Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
Mención Tecnologías de la Información

ApkAudit: App para auditar una app Android utilizando el repositorio App-PIMD

Autor:
Javier García González

Tutores:
M. Mercedes Martínez González
Alejandro Pérez de la Fuente

Agradecimientos

Este trabajo está dedicado a todas las personas que me han ayudado durante el desarrollo de este proyecto y en mi vida.

En especial, me gustaría agradecer enormemente a mi familia, sin ellos no hubiera logrado conseguir la mayoría de mis metas y objetivos en la vida.

También quiero agradecer a mis amigos por su continuo apoyo y la ayuda que me han brindado en todo momento en el desarrollo del proyecto.

Además, también quiero agradecer a los profesores que he tenido durante mi formación académica, su orientación y experiencia ha aportado a mi conocimiento.

Para concluir, quiero compartir una frase que me dijo un profesor muy especial y me ha inspirado en mi vida:

"Nunca dejes de aprender, porque la vida nunca deja de enseñarte."

Resumen

Actualmente, vivimos en un mundo cada vez más digitalizado en el que gran cantidad de información se encuentra en móviles, ordenadores entre otros dispositivos tecnológicos. Esto puede acarrear un gran riesgo para las personas que no son conscientes de las aplicaciones que instalan en sus dispositivos debido a que pueden poner en peligro su privacidad exponiendo información sensible.

En este proyecto se presenta una aplicación con la que sus usuarios pueden analizar y auditar tanto las aplicaciones que se encuentran instaladas en sus móviles como archivos APK. La aplicación facilita a sus usuarios las comprobaciones de la calidad del tratamiento de datos personales en las apps cuyo código faciliten como entrada. De este modo, se indica al usuario los permisos que solicita la aplicación, el correcto cumplimiento de las políticas de privacidad y la seguridad de la aplicación. Para ello, se utilizarán datos procedentes de las declaraciones asociadas a las apps y del análisis de código para este fin. Se ha diseñado para cualquier perfil de usuario, con conocimientos básicos de tecnología.

Abstract

We currently live in an increasingly digitalised world in which a large amount of information is stored on mobile phones, computers and other technological devices. This can pose a significant risk to people who are unaware of the applications they install on their devices, as they may compromise their privacy by exposing sensitive information.

This project presents an application that allows users to analyse and audit both the applications installed on their mobile phones and APK files. The application makes it easy for users to check the quality of personal data processing in the apps whose code they provide as input. In this way, the user is informed of the permissions requested by the application, the correct compliance with privacy policies and the security of the application. To do this, data from the statements associated with the apps and code analysis will be used for this purpose. It has been designed for any user profile with basic technology knowledge.

Índice general

Agradecimientos **3**

Resumen **5**

Abstract **7**

1. Introducción **21**

1.1. Contexto 21

1.2. Motivación 21

1.3. Objetivos 22

1.4. Organización del documento 23

2. Planificación **25**

2.1. Características del proyecto 25

2.2. Metodología empleada 25

2.3. Planificación inicial 27

2.3.1. Plan de actividades 27

2.3.2. Hitos 27

2.3.3. Gestión del tiempo 29

2.4. Plan de riesgos 29

2.5. Riesgos de seguridad 32

2.6. Riesgos de privacidad 33

2.7. Presupuesto 34

3. Fundamentos **35**

3.1. Aplicaciones Android 35

3.2. Permisos en Android 35

3.2.1. Tipos de permisos 35

3.2.2. Grupos de permisos 36

3.3. Trackers . . . 36
3.3.1. Finalidad de los trackers . . . 36
3.3.2. Identificar trackers . . . 37
3.4. Políticas de privacidad . . . 37
3.5. Natural Language Processing . . . 37
3.6. Protocolos y estándares . . . 39
3.6.1. Protocolo HTTPS . . . 39
3.6.2. Protocolo TCP . . . 40
3.6.3. Protocolo TLS . . . 41
3.6.4. Estándar RestAPI . . . 41
3.6.5. Estándar OpenAPI . . . 42

4. Análisis **43**
4.1. Requisitos . . . 43
4.1.1. Requisitos funcionales . . . 43
4.1.2. Requisitos no funcionales . . . 44
4.2. Modelo de dominio . . . 46
4.3. Casos de uso . . . 48
4.3.1. CU01: Analizar aplicación instalada . . . 48
4.3.2. CU02: Analizar un archivo *apk* . . . 51
4.3.3. CU03: Acceder análisis ya realizado . . . 53
4.4. Análisis de las fuentes de datos . . . 54
4.4.1. Repositorio *App-PIMD* . . . 55
4.4.2. *VirusTotal* . . . 55
4.4.3. Servidor de procesamiento . . . 55
4.4.4. Recursos locales de trackers . . . 56

5. Diseño **57**
5.1. Componentes del sistema . . . 57
5.2. Arquitectura de la aplicación . . . 58
5.2.1. Arquitectura MVVM . . . 58

5.2.2. Diseño de acceso a datos . . . 59
5.2.3. Diseño de la base de datos . . . 60
5.2.4. Desarrollo nativo . . . 62
5.2.5. Permisos de la aplicación . . . 63
5.2.6. Diseño arquitectónico de la aplicación . . . 63
5.3. Arquitectura del servidor . . . 67
5.3.1. Arquitectura basada en microservicios . . . 67
5.3.2. Separación de intereses . . . 68
5.3.3. Data Transfer Object (DTO) . . . 69
5.3.4. Arquitectura . . . 69
5.4. Flujo de trabajo del servidor . . . 70
5.5. Diseño detallado de la API del servidor . . . 72
5.6. Tipos de datos básicos del *Data Safety* . . . 75
5.7. Diagramas de secuencia en diseño . . . 76
5.7.1. CU01: Analizar aplicación instalada . . . 76
5.7.2. CU02: Analizar un archivo *apk* . . . 78
5.7.3. CU03: Acceder análisis ya realizado . . . 79
5.8. Diagrama de despliegue . . . 79
5.9. *Mock-ups* de la interfaz de usuario . . . 80
5.9.1. Pantalla principal . . . 81
5.9.2. Pantalla de historial de escaneos . . . 82
5.9.3. Pantalla de resultado de un escaneo . . . 83

6. Implementación **85**
6.1. Entorno de desarrollo . . . 85
6.2. Entorno tecnológico . . . 86
6.3. Documentación . . . 87
6.4. Principales dificultades y retos . . . 88
6.5. Privacidad del usuario . . . 89
6.5.1. RP1: Almacenamiento de datos personales . . . 89
6.5.2. RP2: Uso de datos personales . . . 90

6.5.3. RP3: Transmisión cifrada de la información 90
6.5.4. RP4: Procesamiento de datos no consentido 90

7. Pruebas **91**

7.1. Pruebas de aceptación de usuario 91
7.2. Prueba de los endpoints 100
7.3. Prueba de usabilidad 101
7.3.1. Tareas de la prueba de usabilidad 101
7.3.2. Documento de recopilación de respuestas 102
7.3.3. Resultados 105

8. Seguimiento del proyecto **113**

8.1. Desarrollo de los incrementos 113
8.1.1. Incremento 1: Análisis y planificación inicial 113
8.1.2. Incremento 2: Desarrollo de la funcionalidad básica 113
8.1.3. Incremento 3: Integración con el servidor de procesamiento 113
8.1.4. Incremento 4: Implementación avanzada del servidor de procesamiento . . 114
8.1.5. Incremento 5: Refactorización y documentación final 114
8.1.6. Incremento 6: Pruebas de usabilidad 114
8.2. Evaluación general del desarrollo del proyecto 114
8.3. Riesgos materializados 115
8.3.1. R1: Estimaciones optimistas en la planificación 115
8.3.2. R8: Problemas de rendimiento 115
8.3.3. RP3: Transmisión no cifrada de la información 115
8.3.4. R10: Falta de tiempo para realizar las pruebas de usabilidad 116

9. Conclusiones y líneas de trabajo futuro **117**

9.1. Conclusiones 117
9.2. Líneas de trabajo futuro 118
9.2.1. Funcionalidades a añadir en la aplicación 118
9.2.2. Funcionalidades a añadir en el servidor 118

