Redes sociales, registro de visitas.

- **Deportes**: Comentarios y noticias deportivas, seguimiento de resultados, administración de equipos virtuales, cobertura de partidos.

- **Herramientas**: Herramientas para dispositivos Android.

- **Viajes y guías**: Herramientas para reservar viajes, compartir coche, taxis, guías de ciudades, información sobre empresas locales, herramientas de gestión de viajes, reserva de excursiones.

- **Reproductores de vídeo**: Reproductores y editores de vídeo, almacenamiento multimedia.

- **Tiempo**: Informes meteorológicos.

Las categorías de **juegos** son las siguientes: Acción, aventura, arcade, juegos de mesa, cartas, casino, casual, educativos, música, puzzles, carreras, juegos de rol, simulación, deportes, estrategia, preguntas y respuestas y juegos de vocabulario.

4. Perfilado de aplicaciones

4.1. Utilidad de las categorías de Google Play a la hora del perfilado

Para el trabajo presentado en esta memoria, es interesante la clasificación por categorías, ya que, por lo general, al agrupar las aplicaciones por categorías, se agrupan por funcionalidades similares, lo que significa que los permisos que van a utilizar también deberían de ser parecidos.

De esta manera, podemos estudiar los permisos que suelen usar las aplicaciones de cada categoría para, a la hora de analizar una nueva aplicación, comparar los permisos que esta solicita con los que suelen solicitar las demás aplicaciones de su categoría. Podemos identificar qué aplicaciones piden más permisos de lo normal para su categoría y avisar al usuario de que la aplicación que está descargando es potencialmente maliciosa. El hecho de notificar a los usuarios los permisos concretos que solicita la aplicación que han descargado y que no suelen pedir las aplicaciones similares a ella ayuda a ver qué permisos pueden ser abusivos.

Pero nos encontramos con un gran problema: la categoría a la que pertenece cada aplicación es seleccionada manualmente por su desarrollador sin que haya una comprobación por parte de Google, de tal manera que un desarrollador puede clasificar de manera errónea su aplicación y, por consiguiente, nuestro análisis no estaría mostrando resultados correctos. Por ejemplo, actualmente *Samsung Pay [16]* está categorizada como **Estilo de vida** en lugar de **Finanzas** (que es a la categoría a la que debería pertenecer), de tal forma que, al analizarla de la manera antes mencionada, estaría comparando sus permisos con los permisos típicos de las aplicaciones de Estilo de vida en lugar de las de Finanzas, dando lugar a errores. Como solución a este problema hemos decidido forzar que nuestra herramienta clasifique una aplicación mal categorizada como una aplicación potencialmente maliciosa, ya que no haría lo que, según su categoría, cabría esperar de ella.

Por otro lado, este no es el único problema que nos encontramos, ya que hay categorías (en concreto **Estilo de vida**, **Herramientas** o **Bibliotecas y demos**) que son tan amplias o difusas que contienen aplicaciones de todo tipo. Como tenemos que establecer un criterio para el análisis de las aplicaciones, hemos decidido tratarlas como "*categorías potencialmente maliciosas*" ya que las aplicaciones que contienen son tan variadas que es imposible su perfilado. Esto puede parecer una decisión un tanto drástica, pero como ocurre en el ejemplo antes expuesto de *Samsung Pay*, todas las aplicaciones categorizadas dentro de estas "*categorías potencialmente maliciosas*", pueden clasificarse perfectamente dentro de otras categorías.

Además, de esta manera ciertos desarrolladores pueden sacar ventaja del sistema de categorías, ya que Google Play al filtrar por categorías, hace una clasificación de las aplicaciones más descargadas dentro de cada una de ellas. De esta manera, un desarrollador puede definir su aplicación en una categoría sin aplicaciones similares a la suya para así destacar sobre las demás.

Por último, puede darse el caso de que la gran mayoría de las aplicaciones de una categoría pidan un permiso que no es estrictamente necesario de tal forma que, al realizar el análisis de una nueva aplicación, dicho permiso se tome como frecuente dentro de esta categoría y no salte ningún aviso. Para solucionar esto, hemos realizado un perfilado manual de las categorías en función de qué permisos deberían tener y cuáles no, según nuestro criterio (expuesto y desarrollado en el punto 4.2.2 Criterio común). Este perfilado se usaría para contrastar los resultados de tal manera que al analizar una aplicación no solo se vería si un permiso es usual dentro de la categoría de la aplicación, sino que se vería si es lógico que una aplicación de dicha categoría pida ese permiso.

Cabe destacar que, aunque al analizar una aplicación se reporte que solicita un permiso que dentro de su categoría no es frecuente y que no es razonable que solicite, según nuestro criterio, puede ser que sí sea lícito. Por ejemplo, los juegos de aventura no suelen solicitar el acceso a la cámara y no es lógico que lo soliciten, pero la aplicación **Pokemon Go [17]**, excepcionalmente la usa y es legítimo. Por tanto, este análisis sirve de guía para analizar y prever externamente si una aplicación es potencialmente peligrosa, no se trata de un análisis interno auditando si una aplicación es maliciosa o no. Un usuario recibiría avisos de los permisos que pueden ser peligrosos en la aplicación que está descargando y, junto al análisis de las funcionalidades de la aplicación que dicho usuario realice al usarla, llegaría a la conclusión de si este permiso tiene sentido que se conceda o no.

4.2. Criterios de evaluación de aplicaciones

Una vez comprendido qué hace cada permiso y la utilidad de las categorías a la hora de generar perfiles para el estudio de las aplicaciones, vamos a detallar el criterio que hemos usado para contrastarlo con los permisos más frecuentes de cada categoría. Como hemos reseñado antes, puede haber excepciones, es decir, podemos encontrar alguna aplicación que solicite permisos que según nuestro criterio no sean necesarios y que además no suelan pedir las aplicaciones de su categoría, pero que, sin embargo, sean necesarios y se usen lícitamente.

4.2.1. Categorías potencialmente maliciosas

Como hemos indicado antes, hay tres categorías que clasificaremos como "*categorías potencialmente maliciosas*" ya que su descripción es completamente distinta a las aplicaciones que contiene. Estas categorías son:

- **Estilo de vida**: aunque en la descripción que da Google para los desarrolladores señala que esta categoría está dirigida a aplicaciones como guías de estilo u organización de eventos, esta categoría está repleta de aplicaciones de todo tipo. Aquí encontramos aplicaciones como **Tinder [18]** (la cual por definición debería estar en la categoría *Citas*), multitud de aplicaciones de horóscopo o incluso aplicaciones de mensajería.

- **Herramientas**: al tener una descripción un tanto confusa (*Herramientas para dispositivos Android*), en esta categoría podemos encontrar multitud de aplicaciones que no encajan en las demás categorías y que, por lo tanto, acaban categorizadas aquí. Al tener características tan dispares, el perfilado se complica mucho, lo que podría llevar a errores de análisis de nuestra aplicación.

Por ello, pensamos que una buena solución por parte de Google sería generar más categorías para poder segmentar estas aplicaciones y no llevar a confusiones. Podemos encontrar aplicaciones de seguridad o aplicaciones que extienden la funcionalidad del dispositivo, las cuales no tienen una categoría propia y se acaban mezclando. También encontramos gran cantidad de aplicaciones que, como en el caso de *Estilo de vida*, no están bien categorizadas o simplemente los desarrolladores han optado por asignarles la categoría de *Herramientas* en lugar de la que de verdad les corresponde.

- **Bibliotecas y demos**: esta categoría es muy similar a *Estilo de vida*, ya que, aunque su descripción apunte a que está destinada para bibliotecas de software o demostraciones técnicas, podemos encontrar gran variedad de aplicaciones como juegos o aplicaciones religiosas.

Para concluir, proponemos posibles soluciones para mejorar la segmentación de las categorías.
Una de ellas podría ser, como antes hemos apuntado, la inclusión de nuevas categorías para dar mayor granularidad a la hora de poder clasificar las diferentes aplicaciones. Creemos que el hecho de que existan categorías confusas puede llevar a prácticas poco éticas como mejorar el posicionamiento de una aplicación dentro de la Google Play Store de manera irregular o clasificar erróneamente una aplicación maliciosa para conseguir más descargas.

Otra solución interesante sería la implementación de un sistema que clasifique de manera automática cada aplicación de forma que no sean los propios desarrolladores quienes lo hagan. Mediante este método no solo se libera a los desarrolladores de cada aplicación de la tarea de clasificarla, sino que se podría seguir un criterio único y uniforme para que las categorías sean más efectivas y menos susceptibles de errores.

Una última sugerencia (y la más interesante en nuestra opinión) sería la creación de una entidad independiente que certifique la correcta categorización aportando más seguridad a los usuarios a la hora de descargar aplicaciones.

Figura 5: Categorías en Google Play Store

4.2.2. Criterio común

El criterio que hemos acordado para contrastar el perfilado por categorías lo vamos a desgranar permiso por permiso. Como hemos apuntado antes, al añadir este filtro, podemos conseguir un análisis más profundo y diferenciar con mayor certeza una aplicación con permisos abusivos de una que solicita únicamente los necesarios.

- GRUPO DE PERMISOS DE ALMACENAMIENTO EXTERNO:
    - READ_EXTERNAL_STORAGE: multitud de aplicaciones necesitan este permiso para su correcto funcionamiento, ya que muchas veces es imprescindible acceder a archivos guardados en el almacenamiento externo como imágenes o documentos. Por esta razón hemos considerado que las siguientes categorías no suelen utilizarlo de forma abusiva: *Arte y diseño*, *Automoción*, *Belleza*, *Libros y obras de referencia*, *Economía*, *Cómics*, *Comunicación*, *Ligar*, *Educación*, *Entretenimiento*, *Eventos*, *Finanzas*, *Hogar*, *Medicina*, *Música y audio*, *Personalización*, *Fotografía*, *Productividad*, *Compras*, *Sociedad*, *Viajes y guías* y *Editores y reproductores de vídeo*.
    - WRITE_EXTERNAL_STORAGE: al contrario de lo que ocurre con el permiso anterior, no todas las aplicaciones necesitan modificar o guardar datos en dispositivos externos y por tanto las categorías para las que creemos que este permiso es imprescindible son las mismas que para READ_EXTERNAL_STORAGE.

- GRUPO DE PERMISOS DE UBICACIÓN: estamos acostumbrados a que las aplicaciones nos pidan la ubicación para su correcto funcionamiento. ¿Pero esto es realmente necesario? Como hemos comentado al explicar los permisos, nuestra ubicación puede dar mucha información sobre nosotros y por tanto deberíamos tener cuidado de a quién se la damos. Muchas aplicaciones necesitan saber dónde se encuentran sus usuarios para ofrecerles un mejor servicio, como conectarles a servidores más cercanos o mostrarles productos en su zona. Pero creemos que no es necesario que sean estas aplicaciones quienes recojan la ubicación y desde el punto de vista de la seguridad, es mejor que sean los propios usuarios los que aporten su zona para mejorar su experiencia de uso. A continuación, vamos a describir cuándo son necesarios cada uno de los permisos de este tipo:
    - ACCESS_BACKGROUND_LOCATION: salvo aplicaciones de *navegación o fitness* que pueden necesitar esta funcionalidad para controlar la ruta que se está tomando o para registrar el recorrido de un ejercicio, el resto no deberían necesitar acceder a la ubicación, y menos con la aplicación cerrada o en segundo plano.

- ACCESS_COARSE_LOCATION: aunque este permiso habilita únicamente la consulta de la ubicación aproximada, salvo las aplicaciones de las categorías de *Mapas y navegación* y *Salud y fitness*, creemos que no es imprescindible para el resto de las aplicaciones acceder a nuestra ubicación para poder desempeñar su función. Aunque no sea completamente necesario, entendemos que las aplicaciones de *Comunicación* o *Tiempo* solicitan estos permisos por comodidad del usuario.
- ACCESS_FINE_LOCATION: si no vemos necesario conceder la ubicación aproximada, que una aplicación acceda a nuestra localización precisa es abusivo (salvo nuevamente aplicaciones de *Mapas y navegación* y *Salud y fitness*). Por el mismo motivo que en la ubicación aproximada, las aplicaciones de Comunicación o *Tiempo* también lo usarían de manera legítima.
- ACCESS_MEDIA_LOCATION: siguiendo el argumento de que son los usuarios los que deben de introducir manualmente su ubicación, esto se aplica también a la ubicación de archivos multimedia como fotos personales. No creemos que ninguna aplicación necesite acceder a la geolocalización de los archivos multimedia del usuario.

- GRUPO DE PERMISOS DE TELÉFONO:
  - CALL_PHONE: si no deberíamos dejar que una aplicación conteste llamadas telefónicas por nosotros, tampoco deberíamos permitir que las inicie.
  - READ_PHONE_STATE: salvo aplicaciones de *Sociedad, Comunicación, Mapas y navegación* y *Salud y fitness* no creemos que este permiso pueda encajar en las funciones que ofrecen el resto de las aplicaciones.
  - READ_PHONE_NUMBERS: este permiso revela información sensible del usuario, aunque las aplicaciones de la categoría *Comunicaciones* sí que deberían poder usarlo, ya que, para este tipo de aplicaciones, conocer los números de teléfono del usuario es fundamental para su funcionalidad.
  - READ_CALL_LOG: el registro de llamadas contiene información sensible de los usuarios y no creemos que se deba acceder a él.
  - WRITE_CALL_LOG: como no permite consultar el registro de llamadas y únicamente puede añadir elementos, podría ser una función útil para las aplicaciones de *Comunicaciones*.
  - ADD_VOICEMAIL: creemos que ningún tipo de aplicación debería poder añadir mensajes al buzón de voz en el desarrollo de su actividad.

- ANSWER_PHONE_CALLS: lo mismo ocurre con este permiso, cualquier aplicación que necesite confirmaciones telefónicas de algún tipo debería dejar que los usuarios se encarguen de eso. Si se trata de una aplicación de mensajería, cada aplicación debería tener sus métodos de comunicación y en ningún caso debería poder contestar llamadas en nombre del usuario.
- PROCESS_OUTGOING_CALLS: muy similar a ACCEPT_HANDOVER y CALL_PHONE, creemos que ninguna categoría debería solicitar este permiso porque, aunque esté destinado a facilitar la comunicación entre apps, aporta más riesgos que beneficios. Si se quiere derivar la llamada a otra aplicación o servicio, desde el punto de vista de la seguridad, sería mejor cerrar una de las aplicaciones y cambiar a la otra antes que la deriva sin previa notificación al usuario.

- GRUPO DE PERMISOS DE MENSAJES DE TEXTO: existen ciertos permisos que son susceptibles de ser usados con fines de *phishing* o para realizar estafas, por ello hay que ser conscientes de a qué aplicaciones les son concedidos.
  - READ_SMS: permite a la aplicación leer los SMS entrantes, de modo que se puede leer información sensible o interceptar claves de confirmación de cambio de contraseña o de suscripción a algún servicio.
  - RECEPCIÓN Y ENVÍO DE MENSAJES: como hemos apuntado, las funcionalidades que tienen que ver con la entrada o salida de mensajes en el dispositivo son críticas en lo que a seguridad se refiere. Aunque en un principio puede parecer que las aplicaciones de *Comunicaciones* las necesitan, realmente si se requiere realizar algún intercambio de información, deberían de ser las propias aplicaciones las que se encarguen de esa tarea mediante sus propios medios. Hoy en día está muy extendida la práctica de enviar confirmaciones por SMS, y permitir que las aplicaciones accedan a este recurso supone una gran brecha de seguridad. Por tanto, hemos decidido que ninguna categoría debería solicitar los permisos:
    - RECEIVE_SMS
    - RECEIVE_MMS
    - RECEIVE_WAP_PUSH
    - SEND_SMS
    - USE_SIP

- PERMISO DE ACTIVIDAD FÍSICA
    - ACTIVITY_RECOGNITION: este es otro permiso que combinado con la ubicación puede aportar información sensible de los usuarios y salvo las aplicaciones de navegación o de ejercicio, no creemos que pueda generalizarse su concesión al resto de categorías.

- PERMISO DE MICRÓFONO:
    - RECORD_AUDIO: el uso del micrófono, al poder recopilar información sensible de los usuarios, creemos que debería estar permitido únicamente a aplicaciones de *Arte y diseño, Economía, Ligar, Entretenimiento, Comunicaciones*, *Hogar, Medicina, Crianza de hijos*, *Personalización, Música y Audio* y *Social*; ya que creemos que es necesario su uso para el correcto funcionamiento de estas aplicaciones.

- PERMISO DE SENSORES CORPORALES:
    - BODY_SENSORS: los datos médicos de los usuarios, como los que recaban los recursos a los que protege este permiso solo deberían usarlos aplicaciones de Mapas *y navegación, Medicina* y *Salud y fitness*. Ninguna otra categoría debería acceder a esta información.

- PERMISO DE CÁMARA: aunque el acceso a la cámara es una característica que, como el micrófono, puede recopilar gran cantidad de información de los usuarios, también es una funcionalidad imprescindible para muchas aplicaciones. Por ello creemos que es comprensible que las siguientes categorías pidan permisos de acceso a la cámara:
    - CAMERA: Arte y diseño, Automoción, Belleza, Economía, Comunicaciones, Ligar, Hogar, Crianza de hijos, Productividad, Compras, Medicina, Fotografía, Sociedad y Viajes y Guías**.**

- PERMISOS DE CALENDARIO: el calendario contiene información personal y de importancia para los usuarios, por tanto, ninguna aplicación debería acceder a él. De señalar un evento o concertar una cita, la aplicación podría enviar una notificación de aviso al usuario, o incluso implementar su propio calendario. Además, el usuario siempre puede anotar a mano eventos o recordatorios importantes en el calendario sin necesidad de que una aplicación medie. Permisos a los que aplica:
    - READ_CALENDAR
    - WRITE_CALENDAR

- GRUPO DE PERMISOS DE CONTACTOS: como hemos explicado antes, los contactos pueden ser usados para conocer las personas con las que se relaciona el usuario o incluso para conseguir comunicarse con ellas. Es por esto que solo deberían acceder a esta información aplicaciones de mensajería o redes sociales (en el caso del acceso a la lectura de los contactos, las aplicaciones de Finanzas suelen tener funcionalidades que requieren de los contactos para realizar transferencias o acciones similares):
    - GET_ACCOUNTS: como ocurre en el caso de la cámara, aunque acceder a la información de las cuentas del dispositivo es potencialmente peligroso, para muchas aplicaciones es vital esta funcionalidad, y por tanto hemos decidido que las siguientes categorías lo solicitan de manera legítima: *Libros y obras de consulta, Economía, Comunicaciones, Citas, Compras, Social, Mapas y navegación* y *Salud y bienestar*.
    - READ_CONTACTS: Comunicación, Finanzas y Social.
    - WRITE_CONTACTS: *Comunicación* y *Social*.

5. Diseño e implementación de la aplicación AnPerMis

Uno de los objetivos de este trabajo es desarrollar la aplicación AnPerMis que analiza las aplicaciones descargadas en el dispositivo y reporta cuáles pueden estar abusando del uso de los permisos. En este capítulo vamos a explicar cómo funciona esta aplicación y cómo la hemos implementado. También explicaremos las distintas herramientas que hemos utilizado durante el desarrollo de este trabajo.

5.1. Herramientas de desarrollo, testeo y diseño

Para el desarrollo, el testeo y el diseño de la aplicación y el servidor remoto de tratamiento de datos hemos utilizado las siguientes herramientas:

- **Canva [19]:** Para el diseño del logo se ha utilizado la herramienta Canva, que ofrece de manera intuitiva soluciones para la creación de diseños, infografías e iconos.
- **Android Studio [20]:** Es el entorno de desarrollo oficial para la implementación de aplicaciones Android. Incluye todas las características y herramientas necesarias para diseñar y crear aplicaciones para cualquier dispositivo que cuente con sistema operativo Android. Para el desarrollo de la aplicación Android de este trabajo se ha utilizado la versión 4.1.1 de Android Studio.
- **Emulador de Android AVD [21]:** Para el testeo de la aplicación Android desarrollada decidimos utilizar los dispositivos virtuales de Android AVD integrados en el entorno de desarrollo Android Studio.

Estos AVD definen las características de un teléfono o Tablet Android y nos ofrecen una emulación rápida de la aplicación dentro del entorno de desarrollo. Además, permiten simular muchas características y configuraciones que resultan muy útiles a la hora de desarrollar aplicaciones. Para el testeo y desarrollo de esta aplicación se ha utilizado un AVD que define las características de un smartphone Nexus 5x que ejecuta la versión 9.0 de Android (API 28).

Sin embargo, aunque este es el emulador que finalmente utilizamos para el testeo de nuestra aplicación, para poder tomar esta decisión tuvimos que estudiar y valorar otros emuladores con el objetivo de encontrar aquel que nos ofreciera el mejor rendimiento y cuyas características fuesen las más adecuadas. Entre los emuladores estudiados se encuentran los siguientes:

- **Android x86 [22]:** Es un proyecto que llevó el Android Project Open Source a la plataforma x86 para poder utilizarlo en todos sus dispositivos. Para testear nuestra aplicación tuvimos que instalar este proyecto en una máquina virtual. A

la hora de realizar las pruebas con este emulador surgieron muchos problemas tanto a la hora de la instalación del sistema operativo en la máquina virtual como problemas de rendimiento de las aplicaciones que utilizaban muchos recursos, por lo que tras algunas pruebas decidimos no utilizar este emulador para el testeo.

  - **BlueStacks 4 [23]:** Es un emulador Android basado en la versión 7 del sistema Android pensado para la emulación de videojuegos y aplicaciones en plataformas móviles. A la hora de realizar pruebas con este emulador conseguimos instalar y probar varias aplicaciones sin problemas, pero comprobamos que no dispone de opciones de desarrollo avanzadas que nos permitan testear correctamente nuestra aplicación en tiempo de desarrollo, aunque sí lo podemos utilizar para realizar pruebas con la versión final. Otro de los inconvenientes que presenta el uso de este emulador es que la versión Android en la que se basa (versión 7.0, API 24) es una versión obsoleta del sistema operativo que no incluye todas las características disponibles.

Por último, cabe mencionar que además del emulador integrado en el entorno de desarrollo de Android Studio también hemos utilizado nuestros propios dispositivos móviles para el desarrollo y testeo de la aplicación, que, aunque no nos permiten simular todas las funciones y configuraciones del emulador integrado sí nos permiten realizar muchas tareas de testeo externamente sin consumir tantos recursos del dispositivo donde se está desarrollando la aplicación.

- **Balsamiq Wireframes [24]:** Es una herramienta que permite diseñar interfaces de usuario para móviles o pc simulando el dibujo de estas interfaces en una hoja de papel o pizarra. Utilizaremos esta herramienta para probar los diseños que podría tener la interfaz de usuario final de la aplicación.
- **Eclipse [25]:** Es un entorno de desarrollo para desarrolladores Java que incluye todas las características y herramientas esenciales incluyendo Java IDE, un cliente Git, editor de archivos XML e integración de Maven y Gradle. Para el desarrollo de la aplicación ejecutada en el servidor remoto se ha utilizado la versión 2020-09 (4.17.0) de esta herramienta.
- **PhpMyAdmin [26]:** Es una herramienta escrita en lenguaje PHP que se encarga del manejo y administración de MySQL a través de la Web, Permite la administración de bases de datos, usuarios y permisos, además admite una amplia gama de operaciones en MySQL y MariaDB. Para el manejo y gestión de la base de datos del servidor remoto se ha utilizado la versión 4.9.5deb2 de esta herramienta.
- **Amazon Web Services (AWS) [27]:** Es una plataforma de computación en la nube de Amazon, que ofrece funcionalidades como infraestructuras de cómputo, almacenamiento y bases de datos. Utilizaremos esta plataforma para el alojamiento del servidor remoto de la aplicación.

5.2. Aplicación AnPerMis

La aplicación desarrollada en este trabajo se denomina AnPerMis, en referencia a ***An****droid* ***Per****missions* ***Mis****use*. Su funcionalidad principal consiste en estudiar la aplicación seleccionada por el usuario produciendo los siguientes resultados:

- Indicar si la aplicación utiliza los permisos de forma abusiva, en el caso de que utilice un número de permisos considerados no estándar para su categoría en forma superior a un límite prefijado.
- Mostrar los permisos que esa aplicación solicita y que no se consideran estándar para la categoría a la que pertenecen.
- Permitir al usuario revocar estos permisos y proteger mejor su privacidad.
- Reajustar automáticamente el cálculo de la idoneidad de los tipos de permisos requeridos por las aplicaciones y categorías cuando se produzca un cambio en estos.

Estos avisos se muestran de manera muy visual con un sistema de colores para que, de forma intuitiva el usuario sepa si una aplicación abusa o no del uso de los permisos y, en concreto, cuáles de los permisos no estándar de su categoría está solicitando.

Para ello, es necesario realizar un exhaustivo análisis de los permisos que usan las distintas aplicaciones dentro de una categoría. Aquellos permisos usados por más del 75% de las aplicaciones se considerarán permisos estándar y por lo tanto válidos para esa categoría.

AnPerMis ha sido diseñada para que pueda funcionar correctamente en todos los dispositivos y versiones del sistema Android. Para su desarrollo y testeo hemos utilizado el IDE de desarrollo Android Studio y el SDK de Java.

La herramienta cuenta con una parte software que es la aplicación instalada en el dispositivo Android y una parte remota que es el servidor ejecutado en la nube.

Cuando el usuario lanza AnPerMis (Figura 6) se realiza la conexión con el servidor remoto para descargar las últimas versiones de las tablas de criterios y frecuencias, necesarias para realizar el análisis de los permisos peligrosos de las aplicaciones, después, ya con las tablas actualizadas se muestra al usuario la lista de aplicaciones disponibles para el análisis.

El usuario deberá seleccionar una de las aplicaciones de la lista pulsando sobre ella, después de seleccionar la aplicación, AnPerMis buscará su categoría en Google Play Store y obtendrá una lista con los permisos peligrosos que solicita. Una vez obtenida toda la información sobre la aplicación y actualizadas las tablas locales, AnPerMis analiza la aplicación en busca de permisos que no se consideran válidos para la categoría a la que pertenecen y también envía al servidor la información de la aplicación analizada.

Tras analizar la aplicación AnPerMis muestra los resultados del análisis y permite al usuario la posibilidad de revocar aquellos permisos que considere y comentar en la página de Google Play Store de la aplicación posibles abusos.

Este funcionamiento se describe de forma exhaustiva en las siguientes secciones en base a módulos de funcionamiento.

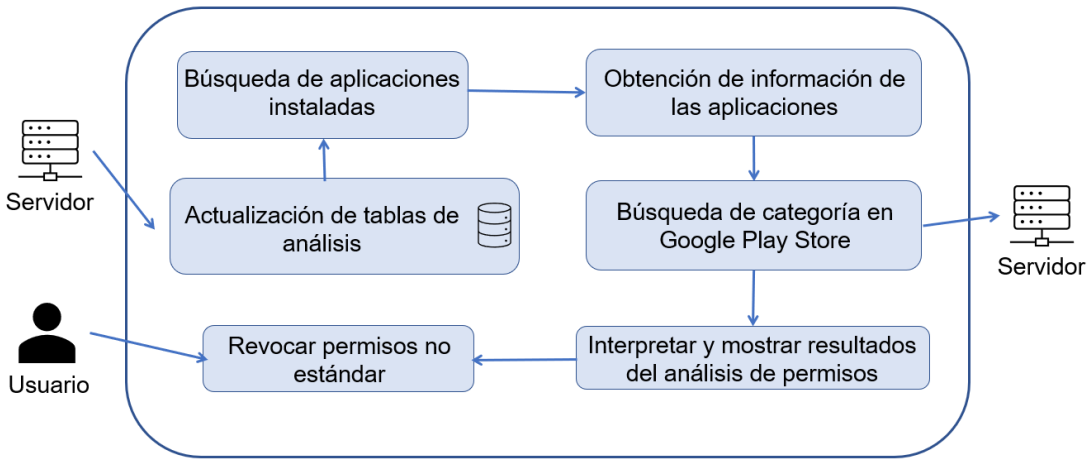


Figura 6: Arquitectura de AnPerMis.

5.2.1. Almacenamiento persistente de información (aplicación)

La aplicación almacena localmente algunos datos relevantes en el dispositivo para poder realizar el análisis de permisos, como información sobre las aplicaciones y las tablas necesarias para el análisis de permisos, ya que almacenar en caché datos relevantes para el funcionamiento de la aplicación es una práctica recomendada que mejorará su rendimiento. De este modo, cuando el dispositivo no pueda acceder a la red, el usuario podrá seguir realizando el análisis de permisos de sus aplicaciones sin ningún tipo de problema, utilizando la información almacenada.

Para manejar este almacenamiento de contenido en una base de datos local utilizaremos Room de Android [28] que "proporciona una capa de abstracción sobre SQLite para manejar la base de datos sin problemas y, al mismo tiempo, aprovechar toda la potencia de SQLite en Android".

Room contiene 3 elementos principales, como puede verse en la Figura 7:

- Base de datos: Contiene la instancia de la base de datos y sirve como punto de acceso principal para la conexión a los datos persistentes de la aplicación.

- Entidad: Representan las tablas dentro de la base de datos.
- DAO: Contiene los métodos utilizados para acceder a la base de datos.

De esta manera, la aplicación utiliza la base de datos Room para obtener los objetos de acceso a los datos DAO, que posteriormente se utilizarán para obtener y almacenar cambios en las entidades de la base de datos.

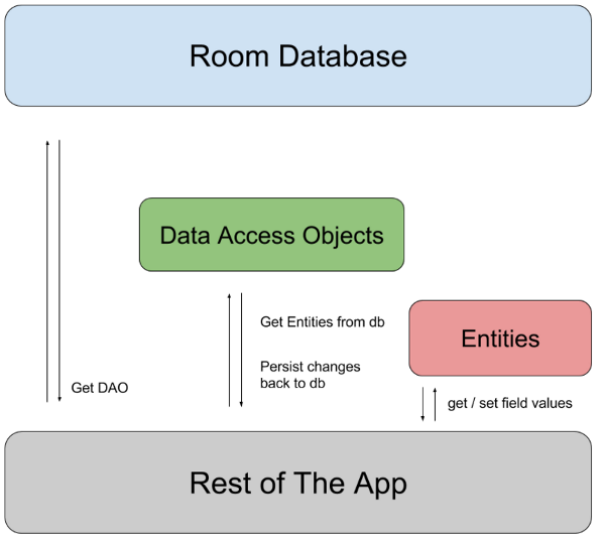


Figura 7. Descripción de los diferentes componentes de Room [28]

Una vez explicado el mecanismo que vamos a utilizar para almacenar localmente los datos relevantes podemos definir las tablas que se almacenarán en la base de datos:

- Tabla **"application"**: almacena la información necesaria de cada aplicación para realizar el análisis de permisos, es decir, información relacionada con la categoría de cada aplicación obtenida de la Google Store, nombres de los paquetes o número de permisos peligrosos. La tabla tiene los siguientes campos:
  - appPackage (primary key) - Nombre del paquete de la aplicación.
  - appCategory - Categoría de la aplicación obtenida directamente de la Google Play Store.
  - numPermisos - Número de permisos peligrosos encontrados en la aplicación.

- Tabla **"categoryFrequencies"**: almacena la información de las frecuencias de aparición de cada permiso en cada categoría disponible en Google Play y tiene los siguientes campos:
  - categoryName (primary key) - Nombre de la categoría.
  - Las siguientes columnas de la tabla representan la frecuencia de aparición de cada uno de los permisos peligrosos en la categoría

categoryName, con el siguiente formato (NombrePermiso – Frecuencia de aparición).

- Tabla "**categoryCriterio**": Esta tabla almacena información sobre el criterio de aparición de cada permiso peligroso en cada categoría y tiene los siguientes campos:
    - categoryName (primary key) – Nombre de la categoría.
    - Las siguientes columnas de la tabla representan el criterio de aparición de cada uno de los permisos peligrosos en la categoría categoryName, es decir, para cada uno de los permisos peligrosos almacena si debe aparecer en la categoría o no. Cada columna tiene el siguiente formato (NombrePermiso – criterio de aparición en la categoría).

Como hemos mencionado al principio de este apartado para poder utilizar la librería Room de Android necesitamos:

- Una base de datos, clase AppDatabase.java que la clase RoomDatabase y que contiene los métodos necesarios para crear y obtener la instancia de la base de datos local siguiendo de esta forma el patrón de diseño Singleton.

```
public static AppDatabase getInstance(Context context){

    //Si la instancia no existe se crea, si no se devuelve la instancia ya creada
    //Singleton
    if(sInstance == null) {
        synchronized (AppDatabase.class) {
            if (sInstance == null) {
                sInstance = Room.databaseBuilder(context.getApplicationContext(),
                        AppDatabase.class, DB_NAME).build();
                //Inicializamos la base de datos con datos precargados
                sInstance.populateInitialData();
            }
        }
    }
    return sInstance;
}
```

- Entidades que definen el esquema de cada una de las tablas de la base de datos, clases como AppEntity.java que define el esquema de la tabla **application**, CategoryFreEntity.java que define el esquema de la tabla **categoryFrequencies** y CategoryCriterioEntity.java que define el esquema de la tabla **categoryCriterio**.

- Objetos DAO para acceder a la información de cada una de estas entidades, clases como AppDao.java que contiene métodos de acceso a la información de la tabla **application**, CategoryFreDao.java que contiene los métodos de acceso a la información de la tabla **categoryFrequencies** y CategoryCriterioDao.java

que contiene los métodos de acceso a la información de la tabla **categoryCriterio**.

Todas estas clases las podemos encontrar dentro del **paquete database** del proyecto que incluye todas las clases necesarias para la creación y manejo de la base de datos.

Para trabajar con la base de datos local de la aplicación utilizando la librería Room de Android no podemos utilizar el hilo principal de la aplicación por motivos de diseño, ya que trabajar con la base de datos en el hilo principal de la aplicación puede provocar inestabilidades o bloqueos en la interfaz de usuario que pueden llegar a ser muy molestos.

Por este motivo hemos decidido crear el **paquete tasks** en el proyecto que contiene las clases necesarias para operar con la base de datos local fuera del hilo principal de la aplicación. Cada una de estas operaciones se han implementado utilizando AsyncTask de Android [29] que permite implementar tareas que deben realizarse en segundo plano fuera del hilo principal de la aplicación.

Las clases contenidas en este paquete son las siguientes:

- InsertAppTask.java – Inserta un objeto **Application** ya creado en la base de datos local.
- InsertCategoryFreTask.java – Inserta un objeto **CategoryFreEntity** ya creado en la tabla application de la base de datos local.
- InsertCategoryCriterioTask.java – Inserta un objeto **CategoryCriterioEntity** ya creado en la tabla categoryCriterio de la base de datos local.
- SearchAppTask.java – Busca y devuelve en caso de que exista un objeto **Application** de la base de datos local.
- SearchCategoryTask.java – Busca y devuelve las frecuencias y el criterio de aparición de cada permiso de una categoría (Objetos **CategoryFreEntity** y **CategoryCriterioEntity**), en caso de que exista la categoría en la base de datos local.

5.2.2. Comunicación con el servidor remoto.

Para realizar el tratamiento de los datos hemos decidido utilizar un servidor remoto que se encargue de procesar todos los datos recibidos por parte de las aplicaciones cliente, de esta forma conseguimos que la aplicación consuma menos recursos en los dispositivos de los usuarios haciéndola más eficiente. Esta decisión la tomamos tras intentar obtener sin éxito la información sobre el criterio de aparición de permisos en cada categoría y la frecuencia de aparición de permisos por categorías de bases de datos de terceros ya creadas, por lo que para realizar el análisis de permisos es necesario que nosotros mismos tratemos estos datos mediante el uso del servidor remoto.

Para implementar esta comunicación entre el servidor y la aplicación utilizaremos Sockets que son un mecanismo de comunicación entre procesos que se pueden ejecutar en dos máquinas distintas, como puede verse en la Figura 8.

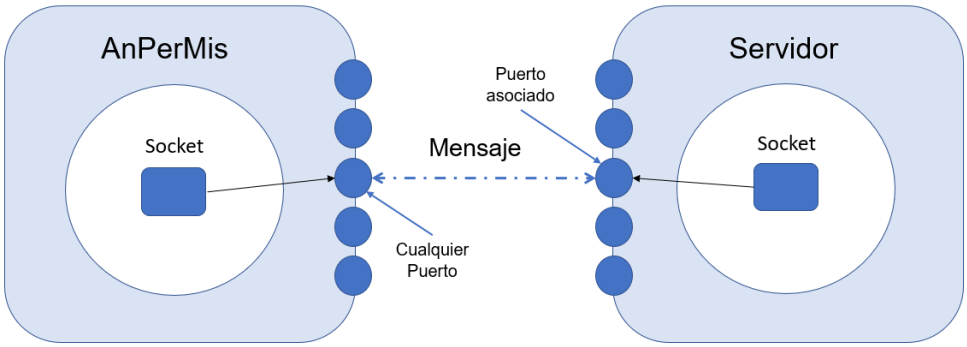


Figura 8: Ejemplo de funcionamiento de Sockets

En una de las primeras versiones de la aplicación se introdujo una implementación de esta comunicación cliente – servidor, basada simplemente en sockets sin seguridad de Android, lo que podría ocasionar graves problemas de seguridad, como la posible interceptación y manipulación de mensajes entre el cliente y el servidor (ataque "Man in the middle") o incluso la comunicación con el servidor de cualquier máquina conectada en red, incluso de aquellas en las que no se está ejecutando la aplicación.

Por este motivo, tras una revisión de este sistema de comunicación decidimos implementar el sistema realizando una conexión SSL sin autenticación de cliente (Figura 9) utilizando SocketsSSL [125] a través de la clase SSLSocket de Android, que es una clase que extiende la clase Socket y que proporciona un socket seguro utilizando protocolos como "Secure Sockets Layer" (SSL) o IETF "Transport Layer Security" (TLS), es decir, estos sockets son sockets de flujo normales que agregan una capa de protección sobre el protocolo de transporte de red como TCP. Entre estas protecciones añadidas se incluyen:

- Protección de Integridad: SSL protege contra la modificación de los mensajes entre el cliente y el servidor por parte de un interlocutor activo.
- Autenticación: SSL proporciona mecanismos de autenticación entre el cliente y el servidor.
- Confidencialidad: SSL cifra la información enviada entre el cliente y el servidor por lo que se evita que los datos puedan ser leídos por personas no autorizadas.

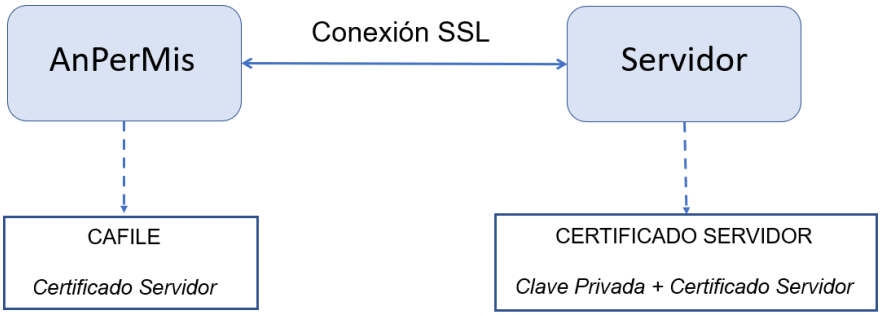


Figura 9: Descripción del funcionamiento de una conexión SSL sin autenticación de cliente

Esta implementación del sistema de comunicación se ha añadido en el **paquete socketFiles** del proyecto que incluye todas las clases necesarias para administrar toda la comunicación segura entre la aplicación cliente y el servidor remoto.

Para realizar la comunicación con el servidor remoto, una vez más, no podemos utilizar el hilo principal de la aplicación por motivos de diseño, ya que esta comunicación podría generar como hemos comentado anteriormente, inestabilidades en la aplicación y bloqueos en la interfaz de usuario. Por este motivo todas las clases incluidas en este paquete son tareas [30] que se ejecutan fuera del hilo principal de la aplicación.

Las clases incluidas en este paquete son las siguientes:

- OpenSocket.java – Esta clase se encarga de crear e iniciar la comunicación segura con el servidor remoto.
- En una primera implementación de esta clase intentamos introducir el certificado de confianza en el sistema haciendo uso del recurso de Java System.setPoperty() que se utiliza en la mayoría de las ocasiones, pero descubrimos que en sistemas Android la introducción de los certificados no se realiza de esta manera por lo que tuvimos que añadir los siguientes métodos en la clase:
    - getUpdatedKeyStore() – Se encarga de procesar e introducir en el almacén de claves el certificado necesario para la comunicación con el servidor, después devuelve el almacén de claves actualizado.

```
private KeyStore getUpdatedKeyStore() {
        KeyStore keyStore = null;
        InputStream is = null;
    //Obtenemos la instancia del almacen de claves
    keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
    //Abrimos el recurso del certificado incluido en la aplicacion
    is = context.getResources().openRawResource(certificate);
    CertificateFactory                certificateFactory                =
    CertificateFactory.getInstance(CERTIFICATE_TYPE);
    Certificate ca;
```

```
        //Generamos el certificado para introducirlo en el almacen de calves
        ca = certificateFactory.generateCertificate(is);
        keyStore.load(null,null);
        keyStore.setCertificateEntry("certTFG",ca);
        //… Captura de excepciones y cierre de flujos
    }
```

Código 1. getUpdateKeyStore()

- getTrustManagerFactory(KeyStore) - Se encarga de inicializar el TrustManager con el almacén de claves actualizado que contiene el nuevo certificado obtenido en el método getUpdatedKeyStore(TrustManagerFactory).

```
private TrustManagerFactory getTrustManagerFactory(KeyStore keyStore) {
        TrustManagerFactory trustManagerFactory = null;
        try {
            trustManagerFactory = TrustManagerFactory.getInstance(
TrustManagerFactory.getDefaultAlgorithm());
            trustManagerFactory.init(keyStore);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyStoreException e) {
            e.printStackTrace();
        }
        return trustManagerFactory;
}
```

Código 2. getTrustManagerFactory()

- getSSLContext() – Se encarga de inicializar el contexto SSL utilizando el TrustManagerFactory obtenido en el método getTrustManagerFactory().

```
private SSLContext getSSLContext(TrustManagerFactory trustManagerFactory) {
        SSLContext sslContext = null;
        try {
            sslContext = SSLContext.getInstance(DEFAULT_TLS_VERSION);
            sslContext.init(null, trustManagerFactory.getTrustManagers(),
null);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (KeyManagementException e) {
            e.printStackTrace();
        }
        return sslContext;
   }
```

Código 3. getSSLContext()

- getSSLSocketFactory() – Se encarga de obtener una factoría de SocketsSSL utilizando los recursos obtenidos en los métodos descritos

anteriormente. Esta factoría se utilizará posteriormente para obtener los SocketsSSL.

```
private SSLSocketFactory getSSLSocketFactory() {
    final KeyStore keyStore = getUpdatedKeyStore();
    final TrustManagerFactory trustManagerFactory =
     getTrustManagerFactory(keyStore);
    final SSLContext sslContext = getSSLContext(trustManagerFactory);
    SSLSocketFactory sslSocketFactory = sslContext.getSocketFactory();
    return sslSocketFactory;
}
```

Código 4. getSSLSocketFactory()

Gracias a estos métodos podremos utilizar la SSLSocketFactory obtenida en el método getSSLSocketFactory() para crear el socket seguro que nos permitirá realizar una comunicación cliente-servidor segura.

```
SSLSocketFactory sslSocketFactory = getSSLSocketFactory();
```

- CloseSocket.java: Esta clase se encarga de cerrar la comunicación con el servidor cuando la aplicación se cierra, es decir, cuando en la actividad principal se ejecuta el método onDestroy().

- SendMessage.java: Esta clase se encarga del envío de mensajes al servidor remoto. Se utilizará para enviar la información relevante de las aplicaciones al servidor utilizando el método sendMsg() que utiliza el flujo de salida del socket para enviar los mensajes tal y como los solicita el servidor remoto.

```
private void sendMsg(){
        output.println("subirapps");
        output.println(message);
        output.println("finsubida");
        Log.d(TAG, "Mensaje enviado: "+message+"\n");
 }
```

Código 5. sendMsg()

- ReceiveMessage.java: Esta clase se encarga de la recepción y procesamiento de los mensajes que envía el servidor remoto utilizando el método getSocketMsg().

```
private void getSocketMsg(){
        try{
            String message = input.readLine();
            //Si el mensaje no es el centinela
            while(!message.equals(centinelaOp)){
                procesarMsgTablaFreq(message);
                //Leemos el nuevo mensaje
                message = input.readLine();
            }
            message = input.readLine();
```

```
                while (!message.equals(centinela)){
                    procesarMsgTablaCriterio(message);
                    //Leemos el nuevo mensaje
                    message = input.readLine();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
```

Código 6. getSocketMsg()

5.2.3. Identificación de las aplicaciones instaladas.

Para conseguir la información de las aplicaciones instaladas en el dispositivo utilizamos la clase PackageManager [31] de Android que permite recuperar toda la información relacionada con los paquetes instalados en el dispositivo. Para la obtención de esta clase utilizamos Context#getPackageManager() con el flag GET_PERMISSIONS, de esta forma obtendremos los permisos que solicitan cada una de las aplicaciones junto con la información básica de los paquetes.

Tras realizar este proceso obtenemos casi toda la información necesaria para realizar el análisis, pero todavía nos falta información relevante que nos permita efectuar este estudio como los permisos peligrosos y la categoría de la Google Play Store de cada una de las aplicaciones.

```
//Crea la lista de aplicaciones que pueden ser analizadas
private void createApplicationList(){
    packageManager = getPackageManager();
    List<PackageInfo> auxAppList = packageManager.getInstalledPackages(
                    packageManager.GET_PERMISSIONS);
    for(int i = 0; i < auxAppList.size(); ++i){
        PackageInfo info = auxAppList.get(i);
        if((info.applicationInfo.flags & ApplicationInfo.FLAG_SYSTEM) == 0){
            if(!info.packageName.equals(this.getPackageName())){
                applicationList.add(new Application(packageManager,
                            info, this, null));
            }
        }
    }
}
```

Código 7. createApplicationList()

5.2.4. Extracción de los permisos peligrosos

Como hemos mencionado anteriormente gracias a la clase PackageManager de Android obtenida con el flag GET_PERMISSIONS podemos obtener diferente información relevante de las aplicaciones instaladas en el dispositivo como los permisos de los que hacen uso, el nombre del paquete, versión, etc. Esta información la obtenemos mediante la clase PackageInfo [32] de Android que proporciona los métodos necesarios para consultar el contenido de cada paquete, que se corresponde con toda la información disponible en el documento AndroidManifest.xml del que se ha hablado en la sección 3.1 Aplicaciones Android.

Una vez obtenida la información sobre los permisos que se solicitan en el AndroidManifest, como Android no distingue en este documento qué permisos son peligrosos y cuáles no, habrá que hacer esta distinción.

Para realizar la extracción de los permisos peligrosos, en primer lugar, estudiamos cuáles de estos permisos son considerados por el sistema operativo Android como peligrosos, tal y como se indica en la guía de desarrollo de Android. Una vez detectados estos permisos los almacenamos en la aplicación para que se puedan identificar con rapidez como peligrosos cuando se realice el análisis de los permisos.

5.2.5. Obtención de la categoría de cada aplicación en Google Play Store

Para realizar el análisis de los permisos es necesario conocer la categoría a la que pertenece cada aplicación en la Google Play Store, por lo que necesitamos saber cómo conseguir esta información. Pero, a la hora de buscar este dato nos encontramos con un problema, esta información no se encuentra disponible en el AndroidManifest de la aplicación sino que es seleccionada manualmente por cada desarrollador a la hora de publicar la aplicación [33], por lo que no podemos obtenerla directamente a partir del contenido del AndroidManifest que nos proporciona la clase PackageInfo.

Como posible solución a este problema se plantearon varias alternativas como obtener esta información de bases de datos de terceros, conectar con una API de Google Play Store para la obtención de información de las aplicaciones o conectar directamente con la página de cada aplicación en Google Play Store.

Finalmente, tras no encontrar ninguna base de datos con información de terceros y tampoco ninguna API en Google APIs en la que se pueda conseguir dicha información, nos decantamos por obtenerla directamente en la página de cada aplicación en la Google Play Store.

En una primera implementación para obtener la información sobre la categoría de la aplicación directamente desde Google Play Store utilizamos un script escrito en

lenguaje Python que se encargaba de conectar directamente con la página de la aplicación y procesar la respuesta obtenida para identificar la categoría de la aplicación.

```
import urllib.request
                import sys
                from bs4 import BeautifulSoup
                app = sys.argv[1];
                datos =
                urllib.request.urlopen('https://play.google.com/store/apps/details?id='+ app
                +'&hl=es&gl=US').read().decode()
                soup = BeautifulSoup(datos,features="lxml")
                tags = soup('a')
                line = ''
                for tag in tags:
                    if tag.get('itemprop') != None:
                        line = str(tag)
                line = line.split('"')
                imprimir = False
                for lin in line:
                    palabras = lin.split('/')
                    for palabra in palabras:
                        if imprimir:
                            print(palabra)
                            imprimir = False
                        if palabra == 'category':
                            imprimir = True
```

Código 8. Script en Python para obtener la categoría de una aplicación desde Google Play Store

Tras realizar varias pruebas y comprobar que este primer script en Python funcionaba correctamente, lo incorporamos a la aplicación y para ello utilizamos dos nuevas clases: la clase CategoryLoaderCallbacks.java que implementa la interfaz LoaderManager.LoaderCallbacks<> y se encarga de realizar las llamadas al cargador de categorías CategoryLoader, y la clase CategoryLoader.java que extiende de AsyncTaskLoader [34] y se encarga principalmente de realizar la carga de la categoría de cada paquete en segundo plano.

Dentro de la clase encargada de realizar la carga en segundo plano, CategoryLoader.java, encontramos algunos métodos relevantes a la hora de obtener la categoría, como son:

- **CategoryLoader#getResponse():** Se encarga de crear y realizar la conexión con Google Play Store utilizando la clase HttpsURLConnection [35] que es una clase que nos permite realizar desde la aplicación AnPerMis una conexión segura con la página de cada aplicación en la Google Play Store; de esta forma obtenemos, en el caso de que esté disponible, toda la información de la página que posteriormente se

procesa para lograr nuestro objetivo. En el caso de que una de las aplicaciones instaladas no esté disponible en la Google Play Store y, por tanto, no disponga de categoría se devolverá la categoría "SIN_INFORMACIÓN" y no se podrá realizar el análisis de los permisos.

```
private String getResponse(){
    final String BASE_URL =
    "https://play.google.com/store/apps/details?id="+appName;
    HttpsURLConnection conn = null;
    String contentAsString = null;
    InputStream is = null;
    Uri builtURI = Uri.parse(BASE_URL).buildUpon().build();
    URL requestURL = new URL(builtURI.toString());
    Log.d(TAG, builtURI.toString());

    conn = (HttpsURLConnection) requestURL.openConnection();
    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);
    conn.setRequestMethod("GET");
    conn.setDoInput(true);
    conn.connect();

    int response = conn.getResponseCode();
    if(response == 200){
        is = conn.getInputStream();
        contentAsString = convertIsToString(is);
    }else{
        contentAsString = "SIN INFORMACION";
    }
    /* … cierre de flujos y captura de excepciones */

}
```

Código 9. getResponse()

- **CategoryLoader#convertIsToString(InputStream):** Se encarga de convertir el flujo de entrada, es decir, la respuesta de la Google Play Store en una cadena que posteriormente se pueda procesar para obtener la categoría.

```
/* Metodo que convierte la respuesta en un string */
private String convertIsToString(InputStream stream){
    StringBuilder builder = null;
    BufferedReader reader = null;
    try{
        builder = new StringBuilder();
        reader = new BufferedReader(new InputStreamReader(stream));
        String line;
        while((line = reader.readLine()) != null){
            builder.append(line+"\n");
        }
        if(builder.length() == 0){
            return null;
        }
    }catch(IOException e){
```

```
            e.printStackTrace();
        }finally{
            if(reader != null){
                try{
                    reader.close();
                }catch(IOException e){
                    e.printStackTrace();
                }
            }
        }
        return builder.toString();
    }
```

Código 10. convertIsToString()

- **CategoryLoader#fromResponse(String):** Se encarga de procesar la cadena obtenida en el método CategoryLoader#convertIsToString() para conseguir la categoría de la aplicación.

```
    private String fromResponse(String s){
        String category = "";
        if(!s.equals("SIN INFORMACION")){
            int pos = s.indexOf("itemprop=\"genre\"");
            if(pos != -1){
                pos = pos + this.offset;
                category = "";
                while(s.charAt(pos) != '"'){
                    category = category + s.charAt(pos);
                    ++pos;
                }
            }
        }else{
            category = s;
        }
        return category;
    }
```

Código 11. fromResponse()

Por último, cabe mencionar que este cargador de la categoría de la aplicación se invoca desde el método AppInfoActivity#searchCategory() siempre y cuando exista conexión con la red; en caso contrario, se buscará la categoría de la aplicación en la base de datos local de la aplicación mediante la clase SearchAppTask.java.

```
public void searchCategory(){
        String pName = currentApplication.getPackageName();
        if(testNetwork()){
             Bundle bundle = new Bundle();
             bundle.putString("package_name", pName);
             LoaderManager.getInstance(this).restartLoader(CATEGORY_LOADER_ID, bundle,
                                categoryLoaderCallbacks);
         }else{
            //No hay conexion, buscamos la aplicacion en la base de datos por si la
             //hubieramos analizado antes
             SearchAppTask search = new SearchAppTask(this,
                        this,currentApplication.getPackageName());
              search.execute();
         }
}
```

Código 12. searchCategory()

5.2.6. Almacenamiento de la información de cada aplicación

Tras obtener toda la información acerca de los paquetes instalados en el dispositivo utilizando la clase PackageManager, la información contenida en cada paquete utilizando la clase PackageInfo y la información de la categoría de cada paquete en la Google Play Store mediante CategoryLoader, podemos almacenar esta información en la tabla AppEntity de la base de datos, de esta forma cuando la aplicación no disponga de conexión a Internet se podrá utilizar esta información para realizar futuros análisis.

Para almacenar esta información la aplicación utiliza la clase InsertAppTask.java que extiende de la clase AsyncTask y se encarga de insertar en segundo plano la información en la base de datos y de comunicar el resultado de la inserción al hilo principal de la aplicación.

Con esta información sobre la aplicación almacenada en la base de datos local también podremos realizar otras comprobaciones necesarias para el análisis de los permisos como detectar cambios de categoría de las aplicaciones o cambios en el número de permisos peligrosos solicitados. Detectar estos cambios ayudará a los usuarios a interpretar mejor los resultados del análisis de los permisos y a entender que la categoría de las aplicaciones y sus permisos se encuentra en constante cambio.

El método AppInfoActivity#onSearchCategoryResult() que se define más adelante es el encargado de realizar estas comprobaciones si en la base de datos local se encontró información de la aplicación perteneciente a análisis realizados anteriormente.

5.2.7. Envío de información de la aplicación al servidor remoto

Tras almacenar la información sobre la aplicación en la base de datos local, se realizará el envío de toda la información relevante obtenida sobre la aplicación al servidor remoto para que este interprete y procese toda esta información con el objetivo de mantener la tabla de frecuencias de aparición de los permisos por categorías actualizada.

Para enviar toda esta información la aplicación utiliza el método AppInfoActivity#uploadAppInfo() que se encarga de obtener el mensaje que se va a enviar utilizando el método Application#getUploadMessage() y de enviar la información sobre la aplicación al servidor utilizando la clase SendMessage.java. Se enviará al servidor un mensaje con el siguiente formato:

(nombre de la aplicación, nombre del paquete, categoría, [lista de permisos peligrosos])

```
//Envia al servidor remoto la informacion sobre la aplicacion
 private void uploadAppInfo(){
     //Si no existe conexion con el servidor no se envia la app
     if(MainActivity.output != null){
         //Si existe un output se envia la app al servidor
         String uploadMessage = currentApplication.getAppUploadMessage();
         Log.d(TAG, "Enviando app: "+uploadMessage);
         new Thread(new SendMessage(uploadMessage, MainActivity.output)).start();
     }
 }
```

Código 13. uploadInfo()

```
public String getAppUploadMessage(){
     //Establecemos el nombre de la app y el nombre del paquete tal y como pide el
      //server.
     String s = this.name + "," + this.packageName + "," + this.category;
     int i = 0;
     if(permisos != null){
         if(this.permisos.length != 0){
             if(diccionarios.getPeligrosos().contains(this.permisos[i])){
                 s = s + "," + this.permisos[i];
             }
             i++;
         }
         while(i < this.permisos.length){
             if(diccionarios.getPeligrosos().contains(this.permisos[i])){
                 s = s + "," + this.permisos[i];
             }
             i++;
         }
     }
     return s;
}
```

Código 14. getUploadMessage()

5.2.8. Clasificación de los permisos peligrosos

Una vez obtenida la información necesaria (categoría de la aplicación y lista de los permisos peligrosos) para realizar el análisis de los permisos, ya podremos realizar el análisis clasificando cada uno de estos permisos en permisos que no deberían aparecer en la aplicación y permisos que pueden aparecer en la aplicación.

Dentro de los permisos de una aplicación considerados como peligrosos, diferenciamos entre los permisos peligrosos que no deberían ser solicitados por la aplicación y los permisos que consideramos que la aplicación sí puede solicitar. Para hacer esta distinción utilizaremos la frecuencia de aparición de cada permiso por categorías y el criterio común de aparición de los permisos peligrosos explicado en la sección 4.2.2.

Esta información acerca de la frecuencia de aparición de los permisos peligrosos se obtiene del servidor remoto que es el encargado de interpretar y procesar la información sobre las aplicaciones que se analizan en la aplicación AnPerMis, generando de esta manera una tabla con las frecuencias de aparición de cada permiso peligroso por categoría.

El servidor también almacena la tabla con el criterio común de aparición de los permisos peligrosos por categoría con el objetivo de que las aplicaciones clientes puedan mantener esta información actualizada.

Estas tablas posteriormente serán enviadas a la aplicación final con el objetivo de que utilizando estas tablas pueda realizar la clasificación de los permisos peligrosos.

La recepción de esta información por parte de la aplicación se realiza con el método getSocketMsg() de la clase ReceiveMessage.java del paquete socketFiles definida anteriormente.

Una vez están disponibles en la aplicación AnPerMis los datos necesarios para realizar este análisis de permisos la aplicación clasifica los permisos siguiendo el siguiente criterio:

- Permiso peligroso que no debería aparecer: Consideramos que un permiso peligroso no debería aparecer en una aplicación si la frecuencia de aparición del permiso en aplicaciones de su categoría es inferior al 75% y además este permiso no debería de ser solicitado por aplicaciones de su categoría siguiendo el criterio común de aparición.

- Permiso peligroso que puede aparecer: Consideramos que un permiso peligroso puede aparecer en una aplicación si la frecuencia de aparición del permiso en aplicaciones de su categoría es superior al 75% y además este permiso puede ser solicitado por aplicaciones de su categoría siguiendo las indicaciones del criterio común de aparición.

Uno de los objetivos de esta clasificación es detectar si una aplicación está utilizando de forma abusiva los permisos peligrosos en comparación con las demás aplicaciones pertenecientes a su categoría, por lo que, una vez realizada la clasificación de los permisos peligrosos podemos determinar si una aplicación utiliza los permisos de forma abusiva si tiene un porcentaje mayor al 60 por ciento de permisos que no deberían aparecer en su categoría. Esta información se representa con una señal de advertencia roja en el caso de que la aplicación utilice los permisos de forma abusiva y una señal amarilla o verde en caso contrario.

El método Application#updateDangId() se encarga de comprobar el análisis global de permisos de una aplicación y de actualizar su señal de advertencia.

```
private void updateDangId(int contAbusivos){
    //Comrpobamos el porcentaje de permisos que son peligrosos
    if(totalPeligro != 0){
        int percent =  (contAbusivos*100) / totalPeligro;
        if(percent >= PORCENTAJE_PERM){
            this.dangId = R.drawable.advertencia_rojo;
        }else{
            this.dangId = R.drawable.advertencia_amarillo;
        }
    }else{
        this.dangId = R.drawable.correcto;
    }
}
```

Código 15. updateDangId()

La clasificación de los permisos peligrosos se implementa en el método getDangerousPermisions() de la clase Application que genera y devuelve una lista con los permisos peligrosos de cada aplicación ya analizados y clasificados.

```
//Obtiene una lista con los permisos peligrosos ya analizados y clasificados
public List<Permission> getDangerousPermissions(String [] permisos,
    CategoryFreEntity categoryFreEntity, CategoryCriterioEntity categoryCriterioEntity){
    Set<String> peligrosos = this.getPeligrosos();
    List<Permission> lista = new LinkedList<>();
    int imageId = 0;
    contAbusivos = 0;
    //Para cada uno de los permisos de la aplicacion actual, comprobamos si es
    peligroso y lo analizamos con los resultados de la base de datos
    if(permisos != null){
        for(int i = 0; i < permisos.length; i++){
            rojo_criterio= false;
            rojo_freq = false;
            naranja = false;
            if(peligrosos.contains(permisos[i])){
                //Obtiene el porcentaje de su categoria
                double porcentaje = categoryFreEntity.getPorcentaje(permisos[i]);
                double criterio = categoryCriterioEntity.getPorcentaje(permisos[i]);
```

```
                    imageId = getImageId(porcentaje, criterio);
                    String msg = getAlertMessage(rojo_freq, rojo_criterio,
                                        naranja, permisos[i], porcentaje);
                    lista.add(new Permission(permisos[i],
                                diccionarios.getPermInfoName().get(permisos[i]),
                                context.getString(diccionarios.getPermInfoMessage().get(permis
                                os[i]), diccionarios.getPermDescription().get(permisos[i]),
                                diccionarios.getPermInfoImage().get(permisos[i]),
                                imageId, porcentaje, msg));
                }
            }
        }
        //EStablecemos la imagen de advertencia a la aplicacion dependiendo del numero de
        //permisos que no deberian aparecer.
        updateDangId(contAbusivos);
        return lista;
    }
```

Código 16. getDangerousPermissions()

5.2.9. Mostrar los resultados del análisis

Una vez clasificados los permisos de la aplicación entre permisos peligrosos que pueden aparecer en la categoría de la aplicación y permisos peligrosos que no deberían aparecer en la categoría de la aplicación se mostrarán los resultados del análisis al usuario siguiendo el siguiente criterio de presentación.

Para cada aplicación aparecerá su icono, nombre de la categoría en la Google Play Store y una señal de advertencia de color rojo, amarillo o verde (Figura 10) con el resultado global del análisis de los permisos peligrosos de la aplicación.

- Señal Roja: Hay un porcentaje superior al 60% de permisos peligrosos en la aplicación que no deberían aparecer.
- Señal amarilla: Hay un porcentaje inferior al 60% de permisos peligrosos en la aplicación que no deberían aparecer.
- Señal verde: La aplicación no solicita ningún permiso peligroso.

A continuación, aparecerá una lista con los permisos peligrosos de la aplicación clasificados entre permisos peligrosos que no debería solicitar la aplicación y permisos peligrosos que puede solicitar la aplicación.

En cada uno de los permisos peligrosos que aparecen en esta lista podemos encontrar su nombre, nombre descriptivo del permiso para Android, y el resultado de la clasificación del permiso peligroso en forma de una señal de color amarillo, naranja o rojo (Figura 10) y la frecuencia de aparición en aplicaciones de la misma categoría.

- Señal roja: El permiso no debería solicitarse en la aplicación o aparece en menos del 50% de aplicaciones de su categoría.
- Señal naranja: El permiso puede solicitarse en la aplicación y aparece entre un 51 y un 60% de las aplicaciones de su categoría.

- Señal amarilla: El permiso puede solicitarse en la aplicación y aparece en más del 75% de las aplicaciones de su categoría.

Figura 10: Señales de advertencia

También, pulsando sobre cada uno de los permisos peligrosos que aparecen en esta lista se puede obtener información sobre el permiso, explicación detallada del resultado del análisis e información sobre cómo revocar el permiso en el caso de que el usuario lo considere necesario.

5.2.10. Revocar permisos abusivos de una aplicación

Una vez presentados los resultados del análisis de los permisos al usuario existe la posibilidad de que este tome la decisión de revocar alguno de estos permisos con el objetivo de mejorar su privacidad y seguridad. Esta funcionalidad redirige al usuario a la página de ajustes de la aplicación analizada con el objetivo de que pueda revocar los permisos que considere necesario. También ofrece al usuario más opciones como desinstalar la aplicación.

Para introducir esta funcionalidad en la parte inferior de la pantalla que muestra el análisis se mostrará el botón Ajustes que realizará la tarea de redirección a los ajustes de la aplicación.

```
//Metodo onclick del boton ajuestes que abre la pagina de ajustes de la aplicacion
    public void onClickajustes (View view){
        pulsadoAjustes = 1;
        Intent intent = new Intent((Settings.ACTION_APPLICATION_DETAILS_SETTINGS));
        intent.setData(Uri.parse("package:"+currentApplication.getPackageName()));
        startActivity(intent);
    }
```

Código 17. onClickAjustes()

Una vez en los ajustes de la aplicación el usuario podrá administrar los permisos pulsando sobre la opción permisos y deshabilitando los permisos que considere tal y como se indica en los resultados del análisis de los permisos.