A. Manual de usuario **125**

A.1. Analizar un fichero APK 126

A.2. Aplicaciones instaladas 127

A.3. Reporte de una aplicación 128

A.3.1. Permisos 130

A.3.2. Trackers 131

A.3.3. VirusTotal 131

A.3.4. Política de privacidad 135

A.4. Historial 139

B. Manual de instalación y despliegue **143**

B.1. Aplicación *ApkAudit* 143

B.2. Servidor de procesamiento 144

C. Ejemplo de JSON devuelto en el endpoint de `/get/compare-privacy` **145**

D. Enlaces adicionales **147**

Índice de figuras

2.1. Diagrama de Gantt con las actividades del proyecto y su estimación del tiempo. . 29
2.2. Matriz de probabilidad e impacto de los riesgos 30

3.1. Embedding que convierte un vector de tamaño 10000 a un vector denso de 300 . . 38
3.2. Petición HTTP y HTTPS[1] 39
3.3. Establecimiento de una conexión TCP [2] 40
3.4. Establecimiento de una conexión segura TLS/SSL sobre TCP [3] 41
3.5. Modelo Rest API[4] 42

4.1. Diagrama de clases del modelo de dominio 46
4.2. Diagrama de casos de uso 48
4.3. Diagrama de secuencia en análisis del caso de uso CU01 51
4.4. Diagrama de secuencia en análisis del caso de uso CU02 53
4.5. Diagrama de secuencia en análisis del caso de uso CU03 54

5.1. Diagrama de componentes del sistema 57
5.2. Patrón arquitectónico MVVM[5] 58
5.3. Patrón Repositorio[6] 59
5.4. Data Access Object (DAO)[7] 59
5.5. Tablas de la base de datos 60
5.6. Diagrama de paquetes de la aplicación 65
5.7. Diagrama de clases del paquete *core* 66
5.8. Diagrama de clases del paquete *data* 66
5.9. Diagrama de clases del paquete *features* 67
5.10. Patrón arquitectónico API Gateway[8] 68
5.11. Patrón de diseño DTO 69
5.12. Diagrama de arquitectura del servicio del servidor de procesamiento 70

5.13. Diagrama de flujo de proceso del servidor para la obtención del análisis de la política de privacidad de una aplicación 71
5.14. Diagrama de secuencia a nivel de diseño del caso de uso *CU01* 76
5.15. Diagrama de secuencia a nivel de diseño del caso de uso *CU02* 78
5.16. Diagrama de secuencia a nivel de diseño del caso de uso *CU03* 79
5.17. Diagrama de despliegue 80
5.18. Pantalla principal de la aplicación 81
5.19. Pantalla de historial de escaneos 82
5.20. Pantalla de resultado de un escaneo 83

6.1. Documentación generada por Dokka 88

7.1. Distribución de los usuarios según su edad 105
7.2. Distribución de los usuarios según sus conocimientos previos en el sistema de Android 105
7.3. Leyenda para el diagrama de dispersión 106
7.4. Diagrama de dispersión para la tarea 1 106
7.5. Diagrama de dispersión para la tarea 2 106
7.6. Diagrama de dispersión para la tarea 2.1 107
7.7. Diagrama de dispersión para la tarea 3 107
7.8. Diagrama de dispersión para la tarea 4 108
7.9. Diagrama de dispersión para la tarea 5 108
7.10. Diagrama de dispersión para la tarea 6 108
7.11. Diagrama de dispersión para la puntuación de facilidad de uso 109
7.12. Diagrama de dispersión para la puntuación de diseño visual 109
7.13. Diagrama de dispersión para la puntuación de comprensibilidad del informe 110
7.14. Diagrama de dispersión para la puntuación de volvería a usar la aplicación 110

A.1. Pantalla de inicio de la aplicación 125
A.2. Pantalla de inicio de la aplicación en modo claro 126
A.3. Pantalla de selección de fichero APK 127
A.4. Pantalla de aplicaciones instaladas 128
A.5. Pantalla de reporte de una aplicación 129

A.6. Pestaña de permisos 130
A.7. Pestaña de trackers 131
A.8. Pestaña de *VirusTotal* 132
A.9. Pantalla de informe completo de *VirusTotal* 133
A.10.Pantalla de detalles de *VirusTotal* 134
A.11.Comparativa del análisis de políticas de privacidad de dos aplicaciones. 136
A.12.Resultado completo del análisis de la política de privacidad de *WhatsApp*. 137
A.13.Pantalla de informe completo de los datos recogidos para *WhatsApp*. 138
A.14.Pestaña de prácticas de seguridad de *WhatsApp*. 139
A.15.Pantalla de historial. 140
A.16.Borrado del reporte de *Strava*. 141

Índice de cuadros

2.1. Planificación de hitos del proyecto 28
2.2. Riesgos generales del proyecto 30
2.3. Calificación de los riesgos. 31
2.4. Plan de mitigación de los riesgos 31
2.5. Riesgos de seguridad del proyecto 32
2.6. Calificación de los riesgos de seguridad. 32
2.7. Plan de mitigación de los riesgos de seguridad. 33
2.8. Riesgos de privacidad del proyecto. 33
2.9. Calificación de los riesgos de privacidad. 34
2.10. Plan de mitigación de los riesgos de privacidad. 34

5.1. Atributos de los elementos de la tabla *scan_results*. 61
5.2. Atributos de los elementos de la tabla *security_tips*. 61
5.3. Atributos de los elementos de la tabla *data_safety_info*. 62
5.4. Atributos de los elementos de la tabla *permission_info*. 62
5.5. Cabecera obligatoria para autenticar peticiones 72
5.6. Parámetros del endpoint `/get/compare-privacy` 73
5.7. Respuestas del endpoint `/get/compare-privacy` 73
5.8. Parámetros del endpoint `/get/security-tips` 73
5.9. Respuestas del endpoint `/get/security-tips` 74
5.10. Parámetros del endpoint `/permissions.json` 74
5.11. Respuestas del endpoint `/permissions.json` 74
5.12. Parámetros del endpoint `/data-safety.json` 74
5.13. Respuestas del endpoint `data-safety-info.json` 75
5.14. Algunos de los tipos de datos básicos del *Data Safety* de *Google Play*. 75

7.1. Prueba UAT 01 91
7.2. Prueba UAT 02 92

7.3. Prueba UAT 03 92
7.4. Prueba UAT 04 92
7.5. Prueba UAT 05 93
7.6. Prueba UAT 06 93
7.7. Prueba UAT 07 93
7.8. Prueba UAT 08 94
7.9. Prueba UAT 09 94
7.10. Prueba UAT 10 95
7.11. Prueba UAT 11 95
7.12. Prueba UAT 12 95
7.13. Prueba UAT 13 96
7.14. Prueba UAT 14 96
7.15. Prueba UAT 15 96
7.16. Prueba UAT 16 97
7.17. Prueba UAT 17 97
7.18. Prueba UAT 18 97
7.19. Prueba UAT 19 98
7.20. Prueba UAT 20 98
7.21. Prueba UAT 21 98
7.22. Prueba UAT 22 99
7.23. Prueba UAT 23 99
7.24. Prueba UAT 24 99
7.25. Pruebas realizadas al endpoint `/get/compare-privacy` 100
7.26. Pruebas realizadas al endpoint `/get/security-tips` 100
7.27. Pruebas realizadas al endpoint `/data-safety.json` 100
7.28. Pruebas realizadas al endpoint `/permissions.info` 101

Capítulo 1

Introducción

1.1. Contexto

En la actualidad, los teléfonos móviles han dejado de ser simples herramientas de comunicación para convertirse en el eje central de nuestra vida personal y profesional. Desde nuestros dispositivos móviles se llevan a cabo numerosas acciones que definen nuestro día a día, como trámites en bancos, compras en línea, redes sociales, uso de aplicaciones de salud y el bienestar.

Debido a esta creciente dependencia, los dispositivos móviles almacenan gran cantidad de información relevante para numerosos individuos y/o entidades. Por tanto, en un mundo donde los datos son uno de los activos más valiosos, la falta de control sobre su uso puede generar riesgos que afectan directamente a la privacidad y seguridad de las personas.

La gestión de datos es un aspecto fundamental en la vida de cualquier persona. Sin embargo, muchos desconocen las medidas y preocupaciones que deben tener en cuenta a la hora de interactuar y transmitir datos en línea. En este contexto, es de gran utilidad saber cómo gestionar quien accede a nuestros datos y cómo proteger la información que almacena.