Tras revocar alguno de los permisos abusivos la aplicación podría dejar de funcionar o volver a solicitar estos permisos durante el arranque de la aplicación como requisito para arrancar por lo que tras regresar de los ajustes, la aplicación avisa al usuario

mediante un cuadro de alerta de que esto es una mala práctica por parte de los desarrolladores que no debería utilizarse y lo anima a informar sobre este hecho a la comunidad de usuarios en la página de la Google Play Store de la aplicación pulsando sobre el botón Comentar que se explica en la siguiente funcionalidad. Este mensaje de alerta se invoca en el método onRestart() de AppInfoActivity, de esta forma esta alerta solo se mostrará al regresar de los ajustes de la aplicación.

```
@Override
protected void onRestart() {
    super.onRestart();
    if (pulsadoAjustes == 1 && !currentApplication.getCategory().equals(defCat)) {
        alertDialog(R.drawable.advertencia_amarillo, "RECOMENDACIÓN",
            getString(R.string.ajustes));
        pulsadoAjustes = 0;
    }
}
```

Código 18. onRestart()

5.2.11. Publicación de comentarios en Google Play Store

Tras interpretar los resultados del análisis de los permisos peligrosos de una aplicación, el usuario, como hemos mencionado anteriormente, tiene la posibilidad de revocar aquellos permisos que considere necesario. Al revocar estos permisos peligrosos puede suceder que la aplicación en cuestión deje de funcionar o vuelva a solicitar estos permisos como requisito para funcionar. Esta funcionalidad redirige directamente al usuario a la página de la Google Play Store de la aplicación para que pueda exponer este problema a la comunidad con el objetivo de que los usuarios se empiecen a concienciar sobre el uso abusivo de algunos permisos por parte de los desarrolladores.

Para implementar esta funcionalidad, se añade en la parte inferior de la pantalla que muestra los resultados del análisis el botón Comentar, que al pulsar sobre él redirige al usuario a la página de la aplicación en Google Play Store.

La implementación de este botón se puede ver en el Código 19.

```
//Metodo Onclick del boton comentar que abre la pagina de la play store en la aplicacion
public void onClickcomentar(View view){
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(
    "https://play.google.com/store/apps/details?id="+currentApplication.getPackageName()
    ));
    startActivity(intent);
}
```

Código 19. onClickComentar()

5.2.12. Modo Offline

Como hemos mencionado anteriormente una buena práctica a la hora de diseñar una aplicación es hacerlo de tal manera que cuando el dispositivo no pueda acceder a la red la aplicación se pueda seguir utilizando; por este motivo, decidimos mejorar la aplicación para que siga funcionando en la medida de lo posible con datos almacenados localmente procedentes de análisis anteriores.

Para la implementación de este modo offline se introdujeron los siguientes cambios en la aplicación:

- Creación de la tabla Application en la base de datos local que almacena datos relevantes para el análisis de los permisos de las aplicaciones como la categoría de las aplicaciones analizadas anteriormente y el número de permisos peligrosos de la aplicación. Con esta información no solo podremos analizar aplicaciones posteriormente sin conexión, sino que también podremos comprobar si los datos de una aplicación han cambiado con respecto a los análisis realizados anteriormente.
- Almacenamiento local de las tablas de frecuencias y criterios de aparición de los permisos peligrosos, con el objetivo de que puedan ser utilizadas para realizar el análisis en el caso de que el dispositivo no disponga de conexión o el servidor esté fuera de servicio.
- Modificación del método AppInfoActivity#updateCategoryResult(), ahora se añade una opción para buscar la aplicación en la base de datos local tras consultar la categoría de la aplicación en la Google Play Store, esto nos servirá para utilizar la categoría de la aplicación almacenada en la base de datos local para realizar el análisis en el caso de que no se haya podido acceder a la Google Play Store.
- Modificación del método AppInfoActivity#searchCategory(), para implementar el modo offline de la aplicación se añade una opción para buscar la categoría de la aplicación en la base de datos local cuando se detecta que el dispositivo no tiene conexión y, por lo tanto, no se podrá obtener la categoría de la Google Play Store.
- Añadido el método AppInfoActivity#onSearchAppResult() que se encarga de comprobar si se ha podido extraer la categoría de una aplicación de la base de datos y utilizarla para realizar el análisis en el caso de que el dispositivo no tenga conexión. También se encarga de comprobar si existen cambios en la información con respecto a análisis anteriores (cambios de categoría o aumento del número de permisos peligrosos), como hemos explicado anteriormente, e introducir la aplicación en la base de datos para poder utilizarla en análisis posteriores.

```
//Metodo que se ejecuta tras buscar la app en la base de datos de la aplicacion
    public void onSearchAppResult(String s, int numPeligrosos){
        //currentApplication.setTotalPeligro(c);
        currentApplication.setTotalPeligro();
        //Si la app estaba en la base de datos comprobamos si la categoria o los
```

```
        //permisos han cambiado.
        if(!s.equals("")){
            if(!currentApplication.getCategory().equals(defCat)){
                //currentApplication.setCategory("PRODUCTIVITY");
                cambioCat = isCambioCat(s);
                cambioNumPerm = isCambioNumPerm(numPeligrosos);
                //Si cambia la categoria o el número de permisos actualizamos la base de
                //datos
                if(cambioCat || cambioNumPerm){
                    //Insertamos la app en la base de datos, lo hacemos en segundo plano.
                    InsertAppTask insert = new InsertAppTask(this,
                            currentApplication.getPackageName(),
                        currentApplication.getCategory(),
                    currentApplication.getTotalPeligro());
                        insert.execute();
                    }
                    //Enviamos la informacion al servidor remoto
                    uploadAppInfo();
                }else{
                   //Modo offline, la app si esta en la bbdd y no se recibió nada de la Google
                   //Play
                    currentApplication.setCategory(s);
                }
            }else{
                //La app no existe en la base de datos
                //Si la PlayStore ha devuelto una categoría la insertamos en la base de datos
                if(!currentApplication.getCategory().equals(defCat)){
                    //Insertamos la app en la base de datos, lo hacemos en segundo plano.
                    InsertAppTask insert = new InsertAppTask(this,
                            currentApplication.getPackageName(),
                            currentApplication.getCategory(),
                            currentApplication.getTotalPeligro());
                    insert.execute();
                    uploadAppInfo();
                }
            }
            /* Buscamos la informacion de la categoria en la BBDD */
            SearchCategoryTask task = new SearchCategoryTask(this, this,
                currentApplication.getCategory());
            task.execute();

    }
```

Código 20. onSearchAppResult()

- Añadidos los métodos MainActivity#testNetwork() y OpenSocket#testIp() que se encargan, respectivamente, de comprobar si la red del dispositivo está activa y de comprobar si el servidor remoto está en línea y de esta forma controlar las conexiones con el servidor remoto de la aplicación y que solo se inicie la comunicación cuando todo funcione correctamente.

- En una primera implementación del método MainAcitivy#testNetwork utilizamos la clase Networkinfo [36] de Android que se encarga de

obtener la información de la conexión de red actual, pero durante el desarrollo de la aplicación nos dimos cuenta de que esta clase se encuentra obsoleta desde la API 29 (Android 10) por lo que cambiamos esta clase por el uso del método ConnectivityManager.getNetworkCapabilities() que realiza una tarea similar.

```
//Metodo que comprueba si la conexion a Internet esta activa
private boolean testNetwork(){
    boolean networkState = false;
    ConnectivityManager connManager =
            (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
    Network network = null;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
        network = connManager.getActiveNetwork();
        if(network != null){
            NetworkCapabilities netCap = connManager.getNetworkCapabilities(network);
            if((netCap != null) &&
               (netCap.hasTransport(NetworkCapabilities.TRANSPORT_WIFI) ||
                    netCap.hasTransport(NetworkCapabilities.TRANSPORT_CELLULAR) ||
                    netCap.hasTransport(NetworkCapabilities.TRANSPORT_ETHERNET) ||
                    netCap.hasTransport(NetworkCapabilities.TRANSPORT_BLUETOOTH))){
                networkState = true;
            }
        }
    }else{
        //La version Android del dispositivo es menor que Android 6.0 por lo que segimos
       //utilizando el metodo antiguo
        NetworkInfo networkInfo = null;
        if(connManager != null){
            networkInfo = connManager.getActiveNetworkInfo();
        }
        if(networkInfo != null && networkInfo.isConnected()){
            networkState = true;
        }
    }
    return networkState;
}
```

Código 21. testNetwork()

Por último, debemos comentar que este análisis de los permisos solo está disponible en modo offline si se ha podido conectar anteriormente con la Google Play Store y por lo tanto la aplicación está disponible en la base de datos local, es decir, el análisis de una aplicación no estará disponible si no se ha podido consultar la información de su categoría en la Google Play Store con anterioridad.

```
//Metodo que comprueba si el servidor esta en linea
private boolean testIp(String ip) throws UnknownHostException, IOException{
     boolean ipState = false;
     InetAddress test = InetAddress.getByName(ip);
     Log.d(TAG, "Enviando PING a "+ip);
     if(test.isReachable(500)){
         Log.d(TAG, ip+" esta Activo");
         ipState = true;
     }else{
         Log.d(TAG, ip+" esta Inactivo");
     }
     return ipState;
 }
```

Código 22. testIp()

5.3. Servidor

AnPerMis utiliza un servidor remoto (Figura 12) que se encarga de:

- Procesar todos los datos recibidos por parte de las aplicaciones cliente para crear y mantener actualizadas las tablas de frecuencias de aparición de los permisos peligrosos de cada categoría.
- Enviar a AnPerMis las tablas obtenidas como resultado del análisis y estudio de los permisos peligrosos para mantener toda la información necesaria para el análisis de aplicaciones actualizada.

De esta manera conseguimos que AnPerMis se mantenga actualizada y consuma menos recursos en los dispositivos de los usuarios haciéndola más eficiente.

En un principio intentamos ejecutar este servidor remoto desde una de nuestras máquinas personales, pero tras investigar el proceso para realizar esta tarea, descubrimos que podrían surgir vulnerabilidades relacionadas con la apertura de puertos en nuestra red local, por lo que finalmente decidimos alojar nuestro servidor en la nube utilizando la plataforma de computación Amazon Web Services AWS [27] que nos permite conectar las aplicaciones al servidor de manera sencilla.

La máquina virtual escogida para alojar el servidor de AnPerMis es la Amazon RDS, que utiliza un solo procesador virtual (vCPU), 1GB de memoria RAM, 50GB de almacenamiento y un sistema operativo Ubuntu Server 20.04, ya que una máquina con estas características permite ejecutar el programa servidor sin problemas.

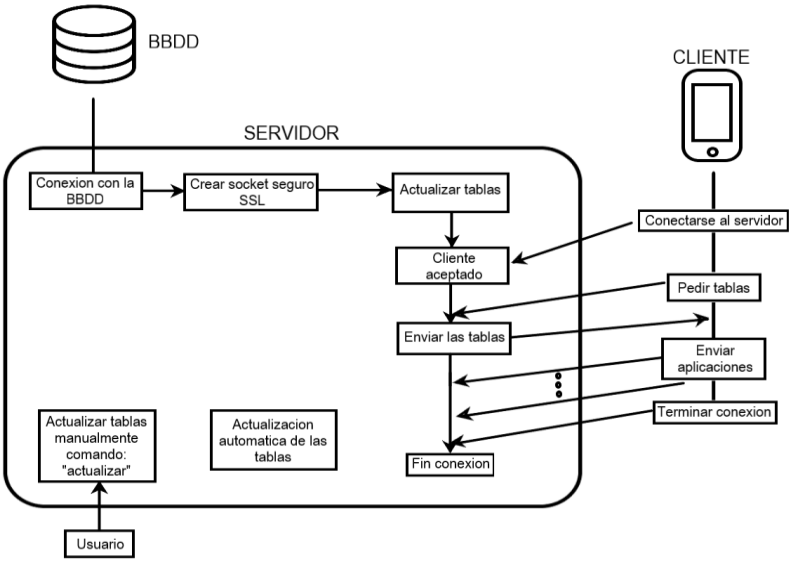


Figura 12: estructura del servidor remoto de AnPerMis

Cuando se inicia el programa servidor, primero se establece la conexión con la base de datos local y se actualizan las tablas para mantener la información actualizada, posteriormente se crea un canal de comunicación seguro con AnPerMis basado en Sockets SSL y el servidor queda a la espera de la conexión con AnPerMis.

Cuando AnPerMis inicia la comunicación con el servidor remoto, este le envía la información actualizada de las tablas de frecuencias y de los criterios necesarias para que AnPerMis pueda realizar el análisis de las aplicaciones, después queda a la espera de recibir información de nuevas aplicaciones analizadas por AnPerMis que posteriormente utilizará para actualizar las tablas.

Este funcionamiento del servidor se describe en las siguientes secciones en base a sus módulos de funcionamiento.

5.3.1. Creación de instancias virtuales en AWS

En esta sección vamos a describir los pasos que hemos seguido para configurar la instancia virtual de AWS utilizada para alojar el servidor de AnPerMis. Para cada uno de estos pasos se describen las acciones realizadas y se muestra una captura de pantalla.

El primer paso para configurar la instancia virtual es seleccionar en la consola de administración de AWS el servicio Amazon Elastic Compute Cloud [124] (Amazon EC2) [37] (Figura 13) que nos permitirá crear la instancia virtual de cómputo.

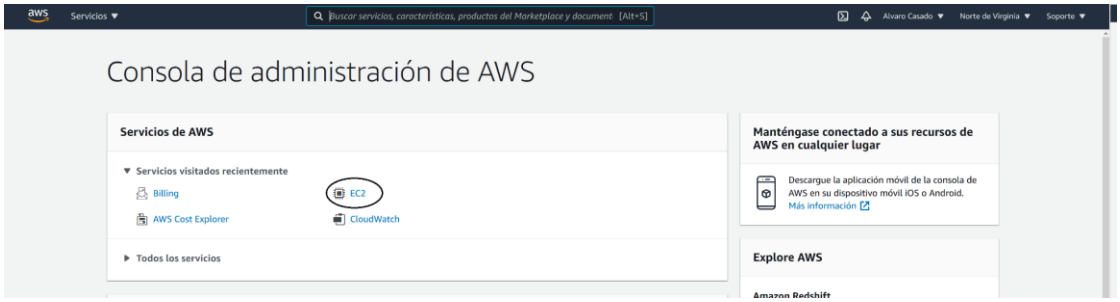


Figura 13: Seleccionar Amazon EC2

Antes de empezar con la configuración de la instancia, es necesario crear una clave público-privada que se utilizará más adelante para conectar con la máquina. Para crear esta clave público-privada seleccionaremos la opción "Crear par de claves" en el menú "Par de claves" (Figura 14).

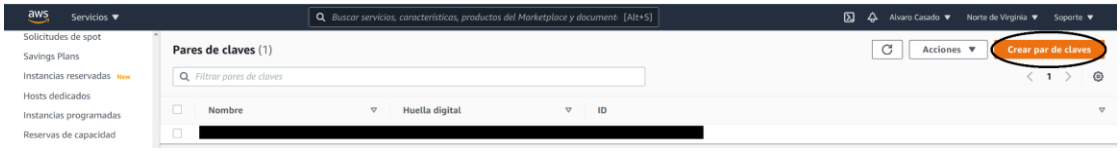


Figura 14: Seleccionar Crear par de claves

Una vez dentro de la opción "Crear par de claves" daremos un nombre a nuestro par de claves y elegiremos un formato de archivo para la clave (Figura 15), en nuestro caso la clave se llamará miClave y utilizará el formato ppk para que podamos utilizar esta instancia a través del cliente para Windows PuTTY.

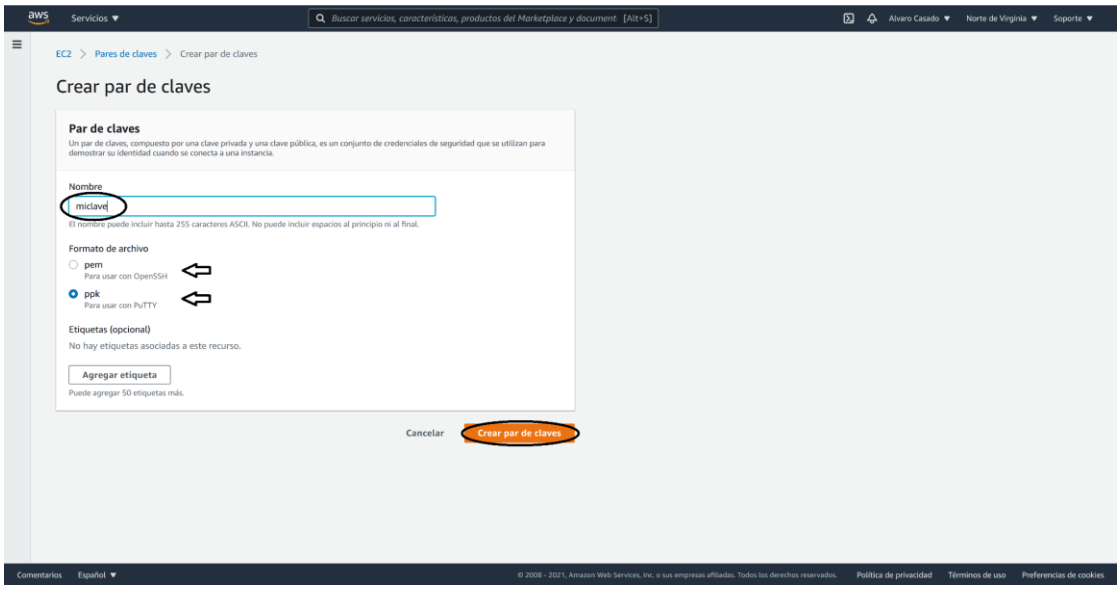


Figura 15: Crear par de claves, dar nombre y elegir formato.

Tras la creación del par de claves pasaremos a la configuración de la instancia virtual que utilizaremos para el alojamiento del servidor remoto. Para ello dentro del menú "Instancias" seleccionaremos la opción "Lanzar instancia" (Figura 16) que abrirá el configurador de instancias.

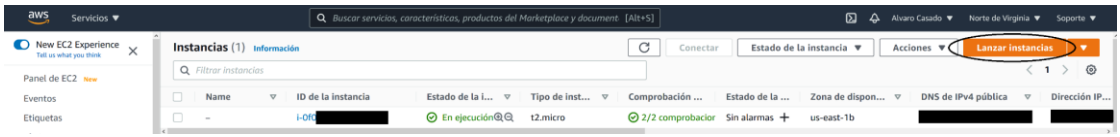


Figura 16: Instancias, lanzar instancia.

Una vez dentro de este configurador de instancias seguiremos los siguientes pasos:

Pasos 1 y 2 (Figuras 17 y 18): Seleccionamos el tipo de instancia virtual y el sistema operativo, en nuestro caso hemos seleccionado una instancia Amazon RDS con sistema operativo Ubuntu Server 20.04.

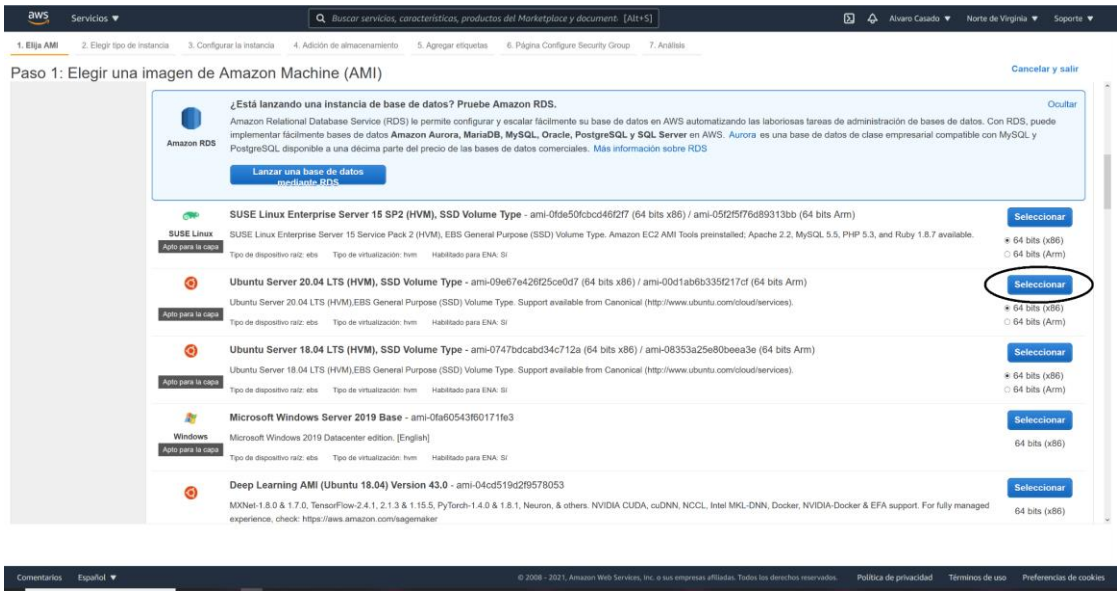


Figura 17: Configuración de la instancia, paso 1.

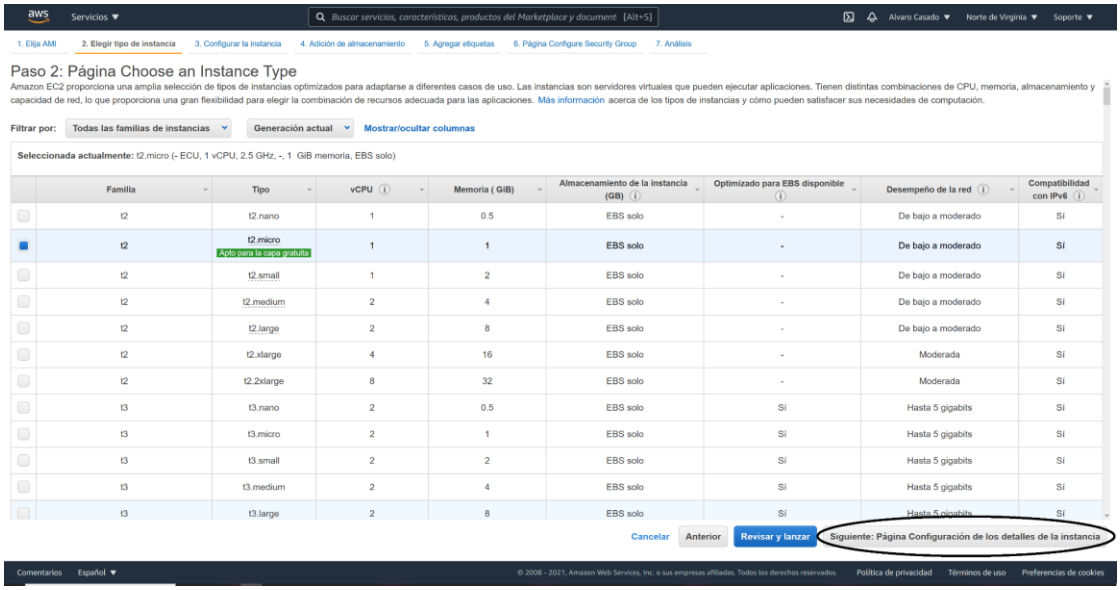


Figura 18: Configuración de la instancia, paso 2.

Paso 3 (Figura 19): En la página en la que se configura la instancia dejaremos las opciones por defecto, ya que son válidas para nuestras necesidades.

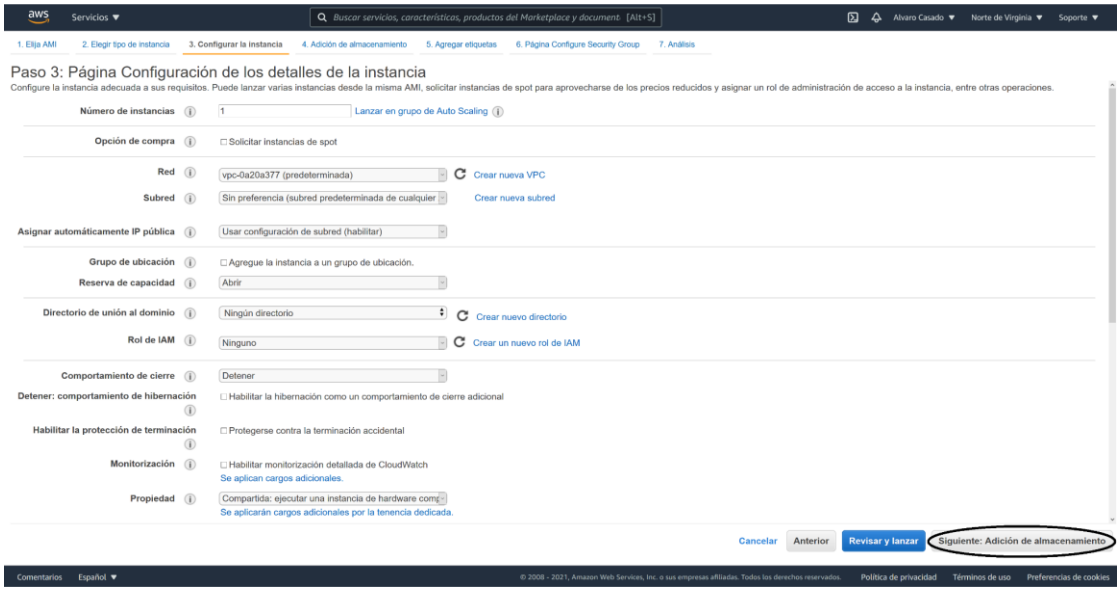


Figura 19: Configuración de la instancia, paso 3.

Paso 4 (Figura 20): Seleccionaremos la capacidad de almacenamiento de la instancia, en nuestro caso elegimos 50GB, ya que este tamaño es suficiente para alojar el programa y las bases de datos. En cualquier caso, si en un futuro se necesitara más capacidad de almacenamiento esta se ampliaría automáticamente.

Figura 20: Configuración de la instancia, paso 4.

Paso 5 (Figura 21): En esta pestaña podemos añadir etiquetas clave-valor a la instancia, para nuestra configuración no es necesario añadir ninguna de estas etiquetas.

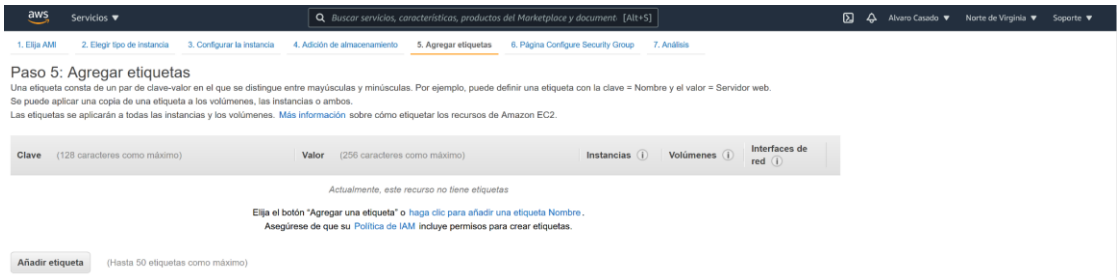


Figura 21: Configuración de la instancia, paso 5.

Paso 6 (Figura 22): En este paso configuraremos el grupo de seguridad de la instancia que es vital para que los usuarios puedan acceder al servidor. En nuestro caso para configurar este grupo de seguridad introducimos la IP de la máquina que se encarga de administrar el servidor y habilitamos el puerto 1234 para que la aplicación cliente AnPerMis pueda acceder al servidor desde cualquier dispositivo.

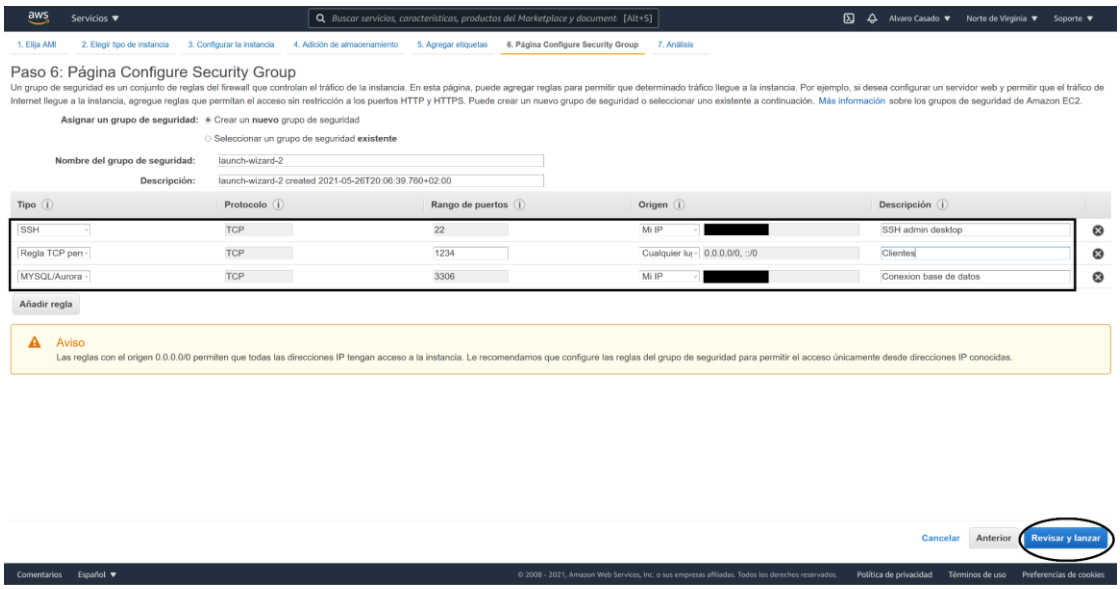


Figura 22: Configuración de la instancia, paso 6.

Paso 7: En este paso se revisa y comprueba la configuración de la instancia virtual antes de lanzar la instancia pulsando sobre la opción "Lanzar".

5.3.2. Control y gestión del servidor

Para gestionar todas sus funcionalidades el servidor remoto utiliza un controlador (Figura 23) que se encarga de realizar tareas como cargar las tablas de la base de datos para enviarlas a las aplicaciones cliente, inicializar el canal de comunicación seguro y programar la tarea que se encarga de actualizar automáticamente la información de las tablas.

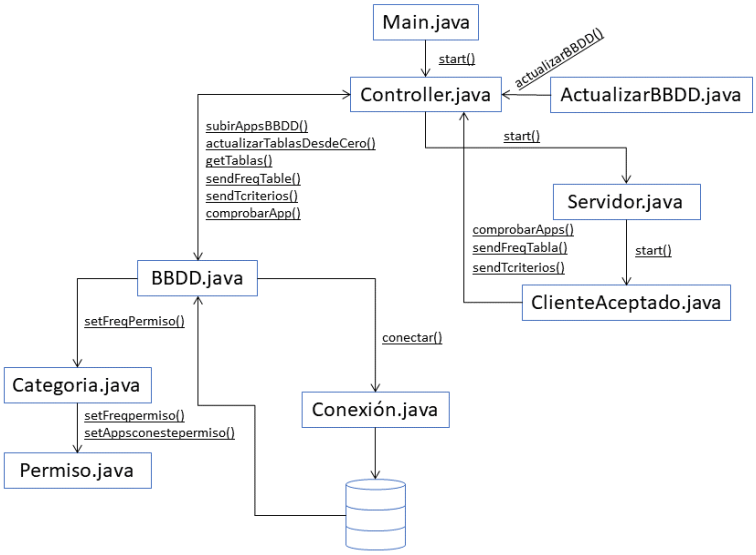


Figura 23: Diagrama de clases del servidor remoto

Para organizar todas las clases relacionados con la gestión y control del servidor hemos decidido crear el **paquete funcionalidad** que contiene las siguientes clases relevantes:

Main.java: Es la clase principal del servidor y se encarga de inicializar el servidor remoto.

Controller.java: Se encarga de la gestión y control de todo el programa servidor. Tiene los siguientes métodos relevantes:

- start(): Se encarga de inicializar el canal de comunicación seguro, cargar la información de las bases de datos e inicializar la tarea programada [38] que se encarga de la actualización automática de la información de las tablas de la base de datos.

```
public void start() throws SQLException, IOException {
    String comando = null;
    bbdd.getTablas(); //carga las tablas de la base de datos
```

```
    server.start();
    BufferedReader reader =  new BufferedReader(new InputStreamReader(System.in));
    Calendar today = Calendar.getInstance();
    today.set(Calendar.HOUR_OF_DAY, 3);
    today.set(Calendar.MINUTE, 0);
    today.set(Calendar.SECOND,0);
    Timer timer = new Timer(true);
    timer.scheduleAtFixedRate(actualizarBBDD, today.getTime(),
        ActualizarBBDD.UNA_VEZ_AL_DIA);
    while(true) {
            comando = reader.readLine();
            System.out.println("Su comando es: " + comando);
            if(comando.replaceAll("\\s","").equalsIgnoreCase("actualizar")) {
                actualizarBBDD();
            }
    }
}
```

Código 23. start()

- actualizarBBDD(): Se encarga de actualizar la información de las tablas de la base de datos cuando vence el temporizador de la tarea programada ActualizarBBDD.java.

```
public void actualizarBBDD() throws SQLException {
    System.out.println("Subiendo apps..");
    bbdd.subirAppsBBDD();
    bbdd.actualizarTablasDesdeCero();
    bbdd.getTablas();
}
```

Código 24.1 actualizarBBDD()

- sendFreqTabla(): Se encarga de obtener los datos de la tabla de frecuencias "tablafreqabs" que posteriormente se enviarán a AnPerMis.

```
public List<String> sendFreqTable() {
    return bbdd.sendFreqTable();
}
```

Código 24.2 sendFreqTable()

- sendTcriterios(): Se encarga de obtener los datos de la tabla de criterios "tcriterios" que posteriormente se enviarán a AnPerMis.

```
public List<String> sendTcriterios(){
    return bbdd.sendTcriterios();
}
```

Código 25. sendTcriterios()

- comprobarApps(): Se encarga de procesar la información de las aplicaciones que envían los clientes para posteriormente insertarlas en la caché mediante el método basededatos#comprobarApp() explicado en la sección 5.3.2.

```
public void comprobarApps(List<String> aplicaciones) {
    Iterator<String> it = aplicaciones.iterator();
    while(it.hasNext()) {
        String aplicacion= it.next();
        bbdd.comprobarApp(aplicacion);
        }
}
```

Código 26. comprobarApps()

5.3.3. Almacenamiento persistente de información en el servidor

El servidor almacena localmente datos relevantes relacionados con el análisis de los permisos obtenidos tras procesar toda la información sobre las aplicaciones analizadas por AnPerMis. De esta forma el servidor podrá actualizar y enviar esta información a AnPerMis cada vez que la solicite.

Para manejar este almacenamiento de contenido utilizaremos una base de datos relacional MySQL [39] y la herramienta PhpMyAdmin [26] que nos permitirá gestionar esta base de datos de manera eficiente.

Una vez explicado el mecanismo que vamos a utilizar para almacenar los datos en el servidor podemos definir las tablas que se almacenarán en la base de datos:

- Tabla permisos de aplicaciones **"permisosapps"**: En esta tabla se almacenan todas las aplicaciones que han sido analizadas junto con sus permisos. De esta manera disponemos de un registro de las aplicaciones con sus permisos y la categoría a la que pertenecen. La tabla tiene los siguientes campos:
    - IdApp (primary key) - Nombre del paquete, que es único, ya que no pueden existir dos aplicaciones con el mismo nombre de paquete en la Google Play Store.
    - NombreApp - Nombre de la aplicación.
    - Categoría - Categoría a la que pertenece la aplicación en la Google Play Store.
    - Las siguientes columnas de la tabla representan los permisos peligrosos que contiene la aplicación con el siguiente formato: Nombre del permiso - (0 ó 1) dependiendo de si el permiso peligroso aparece en la aplicación o no.

En la Tabla 1 podemos ver un ejemplo de los valores de los campos de la tabla de permisos de aplicaciones "permisoapps".

IdApp	Nombre	Categoría	ACCEPT_HANDOVER
com.androiid.vlc	VLC	VÍDEO	0

Tabla 1: Valores de los campos de la tabla de frecuencias

- Tabla de frecuencias **"tablafreqabs"**: Almacena las frecuencias de aparición de los permisos peligrosos en cada categoría. La tabla tiene los siguientes campos:

    - Categoría (primary key) - Categoría de la aplicación en la Google Play Store.
    - NumApps - Número de aplicaciones analizadas de esta categoría.
    - Las siguientes columnas de la tabla representan la frecuencia de aparición de cada uno de los permisos peligrosos en la categoría con formato NombrePermiso - frecuencia de aparición.

En la Tabla 2 podemos ver un ejemplo de los valores de los campos de la tabla de frecuencias "tablafreqabs".

Categoría	NumApps	ACCEPT_HANDOVER
SOCIAL	20	14

Tabla 2: Valores de los campos de la tabla de frecuencias

- Tabla de criterios **"tcriterios"**: Almacena información sobre el criterio de aparición de cada permiso peligroso en cada categoría y tiene los siguientes campos:
    - Categoría (primary key) - Nombre de la categoría.
    - Las siguientes columnas de la tabla representan el criterio de aparición de cada uno de los permisos peligrosos en la categoría Category, es decir, para cada uno de los permisos peligrosos almacena si debe aparecer en la categoría o no. Cada columna tiene el siguiente formato: NombrePermiso – criterio de aparición en la categoría.

En la Tabla 3 podemos ver un ejemplo de los valores de los campos de la tabla de criterios "tcriterios".

Categoría	ACCEPT_HANDOVER
SOCIAL	0

Tabla 3: Valores de los campos de la tabla de criterios

- Tabla de históricos de aplicaciones **"historico"**: como las aplicaciones pueden cambiar sus permisos y además pueden ser movidas de categoría, hemos decidido implementar, una tabla que guarde las anteriores versiones de las aplicaciones y la fecha en la que se añadieron a la tabla "historico".

El esquema de datos de esta tabla es el mismo que la de "permisosapps", solo se le añade una columna (Date) que se rellena automáticamente al insertar la aplicación en la tabla. La tabla tiene los siguientes campos:

  - IdApp (primary key) - Nombre del paquete, que es único, ya que no pueden existir dos aplicaciones con el mismo nombre de paquete en la Google Play Store.
  - NombreApp - Nombre de la aplicación.
  - Date - Fecha en la que se añadió la aplicación a la base de datos.
  - Categoría - Categoría a la que pertenece la aplicación en la Google Play Store.
  - Las siguientes columnas de la tabla representan los permisos peligrosos que contiene la aplicación con el siguiente formato: Nombre del permiso - (0 ó 1) dependiendo de si el permiso peligroso aparece en la aplicación o no.

En la Tabla 4 podemos ver un ejemplo de los valores de los campos de la tabla de históricos "historico".

IdApp	Nombre	Fecha	Categoría	ACCEPT_HANDOVER
com.androiid.vlc	VLC	01/06/2021	VÍDEO	0

Tabla 4: Valores de los campos de la tabla de históricos

Para trabajar con la base de datos local del servidor hemos decidido crear el **paquete basededatos** que contiene todas las clases necesarias para operar con la base de datos local.

Las clases más relevantes contenidas en este paquete son las siguientes:

Conexion.java: Se encarga de realizar la conexión con la base de datos PhpMyAdmin a través del controlador "jdbc".Tiene los siguientes métodos relevantes:

- Conectar(): Se encarga de establecer la conexión con la base de datos y devolver el objeto Connection que después se utilizará para gestionar la base de datos.

```
public Connection conectar() {
    Connection conexion = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (Exception ex) {
        System.out.println("Error, no se ha podido cargar MySQL JDBC Driver");
    }
    try {
         conexion = DriverManager.getConnection(url, user, password);
         System.out.println("conexion establecida con la base de datos");
    } catch (SQLException e) {
       e.printStackTrace();
    }
return conexion;
}
```

Código 27. conectar()

BBDD.java: Se encarga de gestionar las tablas de la base de datos local. Tiene los siguientes métodos relevantes:

- actualizarTablasDesdeCero(): Se encarga de actualizar las tablas "tcriterios" y "tablafreqabs".

```
public void actualizarTablasDesdeCero() throws SQLException {
    ejecutarUpdateQuery(queryParaEliminarTablaFreqEnMySQL());
    ejecutarUpdateQuery(queryParaActualizarTablaFreqEnMySQL());
    getNombreColumnasTablaFreq();
    getNombreColumnasTablaCriterios();
    getNombreColumnasTablaGeneral();
}
```

Código 28. actualizarTablasDesdeCero()

- Init(): Inicia la comunicación con la base de datos y actualiza sus tablas utilizando el método actualizarTablasDesdeCero().

```
private void init() throws SQLException {
    ConRealizada = conectarBBDD.conectar();
    stm = ConRealizada.createStatement();
    actualizarTablasDesdeCero();
}
```

Código 29. init()

- getcriteriosTable(): Obtiene el contenido de la tabla "tcriterios" de la base de datos y lo inserta en la estructura "tcriterios".

```
 private void getCriteriosTable() throws SQLException {
     ResultSet FTable = ejecutarSelectQuery("SELECT * FROM tcriterios");
```

```
            String categoria;
            tcriterios.clear();
            ArrayList<Integer> linea;
            // numero de aplicaciones analizadas en esa categoria
            while (FTable.next()) {
                linea = new ArrayList<Integer>();
                categoria = FTable.getString(nombreColumnastablaFreq.get(0));
                for (int i = 1; i < nombreColumnastablaCriterios.size(); i++) {
                    linea.add(FTable.getInt(nombreColumnastablaCriterios.get(i)));
                }
                tcriterios.put(categoria, linea);
            }
        }
```

Código 30. getCriteriosTable()

- getFreqTable(): Obtiene el contenido de la tabla "tablafreqabs" de la base de datos y lo inserta en la estructura "tablafreqabs".

```
    private void getFreqTable() throws SQLException {
        ResultSet FTable = ejecutarSelectQuery("SELECT * FROM tablafreqabs");
        Categoria cat;
        String categoria, nombrenumappscat;
        String nombrecolumnacategoria = nombreColumnastablaFreq.get(0);
        //numero de aplicaciones analizadas en esa categoria
        while (FTable.next()) {
            categoria = FTable.getString(nombreColumnastablaFreq.get(0));
            nombrenumappscat = nombreColumnastablaFreq.get(1);
            boolean existe = false;
            if ((cat = contieneCategoria(categoria)) != null) {
                existe = true;
            } else {
                cat = new Categoria(FTable.getString(nombrecolumnacategoria));
            }
            cat.setNumdeapps(FTable.getInt(nombrenumappscat));
            for (int i = 2; i < nombreColumnastablaFreq.size(); i++) {
                cat.setFreqPermiso(nombreColumnastablaFreq.get(i),
                    FTable.getInt(nombreColumnastablaFreq.get(i)));
            }
            if (!existe) {
                categoriasfreq.add(cat);
            }
        }
    }
```

Código 31. getFreqTable()

- sendFreqTable(): Devuelve una lista que contiene la información de la tabla "tfreqabs" preparada para enviarla a AnPerMis, con el formato: NombreCategoría; Frecuencia de aparición de permisos peligrosos separados por el carácter ";".

```
    public List<String> sendFreqTable() {
        List<String> headersList = new ArrayList<String>(nombreColumnastablaFreq);
        headersList.remove(1);// quita el elemento en la posicion indicada
        List<String> tabla = new ArrayList<String>();
```

```
        for (int fila = 0; fila < categoriasfreq.size(); fila++) {
            String row = (categoriasfreq.get(fila).getNombreCategoria());
            for (int column = 1; column < headersList.size(); column++) {
                 row = row.concat(";" +
                    String.format("%.2f",categoriasfreq.get(fila).getFreqPermiso(
                            headersList.get(column))));
            }
            tabla.add(row);
        }
        return tabla;
    }
```

Código 32. sendFreqTable()

- sendTcriterios(): Devuelve una lista que contiene la información de la tabla "tcriterios" preparada para enviarla a AnPerMis, en formato: NombreCategoría; Criterio de aparición de cada permiso peligroso separado por el carácter ";".

```
    public List<String> sendTcriterios() {
        List<String> tabla = new ArrayList<String>();
        for (Entry<String, ArrayList<Integer>> cat : tcriterios.entrySet()) {
            String row = "";
            row = row.concat(cat.getKey());
            ArrayList<Integer> permisos = cat.getValue();
            for (Integer p : permisos) {
                row = row.concat(";" + p);
            }
            tabla.add(row);
        }
        return tabla;
    }
```

Código 33. sendTcriterios()

- comprobarApp(): Se encarga de procesar toda la información de una aplicación y adaptarla para poder almacenarla en la caché de aplicaciones si no existía con anterioridad.

```
public void comprobarApp(String informacion) {// Funcion de la cache
    String[] partes = informacion.replaceAll("'", "").replaceAll("´", "").split(",");
    String idapp = partes[1].replaceAll("\\s", "");
    ArrayList<String> row;
    if (!cache.containsKey(idapp) && !partes[2].equalsIgnoreCase("SININFORMACION") &&
                !partes[2].equalsIgnoreCase("")) {
        row = new ArrayList<String>();
        for (int i = 0; i < partes.length; i++) {
            row.add(partes[i].replaceAll("\\s", ""));
        }
        try {
            cierre.lock();
            if (!cache.containsKey(idapp))
                cache.put(idapp, row);
        } finally {
            cierre.unlock();
        }
```

```
    }
}
```

Código 34. comprobarApp()

- compararMismaApp(): Comprueba si ha habido algún cambio en la información almacenada sobre los datos de la aplicación pasada como parámetro, para ello, primero comprueba si la aplicación existe en la base de datos; si existe comprueba si la información sobre la aplicación ha cambiado.

```
private boolean compararMismaApp(ArrayList<String> appbbdd,
        ArrayList<String> appcache) {
    if (appbbdd.size() == appcache.size()) {
        for (int i = 0; i < appbbdd.size(); i++) {
            if (!appbbdd.contains(appcache.get(i))) {
                return false;
            }
        }
        return true;
    } else {
        return false;
    }
}
```

Código 35. comprobarMismaApp()

- formatearResultadoSelect(): Se encarga de dar formato a la información de una aplicación obtenida de la tabla "permisosapps", es decir, la información pasa de tener formato Nombre, id, Categoría, Permisos (0 ó 1, en función de si solicita el permiso o no) a tener formato Nombre, id, Categoría, nombre completo del permiso si se solicita.

```
private ArrayList<String> formatearResultadoSelect(ArrayList<String> appbbdd)
{
    ArrayList<String> appformat = new ArrayList<String>();
    for (int i = 0; i < 3; i++)// nombre, id, categoria
        appformat.add(appbbdd.get(i));
    for (int i = 3; i < appbbdd.size(); i++) {
        int permiso = Integer.parseInt(appbbdd.get(i));
        if (permiso == 1) {
            appformat.add(nombreColumnastablaGeneral.get(i));
        }
    }
    return appformat;
}
```

Código 36. formatearResultadoSelect()

- SubirAppsBBDD(): Se encarga de actualizar las tablas "historico" y "permisosapp" de la base de datos. Para cada una de las aplicaciones que se han almacenado en la caché, comprueba si ha habido algún cambio con

respecto a la información almacenada sobre la aplicación en la base de datos. Si no ha cambiado la información de la aplicación en la base de datos no se realiza ninguna modificación en las tablas; si ha habido algún cambio se inserta la versión anterior en la tabla "histórico" y la nueva versión en la tabla "permisosapp" y si no existe la aplicación se inserta directamente.

```
public void subirAppsBBDD() {
    System.out.println("dentro funcion subirappsbbdd, apps en cache: " + cache.size());
    try {
         ArrayList<String> appbbddnoformat = new ArrayList<String>();
         ArrayList<String> appbbddwithformat;
         for (Map.Entry<String, ArrayList<String>> app : cache.entrySet()) {
            System.out.println("dentro for app: " + app.getKey());
            ResultSet rs =ejecutarSelectQuery(
                querySelectAppPermisosapp(app.getValue().get(1)));
           if (!rs.next()) {
                ejecutarUpdateQuery(queryParaActualizarTablaPermisosappsyHistorico(
                        "permisosapp", app.getValue()));
            } else {
                    for (int i = 1; i <= nombreColumnastablaGeneral.size(); i++)
                        appbbddnoformat.add(rs.getString(i));
                    appbbddwithformat = formatearResultadoSelect(appbbddnoformat);
                    appbbddnoformat.clear();
                     if (!compararMismaApp(appbbddwithformat, app.getValue())) {
                       // inserta la version anterior de la app en la tabla historico
                        ejecutarUpdateQuery(queryParaActualizarTablaPermisosappsyHistorico(
                                "historico", appbbddwithformat));
                       ejecutarUpdateQuery(queryDeleteAppPermisosapp(app.getValue().get(1)));
                        ejecutarUpdateQuery(queryParaActualizarTablaPermisosappsyHistorico(
                                "permisosapp", app.getValue()));
                        appbbddwithformat.clear();
                    }
               }
          }
          cache.clear();
     } catch (SQLException e) {
          System.err.println("Error en la ejecucion del update query en MySQL");
          e.printStackTrace();
     }
}
```

Código 37. subirAppsBBDD()

- queryParaActualizarTablaPermisosappsyHitorico(): Se encarga de generar las consultas SQL para insertar una aplicación en la tabla "permisosapp" o "historico" dependiendo del parámetro de entrada String nombretabla.

```
private String queryParaActualizarTablaPermisosappsyHistorico(String nombretabla,
        ArrayList<String> cache) {
    int max = cache.size();
    String valorpermisos = "";
```

```
    String query = "INSERT IGNORE INTO `" + nombretabla + "`(\r\n" + "
        `Nombre App`,\r\n" + "    `Id App`,\r\n"
            + "    `Categoria`";
    for (int i = 3; i < max; i++) {
        query = query.concat(", `" + cache.get(i) + "`");
        if (i + 1 == max) {
            valorpermisos = valorpermisos.concat("1");
        } else {
            valorpermisos = valorpermisos.concat("1, ");
        }
    }
    query = query.concat(") VALUES (");
    for (int i = 0; i < 3; i++) {
        query = query.concat("'" + cache.get(i) + "',");
    }
    if (max <= 3) {
        query = query.substring(0, query.length() - 1);
    }
    query = query.concat(valorpermisos + " )");
    System.out.println(query);
    return query;
}
```

Código 38. queryParaActualizarTablaPermisosappsyHistorico()

Categoria.java: Almacena la información relacionada con una categoría de la tabla "tablafreqabs", es decir, nombre, número de aplicaciones pertenecientes a la categoría e información sobre todos sus permisos peligrosos. Tiene el siguiente método relevante:

- setPermiso(): Se encarga de establecer un permiso a la categoría y actualizar el número de aplicaciones que utilizan ese permiso.

```
public void setPermiso(String permiso, int numpermisos) {
    if (permisosCategoria.containsKey(permiso)) {
        permisosCategoria.get(permiso).setAppsconestepermiso(numpermisos);
        permisosCategoria.get(permiso).setFreqpermiso(numdeapps);
    } else {
        Permiso p = new Permiso(permiso, numpermisos);
        p.set Freq permiso(numdeapps);
        permisosCategoria.put(permiso, p);
    }
}
```

Código 39. setPermiso()

Permiso.java: Almacena la información sobre un permiso, es decir, nombre, número de aplicaciones de la misma categoría que solicitan el permiso y frecuencia de aparición del permiso en la categoría. Tiene el siguiente método relevante:

- setFreqPermiso(): Se encarga de obtener la frecuencia con la que aparece el permiso dentro de su categoría.

```
public void setFreqPermiso(String permiso, int numpermisos) {
    if (permisosCategoria.containsKey(permiso)) {
        permisosCategoria.get(permiso).setAppsconestepermiso(numpermisos);
        permisosCategoria.get(permiso).setFreqpermiso(numdeapps);
    } else {
        Permiso p = new Permiso(permiso, numpermisos);
        p.set Freq permiso(numdeapps);
        permisosCategoria.put(permiso, p);
    }
}
```

Código 40. setFreqPermiso()

5.3.4. Creación de la tabla de frecuencias

Para que AnPerMis pueda realizar el análisis de los permisos peligrosos de las aplicaciones es necesario que el servidor procese toda la información recibida sobre las aplicaciones analizadas en AnPerMis y genere una tabla con la frecuencia de aparición de los permisos en las aplicaciones de su misma categoría.

Para generar esta tabla de frecuencias "tablafreqabs" se utiliza la tabla "permisosapp" que contiene la información de las aplicaciones enviadas por AnPerMis al servidor. De esta tabla se obtiene la categoría, el número total de aplicaciones analizadas que contiene la categoría y el número de aplicaciones que solicitan cada uno de los permisos en la categoría. De esta manera, podemos calcular el porcentaje de aplicaciones que utilizan cada uno de los permisos según su categoría.

Esta tabla de frecuencias se genera con el método basededatos#actualizarTablasDesdeCero() utilizando la consulta SQL obtenida mediante el método basededatos#queryParaActualizarTablaFreqEnMysql().

5.3.5. Histórico de aplicaciones

La información relevante para el análisis de las aplicaciones suele cambiar constantemente, por lo que, para detectar posibles cambios en la información de las aplicaciones como cambios de categoría, número de permisos y nombre de la aplicación hemos decidido crear la tabla "historico" en la base de datos. En esta tabla se almacena información sobre las versiones anteriores de las aplicaciones ya analizadas junto con la fecha en la que se insertaron en la tabla.

5.3.6. Caché de aplicaciones

Para mejorar el rendimiento del servidor y evitar una variación errónea de las frecuencias de uso de los permisos peligrosos hemos decidido introducir en el servidor una caché que almacena las aplicaciones que analiza AnPerMis en el cliente, de esta

manera si una aplicación ya ha sido analizada con anterioridad no se vuelve a introducir en la base de datos local. De esta forma la información de las tablas del servidor solo se actualizará una vez al día utilizando los datos almacenados en esta caché de aplicaciones mejorando así el rendimiento. Esta tarea se lleva a cabo con el método basededatos#subirApps() que se encarga de actualizar las tablas "histórico" y "permisosapp" con la información de esta caché como hemos explicado en la sección 5.3.2. Almacenamiento persistente de información en el servidor.

5.3.7. Actualización automática de la base de datos

Una vez arrancado el servidor remoto las aplicaciones cliente se pueden comunicar con él para actualizar sus tablas locales que serán utilizadas para realizar el análisis de los permisos y para enviar las nuevas aplicaciones analizadas, de esta forma el servidor podrá utilizar esta nueva información para mantener sus tablas actualizadas y ampliar los resultados del análisis de los permisos.

Para añadir toda esta nueva información recibida de AnPerMis decidimos que el servidor procese y actualice toda esta información automáticamente cada día a las 3:00 de la mañana, de esta forma el servidor mantendrá todas sus tablas de análisis de los permisos actualizadas sin penalizar su rendimiento en las horas de mayor actividad.

Para implementar esta actualización automática de la base de datos del servidor utilizamos la clase ActualizarBBDD.java que extiende de la clase TimerTask [40] que permite programar la ejecución de una tarea utilizando un temporizador. La tarea ActualizarBBDD.java se encargará de iniciar el método Controller#actualizarBBDD() que actualiza todas las tablas de la base de datos con la información disponible en la caché.

```
public void run() {
    try {
        ctrl.actualizarBBDD();
    } catch (SQLException e) {
        System.err.println("Error en la clase ActualizarBBDD");
        e.printStackTrace();
    }
}
```

Código 41. ActualizarBBDD#run()

5.3.8. Comunicación entre la aplicación cliente AnPerMis y el servidor

Para el sistema de comunicación con AnPerMis hemos decidido implementar una conexión SSL sin autenticación de cliente (Código 42) utilizando SocketsSSL a través de las clases SSLServerSocket y ServerSocket de Java. De esta forma la aplicación cliente AnPerMis y el servidor podrán intercambiar información a través del canal de comunicación seguro creado.

Para implementar la conexión SSL [41] sin autenticación de cliente es necesaria la creación de un certificado servidor que incluye el certificado del servidor junto con la clave privada. Para generar este certificado servidor hemos utilizado la herramienta keytool que es una herramienta java que permite el manejo de almacenes de claves, cadenas de confianza y creación y manejo de certificados de confianza.

Esta implementación del sistema de comunicación se ha añadido en el paquete server que incluye todas las clases necesarias para crear y administrar la comunicación segura entre el servidor y la aplicación cliente AnPerMis.

Para realizar esta comunicación con el cliente AnPerMis hemos decidido utilizar tareas que se ejecutan fuera del hilo principal del servidor para que el servidor pueda administrar conexiones con varios clientes simultáneamente y para evitar bloqueos de ejecución. Por este motivo todas las clases incluidas en este paquete serán tareas que se ejecutan fuera del hilo principal.

Las clases incluidas en este paquete son las siguientes:

Servidor.java: Se encarga de configurar e iniciar la comunicación con la aplicación cliente AnPerMis, es decir, prepara un canal de comunicación seguro utilizando el certificado de servidor creado y asigna al cliente AnPerMis un hilo propio a través de la clase ClienteAceptado.java. Para incluir el certificado en el almacén de claves utilizaremos el recurso de Java System.setProperty().

```
public void run()//Método para iniciar el servidor  {
   try {
       File archivo = new File("certificadoTFG.jks");
       if(archivo.exists())
           System.out.println("certificadoTFG.jks existe");
       else
           System.out.println("certificadoTFG.jks NO existe");
           System.setProperty("javax.net.ssl.keyStore","certificadoTFG.jks");
           System.setProperty("javax.net.ssl.keyStorePassword","**********");
           SSLServerSocketFactory sslserversocketfactory
                       =(SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
           ServerSocket sslserversocket =
               sslserversocketfactory.createServerSocket(1234);
           while (true) {
                System.out.println("Esperando..."); // Esperando conexión
                // Accept comienza el socket y espera una conexión desde un cliente
                // se ejecuta un hilo para un solo cliente
                 ClienteAceptado tunel = new
                       ClienteAceptado(sslserversocket.accept(),ctrl);
                  tunel.start();
           }
    } catch (Exception e) {
       System.out.println(e.getMessage());
    }
}
```

Código 42 Servidor#run()

ClienteAceptado.java: Se encarga de realizar todo el proceso de comunicación con el cliente fuera del hilo principal del servidor, es decir, envía las tablas necesarias para que AnPerMis pueda realizar el análisis de los permisos peligrosos de las aplicaciones y recibe los datos de las aplicaciones analizadas. Este proceso se ha implementado mediante un bucle que finaliza cuando se recibe un mensaje nulo o el mensaje "finconexion". Dentro del bucle el servidor espera los mensajes del cliente que pueden ser:

- "enviartabla": Si se recibe este mensaje el servidor enviará las tablas "tfreqabs" y "tcriterios" al cliente.
- "subirapps": Si se recibe este mensaje el servidor se prepara para recibir la información de una aplicación analizada.

```
public void run() {
    String mensajeServidor,texto = "";
    try{
        System.out.println("Cliente en línea " + cliente.getInetAddress());
        //Se obtiene el flujo de salida del cliente para enviarle mensajes
        PrintWriter salidaCliente1 = new PrintWriter(new BufferedWriter(
                new OutputStreamWriter(cliente.getOutputStream())),true);
        //Se le envía un mensaje al cliente usando su flujo de salida
        //salidaCliente1.println("Petición recibida y aceptada");
        //Se obtiene el flujo entrante desde el cliente
        BufferedReader entrada1 = new BufferedReader(
                new InputStreamReader(cliente.getInputStream()));
        mensajeServidor = entrada1.readLine();
        texto = mensajeServidor.replaceAll("\\s","");

        while(mensajeServidor  != null &&
                !texto.equalsIgnoreCase("finconexion")){
            System.out.println(texto);
            if(texto.equalsIgnoreCase("enviartabla")) {
                //El servidor le enviara la tabla actualizada al cliente
                for(String row:ctrl.sendFreqTabla()){
                    salidaCliente1.println(row);
                }
                salidaCliente1.println("finenviotablafreqs");
                for(String row:ctrl.sendTcriterios()){
                    //System.out.println(row);
                    salidaCliente1.println(row);
                }
                salidaCliente1.println("finenvio");
            }else if(texto.equalsIgnoreCase("subirapps")) {
                comprobarApps(salidaCliente1, entrada1);
            }else {
                salidaCliente1.println("Lo siento, su peticion no es valida");
            }
            mensajeServidor = entrada1.readLine();
            if(mensajeServidor != null)
            texto = mensajeServidor.replaceAll("\\s","");
        }
        System.out.println("Fin de la conexión " + cliente.getInetAddress());
        cliente.close();//Se finaliza la conexión con el cliente
    }
    catch (Exception e)
    {
        System.out.println(e.getMessage());
    }
}
```

Código 43. ClienteAceptado#run()

En esta clase también se incluye el método comprobarApps() que se encarga de procesar e insertar en la caché de aplicaciones la información de las aplicaciones recibida cuando el cliente utiliza el mensaje “subirapps”.

```
private void comprobarApps(PrintWriter salida, BufferedReader entrada) throws
        IOException {
    String mensaje;
    List<String> aplicaciones = new ArrayList<String>();
    mensaje = entrada.readLine().replaceAll("\\s","");
    while(!mensaje.equalsIgnoreCase("finsubida")) {
        System.out.println(mensaje);
        aplicaciones.add(mensaje);
        mensaje = entrada.readLine().replaceAll("\\s","");
    }
    if(!aplicaciones.isEmpty())
        ctrl.comprobarApps(aplicaciones);
}
```

Código 44. ClienteAceptado#comprobarApps()

6. Interfaz de usuario y funcionalidades de la aplicación

En este apartado vamos a describir el proceso y los recursos utilizados para realizar el diseño de la aplicación, también se mostrarán los resultados de diseño obtenidos y la funcionalidad de la aplicación.

6.1. Diseño de la interfaz gráfica

Una vez que hemos decidido cuáles van a ser las funcionalidades del sistema y su implementación, comenzamos el diseño de la interfaz de usuario con el objetivo de que sea una interfaz rápida, comprensible y fácil de usar, para que el usuario de la aplicación pueda entender de una forma muy sencilla cuáles son los resultados y los objetivos del análisis de los permisos y que pueda tomar las decisiones que considere oportunas.

Para realizar el primer diseño de esta interfaz de usuario utilizamos la herramienta Balsamiq Wireframes [24] que nos permitió diseñar de una manera rápida la primera versión de la interfaz de usuario de la aplicación (muy similar a la interfaz final). En la Figura 24 se muestran las pantallas utilizadas en este primer diseño.

A partir de este primer diseño de la interfaz se han ido introduciendo distintas mejoras que se describen en el siguiente apartado, hasta llegar al diseño final de la aplicación mostrado en el punto 6.2.

Figura 24: Primer diseño de la interfaz de usuario de AnPerMis

6.1.1. Recursos de diseño utilizados

Una vez definido el primer diseño de la interfaz de usuario de la aplicación vamos a utilizar los siguientes recursos gráficos del sistema Android que nos permitirán diseñar la interfaz final de la aplicación.

- **ConstraintLayout [42]:** Es un recurso gráfico del sistema Android que permite crear diseños complejos y adaptativos mediante el uso de restricciones. Cada restricción representa una conexión o alineación con otra vista, el diseño de nivel superior o una guía invisible. De esta forma haciendo uso de este recurso en los layouts de las actividades de la aplicación conseguimos que la aplicación se vea correctamente en la mayoría de los dispositivos actuales.

- **RecyclerView [43]:** Es un componente gráfico que permite la creación de un contenedor de elementos desplazables, rápido y eficiente para mostrar grandes conjuntos de datos. Como su nombre indica RecyclerView recicla los elementos de la lista de tal forma que cuando los elementos se desplazan fuera de la pantalla no se destruyen, mejorando el rendimiento y la capacidad de respuesta de la aplicación y reduciendo el consumo de energía.

Este recurso lo utilizaremos para crear las listas de aplicaciones en la pantalla principal de la aplicación y para crear la lista de los permisos peligrosos de cada aplicación.

Para crear estas listas de aplicaciones y permisos utilizando RecyclerView, primero agregaremos el elemento RecyclerView a las actividades en las que lo necesitamos, es decir, en MainActivity que muestra la lista de aplicaciones y en AppInfoActivity que muestra la lista de permisos, estableciendo las construcciones necesarias para que este elemento se muestre correctamente en todos los dispositivos donde vayamos a utilizar la aplicación gracias al recurso ConstraintLayout.

Una vez introducidos estos elementos RecyclerView en las actividades donde se necesitan definiremos las siguientes clases que se encargarán de definir el contenedor de vistas y el adaptador para ajustar cada uno de los elementos de la lista. Estas tareas las realizan las clases ApplicationAdapter.java y PermissionAdapter.java.