Por tanto, en este proyecto se ofrece una solución que permite a los usuarios informarse sobre los datos que se recopilan en sus dispositivos móviles. De modo que puedan tomar las precauciones y acciones pertinentes para proteger su información personal y garantizar su privacidad.

1.2. Motivación

En una sociedad en la que gran mayoría de los servicios están informatizados, mucha personas no son plenamente conscientes de la privacidad en el uso de los dispositivos móviles y cometen el error de no proteger correctamente su información personal. En ocasiones, desconocen la seguridad de las aplicaciones que forman parte de su móvil y no son conscientes de los riesgos que suponen para sus datos e intimidad. Ante este habitual desconocimiento, es posible que determinadas aplicaciones accedan a información personal de forma poco transparente, incumpliendo la política de privacidad indicada, solicitando permisos que no son necesarios en el contexto de su uso, implementando trackers[1] y otros servicios que puedan recopilar información personal de los usuarios.

[1]Software que rastrea y monitorea el uso de los dispositivos móviles

Para obtener una solución a este problema existen algunos servicios que ayudan a capacitar y desvelar a los usuarios sobre las aplicaciones que instalan o de las que hacen uso, algunas de ellas son:

- *Exodus Privacy* es una aplicación gratuita que permite a los usuarios analizar aplicaciones aplicaciones Android en busca de información sobre los permisos solicitados y trackers [2].
- *AppCensus* es un servicio de pago con el cual se analiza aplicaciones móviles y proporciona visibilidad sobre el uso de datos y la privacidad de las aplicaciones [3].
- *Tracker Control* es una aplicación móvil gratuita que muestra a los usuarios los trackers presentes en sus aplicaciones a través de realizar un análisis estático y analizar el tráfico de red de las aplicaciones [1].

No obstante, estas soluciones en ocasiones pueden resultar complejas de usar para usuarios con conocimientos poco técnicos o no ofrecen toda la información que pueda interesar a los usuarios acerca de cómo las aplicaciones que usan tratan su privacidad.

Por tanto, ante esta situación junto con el Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid[4] se propone el desarrollo de una aplicación móvil, denominada *ApkAudit*, que permita a los usuarios analizar y auditar aplicaciones móviles. Esta aplicación tendrá como objetivo principal el concienciar y motivar a los usuarios a preocuparse por su privacidad y seguridad. Esto se realizará mediante el estudio de los diferentes datos que se recopilan del usuario en las aplicaciones que usa, qué uso se hace de estos datos y cómo se utilizan para fines comerciales o publicitarios. Se pretende mostrar de una manera amigable y comprensible la información obtenida por las aplicaciones que usa debido a que, actualmente las políticas de privacidad, en las que se declara esta información, pueden resultar extensas y difíciles de comprender para el usuario medio [2].

1.3. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación, *ApkAudit*, que permita a los usuarios verificar las políticas de privacidad de las aplicaciones que se encuentran en sus teléfonos móviles. Para ello, se emplea, entre otras fuentes de datos, el repositorio *App-PIMD*[5].

Los objetivos específicos son los siguientes:

- Facilitar el análisis de aplicaciones móviles con el fin de identificar riesgos relacionados con los permisos, rastreadores y vulnerabilidades que puedan tener.

[2]Exodus Privacy: `https://exodus-privacy.eu.org/en/`
[3]AppCensus: `https://appcensus.io/`
[4]`https://ingpriv.uva.es/`
[5]Repositorio que proporciona un almacenamiento centralizado de metadatos de aplicaciones, accesible a través de una API pública. `https://ingpriv.uva.es/repositorio-app-pimd/`

- Ofrecer una herramienta que ayude a promover la transparencia en las políticas de privacidad de las aplicaciones móviles.
- Asegurar la usabilidad de la aplicación para cualquier usuario mediante una interfaz intuitiva y accesible.
- Fomentar la concienciación del usuario sobre los riesgos asociados a las aplicaciones instaladas.
- Proporcionar a los usuarios mecanismos comprensibles para informar sobre los hallazgos obtenidos en el análisis de una aplicación.

1.4. Organización del documento

Este documento se organiza en varios capítulos en los que se puede encontrar el siguiente contenido:

- **Capítulo 1. Introducción.** Contextualización e introducción al proyecto.
- **Capítulo 2. Planificación.** Se realiza una planificación inicial, se detallan los diversos hitos, los riesgos que pueden materializarse a lo largo del proyecto y el presupuesto.
- **Capítulo 3. Fundamentos.** Se explican los fundamentos teóricos del sistema Android, así como conceptos relacionados con la seguridad de los datos.
- **Capítulo 4. Análisis.** Se analiza el problema a resolver, se definen los requisitos funcionales y no funcionales de la aplicación y se realiza un estudio de los casos de uso principales.
- **Capítulo 5. Diseño.** Se presenta el diseño de la arquitectura del sistema, el diseño de la UI[6] y el diseño de los diferentes componentes del sistema.
- **Capítulo 6. Implementación.** Se detalla los medios tecnológicos utilizados, el proceso de desarrollo de la aplicación y las diferentes decisiones de diseño tomadas.
- **Capítulo 7. Pruebas.** Se presentan las pruebas de software y las pruebas de usabilidad que se han realizado sobre la aplicación.
- **Capítulo 8. Seguimiento del proyecto.** La evolución real del proyecto a lo largo del tiempo en comparación con la planificación inicial y la materialización de los riesgos contemplados en el capítulo 2.
- **Capítulo 9. Conclusiones.** Reflexión final sobre el trabajo realizado y posibles líneas de trabajo futuras.

[6]Interfaz de usuario

Capítulo 2

Planificación

En este capítulo se detalla la planificación del proyecto siguiendo la guía *PMBOK* [3]. En primer lugar se detallan las características del proyecto, se establece una planificación inicial y se detallan los hitos del proyecto. Por último, se incluye un plan de riesgos que se consideran importantes para el proyecto y el presupuesto del proyecto.

2.1. Características del proyecto

El proyecto cuenta con los siguientes componentes: la aplicación móvil, el repositorio de aplicaciones (App-PIMD) [4] y un servidor de procesamiento.

La aplicación móvil se utiliza para que el usuario final pueda encargar el escaneo y/o auditar otra aplicación Android, ya sea en formato APK o aquellas que se encuentren instaladas en el dispositivo del usuario. Esta aplicación se encargará de comunicarse con el servidor de procesamiento y con el repositorio de aplicaciones para, posteriormente, mostrar el resultado de estas operaciones al usuario.

El repositorio de aplicaciones (App-PIMD) [5][6], es un proyecto realizado por el Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid. El repositorio App-PIMD permite estudiar patrones de recopilación de información, evaluar riesgos de privacidad y desarrollar estrategias para mitigar intrusiones, proporcionando acceso estructurado y abierto a metadatos clave. [4]

El servidor de procesamiento se encarga de recibir las aplicaciones representadas por su nombre de paquete y realizar un estudio sobre el correcto cumplimiento de la política de privacidad de dicha aplicación. Además, proporciona información sobre los permisos de Android, consejos de seguridad y descripciones para los datos que se puedan recopilar o compartir del usuario.

2.2. Metodología empleada

Para el desarrollo del proyecto se ha optado por una **metodología incremental**. Mediante la metodología incremental, el sistema se construye y mejora a través de una serie de incrementos o mejoras sucesivas. En cada incremento se añaden nuevas funcionalidades o mejoras a las versiones

anteriores, permitiendo corregir errores pasados y tener una mayor adaptabilidad a cambios de requisitos.

En consecuencia, al emplear una metodología incremental todas las partes del proyecto pueden cambiar durante cualquier momento del desarrollo del mismo. Esto facilita adaptar el proyecto a las necesidades cambiantes y permitir una mayor flexibilidad en el desarrollo. Además, este enfoque resulta especialmente útil para el proyecto, ya que permite desarrollar de forma independiente cada componente del proyecto. A lo largo del proyecto se han establecido una serie de hitos que coinciden con entregas parciales o incrementos:

1. **Incremento 1:** Análisis y planificación inicial.
   Objetivo: Establecer las bases conceptuales y de planificación del proyecto.
   Fecha de finalización: 10/03/2025

   - Listado de requisitos funcionales y no funcionales.
   - Estudio de las fuentes de datos.
   - Diseño preliminar de la aplicación móvil.
   - Planificación temporal del proyecto.