- ApplicationAdapter.java: Extiende de la clase RecyclerView.Adapter y se encarga de definir el contenedor de vistas de las aplicaciones, mediante la clase interna ViewHolder que extiende de RecyclerView.ViewHolder, y el adaptador de las aplicaciones que se definirán dentro de la lista. Utiliza los siguientes métodos relevantes:

ApplicationAdapter#onCreateViewHolder(): Se encarga de definir el layout de cada uno de los elementos aplicación de la lista y devuelve la definición del contenedor de vistas con el layout de cada uno de los elementos aplicación introducidos.

```
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
      View itemView = layoutInflater.inflate(R.layout.layout_app_card,
                parent, false);
     return new ApplicationAdapter.ViewHolder(itemView, this, onAppListener);
}
```

Código 45. ApplicationAdapter#onCreateViewHolder()

ApplicationAdapter#onBindViewHolder Se encarga de establecer los valores de cada una de las aplicaciones de la lista en el contenedor que los mostrará en la lista.

```
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Application application = applicationList.get(position);
        Drawable appIcon = application.getAppIcon();
        appIcon.setBounds(0, 0, 150, 150);
        holder.icon.setCompoundDrawables(appIcon, null, null, null);
```

```
        holder.icon.setCompoundDrawablePadding(30);
        holder.name.setText(application.getName());
        holder.version.setText(application.getVersion());
}
```

Código 46. ApplicationAdapter#onBindViewHolder()

ViewHolder#ViewHolder(): Se encarga de buscar en el layout de cada una de las aplicaciones los identificadores de los campos donde ha de introducirse la información utilizando el método ApplicationAdapter#onBindViewHolder().

```
public ViewHolder(@NonNull View itemView, ApplicationAdapter adapter,
        OnAppListener listener) {
    super(itemView);
    icon = itemView.findViewById(R.id.appIcon);
    name = itemView.findViewById(R.id.appName);
    version = itemView.findViewById(R.id.appVersion);
    this.listener = listener;
    itemView.setOnClickListener(this);
 }
```

Código 47. ApplicationAdapter#ViewHolder#ViewHolder()

- PermissionAdapter.java: Extiende de la clase RecyclerView.Adapter y se encarga de definir el contenedor de vistas de los permisos, mediante la clase interna ViewHolder que extiende de RecyclerView.ViewHolder, y el adaptador de los permisos que se definirán dentro de la lista. Utiliza los siguientes métodos relevantes:

PermissionAdapter#onCreateViewHolder(): Se encarga de definir el layout de cada uno de los permisos de la lista y devuelve la definición del contenedor de permisos con el layout de los elementos permiso introducido.

```
 public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View itemView = layoutInflater.inflate(R.layout.layout_perm_card,
                parent, false);
        return new ViewHolder(itemView, this, onPermListener);
 }
```

Código 48. PermissionAdapter#onCreateViewHolder()

PermissionAdapter#onBindViewHolder Se encarga de establecer los valores de cada uno de los permisos de la lista en el contenedor que los mostrará. En este método, además, se distingue el caso en el que no se dispone de información sobre la categoría de la aplicación y, por lo tanto, no se puede realizar el análisis. Para distinguir este caso solo se mostrará el nombre del permiso y la imagen en el elemento de la lista, el resto de información se ocultará al usuario.

```
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Permission permission = permissionList.get(position);
        holder.permImage.setImageResource(permission.getPermissionImage());
        holder.permName.setText(permission.getName());
        holder.permComplete.setText(permission.getCompleteName());
        if(category.equals("SIN INFORMACION")){
            holder.permCircle.setVisibility(View.GONE);
            holder.permPorcent.setVisibility(View.GONE);
            holder.comentPorcent.setVisibility(View.GONE);
        }else{
           holder.comentPorcent.setText("Aparece en el " +
                        (int)(permission.getPorcentaje()*100)+
                        "% de las apps de esta categoría.");
           holder.permCircle.setImageResource(permission.getCircleImage());
           holder.permPorcent.setText((int)(permission.getPorcentaje()*100)
                 +"%");
        }
  }
```

Código 49. PermissionAdapter#onBindViewHolder()

ViewHolder#ViewHolder(): Se encarga de buscar en el layout de cada uno de los permisos los identificadores de los campos donde ha de introducirse la información utilizando el método PermissionAdapter#onBindViewHolder().

```
public    ViewHolder(@NonNull    View    itemView,    PermissionAdapter    adapter,
OnPermListener listener) {
    super(itemView);
    this.permImage = itemView.findViewById(R.id.permIcon);
    this.permName = itemView.findViewById(R.id.permName);
    this.permComplete = itemView.findViewById(R.id.permComplete);
    this.permCircle = itemView.findViewById(R.id.permCircle);
    this.permPorcent = itemView.findViewById(R.id.permPorcen);
    this.comentPorcent = itemView.findViewById(R.id.comentarioPorcent);
    this.listener = listener;
    itemView.setOnClickListener(this);
}
```

Código 50. PermissionAdapter#ViewHolder#ViewHolder()

- **Cardview [44]:** Este recurso gráfico de Android permite mostrar de manera sencilla la información de cada uno de los elementos de una lista en tarjetas que tienen un estilo coherente para el contenedor. Estas tarjetas se muestran con una elevación sobre el resto de las vistas que las contienen de modo que el sistema puede dibujar sombras debajo de ellas aportando una apariencia bonita a la interfaz de usuario. Utilizaremos estas tarjetas para representar cada uno de los elementos de las listas visuales de la aplicación como Aplicaciones y Permisos.

Para crear estas tarjetas CardView para los elementos Aplicación y Permisos tendremos que crear un nuevo layout para cada uno de ellos (layout_app_card.xml y layout_perm_card.xml) que incluya como contenedor el recurso CardView y posteriormente introducir e inflar estos Layout dentro de la lista RecyclerView correspondiente como explicamos al describir el recurso RecyclerView.

```
View itemView = layoutInflater.inflate(R.layout.layout_app_card, parent, false);
View itemView = layoutInflater.inflate(R.layout.layout_perm_card, parent, false);
```

Código 51. Creación de CardView de aplicaciones y permisos

- **AlertDialog [45]:** Es un recurso gráfico de Android que muestra información al usuario en un cuadro de diálogo que no ocupa toda la pantalla. La clase AlertDialog es una clase que extiende de la clase Dialog de Android y que permite la creación de un diálogo que puede mostrar un título, un contenido, una lista de elementos y hasta 3 botones.

En nuestro diseño se utilizarán estos cuadros de diálogo para mostrar las advertencias sobre, categorías peligrosas, cambio de categoría, cambio del número de permisos peligrosos, recomendaciones y para mostrar la información de los resultados del análisis de cada permiso al usuario sin la necesidad de crear una nueva actividad.

Para la creación de estos cuadros de diálogo en la aplicación se utiliza el método AppInfoActivity#alertDialog() que se encarga de crear y mostrar un cuadro de diálogo en la actividad actual.

```
//Muestra los mensajes de alerta con el boton cerrar
    private void alertDialog(int icono, String title, String message){
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
        alertDialog.setIcon(icono);
        alertDialog.setTitle(title);
        alertDialog.setMessage(message);
        alertDialog.setPositiveButton("Cerrar", null);
        alertDialog.show();
    }
```

Código 52. alertDialog()

- **Splash Screen [46]:** Uno de los aspectos más molestos a la hora de utilizar una aplicación es encontrarnos en el inicio una pantalla que no muestra nada de información al usuario mientras se prepara todo lo necesario para que funcione la aplicación, por este motivo, una buena práctica utilizada en la mayoría de las aplicaciones de esta plataforma es utilizar una pantalla que muestre una imagen de carga o el icono de la aplicación mientras se prepara todo el contenido de la aplicación, de esta manera se indica al usuario que la aplicación está preparando su funcionamiento sin mostrar la molesta pantalla vacía.

Esta Splash Screen en muchas ocasiones se utiliza de manera incorrecta estableciendo simplemente un contador que la muestra desde el inicio un determinado periodo de tiempo preestablecido por el desarrollador convirtiendo esta buena práctica en un antipatrón de diseño.

Un uso correcto de Splash Screen implica que esta pantalla se muestre de forma inmediata al iniciarse la aplicación y solo durante el periodo de tiempo que necesite la aplicación para prepararlo todo. Esta práctica de Splash Screen puede observarse en muchas aplicaciones, como YouTube, Gmail o WhatsApp.

Para introducir esta Splash Screen en la aplicación definiremos un nuevo estilo en la aplicación que contendrá el diseño de esta pantalla Splash, es decir, un recurso drawable (background_splash.xml) con el fondo e imagen a mostrar; de esta forma, al utilizar únicamente recursos que se cargan rápidamente para crear esta pantalla, esta se mostrará sin ningún tipo de retardo al abrir la aplicación.

En la Código 53 se muestra el contenido del fichero /res/values/styles.xml que define los estilos utilizados en la aplicación. En él definimos el estilo de la pantalla Splash indicando en este estilo que se elimine la barra de acción superior y que muestre como fondo del estilo el recurso drawable background_splash.xml.

```
<style name="Splashtheme" parent="Theme.AppCompat.NoActionBar">
    <item name="android:windowBackground">@drawable/background_splash</item>
</style>
```

Código 53. Contenido del fichero /res/values/styles.xml

El contenido del fichero /res/drawable/background_splash.xml que define la imagen que se mostrará de fondo en el estilo SplashTheme se puede ver en el Código 54.

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:drawable="@color/white"/>
    <item>
        <bitmap
            android:gravity="center"
            android:src="@drawable/ic_launcher_ome"/>
    </item>
</layer-list>
```

Código 54. Contenido del fichero /res/drawable/background_splash.xml

Una vez definido el estilo que utilizará la pantalla Splash crearemos una nueva actividad SplashInit.java que no tendrá layout por lo que su carga será inmediata y que se encargará de mostrar la pantalla Splash y lanzar la actividad principal de la aplicación.

De esta forma, mientras se prepara todo lo necesario para iniciar la actividad principal se mostrará la pantalla Splash.

```
public class SplashInit extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        finish();
    }
}
```

Código 55. Definición de la actividad SplashInit.

Por último, actualizaremos el fichero AndroidManifest de la aplicación AnPerMis para que la primera actividad lanzada por la aplicación sea la actividad SplashInit.java en lugar de la actividad MainActivity.java.

```
<activity
    android:name=".SplashInit"
    android:theme="@style/Splashtheme">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Código 56. Actualización del archivo AndroidManifest.xml

6.2. Pantallas y funcionalidad

Una vez definidos los principales recursos de diseño utilizados para la creación de la interfaz de usuario de la aplicación vamos a mostrar los resultados obtenidos para cada una de las pantallas de la aplicación explicando al mismo tiempo su funcionalidad.

- **Pantalla Splash Screen (Figura 25):** Esta pantalla se encarga simplemente de mostrar el icono de la aplicación en pantalla (Splash Screen) mientras se realizan todas las tareas necesarias para iniciar la aplicación, de esta manera se evita mostrar al usuario una molesta pantalla vacía mientras se prepara la aplicación, como se ha explicado en el apartado anterior.

Figura 25. Pantalla Splash Screen

- **Pantalla principal de la aplicación (Figura 26):** Esta pantalla principal muestra al usuario una lista con las aplicaciones instaladas en el dispositivo, para ello utiliza los recursos RecyclerView y CardView explicados en el apartado anterior.

Para acceder al análisis de los permisos de cada una de las aplicaciones de la lista, el usuario simplemente deberá seleccionar y pulsar sobre la aplicación deseada para que se muestre la pantalla Vista de resultados en la que se pueden ver los resultados del análisis de los permisos.

En esta pantalla principal también aparece en la parte inferior derecha un botón flotante que si se pulsa muestra al usuario información sobre cómo utilizar la aplicación y sobre cómo se realiza y se debe interpretar el análisis de los permisos.

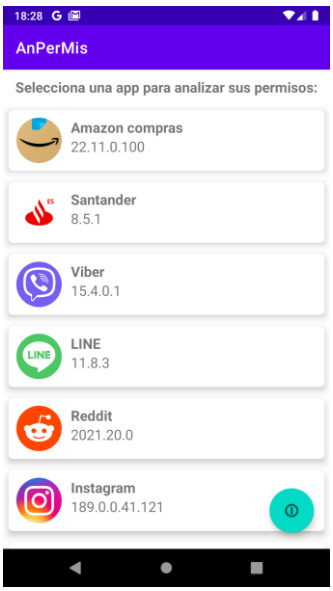


Figura 26. Pantalla principal de la aplicación

- **Pantalla de información (Figura 27):** Esta pantalla muestra al usuario una guía sobre cómo utilizar la aplicación y sobre cómo debe interpretar el análisis de los permisos. Para mostrar esta información por pantalla se utiliza el recurso ScrollView de Android que permite introducir en la actividad una vista con desplazamiento.

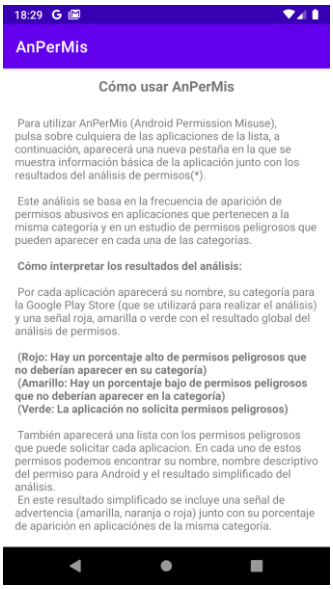


Figura 27. Pantalla de información

Tras pulsar sobre una de las aplicaciones disponibles en la lista, la aplicación puede mostrar diálogos de alerta que avisan al usuario de resultados importantes del análisis de los permisos. Para mostrar por pantalla estos diálogos utilizamos el recurso AlertDialog explicado en el punto anterior. Estos diálogos pueden verse en las siguientes pantallas:

- **Aviso de aplicación potencialmente mal clasificada (Figura 28):** Advierte al usuario de que esta aplicación se encuentra dentro de una categoría peligrosa, como se explica en la sección Categorías peligrosas.

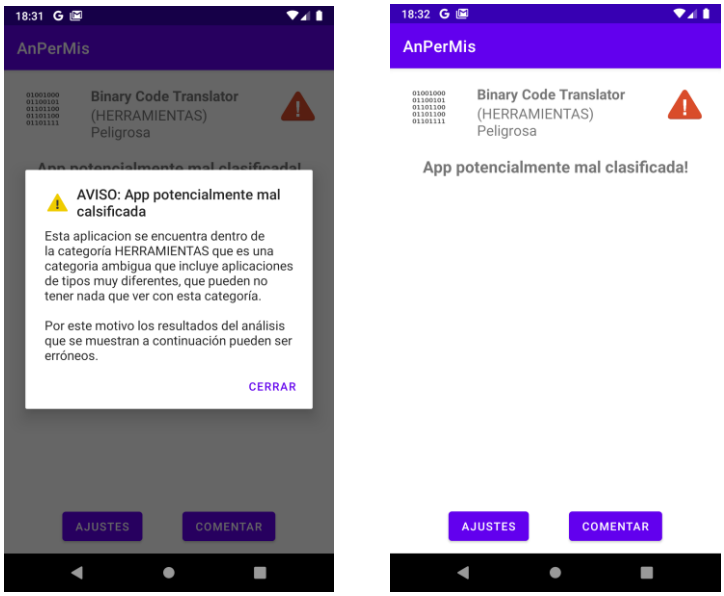


Figura 28. Aviso de aplicación potencialmente mal clasificada

- **Aviso de cambio de categoría (Figura 29):** Advierte al usuario de que la categoría de la aplicación actual ha cambiado con respecto a análisis realizados anteriormente y lo anima a que vuelva a revisar el resultado del análisis de los permisos.

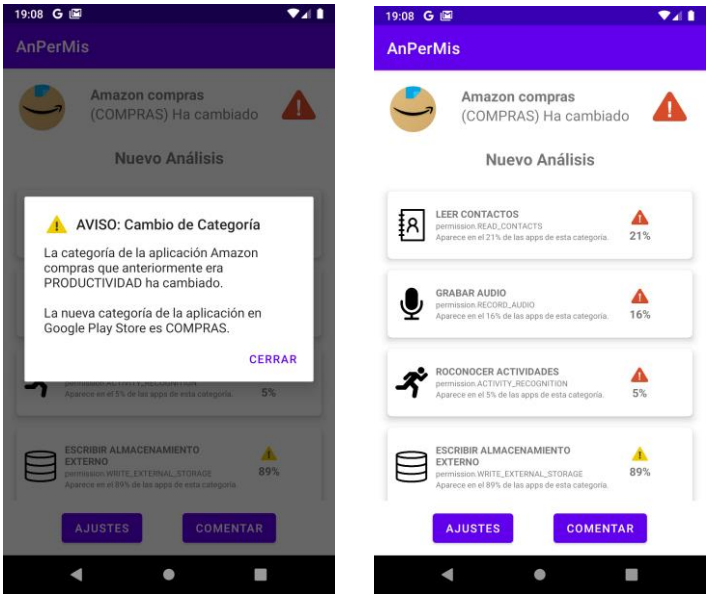


Figura 29. Aviso de cambio de categoría

- **Aviso de cambio del número de permisos peligrosos (Figura 30):** Advierte al usuario de que el número de permisos peligrosos que solicita la aplicación ha cambiado con respecto a análisis realizados anteriormente por lo que el usuario debería revisar de nuevo el resultado del análisis de los permisos.

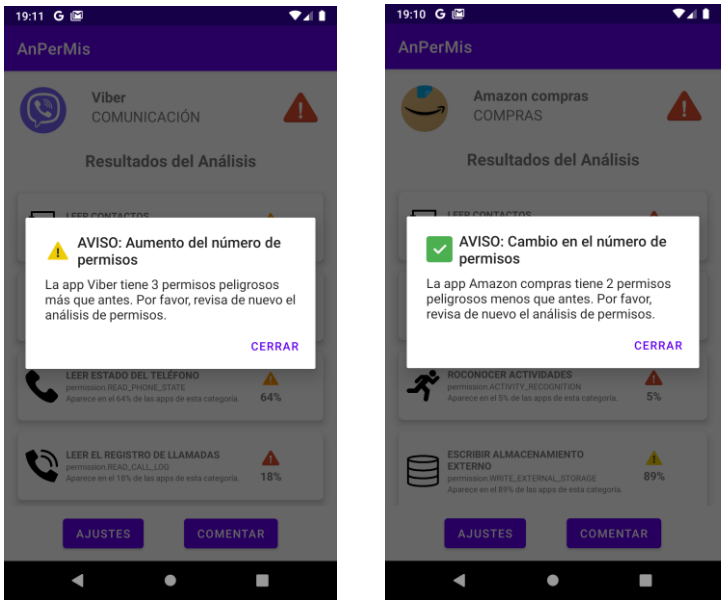


Figura 30. Aviso de cambio de número de permisos peligrosos

- **Pantalla Información Aplicación (Figura 31):** Tras mostrar todos los mensajes de alerta necesarios al usuario se muestra la pantalla con la información sobre el análisis de los permisos peligrosos, en la que podemos ver información sobre la aplicación y su categoría, el resultado global del análisis de la aplicación y una lista con los permisos peligrosos que utiliza a aplicación y que han sido analizados.

En la lista de permisos peligrosos disponible en esta pantalla, creada utilizando los recursos RecyclerView y CardView explicados en el apartado anterior, el usuario podrá pulsar sobre cada uno de estos permisos para obtener información más detallada sobre ellos.

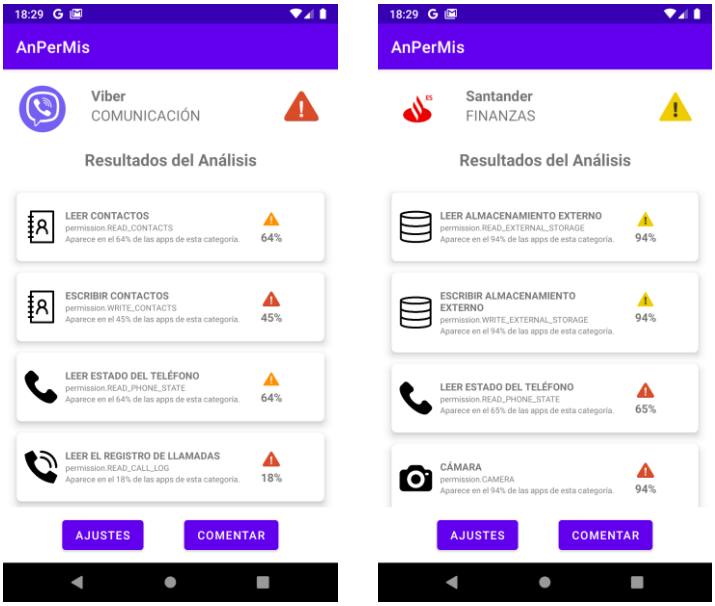


Figura 31. Pantalla información Aplicación

- **Pantalla Información Permisos (Figura 32):** En esta pantalla se muestra haciendo uso del recurso AlertDialog la explicación detallada del permiso peligroso seleccionado y su análisis. Entre esta información el usuario puede ver el nombre del permiso peligroso, su descripción detallada, la explicación detallada del resultado del análisis y cómo actuar en el caso de que el permiso peligroso no debiera aparecer en la aplicación.

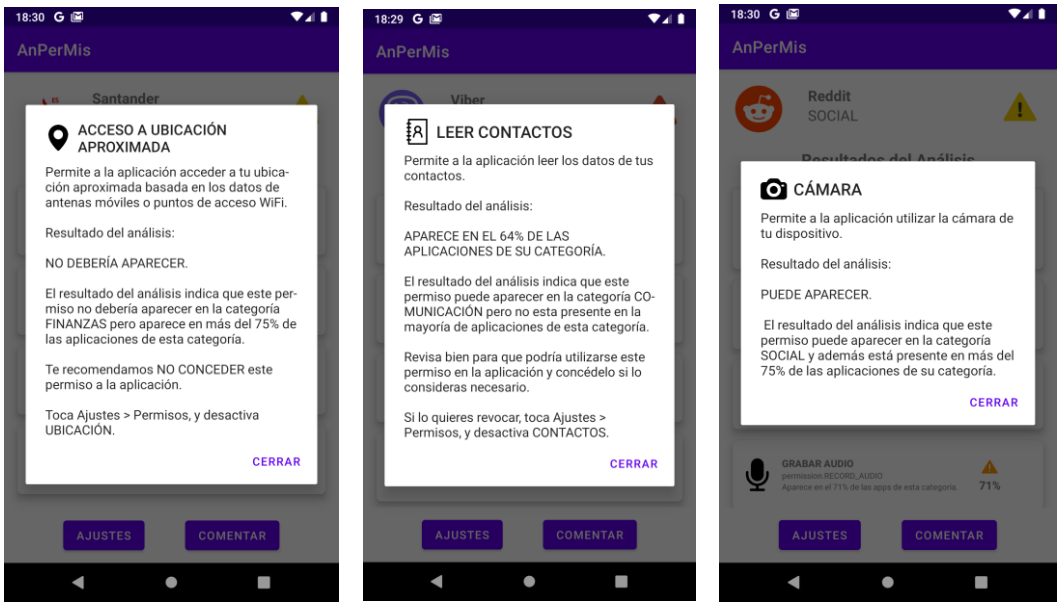


Figura 33. Pantalla Información Permisos

- **Pantalla revocar permisos (Figura 34):** Si tras interpretar el análisis de los permisos peligrosos de la aplicación el usuario decide revocar algún permiso peligroso que no debería aparecer en la aplicación, el usuario lo podrá hacer pulsando sobre el botón Ajustes en la página que muestra la información sobre la aplicación y el análisis de los permisos.

Este botón redirige al usuario a los ajustes de sistema de la aplicación dándole la posibilidad de revocar los permisos que considere oportunos.

Para revocar un permiso peligroso, el usuario deberá pulsar sobre Permisos y retirar los permisos que desee tal y como se indica en la pantalla de información.

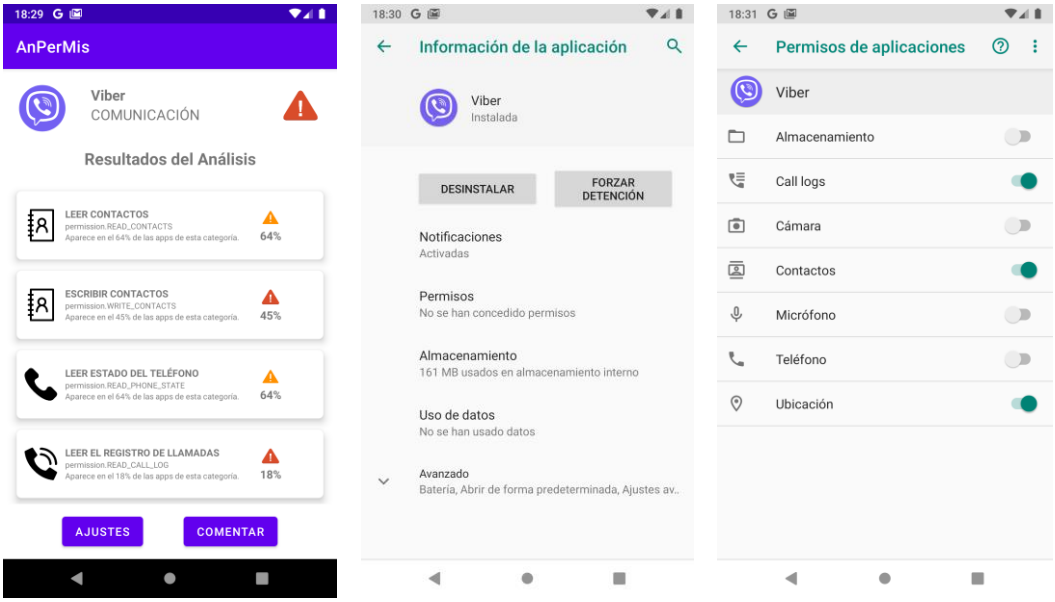


Figura 34. Pantalla revocar permisos

- **Pantalla recomendación tras revocar permisos (Figura 35):** Esta pantalla muestra al usuario una advertencia de lo que puede ocurrir al retirar los permisos peligrosos de la aplicación y le da una recomendación sobre cómo actuar en caso de que esto suceda.

Figura 35. Pantalla Recomendación tras revocar permisos.

- **Pantalla comentar (Figura 36):** En caso de que la aplicación deje de funcionar tras revocar algún permiso el usuario tiene la opción de advertir a la comunidad que utiliza la aplicación sobre este hecho pulsando sobre el botón comentar. Este botón redirige al usuario a la página de la Google Play Store de la aplicación para que pueda realizar esta tarea.

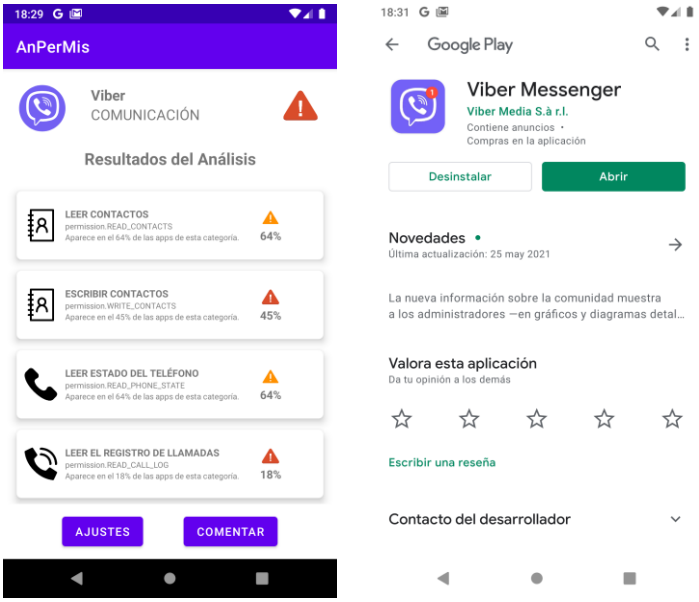


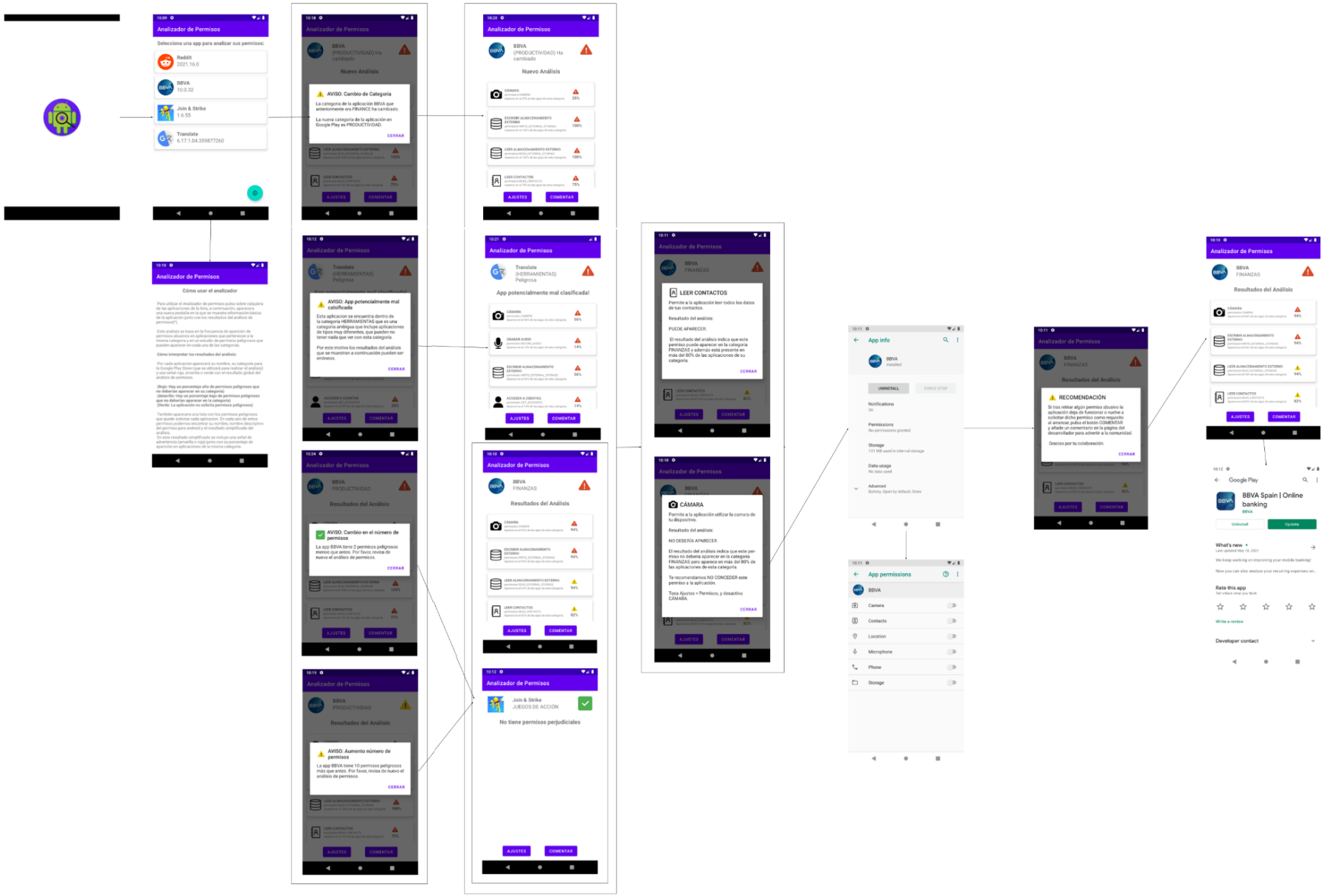
Figura 36. Pantalla Comentar

Por último, cabe mencionar que la interfaz de usuario ha sido probada en dispositivos de varias marcas y versiones del sistema operativo Android, y también en distintas máquinas virtuales Android obteniendo en todos los casos resultados satisfactorios, por lo que podemos decir que la aplicación creada funcionará sin problemas en prácticamente cualquier dispositivo Android.

Tras disponer de una primera versión de la aplicación con la interfaz de usuario descrita, esta se probó con usuarios reales para detectar posibles fallos y realizar correcciones. Durante esta prueba los usuarios reportaron problemas a la hora de interpretar algunos resultados del análisis de los permisos, como el porcentaje de aplicaciones en las que aparece cada permiso por categoría. No se entendía la pantalla dónde se ofrecían estos resultados debido a que el porcentaje de aplicaciones en las que aparece cada permiso por categoría se mostraba solo en formato numérico sin ningún tipo de descripción.

Por este motivo en la revisión de esta interfaz se incorporó la descripción sobre el porcentaje de aplicaciones de la misma categoría en las que aparece un determinado permiso peligroso en las tarjetas que muestran la información sobre los permisos, así esta información resulta más fácil de interpretar para el usuario final.

6.3. Diagrama de flujo (Figura 37)



7. Resultados

Como hemos mencionado en los objetivos de este proyecto, solo vamos a realizar un estudio de los permisos definidos por Android como peligrosos [14], ya que son los únicos permisos cuya concesión puede ser controlada por los usuarios.

En este capítulo presentan los resultados obtenidos del estudio de las aplicaciones Android que se ha tenido que realizar para poder implementar la funcionalidad de la aplicación de AnPerMis. Presentamos por lo tanto los resultados obtenidos del análisis de los permisos peligrosos de un total de 629 aplicaciones procedentes del repositorio oficial de Aplicaciones Android de la Google Play Store. Para realizar este análisis la aplicación AnPerMis se ha ejecutado sobre el dispositivo virtual Android 9 (API 28) integrado en el entorno de desarrollo Android Studio y también en dispositivos de usuarios reales de diferentes marcas y versiones del sistema operativo. En [127] se pueden ver la totalidad de las aplicaciones analizadas. Esta tabla se encuentra alojada en GitHub, ya que debido a la gran cantidad de espacio que ocupa ha resultado imposible incluirla en esta memoria.

7.1. Frecuencia de aparición de los permisos peligrosos

El objetivo de estudiar el uso de los permisos peligrosos es determinar qué permisos utiliza cada aplicación y cuáles de ellos podrían considerarse abusivos siguiendo el criterio que se explicó en el capítulo 5. La heurística seguida consiste en obtener las tablas de frecuencia de aparición de los permisos y la tabla de criterio de aparición de los permisos por categorías.

En el estudio de cada una de las aplicaciones analizadas en esta fase se distinguen dos etapas: una primera etapa en la que se utiliza AnPerMis para extraer los permisos peligrosos que pueden utilizar las aplicaciones y su categoría, y una segunda etapa en la que se analiza y procesa esta información en el servidor remoto con el objetivo de crear las tablas de frecuencia de aparición de los permisos peligrosos [128] y la tabla de criterio de aparición de permisos por categorías [129]. Por la misma razón que la tabla de las aplicaciones, estas tablas se encuentran alojadas en GitHub.

7.2. Análisis de resultados por categorías

En esta sección vamos a explicar los resultados obtenidos en el análisis de permisos peligrosos de las 629 aplicaciones, realizado una comparativa por categorías de las aplicaciones que más y menos permisos peligrosos utilizan respectivamente, en formato descripción de resultados y tabla. Recordamos al lector que durante todo este trabajo definimos permisos peligrosos como aquellos que acceden a los *datos privados del usuario*, es decir, datos restringidos en los que se incluye información que puede ser sensible. Algunos ejemplos de datos privados del usuario incluyen la ubicación y la información de contacto. [47] En esta sección cuando hablamos de permisos nos referimos a este tipo de permisos.

- **Arte y diseño:**

En la categoría arte y diseño que está dedicada a cuadernos de bocetos, herramientas de pintura, herramientas de arte, herramientas de diseño y libros de colorear. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 3 y 4 permisos como indica la tabla 5.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3.8	2
	READ_EXTERNAL_STORAGE 100%, WRITE_EXTERNAL_STORAGE 100%

Tabla 5. Resumen del estudio de permisos para las app de la categoría arte y diseño

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 6). Hemos detectado que la aplicación Behance [48] utiliza en total 7 permisos (la que más permisos peligrosos utiliza dentro de esta categoría). En un estudio de los 5 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceso a la ubicación aproximada, acceso a la ubicación detallada y acceso a cuentas. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 38). En contraposición a esta aplicación tenemos Photoshop Scketch [49], que es una de las que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Behance	7	2 (29%)	5 (71%)
Photoshop Sketch	3	2 (67%)	1 (33%)

Tabla 6. Estudio de permisos para dos apps específicas de la categoría arte y diseño

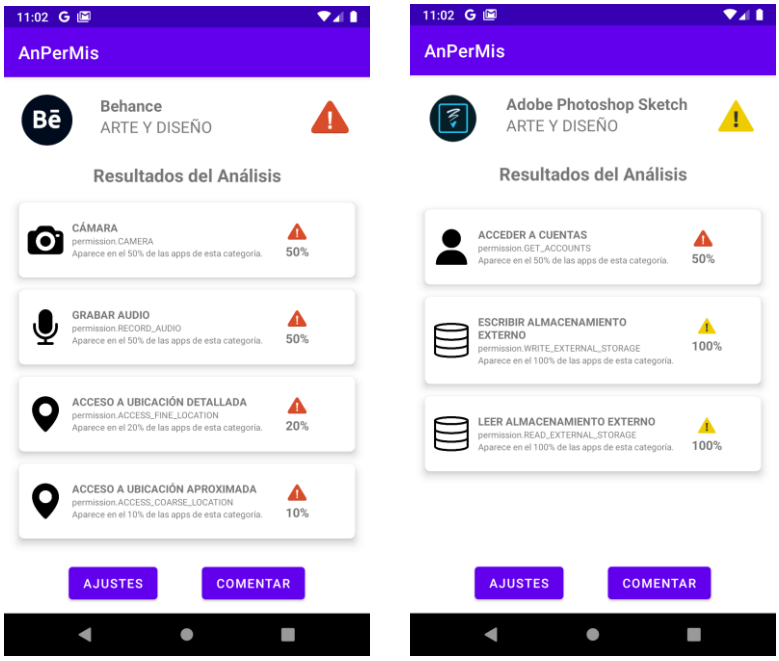


Figura 38. Resultados del análisis de dos apps específicas de la categoría arte y diseño en AnPerMis

- **Automoción**

En la categoría automoción que está dedicada a venta de coches, seguros de coches, comparación de precios de coches, seguridad vial, noticias y opiniones sobre vehículos. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE y ACCESS_COARSE_LOCATION. La mayoría de las aplicaciones de esta categoría utilizan 4 permisos como indica la tabla 7.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4.1	3
	READ_EXTERNAL_STORAGE 80%, WRITE_EXTERNAL_STORAGE 100%, ACCESS_COARSE_LOCATION 80%

Tabla 7. Resumen del estudio de permisos para las app de la categoría automoción

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 8). Hemos detectado que la aplicación Coches.net [50] utiliza en total 6 permisos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 4 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación detallada, acceder a ubicación aproximada y leer el estado del teléfono. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Carista OBD2 [51], que es una de las aplicaciones que menos permisos utiliza en la categoría, entre los permisos pocos comunes que utiliza se encuentran acceso a ubicación aproximada y acceso a ubicación detallada cuyo uso puede ser considerado un abuso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Coches.net	6	2 (33%)	4 (67%)
Carista OBD	2	0 (0%)	2 (100%)

Tabla 8. Estudio de permisos para dos apps específicas de la categoría automoción

- **Belleza**

En la categoría belleza que está dedicada a tutoriales de maquillaje, utensilios de maquillaje, peluquería, compra de productos de belleza y simuladores de maquillaje. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 4 y 5 permisos como indica la tabla 9.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4.54	2
	READ_EXTERNAL_STORAGE 100%, WRITE_EXTERNAL_STORAGE 100%,

Tabla 9. Resumen del estudio de permisos para las app de la categoría belleza

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 10). Hemos detectado que la aplicación Booksy BIZ [52] utiliza en total 8 permisos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 6 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, acceder a la ubicación detallada, leer el estado del teléfono y realizar llamadas. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Mi cronograma capilar [53], que es una de las que menos permisos utiliza en la categoría, entre los permisos válidos que utiliza se encuentran, leer almacenamiento externo y escribir almacenamiento externo.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Booksy BIZ	8	2 (25%)	6 (75%)
Mi cronograma capilar	2	2 (100%)	0 (0%)

Tabla 10. Estudio de permisos para dos apps específicas de la categoría belleza

- **Empresa**

En la categoría empresa que está dedicada a lectores y editores de documentos, seguimiento de paquetes, administración de correo electrónico y búsqueda de empleo. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son CAMERA, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan 6 permisos como indica la tabla 11.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
6,4	3
	CAMERA 78%, READ_EXTERNAL 86%, WRITE_EXTERNAL 86%,

Tabla 11. Resumen del estudio de permisos para las app de la categoría empresa

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 12). Hemos detectado que la aplicación Zoom [54] utiliza en total 11 permisos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 8 permisos peligrosos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación aproximada, acceder a ubicación detallada, leer estado del teléfono, realizar llamadas, leer calendario, leer contactos y leer números de teléfono. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 39). En contraposición a esta aplicación tenemos InfoJobs [55], que es la aplicación que menos permisos utiliza en la categoría, aunque el único permiso peligroso que utiliza, leer estado del teléfono, puede ser considerado como un permiso utilizado de forma abusiva.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Zoom	11	3 (27%)	8 (73%)
InfoJobs	1	0 (0%)	1 (100%)

Tabla 12. Estudio de permisos para dos apps específicas de la categoría empresa.

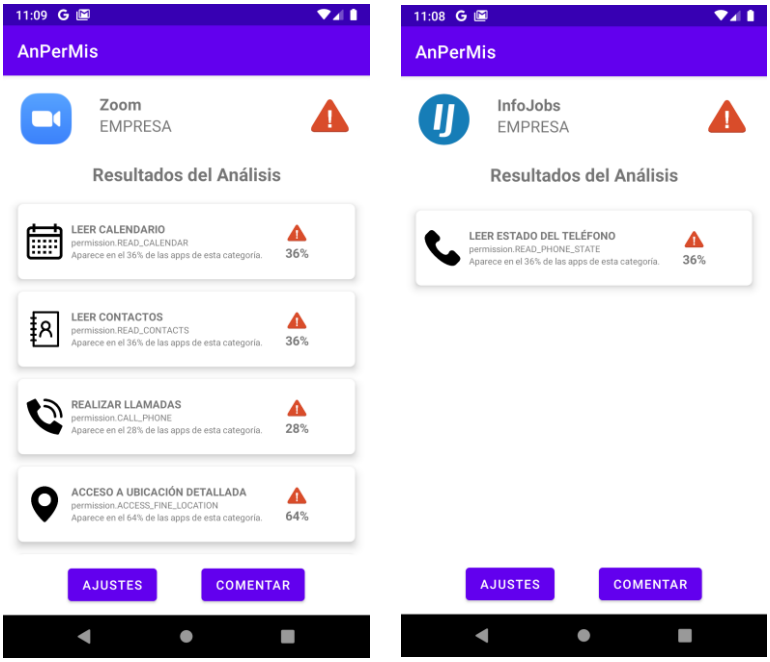


Figura 39. Resultados del análisis de dos apps específicas de la categoría empresa en AnPerMis

- **Cómics**

En la categoría cómics que está dedicada a Personajes de cómics y libros de cómics. El estudio que hemos realizado muestra que ningún permiso peligroso aparece en más del 75% de aplicaciones, además, la mayoría de las aplicaciones de esta categoría utilizan entre 1 y 2 permisos como indica la tabla 13.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
1,75	0

Tabla 13. Resumen del estudio de permisos para las app de la categoría cómics

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 14). Hemos detectado que la aplicación Webtoon [56] utiliza en total 5 permisos (la que más permisos utiliza dentro de esta categoría). Un estudio de los 5 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, leer el estado del teléfono y utilizar la cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Mango [57], que no utiliza ningún permiso peligroso al igual que el 42% de aplicaciones de esta categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
WebToon	5	0 (0%)	5 (100%)
Mango	0	0	0

Tabla 14. Estudio de permisos para dos apps específicas de la categoría cómics.

- **Libros y obras de consulta**

En la categoría libros y obras de consulta que está dedicada a lectores de libros, libros de referencia, diccionarios, tesauros y wikis. El estudio que hemos realizado muestra que el permiso peligroso que aparece con mayor frecuencia es READ_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 2 y 3 permisos como indica la tabla 15.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
2,72	1
	,READ_EXTERNAL_STORAGE 83%,

Tabla 15. Resumen del estudio de permisos para las app de la categoría libros y obras de consulta

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 16). Hemos detectado que la aplicación My Heritage [58] utiliza en total 6 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 4 permisos poco comunes para esta categoría observamos que nos pide permiso para: utilizar la cámara, grabar audio y leer contactos. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos BMW Drivers Guide [59], que al ser una guía para conductores no utiliza ningún permiso peligroso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
My Heritage	6	2 (33%)	4 (67%)
BMW Drivers Guide	0	0	0

Tabla 16. Estudio de permisos para dos apps específicas de la categoría libros y obras de consulta.

- **Comunicación**

En la categoría comunicación de consulta que está dedicada a mensajes, chats/mensajería instantánea, marcadores, libretas de direcciones, navegadores y gestión de llamadas. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 8 y 9 permisos como indica la tabla 17.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
8,9	2
	,READ_EXTERNAL 91%, WRITE_EXTERNAL 91%

Tabla 17. Resumen del estudio de permisos para las app de la categoría comunicación

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 18). Hemos detectado que la aplicación WhatsApp [60] utiliza en total 15 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 13 permisos poco comunes para esta categoría observamos que nos pide permiso para: leer números de teléfono, enviar SMS, realizar llamadas, contestar llamadas y leer el registro de llamadas. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 40). En contraposición a esta aplicación tenemos Orbot [61], que no utiliza ningún permiso peligroso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
WhatsApp	15	2 (13%)	13 (87%)
Orbot	0	0	0

Tabla 18. Estudio de permisos para dos apps específicas de la categoría comunicación.

Como podemos observar, aunque las dos aplicaciones pertenecen a la misma categoría mientras que WhatsApp es una aplicación relacionada con mensajería, la aplicación Orbot es una aplicación proxy, por lo que, aunque la aplicación se encuentra dentro de la categoría correcta, los resultados obtenidos al realizar esta comparación serán poco útiles. Por este motivo y tras realizar un análisis más profundo de los resultados obtenidos en la categoría Comunicación hemos decidido comparar las aplicaciones WhatsApp y Line [62] (Tabla 19) que son aplicaciones más parecidas.

En esta nueva comparación, hemos detectado que la aplicación Line utiliza en total 12 permisos peligrosos (3 permisos menos que la aplicación WhatsApp). Un estudio de los 10 permisos poco comunes para esta categoría observamos que nos pide permiso para: leer números de teléfono y realizar llamadas. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 40).

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
WhatsApp	15	2 (13%)	13 (87%)
Line	12	2 (17%)	10 (83%)

Tabla 19. Estudio de permisos para dos apps específicas de la categoría comunicación.

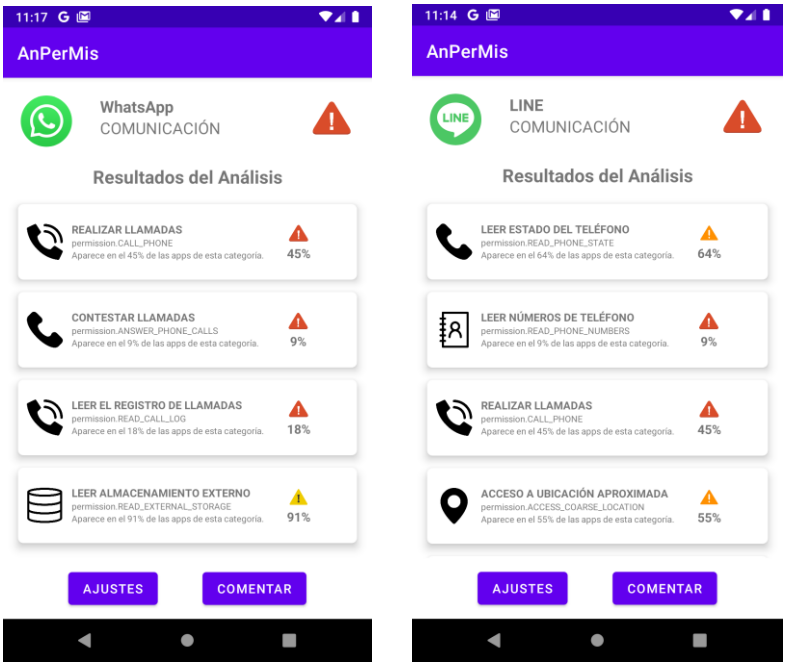


Figura 40. Resultados del análisis de dos apps específicas de la categoría comunicación en AnPerMis

- **Citas**

En la categoría citas que está dedicada a emparejamiento, noviazgo, mantener una relación, conocer nuevas personas y enamorarse. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son CAMERA, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 5 y 6 permisos como indica la tabla 20.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5,3	3
	CAMERA 92%, READ_EXTERNAL 100%, WRITE_EXTERNAL 92%,

Tabla 20. Resumen del estudio de permisos para las app de la categoría citas

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 21). Hemos detectado que la aplicación Badoo [63] utiliza en total 9 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a todos los permisos relacionados con la ubicación y leer contactos. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Idilium [64], que es una de las que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Badoo	9	2 (22%)	7 (78%)
Idilium	3	3 (100%)	0 (0%)

Tabla 21. Estudio de permisos para dos apps específicas de la categoría citas.

Por último, cabe mencionar que más del 75% de las aplicaciones analizadas de esta categoría utilizan de forma abusiva permisos relacionados con la ubicación del usuario, ya que existen

alternativas al uso de estos permisos peligrosos como la utilización de formularios para conocer la ubicación.

- **Educación**

En la categoría educación que está dedicada a preparación de exámenes, ayudas para el estudio, vocabulario, juegos educativos y aprendizaje de idiomas. El estudio que hemos realizado muestra que ningún permiso peligroso aparece en más del 75% de aplicaciones, además, la mayoría de las aplicaciones de esta categoría utilizan entre 1 y 2 permisos como indica la tabla 22.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
1,4	0

Tabla 22. Resumen del estudio de permisos para las app de la categoría educación

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 23). Hemos detectado que la aplicación Duolingo [65] utiliza en total 4 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 4 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a cuentas y utilizar la cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Google Classroom [66], que no utiliza ningún permiso peligroso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Duolingo	4	0 (0%)	4 (100%)
Google Classroom	0	0	0

Tabla 23. Estudio de permisos para dos apps específicas de la categoría educación

- **Entretenimiento**

En la categoría entretenimiento que está dedicada a reproducción de vídeo, películas, programas de TV y contenido de ocio interactivo en streaming. El estudio que hemos realizado

muestra que ningún permiso peligroso aparece en más del 75% de aplicaciones, además, la mayoría de las aplicaciones de esta categoría utilizan 3 permisos como indica la tabla 24.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3	0

Tabla 24. Resumen del estudio de permisos para las app de la categoría entretenimiento

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 25). Hemos detectado que la aplicación Bitmoji [67] utiliza en total 7 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: leer contactos, escribir contactos, utilizar la cámara, acceder a cuentas y leer el estado del teléfono. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 41). En contraposición a esta aplicación tenemos Disney+ [68], que no utiliza ningún permiso peligroso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Bitmoji	7	0 (0%)	7 (100%)
Disney +	0	0	0

Tabla 25. Estudio de permisos para dos apps específicas de la categoría entretenimiento

En la categoría Entretenimiento solo deberían aparecer aplicaciones de reproducción de vídeo, películas, programas de TV y contenido de ocio interactivo en streaming por lo que podríamos concluir que esta aplicación se encuentra mal clasificada dentro de esta categoría.

Por este motivo y tras realizar un nuevo análisis de resultados obtenidos en esta categoría decidimos comparar las aplicaciones Prime Video [69] y Disney+ (Tabla 26) que son aplicaciones populares que se ajustan más a la definición de esta categoría.

En esta nueva comparación, hemos detectado que la aplicación Prime Video utiliza en total 6 permisos peligrosos (la que más permisos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación detallada, acceder a ubicación aproximada, acceder a cuentas y leer el estado del

teléfono. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 41).

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Prime Video	6	0 (0%)	7 (100%)
Disney +	0	0	0

Tabla 26. Estudio de permisos para dos apps específicas de la categoría entretenimiento

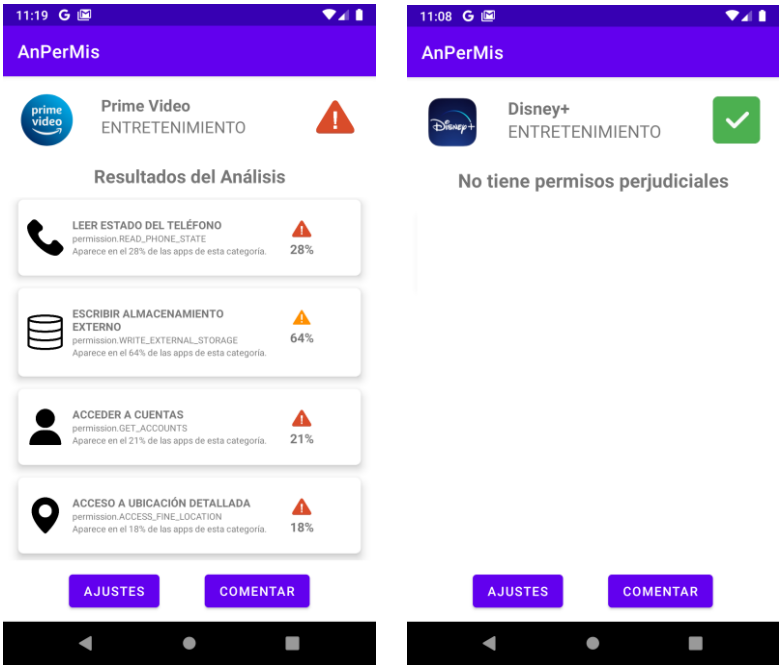


Figura 41. Resultados del análisis de dos apps específicas de la categoría entretenimiento en AnPerMis

- **Eventos**

En la categoría eventos que está dedicada a entradas de conciertos, de eventos deportivos y de cine, reventa de entradas, tarjetas de cumpleaños. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 3 y 4 permisos como indica la tabla 27.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3,3	2
	READ_EXTERNAL_STORAGE 82%, WRITE_EXTERNAL_STORAGE 82%,

Tabla 27. Resumen del estudio de permisos para las app de la categoría eventos

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 28). Hemos detectado que la aplicación Greetings Island [70] utiliza en total 7 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 5 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación aproximada, acceder a ubicación detallada, utilizar la cámara, grabar audio y acceder a cuentas. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a están las aplicaciones tenemos Backpack Mod [71] y Avengers Mod [72], que no utilizan ningún permiso peligroso

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Greetings Island	7	2 (29%)	5 (71%)
Backpack Mod	0	0	0
Avengers Mod	0	0	0

Tabla 28. Estudio de permisos para dos apps específicas de la categoría eventos

En los resultados de esta categoría podemos observar que las dos aplicaciones que menos permisos utilizan son aplicaciones que permiten instalar modificaciones en videojuegos, por lo que, no tienen nada que ver con la categoría eventos. Por este motivo y tras realizar un nuevo análisis de los resultados obtenidos en esta categoría decidimos comparar las aplicaciones Greetings Island y Ticketmaster [73] que por definición si pertenecen a la categoría eventos (Tabla 29).

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Greeting Island	7	2 (29%)	5 (71%)
TicketMaster	2	2 (100%)	0

Tabla 29. Estudio de permisos para dos apps específicas de la categoría eventos

- **Finanzas**

En la categoría finanzas que está dedicada a banca, pagos, buscadores de cajeros, noticias financieras, seguros, impuestos, cartera y comercio, calculadoras de propinas. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, CAMERA, READ_CONTACTS, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 7 y 8 permisos como indica la tabla 30.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
7,2	6
	ACCESS_COARSE_LOCATION 83%, ACCESS_FINE_LOCATION 88%, CAMERA 94%, READ_CONTACTS 83%, READ_EXTERNAL 94%, WRITE_EXTERNAL 94%,

Tabla 31. Resumen del estudio de permisos para las app de la categoría finanzas

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 32). Hemos detectado que la aplicación Imagin [74] utiliza en total 11 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 8 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación aproximada, acceder a ubicación detallada, acceder a ubicación en segundo plano, realizar llamadas, leer estado del teléfono y utilizar la cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Coinbase [75], que es una de las que menos permisos utiliza en la categoría

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Imagin	11	3 (27%)	8 (73%)
Coinbase	3	3 (100%)	0 (0%)

Tabla 32. Estudio de permisos para dos apps específicas de la categoría finanzas

Como se puede observar, aunque las dos aplicaciones pertenecen a la misma categoría mientras que Imagine es una aplicación de banca, Coinbase es una aplicación de compra y venta de divisas digitales por lo que, los resultados de realizar esta comparación serían al igual que en casos anteriores poco útiles, aunque, la aplicación Coinbase estaría dentro de la categoría correcta.

Por este motivo y tras realizar un nuevo análisis de los resultados obtenidos en la categoría Finanzas hemos decidido comparar las aplicaciones Imagin y Banco Santander [76] que son aplicaciones más parecidas (Tabla 33).

La aplicación Santander a pesar de utilizar menos permisos que Imagin, solicita permisos peligrosos que tampoco deberían aparecer en su categoría. Un estudio de los 4 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación aproximada, acceder a ubicación detallada, leer estado del teléfono y utilizar la cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos también a esta aplicación como potencialmente peligrosa

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Imagin	11	3 (27%)	8 (73%)
Santander	7	3 (43%)	4(57%)

Tabla 33. Estudio de permisos para dos apps específicas de la categoría finanzas

- **Comer y beber**

En la categoría comer y beber que está dedicada a recetas, restaurantes, guías gastronómicas, descubrimiento y cata de vinos, y recetas de bebidas. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, CAMERA, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan 5 permisos como indica la tabla 34.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5	5
	ACCESS_COARSE_LOCATION 84%, ACCESS_FINE_LOCATION 92%, CAMERA 85%, READ_EXTERNAL_STORAGE 77%, WRITE_EXTERNAL_STORAGE 74%,

Tabla 34. Resumen del estudio de permisos para las app de la categoría comer y beber

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 35). Hemos detectado que la aplicación McDo+ [77] utiliza en total 8 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 8 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, acceder a la ubicación detallada, acceder a la ubicación en segundo plano, leer estado del teléfono, cámara, realizar llamadas y almacenamiento externo. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Burger King [78], que es una de las que menos permisos utiliza en la categoría, de entre los permisos poco comunes que utiliza podemos encontrar acceso a la ubicación aproximada, acceso a la ubicación detallada y cámara.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
McDo+	8	0 (0%)	8 (100%)
Burger King	3	0 (0%)	3 (100%)

Tabla 35. Estudio de permisos para dos apps específicas de la categoría comer y beber

- **Salud y bienestar**

En la categoría salud y bienestar que está dedicada a bienestar personal, seguimiento de ejercicios, dietas y consejos nutricionales, salud y seguridad. El estudio que hemos realizado muestra que el permiso peligroso que aparece con mayor frecuencia es READ_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 5 y 6 permisos como indica la tabla 36.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5,78	1
	READ_EXTERNAL_STORAGE 87%

Tabla 36. Resumen del estudio de permisos para las app de la categoría salud y bienestar

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 37). Hemos detectado que la aplicación Fitbit [79] utiliza en total 16 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 16 permisos poco comunes para esta categoría observamos que nos pide permiso para: leer calendario, utilizar la cámara, leer almacenamiento externo, escribir almacenamiento externo y leer contactos. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Contador de pasos [80], que no utiliza ningún permiso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Fitbit	16	0 (0%)	16 (100%)
Contador de pasos	0	0	0

Tabla 37. Estudio de permisos para dos apps específicas de la categoría salud y bienestar

- **Casa y hogar**

En la categoría casa y hogar que está dedicada búsqueda de casa y piso, reformas, interiorismo, hipotecas e inmobiliarias. El estudio que hemos realizado muestra que el permiso peligroso que aparece con mayor frecuencia es ACCESS_FINE_LOCATION. La mayoría de las aplicaciones de esta categoría utilizan entre 4 y 5 permisos como indica la tabla 38.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4,25	1
	ACCESS_FINE_LOCATION 83%,

Tabla 38. Resumen del estudio de permisos para las app de la categoría casa y hogar

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 39). Hemos detectado que la aplicación Housing [81] utiliza en total 9 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación detallada, acceder a la ubicación aproximada, leer el estado del teléfono, leer el registro de llamadas, contestar llamadas y realizar llamadas. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Trovit Casas [82], que es una de las que menos permisos utiliza en la categoría, de entre los permisos poco comunes que utiliza podemos encontrar acceso a la ubicación aproximada y acceso a la ubicación detallada.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Housing	9	2 (23%)	7 (77%)
Trovit Casas	2	0	2 (100%)

Tabla 39. Estudio de permisos para dos apps específicas de la categoría casa y hogar

- **Bibliotecas y demos**

En la categoría bibliotecas y demos que está dedicada a bibliotecas de software y demostraciones técnicas. El estudio que hemos realizado muestra que ningún permiso peligroso aparece en más del 75% de aplicaciones, además, la mayoría de las aplicaciones de esta categoría utilizan 2 permisos como indica la tabla 40.

Media de permiso utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
2	0

Tabla 40. Resumen del estudio de permisos para las app de la categoría bibliotecas y demos

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 41). Hemos detectado que la aplicación IP Pro [83] utiliza en total 9 permisos peligrosos (la

que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 9 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación detallada, acceder a la ubicación aproximada, leer el estado del teléfono, realizar llamadas, utilizar la cámara, acceso a cuentas, leer almacenamiento externo, escribir almacenamiento externo y grabar audio. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 42). En contraposición a esta aplicación tenemos Mirror Link Screen Conector [84], que no utiliza ningún permiso peligroso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
IP Pro	9	0 (0%)	9 (100%)
Mirror Link Screen Conector	0	0	0

Tabla 41. Estudio de permisos para dos apps específicas de la categoría bibliotecas y demos

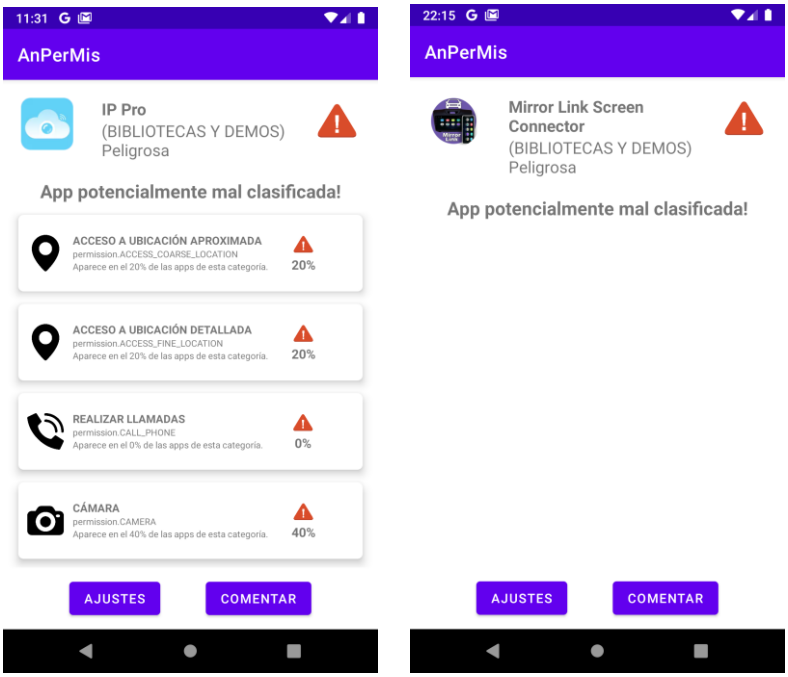


Figura 42. Resultados del análisis de dos apps específicas de la categoría bibliotecas y demos en AnPerMis

Además, en los resultados obtenidos en esta categoría podemos observar que tanto la aplicación que más permisos peligrosos utiliza IP Pro, que es una aplicación de videovigilancia como la aplicación que menos utiliza Mirror Link que es una aplicación que permite proyectar la pantalla del dispositivo son aplicaciones de tipos muy diferentes que no deberían aparecer

en esta categoría, lo que confirma que la categoría Bibliotecas y demos es una categoría que incluye una gran cantidad de aplicaciones potencialmente mal clasificadas.

- ## Estilo de vida

En la categoría estilo de vida que está dedicada a guías de estilo, organización de fiestas y bodas y guías de instrucciones. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, CAMERA, READ_EXTERNAL_STORAGE Y WRITE EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 6 y 7 permisos como indica la tabla 42.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
6,5	5
	ACCESS_COARSE_LOCATION 76%, ACCESS_FINE_LOCATION 76%, CAMERA 86%, READ_EXTERNAL_STORAGE 91%, WRITE_EXTERNAL_STORAGE 90%

Tabla 42. Resumen del estudio de permisos para las app de la categoría estilo de vida

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 43). Hemos detectado que la aplicación Amazon Alexa [85] utiliza en total 16 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 16 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación detallada, acceder a la ubicación aproximada, acceder a la ubicación en segundo plano, contestar llamadas, realizar llamadas, leer contactos, leer el estado del teléfono, leer SMS, recibir MMS, recibir SMS, utilizar la cámara, acceso a cuentas, leer almacenamiento externo, escribir almacenamiento externo y grabar audio Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 43). En contraposición a esta aplicación tenemos Flying Robot Hero [86], que no utiliza ningún permiso peligroso.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Amazon Alexa	16	0 (0%)	16 (100%)
Flying Robot Hero	0	0	0

Tabla 43. Estudio de permisos para dos apps específicas de la categoría estilo de vida

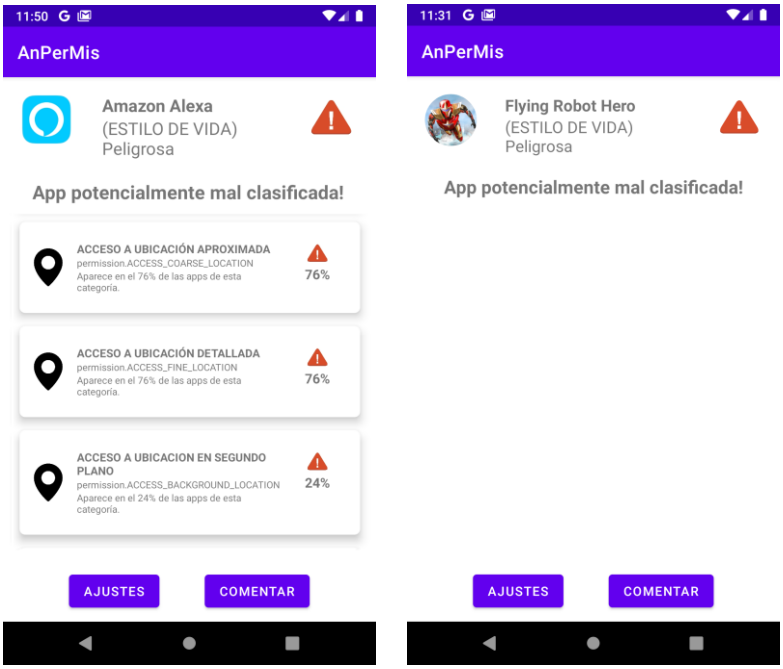


Figura 43. Resultados del análisis de dos apps específicas de la categoría estilo de vida en AnPerMis

Al igual que sucede en la categoría anterior las dos aplicaciones que podemos ver en la tabla 43 son aplicaciones que no deberían aparecer en esta categoría ya que mientras que la aplicación Amazon Alexa es un asistente personal, Flying Robot Hero es un juego, lo que confirma que la categoría Estilo de vida también es una categoría repleta de aplicaciones potencialmente mal clasificadas.