2. **Incremento 2:** Desarrollo de la funcionalidad básica.
   Objetivo: Implementar el análisis inicial de aplicaciones.
   Fecha de finalización: 14/04/2025

   - Funcionalidad de análisis de aplicaciones en formato APK.
   - Funcionalidad de análisis de aplicaciones instaladas.

3. **Incremento 3:** Integración con el servidor de procesamiento.
   Objetivo: Establecer la comunicación entre la aplicación *ApkAudit* y el servidor.
   Fecha de finalización: 07/05/2025

   - Desarrollo de la funcionalidad principal del servidor de procesamiento.
   - Integración del servidor con la aplicación móvil.

4. **Incremento 4:** Implementación avanzada del servidor de procesamiento.
   Objetivo: Ampliar la funcionalidad del servidor.
   Fecha de finalización: 21/05/2025

   - Reorganización y refactorización del código.
   - Creación de endpoints para ofrecer documentación de Android.
   - Tratamiento de errores en los endpoints.

5. **Incremento 5:** Refactorización y documentación final.
   Objetivo: Consolidar la aplicación y dejarla lista para pruebas.
   Fecha de finalización: 27/05/2025

- Refactorización del código de la aplicación móvil.
- Implementación del tratamiento de errores.
- Documentación técnica y de usuario.

6. **Incremento 6:** Pruebas de usabilidad.
   Objetivo: Validar la aplicación con usuarios reales.
   Fecha de finalización: 05/06/2025

   - Realización de las pruebas de usabilidad.
   - Análisis de resultados y retroalimentación.
   - Corrección de errores detectados.

Cada incremento se apoya en los anteriores y produce un sistema funcional y más complejo.

2.3. Planificación inicial

El proyecto se llevará a cabo desde el 13 de febrero de 2025 hasta el 12 de junio de 2025, es decir, 88 días (17 semanas) de diferencia. El trabajo tiene una duración de 300 horas que, teniendo en cuenta los días de trabajo disponibles se realiza una media de 3 horas y media por día.

2.3.1. Plan de actividades

En este apartado, se hace un desglose de las actividades que se llevarán a cabo durante el desarrollo del proyecto. Se utiliza el programa gratuito y de código abierto *Gantt Project*[7] en su versión 3.3 para Windows para la realización del diagrama Gantt en el que se muestra el reparto de actividades y su respectiva estimación del tiempo. Además, en la tabla 2.1 se presentan los distintos hitos del proyecto que servirán como puntos de control.

2.3.2. Hitos

Para asegurar un desarrollo ordenado y progresivo del proyecto *ApkAudit*, se han establecido los siguientes hitos con fechas específicas:

Hito	Semana	Fecha
Lista de requisitos iniciales	2	17/02/2025
Diseño de la aplicación móvil	5	10/03/2025
Implementación del servicio básico e integración con App-PIMD	8	14/04/2025
Integración con servidor de procesamiento	11	07/05/2025
Tests de usabilidad realizados	14	27/05/2025
Fin del proyecto	17	12/06/2025

Tabla 2.1: Planificación de hitos del proyecto

Estos hitos servirán como puntos de control para evaluar el progreso del proyecto y asegurar que se cumplen los objetivos establecidos dentro del plazo previsto. Cada hito representa un logro significativo en el desarrollo del sistema ApkAudit y permite verificar que el proyecto avanza según lo planificado. A continuación, se explica detalladamente cada hito:

1. **Lista de requisitos iniciales:** Se recogen los requisitos funcionales y no funcionales para el proyecto.
2. **Diseño de la aplicación móvil:** Se realizan los wireframes y mockups de la interfaz de usuario de la aplicación móvil.
3. **Implementación del servicio básico e integración con App-PIMD [4]:** Se han implementado las funcionalidades básicas de ApkAudit, esto incluye el escaneo de aplicaciones en formato APK y la integración con el repositorio de aplicaciones *App-PIMD* [4].
4. **Integración con servidor de procesamiento:** Se integra en los resultados del escaneo el resultado de realizar el estudio del correcto cumplimiento de la política de privacidad de la aplicación.
5. **Implementación de funcionalidad completa:** Se finaliza el desarrollo de las funcionalidades de *ApkAudit*.
6. **Tests de usabilidad realizados:** Se han realizado distintos tests de usabilidad para evaluar la usabilidad de *ApkAudit*
7. **Fin del proyecto:** Se ha completado el proyecto.

2.3.3. Gestión del tiempo

Como se ha mencionado en el apartado 2.3.1, se utiliza el software gratuito y de código abierto *Gantt Project*[7] para realizar el diagrama de Gantt en el que se evalúa y se presenta las distintas actividades que se llevarán a cabo durante el proyecto, junto con la estimación del tiempo total que tendrán dichas actividades.

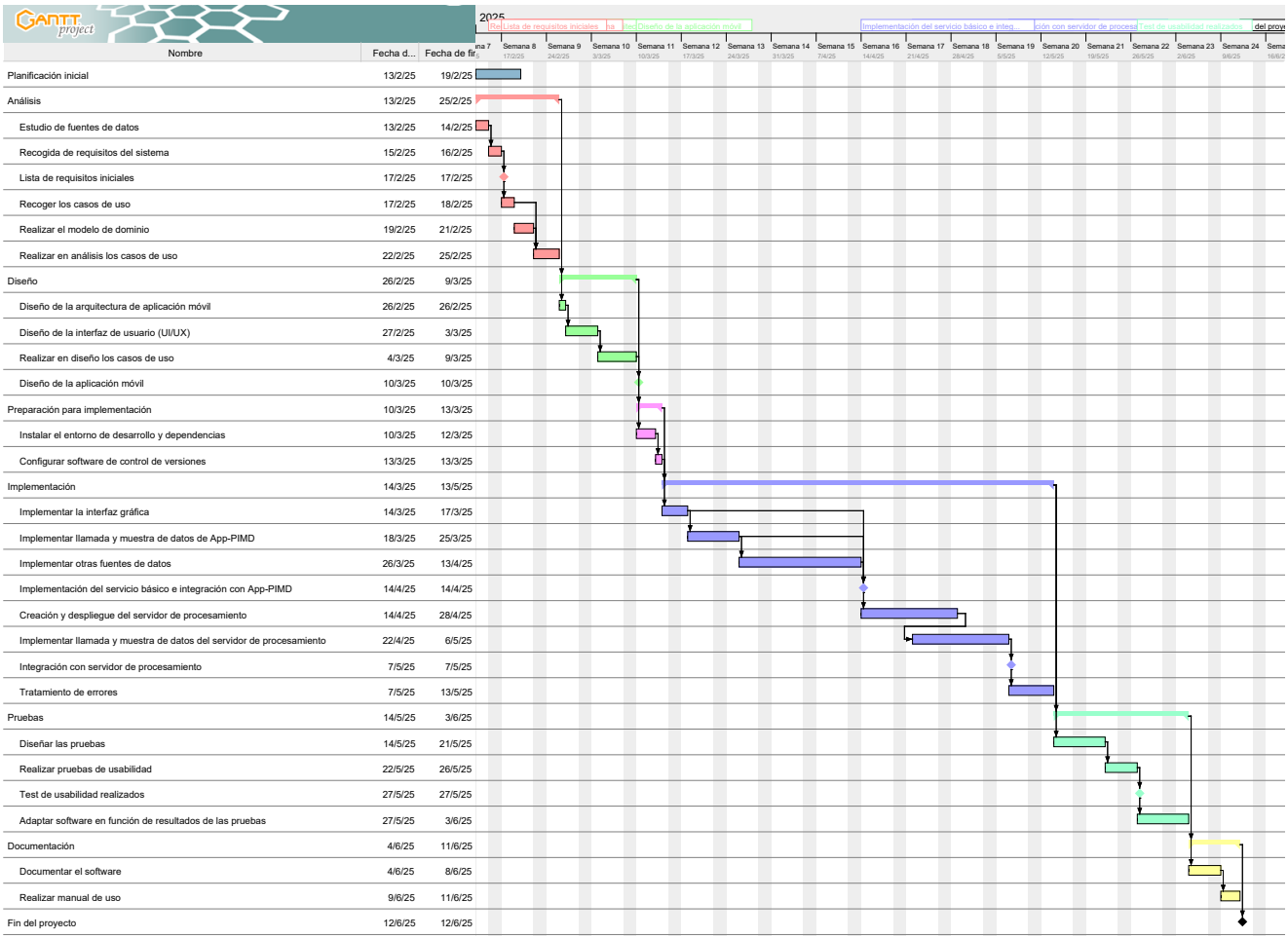


Figura 2.1: Diagrama de Gantt con las actividades del proyecto y su estimación del tiempo.

2.4. Plan de riesgos

En este apartado se realiza una identificación de los posibles riesgos que pueden afectar a este proyecto y materializarse durante la ejecución del mismo. Además, se realiza un análisis de dichos riesgos acompañado de una estimación de la probabilidad de que ocurran y el plan de mitigación para cada unos de ellos.

Para establecer la probabilidad de ocurrencia y el impacto que supondrían los riesgos, se ha utilizado la siguiente matriz de riesgos[1]:

[1]Una matriz de riesgos es una herramienta que permite cuantifica la probabilidad de ocurrencia y el impacto de los posibles riesgos en un proyecto.

Figura 2.2: Matriz de probabilidad e impacto de los riesgos

La tabla 2.2 se puede encontrar en el capítulo 11 de *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)* [3]

Por tanto, en la tabla 2.2 se muestran los riesgos generales identificados en el proyecto junto con su descripción.

ID Riesgo	Descripción
R1	Estimaciones optimistas en la planificación
R2	Modificación de los requisitos
R3	Retrasos causados por el uso de tecnologías desconocidas
R4	Fallos en la comunicación entre la aplicación y el servidor externo
R5	Encontrar un fallo de diseño tardío
R6	Menor disponibilidad de la necesaria
R7	Problemas de integración con el repositorio de metadatos de aplicaciones
R8	Problemas de rendimiento
R9	Problemas de integración con el servidor de procesamiento
R10	Falta de tiempo para realizar las pruebas de usabilidad

Tabla 2.2: Riesgos generales del proyecto

En la tabla 2.3, se muestra para cada uno de los riesgos identificados en la tabla 2.2, la probabilidad de ocurrencia y el impacto que supondría en el proyecto.

ID Riesgo	Probabilidad	Impacto	Calificación
R1	0,70 (Alta)	0,40 (Alto)	0,28 (Alto)
R2	0,90 (Muy alta)	0,20 (Moderado)	0,18 (Moderado)
R3	0,10 (Muy baja)	0,20 (Moderado)	0,02 (Moderado)
R4	0,70 (Alta)	0,20 (Moderado)	0,10 (Moderado)
R5	0,30 (Baja)	0,80 (Muy alto)	0,24 (Alto)
R6	0,70 (Alta)	0,20 (Moderado)	0,14 (Moderado)
R7	0,30 (Baja)	0,40 (Alto)	0,12 (Moderado)
R8	0,50 (Mediana)	0,20 (Moderado)	0,10 (Moderado)
R9	0,70 (Alta)	0,40 (Alto)	0,28 (Alto)
R10	0,90 (Muy alta)	0,20 (Moderado)	0,18 (Moderado)

Tabla 2.3: Calificación de los riesgos.

Finalmente, para cada uno de los riesgos generales identificados se muestra su plan de mitigación, es decir, las acciones que han de llevarse a cabo en el caso de que estos riesgos lleguen a materializarse.

ID Riesgo	Plan de mitigación
R1	Realizar revisiones contínuas de la planificación con los tutores
R2	Analizar los nuevos requisitos y comprobar en que grado modifican el proyecto y, si fuera necesario, modificar la planificación
R3	Estudiar exhaustivamente la documentación de las tecnologías a emplear.
R4	Consultar los errores correspondientes, estudiar posibles soluciones y, en caso de ser necesario, consultar a los tutores.
R5	Realizar revisiones frecuentes en las fases tempranas del proyecto involucrando a los tutores para asegurar que se cumple el diseño correcto.
R6	Proporcionar y estudiar los márgenes y flexibilidad en la planificación del proyecto.
R7	Consultar con el desarrollador del repositorio de datos *App-PIMD*
R8	Utilizar estructuras de datos y patrones de diseño para evitar problemas de rendimiento.
R9	Identificar los errores y adaptar las llamadas desde la aplicación en caso de ser necesario.
R10	Realizar revisiones frecuentes y actualizaciones de la planificación inicial

Tabla 2.4: Plan de mitigación de los riesgos

2.5. Riesgos de seguridad

En este apartado se indican los riesgos de seguridad que pueden materializarse y afectar a la seguridad del sistema de *ApkAudit*. Se tienen en cuenta estos riesgos de seguridad debido a que se trata de un proyecto en el que se interactúa y recibe entradas por parte de terceros, es decir, del usuario final quienes pueden llevar a cabo ataques que atenten contra la seguridad del mismo.

En la tabla 2.5 se muestran los riesgos de seguridad identificados en el proyecto junto con su descripción.

ID Riesgo	**Descripción**
RS1	Dependencias de terceros vulnerables: es posible que las dependencias y/o bibliotecas externas utilizadas en el código del proyecto presenten vulnerabilidades o no estén actualizadas
RS2	Entrada de datos maliciosos: el usuario puede introducir APKs maliciosas o aplicaciones falsas para vulnerar la seguridad de los servidores y/o el repositorio de datos
RS3	Ataque de denegación de servicio: un usuario puede realizar de forma intencionada multitud de peticiones que afecten al rendimiento de los servidores y, en consecuencia, denegar el acceso al servicio
RS4	Fuga de claves API: debido al uso de claves API es posible que estas se vean reveladas

Tabla 2.5: Riesgos de seguridad del proyecto

A continuación, en la tabla 2.6, se muestra la probabilidad e impacto que tendrían en el proyecto los distintos riesgos de seguridad que se han identificado en la tabla 2.5.

ID Riesgo	**Probabilidad**	**Impacto**	**Calificación**
RS1	0,30 (Baja)	0,40 (Alto)	0,12 (Moderado)
RS2	0,30 (Baja)	0,80 (Muy alto)	0,24 (Alto)
RS3	0,50 (Mediana)	0,40 (Alto)	0,20 (Moderado)
RS4	0,10 (Muy baja)	0,80 (Muy alto)	0,08 (Moderado)

Tabla 2.6: Calificación de los riesgos de seguridad.

En la siguiente tabla 2.7, se presenta el plan de mitigación de los riesgos de seguridad identificados en la tabla 2.5.

ID Riesgo	Plan de mitigación
RS1	Realizar actualizaciones frecuentes de la aplicación para parchear las posibles vulnerabilidades.
RS2	Validar los datos de entrada del usuario tanto en frontend como en el backend.
RS3	Limitar las peticiones que puede realizar un mismo usuario sin que llegue a afectar a la funcionalidad de la aplicación.
RS4	Utilizar ofuscación[2] y variables de entorno para evitar que se filtren las claves API. En caso de que se hayan obtenido, realizar un rotado de las claves correspondientes.

Tabla 2.7: Plan de mitigación de los riesgos de seguridad.

2.6. Riesgos de privacidad

La privacidad es un tema que tiene un gran impacto en los usuarios y que, en muchas ocasiones, se pasa por alto. Por tanto, en este apartado se indican los riesgos de privacidad que puede materializarse y afectar a la privacidad de los usuarios que hacen uso de *ApkAudit*.

En la tabla 2.8 se hace un listado de los riesgos de privacidad identificados junto con la correspondiente descripción.

ID Riesgo	Descripción
RP1	Almacenamiento de datos personales sin anonimización: los datos que se extraen de las APK pueden contener información sensible para el usuario.
RP2	Acceso a más recursos o datos del usuario de los necesarios.
RP3	Transmisión no cifrada de la información: la comunicación entre el cliente y el servidor de procesamiento puede no estar cifrada mediante protocolos como TLS
RP4	Procesamiento no consentido de datos, es decir, que se envíen datos a los servidores necesarios para hacer el análisis de las aplicaciones sin antes solicitar el consentimiento del usuario.

Tabla 2.8: Riesgos de privacidad del proyecto.

En la tabla 2.9, se muestra la probabilidad e impacto que tendrían los riesgos de privacidad presentados en la tabla 2.8.

[2]Ofuscación: es el proceso de alterar el código de la app para hacerlo más difícil de entender para cualquier persona que intente analizarlo o descompilarlo.