- **Mapas y navegación**

En la categoría mapas y navegación que está dedicada a herramientas de navegación, GPS, creación de mapas, herramientas para la circulación y transporte público. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION. La mayoría de las aplicaciones de esta categoría utilizan entre 5 y 6 permisos como indica la tabla 44.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5,3	2
	ACCESS_COARSE_LOCATION 87%, ACCESS_FINE_LOCATION 95%

Tabla 44. Resumen del estudio de permisos para las app de la categoría mapas y navegación

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 45). Hemos detectado que la aplicación Uber [87] utiliza en total 12 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 10 permisos poco comunes para esta categoría observamos que nos pide permiso para: utilizar la cámara, leer SMS, enviar SMS, realizar llamadas y leer contactos. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Spain Travel Health [88], que es una de las que menos permisos utiliza en la categoría, de entre los permisos poco comunes que utiliza podemos encontrar el permiso para utilizar la cámara.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Uber	12	2 (17%)	10 (83%)
Spain Travel Health	1	0 (0%)	1 (100%)

Tabla 45. Estudio de permisos para dos apps específicas de la categoría mapas y navegación

- **Medicina**

En la categoría medicina que está dedicada a referencias clínicas y farmacológicas, calculadoras, manuales para el personal sanitario, noticias y revistas médicas. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION. La mayoría de las aplicaciones de esta categoría utilizan entre 4 y 5 permisos como indica la tabla 46.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4,5	2
	ACCESS_COARSE_LOCATION 87%, ACCESS_FINE_LOCATION 95%

Tabla 46. Resumen del estudio de permisos para las app de la categoría medicina

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 47). Hemos detectado que la aplicación Quirón Salud [89] utiliza en total 10 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los

10 permisos poco comunes para esta categoría observamos que nos pide permiso para: leer el estado del teléfono, reconocer actividades, acceder a la ubicación detallada, acceder a la ubicación aproximada, y leer el calendario. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Radar COVID [90], que no utiliza permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Quirón Salud	10	0 (0%)	10 (100%)
Radar COVID	0	0	0

Tabla 47. Estudio de permisos para dos apps específicas de la categoría medicina

- **Música y audio**

En la categoría música y audio que está dedicada a servicios musicales, radios y reproductores de música. El estudio que hemos realizado muestra que ningún permiso peligroso aparece en más del 75% de aplicaciones, además, la mayoría de las aplicaciones de esta categoría utilizan entre 3 y 4 permisos como indica la tabla 48.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3,28	0

Tabla 48. Resumen del estudio de permisos para las app de la categoría música y audio

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 49). Hemos detectado que la aplicación Castbox [91] utiliza en total 7 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación detallada, acceder a la ubicación aproximada, leer el estado del teléfono y utilizar la cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Podcast de Google [92], que no utiliza permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Castbox	7	0 (0%)	7 (100%)
Podcast de Google	0	0	0

Tabla 49. Estudio de permisos para dos apps específicas de la categoría música y audio

- **Noticias y revistas**

En la categoría noticias y revistas dedicada a Periódicos, agregadores de noticias, revistas, blogs. El estudio que hemos realizado muestra que el permiso peligroso que aparece con mayor frecuencia es READ_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 3 y 4 permisos como indica la tabla 50.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3,45	1
	READ_EXTERNAL_STORAGE 91%

Tabla 50. Resumen del estudio de permisos para las app de la categoría noticias y revistas

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 51). Hemos detectado que la aplicación Microsoft News [93] utiliza en total 8 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 8 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación detallada, acceder a la ubicación aproximada, leer el estado del teléfono, utilizar la cámara, grabar audio, acceder a cuentas, leer almacenamiento externo y escribir almacenamiento externo. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 44). En contraposición a esta aplicación tenemos GeekTech [94], que no utiliza permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Microsoft News	8	0 (0%)	8 (100%)
GeekTech	0	0	0

Tabla 51. Estudio de permisos para dos apps específicas de la categoría noticias y revistas

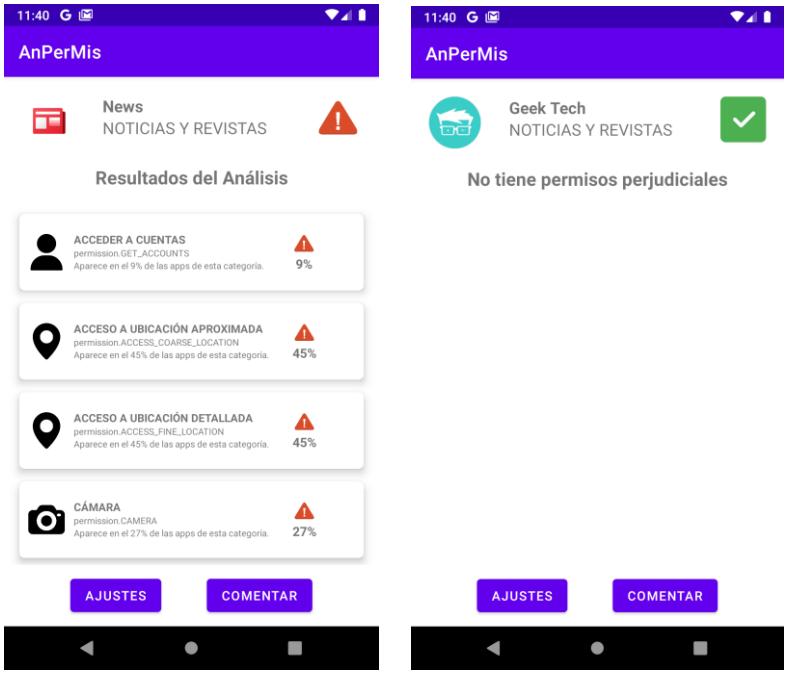


Figura 44. Resultados del análisis de dos apps específicas de la categoría noticias y revistas en AnPerMis

- **Ser padres**

En la categoría ser padres dedicada a embarazo, cuidado de bebés y de niños. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia es CAMERA, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 4 y 5 permisos como indica la tabla 52.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4,28	3
	CAMERA 79%, READ_EXTERNAL_STORAGE 93%, WRITE_EXTERNAL_STORAGE 86%

Tabla 52. Resumen del estudio de permisos para las app de la categoría ser padres

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 53). Hemos detectado que la aplicación Control parental Kroha [95] utiliza en total 10 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 9 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación detallada, acceder a la ubicación aproximada, acceder a la ubicación en segundo plano, leer el estado del teléfono, utilizar la cámara, grabar audio, acceder a cuentas, leer almacenamiento externo y escribir almacenamiento externo. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Consejos para padres [96], que no utiliza permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Control parental Kroha	10	1 (10%)	9 (90%)
Consejos para padres	0	0	0

Tabla 53. Estudio de permisos para dos apps específicas de la categoría ser padres

- **Personalización**

En la categoría personalización dedicada a fondos de pantalla, fondos de pantalla animados, pantalla de inicio, pantalla de bloqueo, tonos de llamada. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 4 y 5 permisos como indica la tabla 54.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4,33	2
	READ_EXTERNAL_STORAGE 87%, WRITE_EXTERNAL_STORAGE 80%

Tabla 54. Resumen del estudio de permisos para las app de la categoría personalización

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 55). Hemos detectado que la aplicación Square Home [97] utiliza en total 8 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 6 permisos poco comunes para esta categoría observamos que nos pide permiso para: utilizar la cámara, leer el estado del teléfono, acceder a cuentas, leer el calendario, leer contactos, realizar llamadas, leer almacenamiento externo y escribir almacenamiento externo. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Notify Buddy [98], que no utiliza permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Square Home	8	2 (25%)	6 (75%)
Notify Buddy	0	0	0

Tabla 55. Estudio de permisos para dos apps específicas de la categoría personalización

Por último, cabe mencionar que en esta categoría se encuentran un gran número de aplicaciones relacionadas con capas de personalización de la interfaz del sistema Android que aunque están bien categorizadas, hacen uso de permisos peligrosos como realizar llamadas, leer contactos o leer estado del teléfono necesarios para su funcionamiento, pero que en otras aplicaciones de esta categoría son poco comunes, por ejemplo, en aplicaciones de fondos de pantalla, por lo que podemos concluir que esta categoría a pesar de que contiene aplicaciones bien clasificadas, tiene una definición muy general.

- **Fotografía**

En la categoría fotografía dedicada a cámaras, herramientas de edición fotográfica y aplicaciones para administrar y compartir fotos. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan 4 permisos como indica la tabla 56.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
4,11	2
	READ_EXTERNAL_STORAGE 100%, WRITE_EXTERNAL_STORAGE 100%

Tabla 56. Resumen del estudio de permisos para las app de la categoría fotografía

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 57). Hemos detectado que la aplicación Pro Cam X [99] utiliza en total 9 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, acceder a la ubicación detallada, acceder a la ubicación en segundo plano, reconocer actividades y grabar audio. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Lumii [100], que es una de las aplicaciones que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Pro Cam X	9	2 (22%)	7 (78%)
Lumii	2	2 (100%)	0 (0%)

Tabla 57. Estudio de permisos para dos apps específicas de la categoría fotografía

- **Productividad**

En la categoría productividad dedicada a blocs de notas, listas de tareas, teclados, impresión, calendarios, copias de seguridad, calculadoras, herramientas de conversión. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 5 y 6 permisos como indica la tabla 58.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5,22	2
	READ_EXTERNAL 94%, WRITE_EXTERNAL 88%

Tabla 58. Resumen del estudio de permisos para las app de la categoría productividad

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 59). Hemos detectado que la aplicación Airtel [101] utiliza en total 14 permisos peligrosos (la

que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 12 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a cuentas, leer y enviar SMS, leer estado del teléfono, leer y escribir contactos, realizar llamadas, acceder a la ubicación aproximada, acceder a la ubicación detallada, grabar audio, leer calendario y utilizar la cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Cl@vePIN [102], que no utiliza permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Airtel	14	2 (14%)	12 (86%)
Cl@vePIN	0	0	0

Tabla 59. Estudio de permisos para dos apps específicas de la categoría productividad

El uso de permisos peligrosos de la aplicación Airtel, junto con su descripción, nos hace ver que nos encontramos ante una aplicación que se encuentra dentro de una categoría que no le corresponde.

Por este motivo y tras realizar un nuevo análisis de los resultados obtenidos en la categoría Social hemos decidido comparar las aplicaciones Google Drive [103] y Cl@vePIN (Tabla 60) que son aplicaciones más parecidas.

Google Drive utiliza en total 5 permisos peligrosos. Un estudio de los 3 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a cuentas y leer contactos. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Google Drive	5	2 (40%)	3 (60%)
Cl@vePIN	0	0	0

Tabla 60. Estudio de permisos para dos apps específicas de la categoría productividad

- **Compras**

En la categoría compras dedicada a compras online, subastas, cupones, comparaciones de precios, listas de la compra y reseñas de productos. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_FINE_LOCATION, CAMERA, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 5 y 6 permisos como indica la tabla 61.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5,42	4
	ACCESS_FINE_LOCATION 79%, CAMERA 95%, READ_EXTERNAL 89%, WRITE_EXTERNAL 89%

Tabla 61. Resumen del estudio de permisos para las app de la categoría compras

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 62). Hemos detectado que la aplicación Amazon compras [104] utiliza en total 11 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 8 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceso a la ubicación aproximada, acceso a la ubicación detallada, acceso a la ubicación en segundo plano, grabar audio, reconocer actividades y leer el estado del teléfono. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura NN). En contraposición a esta aplicación tenemos Mister Auto [105], que es una de las que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Amazon Compras	11	3 (27%)	8 (73%)
Mister Auto	1	1 (100%)	0 (0%)

Tabla 62. Estudio de permisos para dos apps específicas de la categoría compras

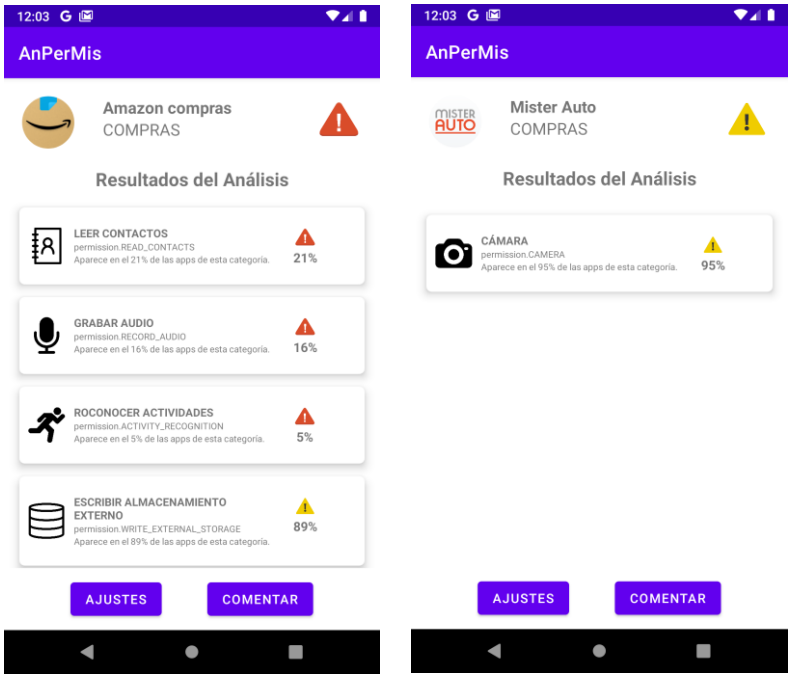


Figura 45. Resultados del análisis de dos apps específicas de la categoría herramientas en AnPerMis

- **Social**

En la categoría social dedicada a redes sociales y registro de visitas. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son CAMERA, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 6 y 7 permisos como indica la tabla 63.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
6,3	3
	CAMERA 76%, READ_EXTERNAL 88%, WRITE_EXTERNAL 76%

Tabla 63. Resumen del estudio de permisos para las app de la categoría social

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 64). Hemos detectado que la aplicación Facebook [106] utiliza en total 13 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 10 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceso a la ubicación aproximada, acceso a la ubicación detallada, acceso a la ubicación en segundo plano, leer el estado del teléfono y escribir contactos. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente

peligrosa (ver Figura 46). En contraposición a esta aplicación tenemos Pi [107], que es una de las aplicaciones que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Facebook	13	3 (23%)	10 (77%)
Pi	1	1 (100%)	0 (0%)

Tabla 64. Estudio de permisos para dos apps específicas de la categoría social

Como se puede observar en los resultados obtenidos, aunque las dos aplicaciones pertenecen a la misma categoría, mientras que Facebook es una red social, Pi es una aplicación que permite administrar la moneda digital Pi, por lo que los resultados de realizar esta comparación serían poco relevantes y hasta se podría considerar que la aplicación Pi está mal clasificada dentro de esta categoría.

Por este motivo y tras realizar un análisis de los resultados obtenidos en la categoría Social hemos decidido comparar las aplicaciones Facebook y Reddit [108] (Tabla 65) que son las aplicaciones más parecidas.

Reddit utiliza 5 permisos (8 menos que la aplicación Facebook). Un estudio de los 2 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación en segundo plano, y leer estado del teléfono. Aunque Reddit utiliza 2 permisos poco comunes para esta categoría, un 60% de sus permisos peligrosos (utilizar la cámara, grabar audio y escribir almacenamiento externo) son permisos comunes en esta categoría por lo que no catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 46).

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Facebook	13	3 (23%)	10 (77%)
Reddit	5	3 (60%)	2 (40%)

Tabla 65. Estudio de permisos para dos apps específicas de la categoría social

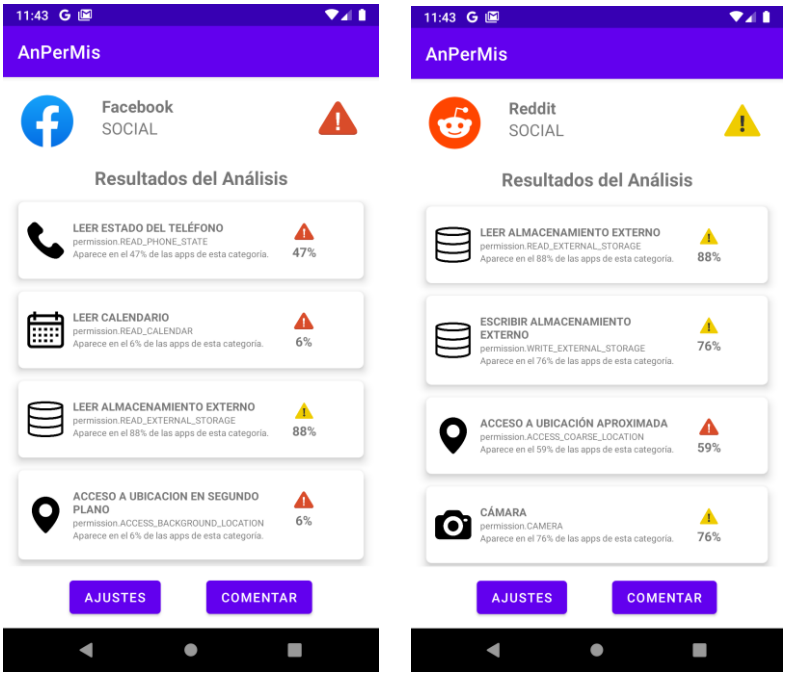


Figura 46. Resultados del análisis de dos apps específicas de la categoría social en AnPerMis

- **Deportes**

En la categoría deportes dedicada a comentarios y noticias deportivas, seguimiento de resultados, administración de equipos virtuales, cobertura de partidos. El estudio que hemos realizado muestra que ningún permiso peligroso aparece en más del 75% de aplicaciones, además, la mayoría de las aplicaciones de esta categoría utilizan 3 permisos como indica la tabla 66.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3	0

Tabla 66. Resumen del estudio de permisos para las app de la categoría deportes

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 67). Hemos detectado que la aplicación NBA [109] utiliza en total 7 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, acceder a la ubicación detallada, acceder a la ubicación en segundo plano, acceder a cuentas y leer y escribir almacenamiento externo. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como

potencialmente peligrosa. En contraposición a esta aplicación tenemos las aplicaciones AS [110] y F1 TV [111] que no utilizan permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
NBA	7	0 (0%)	7 (100%)
AS	0	0	0
F1 TV	0	0	0

Tabla 67. Estudio de permisos para dos apps específicas de la categoría deportes

- **Herramientas**

En la categoría herramientas dedicada a herramientas para dispositivos Android. El estudio que hemos realizado muestra que el permiso peligroso que aparece con mayor frecuencia es READ_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan 3 permisos como indica la tabla 68.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3	1
	READ_EXTERNAL_STORAGE 77%

Tabla 68. Resumen del estudio de permisos para las app de la categoría herramientas

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 69). Hemos detectado que la aplicación Mi Remoto [112] utiliza en total 8 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 8 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, acceder a la ubicación detallada, acceder a cuentas, leer el estado del teléfono, grabar audio, utilizar la cámara y leer y escribir almacenamiento externo. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa (ver Figura 47). En contraposición a esta aplicación tenemos las aplicaciones Binary Code Translator[126] y Google Play Services AR [113] que no utilizan permisos peligrosos.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Mi Remoto	8	0 (0%)	8 (100%)
Binary Code Translator	0	0	0
Google Play Services AR	0	0	0

Tabla 69. Estudio de permisos para dos apps específicas de la categoría herramientas

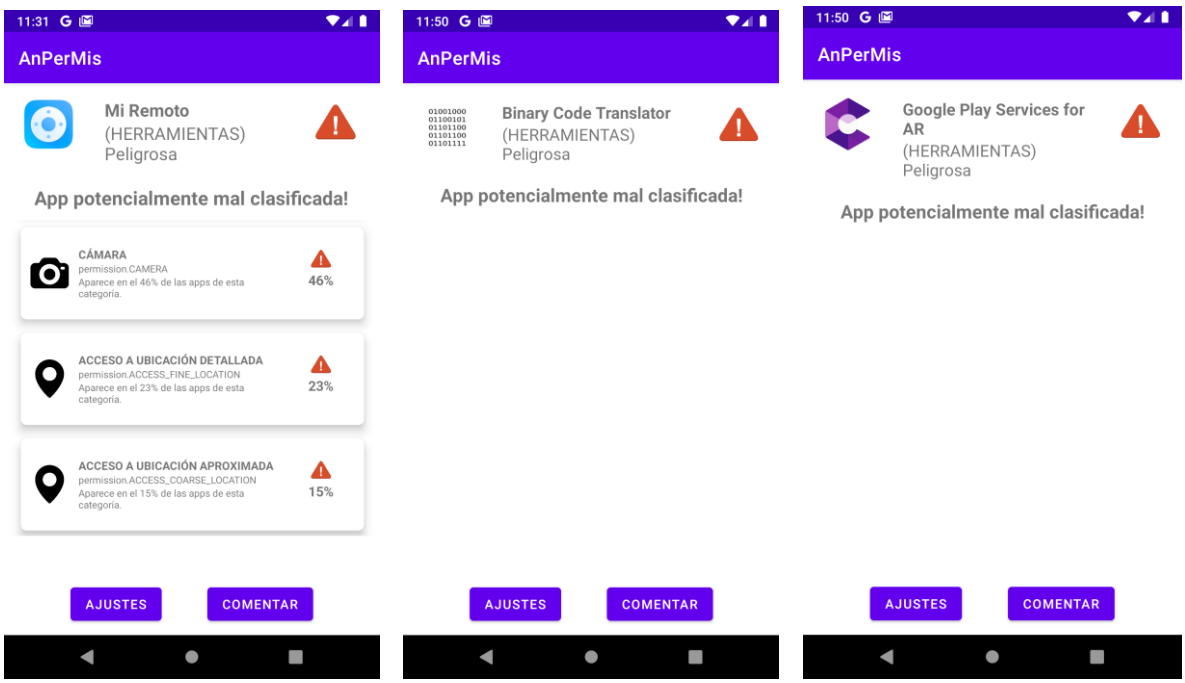


Figura 47. Resultados del análisis de dos apps específicas de la categoría herramientas en AnPerMis

Una gran cantidad de aplicaciones de la Google Play Store se pueden considerar herramientas, por lo que, esta categoría incluye aplicaciones de tipos muy diferentes que podrían considerarse como aplicaciones mal clasificadas.

La definición ambigua de esta categoría invita a los desarrolladores a clasificar sus aplicaciones dentro de esta cuando no saben muy bien cómo clasificarla, por este motivo, AnPerMis considera que todas las aplicaciones de esta categoría utilizan los permisos peligrosos de forma abusiva.

- ## Viajes y guías

En la categoría viajes y guías dedicada a herramientas para reservar viajes, compartir coche, taxis, guías de ciudades, información sobre empresas locales, herramientas de gestión de viajes y reserva de excursiones. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son ACCESS_FINE_LOCATION, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 5 y 6 permisos como indica la tabla 70.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
5,6	3
	ACCESS_FINE_LOCATION 87%, READ_EXTERNAL 87%, WRITE_EXTERNAL 87%

Tabla 70. Resumen del estudio de permisos para las app de la categoría viajes y guías

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 71). Hemos detectado que la aplicación Airbnb [114] utiliza en total 9 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 7 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a ubicación aproximada, acceso a ubicación detallada, acceso a cuentas, leer contactos y grabar audio. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Trivago [115], que es una de las que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Airbnb	9	2 (22%)	7 (78%)
Trivago	1	1 (100%)	0 (0%)

Tabla 71. Estudio de permisos para dos apps específicas de la categoría viajes y guías

- ## Reproductores de vídeo

En la categoría reproductores de vídeo dedicada a reproductores y editores de vídeo, y almacenamiento multimedia. El estudio que hemos realizado muestra que los permisos peligrosos que aparecen con mayor frecuencia son READ_EXTERNAL_STORAGE y

WRITE_EXTERNAL_STORAGE. La mayoría de las aplicaciones de esta categoría utilizan entre 3 y 4 permisos como indica la tabla 72.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3,5	2
	READ_EXTERNAL_STORAGE 100%, WRITE_EXTERNAL_STORAGE 100%

Tabla 72. Resumen del estudio de permisos para las app de la categoría reproductores de vídeo

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 73). Hemos detectado que la aplicación Reproductor MX [116] utiliza en total 5 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 3 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación aproximada, acceder a la ubicación detallada y cámara. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Downloader from Facebook [117], que es una de las que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Reproductor MX	5	2 (40%)	3 (60%)
Downloader from Facebook	2	2 (100%)	0 (0%)

Tabla 73. Estudio de permisos para dos apps específicas de la categoría reproductores de vídeos

- **Tiempo**

En la categoría tiempo dedicada a informes meteorológicos. El estudio que hemos realizado muestra que el permiso peligroso que aparece con mayor frecuencia es ACCESS_FINE_LOCATION. La mayoría de las aplicaciones de esta categoría utilizan entre 3 y 4 permisos como indica la tabla 74.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
3,6	1
	ACCESS_FINE_LOCATION 100%

Tabla 74. Resumen del estudio de permisos para las app de la categoría tiempo

Dentro de las aplicaciones de esta categoría analizamos dos aplicaciones particulares (Tabla 75). Hemos detectado que la aplicación AccuWeather [118] utiliza en total 7 permisos peligrosos (la que más permisos peligrosos utiliza dentro de esta categoría). Un estudio de los 6 permisos poco comunes para esta categoría observamos que nos pide permiso para: acceder a la ubicación en segundo plano, reconocer actividades, leer almacenamiento externo, escribir almacenamiento externo y leer el estado del teléfono. Consideramos que estos permisos están siendo utilizados de forma abusiva y catalogaríamos a esta aplicación como potencialmente peligrosa. En contraposición a esta aplicación tenemos Windfinder [119], que es una de las que menos permisos utiliza en la categoría.

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Accuweather	7	1 (14%)	6 (86%)
Windfinder	1	1 (100%)	0 (0%)

Tabla 75. Estudio de permisos para dos apps específicas de la categoría tiempo

- **Juegos**

Google Play Store ofrece una gran cantidad de categorías para clasificar cada uno de los juegos disponibles, por lo que, para el análisis de los resultados hemos decidido englobar todas estas categorías dentro de la sección Juegos donde se van a mostrar resultados sobre las aplicaciones de juego que más y menos permisos peligrosos utilizan. La mayoría de las aplicaciones de esta categoría utilizan entre 1 y 2 permisos como indica la tabla 76.

Media de permisos utilizados en las aplicaciones	Nº de permisos que aparecen en más del 75%
1,2	0

Tabla 76. Resumen del estudio de permisos para las app de la categoría Juegos

Dentro de las aplicaciones de esta categoría analizamos cuatro aplicaciones particulares (Tabla 77). Deer Hunter Classic que utiliza un total de 6 permisos peligrosos, Minecraft [121] que utiliza 4 permisos peligrosos y AmongUs [122] y Fruit Ninja que no utilizan permisos peligrosos [123].

Nombre	Nº de permisos totales	Nº de permisos que pueden aparecer	Nº de permisos que no pueden aparecer
Deer Hunter Classic	6	0 (0%)	6 (100%)
Minecraft	4	0 (0%)	4 (100%)
AmongUs	0	0	0
Fruit Ninja	0	0	0

Tabla 77. Estudio de permisos para dos apps específicas de la categoría tiempo

8. Trabajo individual

En este capítulo hablaremos de los trabajos y tareas que han sido realizadas por cada uno de los miembros del grupo. La fase previa de organización y maduración de la idea, así como la puesta en común de las propuestas para el análisis de aplicaciones se han realizado por parte de los tres integrantes del grupo, al igual que la documentación previa al inicio del trabajo.

8.1. Andrés Ramiro Ramiro

Andrés Ramiro Ramiro se ha encargado principalmente de todo lo relacionado con el desarrollo de AnPerMis.

Realizó el estudio, investigación y pruebas de los emuladores del sistema Android con el objetivo de obtener el emulador que nos ofreciera el mejor rendimiento y características para utilizar durante la fase de desarrollo y pruebas de AnPerMis.

Tras finalizar esta investigación, comenzó el desarrollo de la aplicación AnPerMis, para ello primero estudió la documentación sobre instalación y configuración de la herramienta Android Studio, y sobre la creación de aplicaciones Android desde cero, obtenida de la guía de desarrolladores de Android.

Una vez creado el proyecto de la aplicación, estudió la implementación de las clases PackageManager y PackageInfo de Android con el objetivo de extraer la lista de paquetes instalados en el dispositivo junto con toda su información. Tras finalizar el estudio de esta documentación, implementó todas las funcionalidades de los módulos Identificación de aplicaciones instaladas y Extracción de permisos peligrosos utilizando estas clases.

Tras una primera implementación de estas funcionalidades básicas y tras darnos cuenta de que no podíamos obtener la categoría de la aplicación, necesaria para realizar el análisis de permisos, desde la información que nos proporcionaban las clases PackageManager y PackageInfo, estudió las clases AsyncTaskLoader y HttpsURLConnection de Android para implementar el módulo Obtención de la categoría de cada aplicación en Google Play Store utilizando como base el Script Python creado por Álvaro Casado. Después de realizar este estudio de clases implementó todas las funcionalidades de este módulo.

Implementó y probó el módulo Clasificación de permisos peligrosos en AnPerMis utilizando la información de las aplicaciones obtenidas en los módulos de funcionamiento anteriores.

Tras decidir que AnPerMis iba a utilizar almacenamiento local de información para poder funcionar sin conexión y mejorar su rendimiento, estudió cómo implementar el módulo Almacenamiento persistente de información, llegando a la conclusión de que lo mejor era utilizar una base de datos Room que permite utilizar toda la potencia que brinda SQLite en Android. Después realizó un estudio de bases de datos Room y diseñó e implementó todo lo necesario para poder crear esta base de datos local.

Una vez implementada en la aplicación la base de datos local utilizando Room estudió cómo utilizar esta base de datos desde la aplicación llegando a la conclusión de que no se podía utilizar el hilo principal de la aplicación para operar con esta base de datos local. Para solucionar este problema decidió implementar todas las clases necesarias para operar con la base de datos local utilizando tareas asíncronas de Android o AsyncTask.

Tras implementar y probar todos estos módulos de funcionalidad básica y acordar el uso de un servidor remoto para obtener las tablas de análisis, implementó todas las clases del módulo Comunicación con el servidor remoto utilizando Sockets de Android. Una vez probado todo este sistema de comunicación con el servidor utilizando Sockets sin seguridad de Android decidimos utilizar un nuevo sistema de comunicación más seguro para conectar AnPerMis con el servidor remoto.

Después de llegar a la conclusión de que lo mejor para implementar este sistema de comunicación, era utilizar una conexión SSL sin autenticación de cliente, implementó de nuevo este módulo de Comunicación con el servidor remoto utilizando SocketsSSL de Android, además, una vez probado el nuevo sistema de comunicación implementó el módulo Envío de información de la aplicación al servidor.

Diseñó e implementó el modo de funcionamiento sin conexión, con el objetivo de que AnPerMis funcionará en la mayor cantidad de casos de uso posibles.

Tras acordar cuál iba a ser el diseño de la interfaz de usuario de AnPerMis estudió los recursos necesarios para poder implementar dicha interfaz y tras realizar este estudio implementó la interfaz de usuario utilizando estos recursos.

Una vez implementada esta interfaz de usuario y tras realizar pruebas en usuarios reales, en las que se encontraron errores, realizó las correcciones de estos errores llegando a la interfaz de usuario final. Después también implementó las funcionalidades de Publicación de comentarios en Google Play Store y Revocar permisos abusivos de las aplicaciones.

Finalmente colaboró durante la fase de pruebas de AnPerMis, entrenamiento del servidor remoto, corrección de errores, análisis de resultados, y también en el desarrollo de la memoria añadiendo los puntos 5.1, 5.2, 6 y 7.

8.2. Álvaro Casado Molinero

Álvaro Casado se ha encargado principalmente del diseño e implementación del servidor.

Realizó un programa en Python para obtener la categoría de una aplicación buscando directamente en la página de Google Play Store utilizando su identificador que sirvió como aproximación a lo que sería la implementación final de esta funcionalidad en AnPerMis.

Estudió toda la documentación necesaria para implementar una conexión cliente-servidor sin autenticación de cliente basada en el uso de sockets SSL. Generó el par de claves público-privadas y creó el certificado del servidor para la conexión segura del mismo utilizando la herramienta keytool de Java.

Luego implementó e hizo pruebas en local de la conexión mediante sockets-tcp con el esquema *accept-and-fork* que permite varios clientes al mismo tiempo en un solo servidor, también para realizar pruebas creó una base de datos local.

Programó una primera versión en Java del servidor diseñando una base de datos relacional para el almacenamiento persistente de la información obtenida por la aplicación AnPerMis, también creó todas las consultas necesarias para obtener e insertar la información en la base de datos. Además, configuró todo lo relacionado con la instancia virtual de Amazon AWS y la base de datos, instaló MySQL, PhpMyAdmin, y configuró los permisos de acceso a la instancia.

Implemento las funcionalidades relevantes del servidor como crear la tabla de frecuencias a partir de la información de la tabla “permisosapp”, crear la caché de información de aplicaciones para mejorar los resultados y el rendimiento del servidor, añadir a la base de datos la tabla de históricos “historico” para detectar posibles cambios de información en las aplicaciones y la actualización automática de todas las tablas en la base de datos del servidor con el objetivo de mantener siempre actualizada la información que utiliza AnPerMis para realizar el análisis de uso de permisos.

Por último, colaboró durante la fase de pruebas de AnPerMis, entrenamiento del servidor remoto, análisis de resultados y también en el desarrollo de la memoria añadiendo el punto 5.3.