ID Riesgo	Probabilidad	Impacto	Calificación
RP1	0,70 (Alta)	0,80 (Muy alto)	0,56 (Muy alto)
RP2	0,70 (Alta)	0,80 (Muy alto)	0,56 (Muy alto)
RP3	0,30 (Baja)	0,20 (Moderado)	0,06 (Moderado)
RP4	0,70 (Alta)	0,40 (Alto)	0,28 (Alto)

Tabla 2.9: Calificación de los riesgos de privacidad.

A continuación, en la tabla 2.10, se presentan las acciones que se deberían llevar a cabo en caso de que los riesgos de privacidad se llegaran a materializar.

ID Riesgo	Plan de mitigación
RP1	Evitar almacenar información que pueda identificar directa o indirectamente al usuario. Si es necesario aplicar técnicas de anonimización o pseudoanonimización antes de almacenar los datos.
RP2	Aplicar el principio de minimización de datos, es decir, solo se debe solicitar los recursos y datos estrictamente necesarios para desempeñar las funciones. Por ejemplo, que la aplicación solicite solo los permisos que necesita.
RP3	Utilizar siempre protocolos de comunicación seguros como HTTPS para cifrar los datos en tránsito.
RP4	Analizar solo las aplicaciones que haya elegido previamente el usuario.

Tabla 2.10: Plan de mitigación de los riesgos de privacidad.

2.7. Presupuesto

En este apartado se detalla el presupuesto del proyecto exponiendo los distintos gastos que tendrán lugar durante el desarrollo del mismo.

El gasto de personal es de 0 euros debido a que el proyecto se lleva a cabo como trabajo de fin de grado y no se cobrará por el desarrollo del mismo. De todos modos, se tiene en cuenta que el salario de un ingenierio de software junior es de aproximadamente 23500 euros al año, es decir, 12 euros la hora [8]. Teniendo en cuenta que el proyecto tiene una duración de 300 horas el gasto en personal es de 3600 euros.

El gasto en hardware es de 0 euros debido a que los equipos necesarios para el trabajo es el ordenador personal de desarrollador.

El gasto en software es de 0 euros debido a que se utilizan herramientas gratuitas y de código abierto. Además, se utiliza una máquina virtual cuyo servicio lo brinda la Escuela de Ingeniería Informática. Por tanto, el coste también sería de 0 euros aunque, si se tuviese que utilizar un servidor supondría un gasto de 158,50 dólares al mes [9], este precio se ha obtenido a partir de la calculadora del coste de una máquina EC2 de Amazon Web Services.

Capítulo 3

Fundamentos

En este capítulo se presentan los fundamentos teóricos necesarios para contextualizar el proyecto y entender el motivo de su desarrollo.

3.1. Aplicaciones Android

El sistema operativo *Android*, desarrollado por Google, es el sistema operativo móvil más utilizado a nivel mundial, con una cuota de mercado aproximadamente del 74 % [10]. Las aplicaciones que se ejecutan en los sistemas operativos Android son empaquetadas en archivos con extensión *.apk*, es decir, ficheros que contienen, entre otros recursos, el código fuente compilado de la aplicación. A través, de estos ficheros *.apk*, los usuarios pueden instalar aplicaciones en sus dispositivos móviles y agregar, de este modo, nuevas funcionalidades a sus equipos.

Sin embargo, tambien existe la opción de descargar e instalar las aplicaciones desde instaladores de aplicaciones como *Google Play Store* [11], *Amazon Appstore* [12], *HUAWEI AppGallery* [13], entre otros. Estos instaladores facilitan a los usuarios la descarga de aplicaciones y contribuyen a mejorar la seguridad del sistema debido a las políticas de privacidad y seguridad que suelen establecer principalmente las tiendas oficiales de aplicaciones.

3.2. Permisos en Android

Las aplicaciones Android necesitan los permisos para acceder a ciertas funcionalidades o recursos del usuario y/o del dispositivo del usuario, es decir, ayudan a mantener segura la privacidad del usuario. Android categoriza los permisos en varios tipos ([14]):

3.2.1. Tipos de permisos

- Permisos en el momento de instalación: se conceden de forma automática cuando el usuario instala la aplicación.

- Permisos normales: son aquellos que permiten el acceso a datos y acciones que se extienden más allá de la zona de pruebas de la aplicación.

- Permisos de firma: es el permiso que se concede a una app solo cuando está firmada por el

mismo certificado que la app o el SO que define el permiso.

- Permisos en tiempo de ejecución: son aquellos permisos que brindan a la app de un acceso a datos o recursos más restringidos y que afectan significativamente al sistema y al usuario. Requieren que el usuario conceda explícitamente el permiso. Tambien son conocidos como permisos peligrosos.
- Permisos especiales: son permisos que corresponden a funcionalidades u operaciones particulares de la app.

3.2.2. Grupos de permisos

Los permisos pueden pertenecer a grupos de permisos, que se relacionan de forma lógica. Por ejemplo, los permisos para enviar y recibir SMS pueden pertenecer al mismo grupo, ya que ambos se relacionan con la interacción de la aplicación con SMS. Los grupos de permisos se presentan juntos en la interfaz de usuario de la aplicación.

Estos grupos de permisos se declaran en el archivo *AndroidManifest.xml* mediante la etiqueta `<permission-group>`.

3.3. Trackers

En el contexto de las aplicaciones Android, los *trackers* o rastreadores son servicios o componentes encargados de recopilar información sobre el comportamiento del usuario en el uso de una *app*. En torno al 50 % de las aplicaciones Android utilizan algún tipo de tracker [15].

3.3.1. Finalidad de los trackers

Los trackers pueden tener finalidades variadas, algunas de ellas son:

- Venta a terceros, como anunciantes o empresas de publicidad. Se pueden utilizar para recopilar información sobre los usuarios y a partir de esta información, venderla a terceros. Estos terceros normalmente con fines comerciales para enfocar por ejemplo sus campañas publicitarias a grupos específicos de usuarios.
- Monitoreo y análisis de datos de uso. Se pueden utilizar para identificar patrones de uso de las aplicaciones por parte de los usuarios y detectar posibles incidencias como errores, fallos o comportamientos anómalos.
- Mejora de la experiencia del usuario. Los desarrolladores pueden utilizar los datos recabados por los trackers para entender cómo los usuarios interactúan con su producto y, así enfocar las funcionalidades nuevas y existentes hacia la correcta dirección para mejorar la experiencia del usuario.

3.3.2. Identificar trackers

Para identificar los trackers que se encuentran presentes en una aplicación Android, se pueden emplear varios métodos, como:

- **Análisis de estático:** se descompila el fichero *.apk* para identificar las clases y métodos de la misma. En las clases *DEX* se busca las coincidencias con las firmas de los trackers o rastreadores que se desee buscar.
- **Análisis dinámico:** se ejecuta la aplicación en un dispositivo Android y, a través del funcionamiento normal de la aplicación se puede monitorear, por ejemplo, el tráfico de red identificando a los servidores o hosts a los que se envían las peticiones desde la aplicación.

El método empleado en *ApkAudit* es el análisis estático utilizando como base de datos para comprobar las firmas de los tracker más conocidos la proporcionada por *Tracker Control* [1].

3.4. Políticas de privacidad

Las políticas de privacidad son documentos legales que establecen los datos que, en este caso, recopila una aplicación, cómo los utiliza, almacena y protege estos datos de los usuarios. Por tanto, son un elemento fundamental para garantizar la transparencia en el tratamiento de los datos personales por parte de las aplicaciones.

Aunque, las políticas de privacidad pueden resultar difíciles de comprender, interpretar o ser demasiado extensas para gran cantidad de usuarios, que, optan por ignorar esta información aunque sean conscientes de su importancia. [16]

Para tratar de poner solución a esto la tienda oficial de Android, *Google Play Store*, ofrece a los desarrolladores la opción de establecer, en un campo llamado *Data Safety* [17], los datos de forma resumida que recopilan mediante el uso de su aplicación por parte de los usuarios. De este modo, los usuarios pueden consultar rápidamente qué datos recopilan las aplicaciones y cómo los utilizan.

Sin embargo, en muchas ocasiones existen discrepancias entre lo que se declara en la política de privacidad oficial y lo que se informa en la correspondiente sección de *Data Safety* de la tienda oficial. Según un estudio académico, el 14.2 % de las aplicaciones analizadas de un grupo de 11.430 aplicaciones, no coincidían en la información que se declaraba en la política de privacidad oficial y la que se informaba en la sección *Data Safety* de la tienda oficial. [18]

3.5. Natural Language Processing

El análisis de las políticas de privacidad se ha convertido en un área de interés debido al auge de la necesidad de un correcto tratamiento de los datos por parte de las empresas. Por tanto, en

este proyecto se ha desarrollado un mecanismo para simplificar a los usuarios la tarea de entender las políticas de privacidad.

Para lograr esto, se plantea el uso de un modelo de procesamiento basado en NLP (Natural Language Processing), es decir, un modelo capaz de entender, manipular y analizar el lenguaje natural. En concreto, se hace uso del Reconocimiento de Entidades Nombradas (NER, por sus siglas en inglés). El Reconocimiento de Entidades Nombradas es un proceso de NLP que consiste en identificar y clasificar entidades nombradas en un texto. En este caso, las entidades nombradas son aquellas que hacen referencia a una serie de tipos básicos definidos a partir de los datos que se pueden recopilar de los usuarios. [19]

El texto a analizar, en primer lugar, pasa por el proceso llamado tokenización, mediante el cual se divide el texto en *tokens*, es decir, fragmentos de texto que son más pequeños y más fáciles de procesar.

Tras esto, se realiza un proceso de etiquetado en el cual se asigna a los distintos tokens una etiqueta que los identifica sintácticamente y semánticamente en la oración.

Posteriormente cada token se transforma en una representación numérica conocida como *vector de palabras* o *embeddings*. Estos *embeddings*, en este caso preentrenados, asignan a cada palabra un vector de números que representa su significado. Estos vectores ayudan al modelo a relacionar aquellas palabras con significados similares. [20] [21]

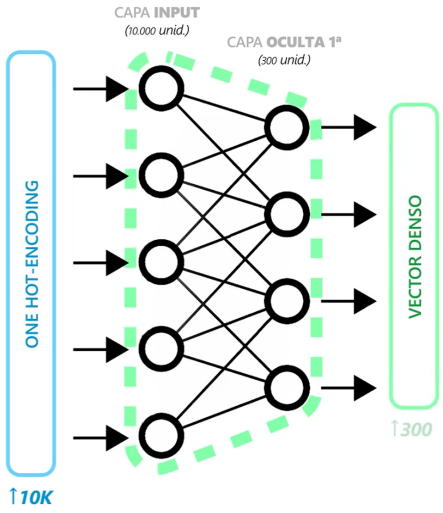


Figura 3.1: Embedding que convierte un vector de tamaño 10000 a un vector denso de 300

Tras obtener estas representaciones vectoriales de los tokens, se aplica la técnica NER (Named Entity Recognition), mediante la cual se clasifican fragmentos de texto en entidades que, en este caso, son aquellas que hacen referencia a los tipos de datos básicos que se recopilan según el *Data Safety* de la tienda oficial de Android. [19]

3.6. Protocolos y estándares

A continuación, se ofrece una breve descripción de los principales protocolos y estándares que se utilizan en el proyecto desarrollado.

3.6.1. Protocolo HTTPS

El protocolo HTTPS (*HyperText Transfer Protocol*) es la versión segura del protocolo HTTP, que el principal protocolo utilizado para enviar datos entre un navegador web y un sitio web, es decir, es un protocolo de nivel de aplicación. Sin embargo, el protocolo HTTPS utiliza un crifrado para aumentar la seguridad de los datos transmitidos.

El protocolo HTTP y HTTPS se comunica mediante mensajes de texto. A continuación, en la figura 3.2 se puede ver un ejemplo de una petición HTTP y una petición HTTPS con la correspondiente respuesta.

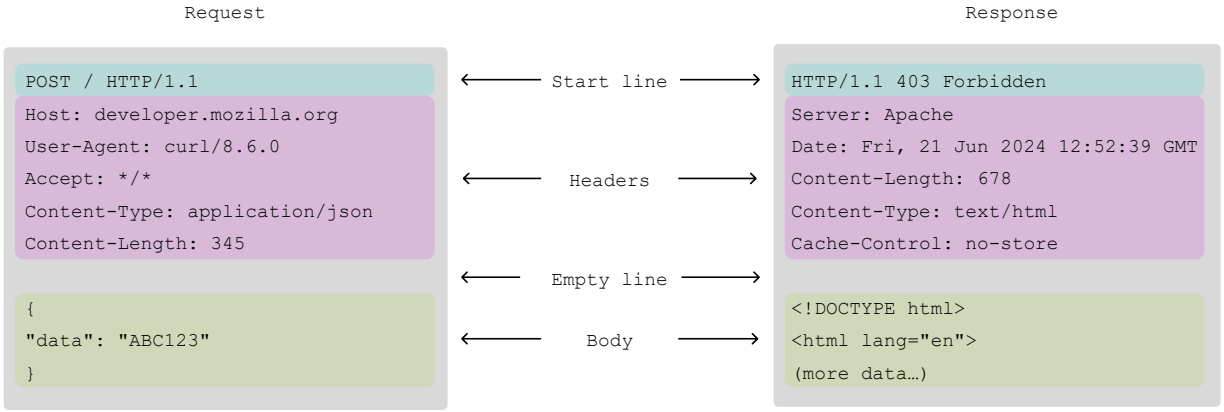


Figura 3.2: Petición HTTP y HTTPS[1]

En una petición HTTP y/o HTTPS, se definen una serie de elementos que son:

- Método HTTP. Existen varios que realizan diferentes operaciones sobre el recurso del servidor web.
- Recurso: es el recurso del servidor web solicitado por el cliente.
- Headers: son una serie de metadatos que describen el mensaje enviado. Por ejemplo, indicando el formato del mensaje. Es un campo opcional.
- Body: es el contenido de la petición que se envía al servidor. También se trata de un campo opcional.

A continuación, se muestran los distintos métodos HTTP [23].

[1]Imagen tomada de [22]

- GET: solicita una representación de un recurso específico.
- HEAD: pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
- POST: se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
- PUT: reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
- DELETE: borra un recurso en específico.
- CONNECT: establece un túnel hacia el servidor identificado por el recurso.
- OPTIONS: utilizado para describir las opciones de comunicación para el recurso de destino.
- TRACE: realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.
- PATCH: utilizado para aplicar modificaciones parciales a un recurso.

3.6.2. Protocolo TCP

El protocolo TCP (Transmission Control Protocol), actúa como protocolo de transporte para HTTP. Es un protocolo orientado a conexiones que proporciona un servicio de entrega de paquetes confiable y ordenado. Para iniciar una comunicación mediante el protocolo TCP, primero se establece una conexión entre el cliente y el servidor con el llamado *Three-Way Handshake* como se puede ver en la figura 3.3.

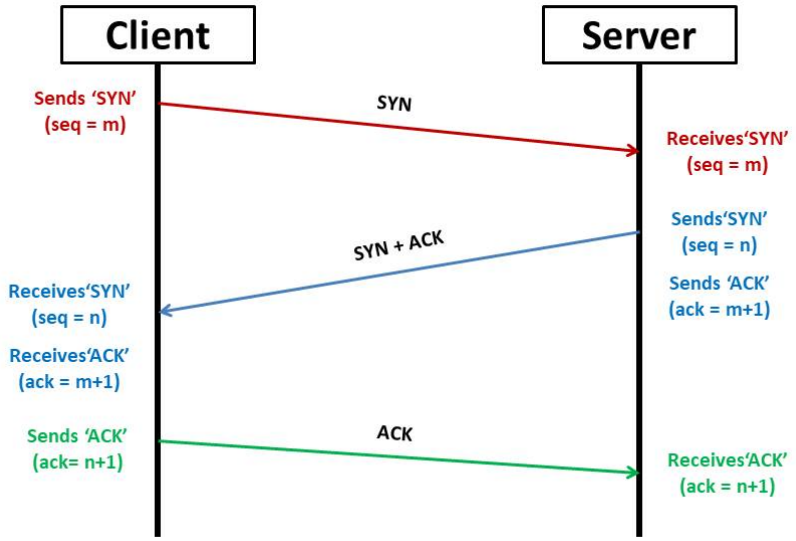


Figura 3.3: Establecimiento de una conexión TCP [2]

3.6.3. Protocolo TLS

Como se ha mencionado en el apartado anterior, el protocolo HTTPS utiliza un protocolo de encriptación para encriptar las comunicaciones y, este protocolo es conocido como *Transport Layer Security* (TLS), previamente SSL.