8.3. Alejandro Ortiz Pintado

En una primera fase del proyecto, Alejandro se encargó de recopilar toda la información relacionada con el funcionamiento y estructura de las aplicaciones de Android, el sistema de permisos y la categorización de aplicaciones en Google Play Store para exponerla en la memoria y explicar los conceptos de manera concisa para la comprensión del funcionamiento de la herramienta.

Tras esto, se puso en contacto con creadores de repositorios de metadatos de aplicaciones (estos recopilatorios contenían miles de aplicaciones con los permisos solicitados) para empezar a entrenar el servidor, pero ante la imposibilidad de obtener estos recopilatorios, creó manualmente una primera versión de la tabla de aplicaciones con más de 130 aplicaciones incluyendo su categoría y permisos peligrosos solicitados. De esta manera logramos un primer punto de partida para empezar a generar los análisis.

Colaboró con Álvaro Casado en el diseño del algoritmo del servidor que calcula las frecuencias de aparición de los permisos peligrosos en las distintas aplicaciones según su categoría.

Realizó un estudio de las categorías disponibles en Google Play Store con el objetivo de obtener una primera aproximación al criterio común para el análisis de los posibles abusos de permisos por parte de las aplicaciones. Detalló casos de uso en los que los permisos son utilizados por las aplicaciones de forma abusiva y detalló qué categorías de aplicaciones (según la definición de Google) necesitan estos permisos para su funcionamiento.

Estudió las categorías consideradas como peligrosas para evitar errores en el perfilado de la herramienta. Estas categorías son las que, ya sea por mala descripción o por contener todo tipo de aplicaciones mal categorizadas, pueden llevar a AnPerMis a dar resultados erróneos.

Diseñó las distintas pantallas de la aplicación generando ejemplos con Balsamiq Wireframes para su organización y posterior implementación en la aplicación final.

Finalmente colaboró durante la fase de pruebas de AnPerMis, implementación de la comunicación cliente-servidor, entrenamiento del servidor remoto, corrección de errores, análisis de resultados y también en el desarrollo de la memoria.

9. Conclusiones, asignaturas utilizadas y trabajo futuro

En este capítulo desarrollaremos las conclusiones obtenidas tras la realización de este trabajo, las asignaturas utilizadas para su desarrollo y las posibles tareas a realizar en un futuro de cara a mejorar nuestra herramienta.

9.1. Conclusiones

AnPerMis es una herramienta completamente funcional y preparada para procesar simultáneamente peticiones de numerosos usuarios. Como hemos expuesto en capítulos anteriores, realiza un correcto análisis de los permisos peligrosos solicitados por las aplicaciones instaladas en el dispositivo del usuario y detecta eficazmente el uso abusivo de estos permisos exponiendo los factores por los que las aplicaciones pueden estar haciendo este uso indebido de ellos. Además, ha sido probada por usuarios reales y cuenta con compatibilidad total para todas las versiones de Android. Aunque el análisis generado por AnPerMis es bastante completo, se necesita una interpretación de los resultados por parte del usuario, ya que, aunque un permiso pueda parecer abusivo, su uso puede ser completamente lícito y justificado y debe ser el propio usuario quien decida qué permisos revocar a las aplicaciones y cuáles decide mantener.

En relación con los resultados obtenidos del análisis de las 629 aplicaciones realizado en este trabajo, hemos detectado que 422, un 66,98%, utilizan de forma abusiva más de un 60% de los permisos que solicitan y, además, 326 de estas utilizan de forma abusiva todos sus permisos. Las aplicaciones analizadas utilizan de media entre 3 y 4 permisos, siendo los más comunes, READ_EXTERNAL_EXTORAGE (453 aplicaciones), WRITE_EXTERNAL_STORAGE (408 aplicaciones), CAMERA (259 aplicaciones) y ACCESS_COARSE_LOCATION (259 aplicaciones). También cabe destacar que en ninguna de las categorías analizadas aparecen más de 6 permisos solicitados por más del 75% de sus aplicaciones.

Un posible resultado de la utilización de la herramienta por parte de los usuarios es que podría forzar a Google a que mejore el sistema de clasificación por categorías de las aplicaciones en Google Play Store. Este sistema de clasificación afecta directamente a la precisión de los resultados generados por AnPerMis, ya que, si la aplicación a analizar está mal categorizada, los resultados no van a ser tan precisos como serían si se realizase el perfilado con la categoría correcta. Muchas categorías son bastante confusas y acaban conteniendo aplicaciones de todo tipo que distan mucho de la descripción que da Google de estas categorías. Una posible mejora para este sistema de clasificación sería la inclusión de más categorías de manera que se acote con mayor granularidad la finalidad de las aplicaciones y así evitar su mala categorización. Aunque durante el desarrollo de este trabajo, Google ha añadido más categorías o ha renombrado las existentes para mejorar su comprensión, el problema de las

aplicaciones mal categorizadas persiste. Por tanto, otra mejora para este sistema es que sea Google quien confirme la categorización propuesta por los desarrolladores o directamente categorice las aplicaciones. De esta manera no habría tantas aplicaciones mal categorizadas con lo que el perfilado de AnPerMis mejoraría mucho y se obtendrían resultados considerablemente mejores. Además, nuestra herramienta seguiría funcionando perfectamente, aunque se realizasen estas optimizaciones en las categorías, ya que ha sido diseñada para realizar el análisis sin importar la estructura de dicho sistema de categorización de las aplicaciones.

El trabajo realizado nos ha permitido averiguar que muchas aplicaciones solicitan un permiso peligroso para realizar una acción muy concreta y puntual. Por ejemplo, las aplicaciones de música solicitan el acceso a la lectura del estado del teléfono para detectar si está recibiendo una llamada y pausar la reproducción. Pero al conceder este permiso no solo se da acceso a detectar si hay una llamada entrante, sino que puede consultar el estado del teléfono constantemente pudiendo detectar si el dispositivo está en reposo o realizando alguna actividad. Esto se podría solucionar si, como en el caso de las categorías, se añadiese mayor granularidad en la definición de los permisos. Si en lugar de un permiso para leer el estado del teléfono, este se divide en varios para consultar los distintos estados del mismo, se podría conceder a las distintas aplicaciones acceso únicamente a las funcionalidades necesarias para su operación, conservando el principio de mínimo privilegio. El problema de esta cuestión radica en que Android define una serie de buenas prácticas a la hora de desarrollar aplicaciones, de manera que los desarrolladores únicamente deberían solicitar los permisos imprescindibles para el uso de la aplicación, pero luego no existe ningún proceso de auditoría para comprobar si realmente se está cumpliendo este principio de mínimo privilegio. De esta manera, se pueden solicitar permisos de más sin ningún tipo de penalización, obteniendo acceso a más funcionalidades protegidas de las realmente necesarias. Creemos que, si la inclusión de los permisos más concretos se acompaña de la declaración al usuario de los permisos que se solicitan, junto con una explicación de su solicitud, podría aumentar considerablemente la transparencia del uso de permisos y la experiencia sería más amigable para los usuarios.

Aunque finalmente nos centramos en el análisis de permisos peligrosos de aplicaciones, este trabajo comenzó tratando sobre análisis de malware en Android. Al empezar la fase documentación de este nos dimos cuenta de que el uso abusivo de estos permisos peligrosos puede suponer un riesgo para la privacidad de los usuarios y un vector de entrada de todo tipo de malware, por este motivo, decidimos enfocar este trabajo en el desarrollo de AnPerMis.

9.2. Aprendizajes del grado empleados en el desarrollo de este proyecto

En esta sección desatacamos los aprendizajes, asociados a asignaturas particulares, adquiridos durante el desarrollo del grado que nos han ayudado a desarrollar este trabajo.

- **Programación de aplicaciones para dispositivos móviles**, lo aprendido en esta asignatura ha sido utilizado para el diseño y desarrollo de la aplicación AnPerMis.
- **Tecnología de la programación**, los conocimientos adquiridos nos han servido para la programación de AnPerMis y del servidor remoto.
- **Diseño de sistemas interactivos,** ha sido de utilidad para crear el diseño de la interfaz de usuario de la aplicación.
- **Cloud y Big Data**, lo impartido en esta asignatura ha sido utilizada para configurar y utilizar las herramientas de computación en la nube necesarias para el funcionamiento del servidor remoto.
- **Bases de datos**, lo aprendido nos ha servido para diseñar y manejar las bases de datos de la aplicación y del servidor remoto.
- **Estructuras de datos**, nos ha ayudado a manejar todas las estructuras de datos necesarias en el desarrollo de la aplicación y del servidor remoto.
- **Ética, legislación y profesión**, los conceptos estudiados en esta asignatura han sido muy útiles a la hora de estudiar y crear la tabla de criterios de aparición de los permisos en las categorías.
- **Ingeniería del software**, los conocimientos adquiridos han sido utilizados para planear, desplegar y organizar el desarrollo del trabajo durante todas sus etapas.

9.3. Trabajo futuro

En el transcurso de este proyecto, tanto en la fase inicial como una vez comenzado el desarrollo de la herramienta, han surgido ideas para mejorar o ampliar la funcionalidad de AnPerMis. Aunque esta herramienta es completamente funcional y está preparada para su uso, no deja de ser una primera versión con mucho margen de mejora. Las principales tareas a realizar en un futuro serían las siguientes:

- AnPerMis realiza el análisis de todas las aplicaciones instaladas en el dispositivo del usuario sin comprobar si son descargadas desde el repositorio oficial de Google Play Store. Una extensión de la funcionalidad interesante sería comprobar si los paquetes de la aplicación provienen de este repositorio o de repositorios de terceros como Aptoide. Estas aplicaciones de orígenes externos pueden incluir malware o solicitar

permisos distintos a los de las aplicaciones oficiales, por lo que sería interesante que nuestra herramienta pudiese avisar de esto al usuario una vez realizado el análisis.

- Para publicar una aplicación en Google Play Store, hace falta que el paquete de instalación de esta esté firmado. Existe la posibilidad de que un desarrollador autofirme su aplicación de manera que pueda incluir malware en el código o solicite permisos de forma abusiva. Una tarea interesante sería estudiar si las aplicaciones están autofirmadas, añadiendo así mucho más valor al análisis.

- Al planear el desarrollo de AnPerMis, valoramos la posibilidad de implementar la funcionalidad de conceder permisos por un tiempo limitado, de manera que se revoquen automáticamente pasado este tiempo. A pesar de que, durante las primeras fases de este proyecto, con la aparición de Android 11 ya se incluyó esta funcionalidad, aún nos sigue pareciendo interesante su inclusión en nuestra herramienta, ya que, de esta manera, como AnPerMis es compatible con todas las versiones de Android, los usuarios con versiones anteriores del sistema operativo podrían hacer uso de esta opción.

- Una de las funcionalidades a implementar en un futuro que nos parece más interesante es el análisis del uso que hacen las aplicaciones de los permisos consultando las llamadas al sistema. De esta manera no solo comprobamos los permisos que solicitan las aplicaciones, sino que también estudiamos si se está realizando un uso excesivo de los mismos o si se están utilizando sin conocimiento del usuario. Esto permitiría generar un análisis mucho más profundo, pudiendo diferenciar cuándo una aplicación solicita un permiso peligroso por necesidad o si está haciendo un uso excesivo del mismo.

- En un futuro también se podría añadir a la clasificación de los permisos de AnPerMis un nuevo tipo de permisos, que serían los que históricamente han sido constatados como permisos típicos de aplicaciones que han supuesto la introducción de malware en dispositivos, esto mejoraría sustancialmente la clasificación y el análisis de los permisos.

- Por último, una mejora considerable sería analizar una mayor cantidad de aplicaciones, porque cuantas más aplicaciones haya analizado AnPerMis, más fiables son los resultados. Esto es así ya que, al añadir nuevas aplicaciones a la base de datos, se puede realizar un mejor perfilado de los permisos que solicitan las aplicaciones de cada categoría.

10. Conclusions and Future Work

In this chapter we will develop the conclusions obtained after carrying out this project and the possible tasks to be accomplished in the future to improve our tool.

10.1. Conclusions

AnPerMis is a fully functional tool designed to process several user analyses requests simultaneously. As we have explained in previous chapters, it performs a correct analysis of the dangerous permissions requested by the applications installed on the user's device and effectively detects the abuse of these permissions, exposing the factors why the applications may be making this improper use of permissions. In addition, it has been tested by real users and has full compatibility with all Android versions. Although the analysis generated by AnPerMis is quite complete, an interpretation of the results by the user is needed, because even if a permission may seem abusive, its use can be completely justified, and it must be the user himself who decides what permissions he wants to grant to the applications and what permissions he wants to revoke.

Regarding the results obtained from analyzing 629 applications carried out in this study, we have detected that 422 of these applications, about 66.98%, abuse in more than 60% of the permissions requested, in addition, 326 of these applications abuse in all their permissions. The applications analyzed use on average between 3 and 4 permissions, the most common being READ_EXTERNAL_EXTORAGE (453 applications), WRITE_EXTERNAL_STORAGE (408 applications), CAMERA (259 applications) and ACCESS_COARSE_LOCATION (259 applications). It should also be noted that none of the categories analyzed use more than 6 permissions requested by more than 75% of their applications.

One possible result of the use of the tool by the users is that it could force Google to improve the classification system by categories of applications in the Google Play Store. This classification system directly affects the accuracy of the results generated by AnPerMis, because if the application to be analyzed is wrongly categorized, the results will not be as precise as they would be if profiling were performed with the application categorized as it should be. Many categories are quite confusing and end up containing applications of all kinds that are far from the description given by Google for these categories. A possible solution to this problem would be the inclusion of more categories to limit the purpose of the applications with greater granularity and avoid the wrong categorization of many applications. Although during the development of this work, Google has added more categories or has renamed the existing ones to improve its understanding, the problem of miscategorized already existing applications persists. Therefore, another possible solution to this problem is for Google to confirm the categorization proposed by the developers or directly categorize the applications themselves. In this way, there would not be so many badly categorized applications and the

profiling of AnPerMis would improve a lot, enhancing the results considerably, since, although changes are made in the category system, our tool is designed to perform the analysis regardless of the structure of the category system.

This project has allowed us to find out that many applications request a dangerous permission to perform a very concrete and punctual action. For example, music applications request access to read the phone status to detect if you are receiving a call and pause playback. But by granting this permission, you do not only give access to detect if there is an incoming call, but you give the application permission to constantly check the status of the phone, being able to detect if the device is in sleep mode or performing some activity. This could be solved if, as in the case of categories, greater granularity was added in the definition of permissions. If instead of defining a permission to read the state of the phone, it is divided into several different permissions to consult the different states of the phone, the applications could be granted access only to the functionalities necessary for their performing, preserving the principle of least privilege. The problem is that Android defines good practices to develop applications, so that developers should only request necessary permissions to use the application, but there is no audit process to verify if this principle of least privilege is really being kept. In this way, developers can request more permissions without any type of penalty, obtaining access to more protected functionalities than necessary. We believe that if the inclusion of more specific permissions is accompanied by the declaration to the user of the requested permissions along with an explanation of their request, it could considerably increase the transparency of the use of permissions and the experience would be more user-friendly.

Although we finally focused on the analysis of dangerous permissions, this work started being about malware on Android. At the beginning of the documentation phase we realised that the misuse of these dangerous permissions can cause a risk to the users privacy and can be an entry vector for all types of malware.

10.2. Degree subjects used in the development of this project

In this section we highlight the knowledge acquired from specific subjects during the course of our degrees that have helped us to develop this project.

- **Application Programming for Mobile Devices**: the acquired knowledge in this subject has been used for the design and development of AnPerMis.
- **Computer Programming Technology:** this subject has helped for the programming of AnPerMis and the server.
- **Evaluation of Computer Systems:** it has been useful to create the design of the application's user interface.

- **Cloud and Big Data**: applying what we have learned in this subject has vital to configure and use the cloud computing tools necessary for the operation of the remote server.
- **Databases:** this subject has helped us to design and manage the databases used in the application and the remote server.
- **Data structures**: it has helped us to manage all the necessary data structures in the development of the application and the remote server.
- **Ethics, legislation and profession**: the concepts studied in this subject have been very useful when studying and creating the criteria for the appearance of permissions in the categories.
- **Software engineering**: the knowledge acquired has been used to plan, deploy and organize the development of the work during all its stages.

10.3. Future Work

During this project, both in the initial phase and once the development of the tool began, ideas have arisen to improve or extend AnPerMis functionality. Although this tool is fully functional and ready for its use, it is still a first version with a lot of room for improvement. The main tasks to be carried out in the future would be the following:

- AnPerMis performs its analysis on all applications installed on the user's device without checking if they are downloaded from the official Google Play Store repository. An interesting extension of the functionality would be to check if the application packages come from this repository or from third party repositories like Aptoide. These applications from external sources can include malware or request different permissions than the official applications and warning the user about this when performing the analysis would be a good inclusion.

- In order to publish an application in Google Play Store, the package of the application must be signed. A developer can self-sign their application, so it could include malware in the code or request permissions in an abusive way. An interesting task would be to study if the applications are self-signed adding much more value to the analysis.

- When planning the development of AnPerMis, we thought about possibility of implementing the functionality to grant permissions for a limited time, so that they are automatically revoked after this time has passed. During the first phases of the project, with the Android 11 update, this functionality was included, although its implementation is still interesting since in this way, as AnPerMis is compatible with all

Android versions, users with previous versions of the operating system could make use of these functionalities.

- One of the functionalities to be implemented in the future that seems more interesting to us is the analysis of the use that applications make of the permissions by consulting the system calls. In this way, we do not only check the permissions requested by the applications, but we also study if they are being used excessively or if they are being used without the user's knowledge. This would allow to generate a much deeper analysis, being able to differ when an application requests a dangerous permission out of necessity or if it is making excessive use of it.

- In a near future, a new type of permissions could also be added to the classification of AnPerMis permissions, which would be those that have historically been verified as typical permissions requested by applications that have led to the deployment of malware in devices. This would substantially improve the classification and the analysis of permissions.

- Finally, a considerable improvement would be to analyze a greater number of applications, because the more applications AnPerMis has analyzed, the more reliable the results are. When adding new applications to the database, a better profiling of the permissions requested by the applications of each category can be carried out.

BIBLIOGRAFÍA

[1] Los datos ocultos en tu teléfono que pueden traicionar tu privacidad.
https://www.bbc.com/mundo/noticias/2015/02/150213_vert_fut_metadata_telefono_privacidad_yv

[2] Adrienne Porter Felt, Kate Greenwood and David Wagner (2011). The Effectiveness of Application Permissions. Proceddings of the 2nd USENIX conference on Web application development.

[3] Zheran Fang, WeiliHan and Yingjiu Li. Permission based Android security: Issues and countermeasures. Computers & Security. Volume 43, June 2014, Pages 205-218

[4] Biniam Fisseha Demissie, Mariano Ceccato & Lwin Khin Shar. Security analysis of permission re-delegation vulnerabilities in Android apps. Empirical Software Engineering volume 25, pages5084–5136 (2020)

[5] Actualizaciones de permisos en Android 11
https://developer.android.com/about/versions/11/privacy/permissions?hl=es-419

[6] Acerca de las actualizaciones de iOS 14.
https://support.apple.com/es-lamr/HT211808#145

[7] Conan Mobile en Google Play Store.
https://play.google.com/store/apps/details?id=es.inteco.conanmobile&hl=es&gl=US

[8] Aspectos fundamentales de una aplicación Android.
https://developer.android.com/guide/components/fundamentals?hl=es-419

[9] Archivo Android Manifest. https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

[10] Flujo de trabajo para el uso de permisos.
https://developer.android.com/guide/topics/permissions/overview?hl=es-419#workflow

[11] Prácticas recomendadas para el uso de permisos en Android.
https://developer.android.com/guide/topics/permissions/overview?hl=es-419#best-practices

[12] Solicitar permisos en tiempo de ejecución.
https://developer.android.com/training/permissions/requesting?hl=es-419

[13] Permisos en Android.
https://developer.android.com/guide/topics/permissions/overview?hl=es-419

[14] Página de referencia de la API de permisos, Developer Android.
https://developer.android.com/reference/android/Manifest.permission?hl=es-419

[15] Categorías en Google Play Store
https://support.google.com/googleplay/android-developer/answer/9859673?hl=es#zippy=%2Caplicaciones

[16] Samsung Pay en Google Play Store
https://play.google.com/store/apps/details?id=com.samsung.android.spay&hl=es&gl=US

[17] Pokemon Go en Google Play Store
https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=es&gl=US

[18] Tinder en Google Play Store
https://play.google.com/store/apps/details?id=com.tinder&hl=es&gl=US

[19] Canva. https://www.canva.com/es_es/

[20] Android Studio. https://developer.android.com/studio

[21] Android Virtual Device (AVD)
https://developer.android.com/studio/run/managing-avds?hl=es-419

[22] Android x86 Project. https://www.android-x86.org/

[23] Bluestacks 4. https://www.bluestacks.com/es/index.html

[24] Balsamiq Wireframes. https://balsamiq.com/wireframes/

[25] Eclipse IDE. https://www.eclipse.org/ide/

[26] PhpMyAdmin. https://www.phpmyadmin.net/

[27] Amazon Web Services (AWS). https://aws.amazon.com/es/

[28] Guardar contenido en una base de datos local con Room
https://developer.android.com/training/data-storage/room

[29] Clase AsyncTask Android
https://developer.android.com/reference/android/os/AsyncTask

[30] Clase Task Android
https://developer.android.com/reference/com/google/android/play/core/tasks/Task

[31] Clase PackageManager Android
https://developer.android.com/reference/android/content/pm/PackageManager?hl=es

[32] Clase PackageInfo Android
https://developer.android.com/reference/android/content/pm/PackageInfo

[33] Elegir la categoría y las etiquetas de una aplicación o un juego
https://support.google.com/googleplay/android-developer/answer/9859673?hl=es

[34] Clase AsyncTaskLoader Android
https://developer.android.com/reference/androidx/loader/content/AsyncTaskLoader

[35] Clase HttpsURLConnection Android
https://developer.android.com/reference/kotlin/javax/net/ssl/HttpsURLConnection

[36] Clase NetworkInfo Android
https://developer.android.com/reference/android/net/NetworkInfo

[125] SSLSocket Android
https://developer.android.com/reference/javax/net/ssl/SSLSocket

[38] Tareas programadas en Java.
https://programandoointentandolo.com/2014/09/ejecutar-metodo-a-hora-especifica-java.html

[39] Guía para instalar MySQL en Ubuntu 20.04
https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04-es

[40] Clase TimerTask Java
https://docs.oracle.com/javase/7/docs/api/java/util/TimerTask.html

[41] SocketsSSL en Java
http://chuwiki.chuidiang.org/index.php?title=Socket_SSL_con_Java

[42] Cómo crear una IU responsiva con ConstraintLayout
https://developer.android.com/training/constraint-layout?hl=es-419

[43] Cómo crear listas dinámicas con RecyclerView
https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=es

[44] Cómo crear un diseño basado en tarjetas
https://developer.android.com/guide/topics/ui/layout/cardview?hl=es

[45] Cuadros de diálogo
https://developer.android.com/guide/topics/ui/dialogs?hl=es-419

[46] Splash Screen
https://bignerdranch.com/blog/splash-screens-the-right-way/

[47] Permisos peligrosos android
https://developer.android.com/guide/topics/permissions/overview?hl=es-419#dangerous_permissions

[48] Behance en Google Play Store
https://play.google.com/store/apps/details?id=com.behance.behance&hl=es_419&gl=US

[49] Photoshop Sketch en Google Play Store
https://play.google.com/store/apps/details?id=com.adobe.creativeapps.sketch&hl=es_419&gl=US

[50] Coches.net en Google Play Store
https://play.google.com/store/apps/details?id=coches.net&hl=es&gl=US

[51] Carista OBD en Google Play Store
https://play.google.com/store/apps/details?id=com.prizmos.carista&hl=es&gl=US

[52] Booksy BIZ en Google Play Store
https://play.google.com/store/apps/details?id=net.booksy.business&hl=es&gl=US

[53] Mi cronograma capilar en Google Play Store
https://play.google.com/store/apps/details?id=br.art.code.meucronogramacapilar&hl=es&gl=US

[54] Zoom en Google Play Store
https://play.google.com/store/apps/details?id=us.zoom.videomeetings&hl=es&gl=US

[55] InfoJobs en Google Play Store
https://play.google.com/store/apps/details?id=net.infojobs.mobile.android&hl=es&gl=US

[56] WebToon en Google Play Store
https://play.google.com/store/apps/details?id=com.naver.linewebtoon&hl=es&gl=US

[57] Mango en Google Play Store
https://play.google.com/store/apps/details?id=com.jegulabs.mango&hl=es&gl=US

[58] My Heritage en Google Play Store

https://play.google.com/store/apps/details?id=air.com.myheritage.mobile&hl=es&gl=US

[59] BMW Drivers Guide en Google Play Store
https://play.google.com/store/apps/details?id=com.bmwgroup.driversguide.usa&hl=es&gl=US

[60] WhatsApp en Google Play Store
https://play.google.com/store/apps/details?id=com.whatsapp&hl=es&gl=US

[61] Orbot en Google Play Store
https://play.google.com/store/apps/details?id=org.torproject.android&hl=es&gl=US

[62] Line en Google Play Store
https://play.google.com/store/apps/details?id=jp.naver.line.android&hl=es&gl=US

[63] Badoo en Google Play Store
https://play.google.com/store/apps/details?id=com.badoo.mobile&hl=es&gl=US

[64] Idilium en Google Play Store
https://play.google.com/store/apps/details?id=meet.new.people.make.friends.free.dating.video.chat

[65] Duolingo en Google Play Store
https://play.google.com/store/apps/details?id=com.duolingo

[66] Google Classroom en Google Play Store
https://play.google.com/store/apps/details?id=com.google.android.apps.classroom

[67] Bitmoji en Google Play Store
https://play.google.com/store/apps/details?id=com.bitstrips.imoji

[68] Disney+ en Google Play Store
https://play.google.com/store/apps/details?id=com.disney.disneyplus

[69] Prime Video en Google Play Store
https://play.google.com/store/apps/details?id=com.amazon.avod.thirdpartyclient

[70] Greetings Island en Google Play Store
https://play.google.com/store/apps/details?id=com.greetingsisland.sam

[71] Backpack Mod en Google Play Store
https://play.google.com/store/apps/details?id=minecraft.mk.mods.h

[72] Avengers Mod en Google Play Store
https://play.google.com/store/apps/details?id=minecraft.mk.mods.j

[73] Ticketmaster en Google Play Store
https://play.google.com/store/apps/details?id=com.ticketmaster.tickets.international

[74] Imagin en Google Play Store
https://play.google.com/store/apps/details?id=com.imaginbank.app

[75] Coinbase en Google Play Store
https://play.google.com/store/apps/details?id=com.coinbase.android

[76] Banco Santander en Google Play Store
https://play.google.com/store/apps/details?id=es.bancosantander.apps

[77] McDo+ en Google Play Store
https://play.google.com/store/apps/details?id=com.md.mcdonalds.gomcdo

[78] Burger King en Google Play Store
https://play.google.com/store/apps/details?id=es.burgerking.android

[79] Fitbit en Google Play Store
https://play.google.com/store/apps/details?id=com.fitbit.FitbitMobile

[80] Contador de pasos en Google Play Store
https://play.google.com/store/apps/details?id=com.sudo.den.stepcounter

[81] Housing en Google Play Store
https://play.google.com/store/apps/details?id=com.locon.housing

[82] Trovit Casas en Google Play Store
https://play.google.com/store/apps/details?id=es.roid.and.trovit

[83] IP Pro en Google Play Store
https://play.google.com/store/apps/details?id=com.specialyg.ippro

[84] Mirror Link Screen Connector en Google Play Store
https://play.google.com/store/apps/details?id=com.connection.wireless.wirelessconnectortv

[85] Amazon Alexa en Google Play Store
https://play.google.com/store/apps/details?id=com.amazon.dee.app

[86] Flying Robot Hero en Google Play Store
https://play.google.com/store/apps/details?id=com.TGP.FlyingRobotGame.SuperheroIronRobotFighting

[87] Uber en Google Play Store
https://play.google.com/store/apps/details?id=com.ubercab

[88] Spain Travel Health en Google Play Store
https://play.google.com/store/apps/details?id=com.atos.spain.th

[89] Quirón Salud en Google Play Store
https://play.google.com/store/apps/details?id=com.divisait.quironsalud

[90] Radar COVID en Google Play Store
https://play.google.com/store/apps/details?id=es.gob.radarcovid

[91] Castbox en Google Play Store
https://play.google.com/store/apps/details?id=fm.castbox.audiobook.radio.podcast

[92] Podcast de Google en Google Play Store
https://play.google.com/store/apps/details?id=com.google.android.apps.podcasts

[93] Microsoft News en Google Play Store
https://play.google.com/store/apps/details?id=com.microsoft.amp.apps.bingnews

[94] GeekTech en Google Play Store
https://play.google.com/store/apps/details?id=geektech.technewsapp

[95] Control Parental Kroha en Google Play Store
https://play.google.com/store/apps/details?id=ua.com.tim_berners.parental_control

[96] Consejos para padres en Google Play Store
https://play.google.com/store/apps/details?id=com.andromo.dev616791.app750403

[97] Square Home en Google Play Store
https://play.google.com/store/apps/details?id=com.ss.squarehome2

[98] Notify Buddy en Google Play Store
https://play.google.com/store/apps/details?id=com.xander.android.notifybuddy

[99] Pro Cam X en Google Play Store
https://play.google.com/store/apps/details?id=com.intermedia.hd.camera.professional

[100] Lumii en Google Play Store
https://play.google.com/store/apps/details?id=photo.editor.photoeditor.filtersforpictures

[101] Airtel en Google Play Store
https://play.google.com/store/apps/details?id=com.myairtelapp

[102] Cl@ve Pin en Google Play Store
https://play.google.com/store/apps/details?id=es.aeat.pin24h

[103] Google Drive en Google Play Store
https://play.google.com/store/apps/details?id=com.google.android.apps.docs

[104] Amazon Compras en Google Play Store
https://play.google.com/store/apps/details?id=com.amazon.mShop.android.shopping

[105] Mister Auto en Google Play Store
https://play.google.com/store/apps/details?id=com.misterauto.misterauto

[106] Facebook en Google Play Store
https://play.google.com/store/apps/details?id=com.facebook.katana

[107] Pi en Google Play Store
https://play.google.com/store/apps/details?id=com.blockchainvault

[108] Reddit en Google Play Store
https://play.google.com/store/apps/details?id=com.reddit.frontpage

[109] NBA en Google Play Store
https://play.google.com/store/apps/details?id=com.nbaimd.gametime.nba2011

[110] AS en Google Play Store
https://play.google.com/store/apps/details?id=es.mmip.prisacom.as

[111] F1 TV en Google Play Store
https://play.google.com/store/apps/details?id=com.formulaone.production

[112] Mi Remoto en Google Play Store
https://play.google.com/store/apps/details?id=com.duokan.phone.remotecontroller

[113] Google Play Services AR en Google Play Store
https://play.google.com/store/apps/details?id=com.google.ar.core

[114] Airbnb en Google Play Store
https://play.google.com/store/apps/details?id=com.airbnb.android

[115] Trivago en Google Play Store
https://play.google.com/store/apps/details?id=com.trivago

[116] Reproductor MX en Google Play Store
https://play.google.com/store/apps/details?id=com.mxtech.videoplayer.ad

[117] Downloader from Facebook en Google Play Store
https://play.google.com/store/apps/details?id=facebook.video.downloader.savefrom.fb

[118] AccuWeather en Google Play Store
https://play.google.com/store/apps/details?id=com.accuweather.android

[119] Windfinder en Google Play Store
https://play.google.com/store/apps/details?id=com.studioeleven.windfinder

[120] Deer Hunter Classic en Google Play Store
https://play.google.com/store/apps/details?id=com.glu.deerhunt2

[121] Minecraft en Google Play Store
https://play.google.com/store/apps/details?id=com.mojang.minecraftpe

[122] AmongUs en Google Play Store
https://play.google.com/store/apps/details?id=com.innersloth.spacemafia

[123] Fruit Ninja en Google Play Store
https://play.google.com/store/apps/details?id=com.halfbrick.fruitninjafree

[124] Amazon EC2
https://aws.amazon.com/es/ec2/

[125] Crear y configurar una aplicación.
https://support.google.com/googleplay/android-developer/answer/9859152?hl=es#zippy=%2Cfirmar-la-aplicaci%C3%B3n

[126] Binary Code Translator en Google Play Store
https://play.google.com/store/apps/details?id=com.halfbrick.fruitninjafree

[127] Tabla de aplicaciones analizadas.
https://github.com/Caxious77/AnPerMis/blob/main/Tablas/Tabla%20de%20aplicaciones%20analizadas.xlsx

[128] Tabla de frecuencias
https://github.com/Caxious77/AnPerMis/blob/main/Tablas/Tabla%20de%20frecuencias.xlsx

[129] Criterios de aparición de permisos
https://github.com/Caxious77/AnPerMis/blob/main/Tablas/Criterio%20de%20aparici%C3%B3n.xlsx

[130] Repositorio del proyecto
https://github.com/Caxious77/AnPerMis