Estos protocolos aseguran la confidencialidad y autenticidad de los datos transmitidos en una comunicación entre un cliente y un servidor web. [25]

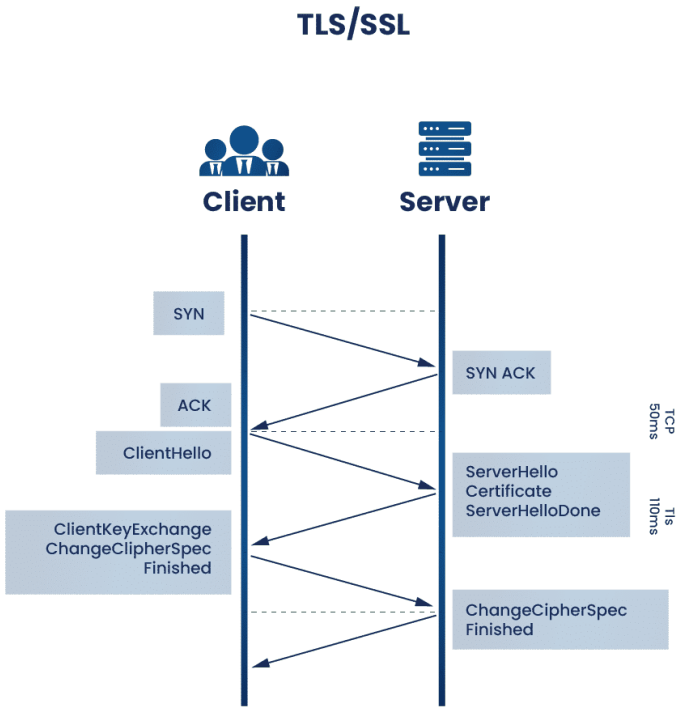


Figura 3.4: Establecimiento de una conexión segura TLS/SSL sobre TCP [3]

3.6.4. Estándar RestAPI

Una REST API es una interfaz que permite la comunicación entre sistemas utilizando un protocolo HTTP y operaciones o métodos estándares como GET, POST, PUT o DELETE.

[2]Imagen tomada de [24]
[3]Imagen tomada de [26]

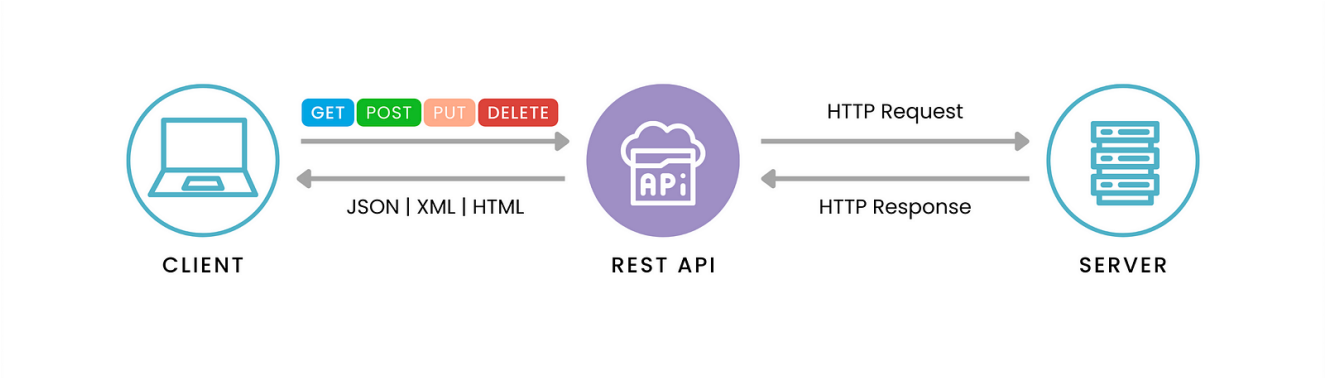


Figura 3.5: Modelo Rest API[4]

3.6.5. Estándar OpenAPI

OpenAPI es un estándar para describir APIs REST de forma estructurada. Este estándar permite documentar los endpoints, parámetros, respuestas y esquemas de datos de una API. El uso de *OpenAPI* facilita la generación de la documentación acerca de la API. [28]

[4]Imagen tomanda de [27]

Capítulo 4

Análisis

En este capítulo se analiza el problema que resolverá el proyecto. Para ello, se describen los requisitos funcionales y no funcionales del proyecto para determinar el alcance del mismo. Además, se establecen y analizan una serie de casos de uso correspondientes con el comportamiento y funcionalidad principal del sistema.

4.1. Requisitos

En esta sección se detallan los requisitos funcionales (RFXX) y no funcionales del proyecto (RNFXX).

4.1.1. Requisitos funcionales

A continuación, se muestran los requisitos funcionales contemplados para el proyecto.

- **RF01**: El sistema permitirá analizar archivos *apk* proporcionados por los usuarios.
- **RF02**: El sistema permitirá a los usuarios analizar las aplicaciones instaladas en su dispositivo.
- **RF03**: El sistema almacenará los datos de cada escaneo realizado por el usuario en una base de datos local.
- **RF04**: El sistema permitirá a los usuarios acceder al historial de escaneos realizados.
- **RF05**: El sistema permitirá al usuario conocer qué trackers utiliza la aplicación que está analizando.
- **RF06**: El sistema identificará posibles indicios de malware en las aplicaciones analizadas.
- **RF07**: El sistema mostrará al usuario los resultados del análisis realizado.
- **RF08**: El sistema permitirá a los usuarios verificar el correcto cumplimiento de las políticas de privacidad de las aplicaciones analizadas.
- **RF09**: El sistema clasificará los permisos por nivel de riesgo para la privacidad del usuario.

- **RF10**: El sistema permitirá al usuario verificar si una aplicación ha sido instalada desde una fuente desconocida.
- **RF11**: El sistema mostrará al usuario los metadatos del fichero *apk* correspondiente a la aplicación analizada.
- **RF12**: El sistema permitirá al usuario buscar aplicaciones específicas entre las aplicaciones instaladas.
- **RF13**: El sistema permitirá al usuario filtrar según diferentes parámetros los escaneos realizados.
- **RF14**: El sistema solicitará el consentimiento del usuario antes de enviar datos necesarios para el análisis a servidores externos o APIs de terceros.
- **RF15**: El sistema mostrará mensajes de error en el caso de darse alguno, indicando de manera comprensible la causa del problema al usuario.

4.1.2. Requisitos no funcionales

A continuación, se listan los requisitos no funcionales correspondientes para el proyecto.

- **RNF01**: El sistema deberá ser compatible con dispositivos Android 10 (SDK: 29) y versiones superiores.
- **RNF03**: El sistema realizará el análisis de la aplicación en menos de 30 segundos en el 90 % de los análisis realizados.
- **RNF04**: El sistema mostrará los datos del análisis de una manera legible y accesible para usuarios sin conocimientos técnicos.
- **RNF05**: El sistema limitará las conexiones simultáneas a 10 para prevenir posibles ataques de denegación de servicio.
- **RNF06**: El sistema permitirá al usuario acceder a los análisis realizados de manera *offline*[1].
- **RNF07**: El sistema tendrá un tamaño de instalación inferior a 30MB.
- **RNF08**: El sistema podrá ejecutarse en móviles de gama baja, con al menos 2GB de RAM.
- **RNF09**: El sistema se comunicará con el servidor de procesamiento de una manera cifrada con el protocolo HTTPS.
- **RNF10**: El sistema solicitará los permisos estrictamente necesarios para su funcionamiento.

[1]Offline: que no es necesario disponer de conexión a Internet.

- **RNF11**: El sistema tendrá una arquitectura escalable y permitirá la incorporación de nuevas funcionalidades sin grandes reestructuraciones.
- **RNF12**: El sistema consultará el repositorio de datos *App-PIMD*.
- **RNF13**: El sistema no almacenará información que pueda identificar a los usuarios como nombres, correos electrónicos o direcciones IP.

4.2. Modelo de dominio

En la figura 4.1 se muestra el modelo de dominio del sistema planteado.

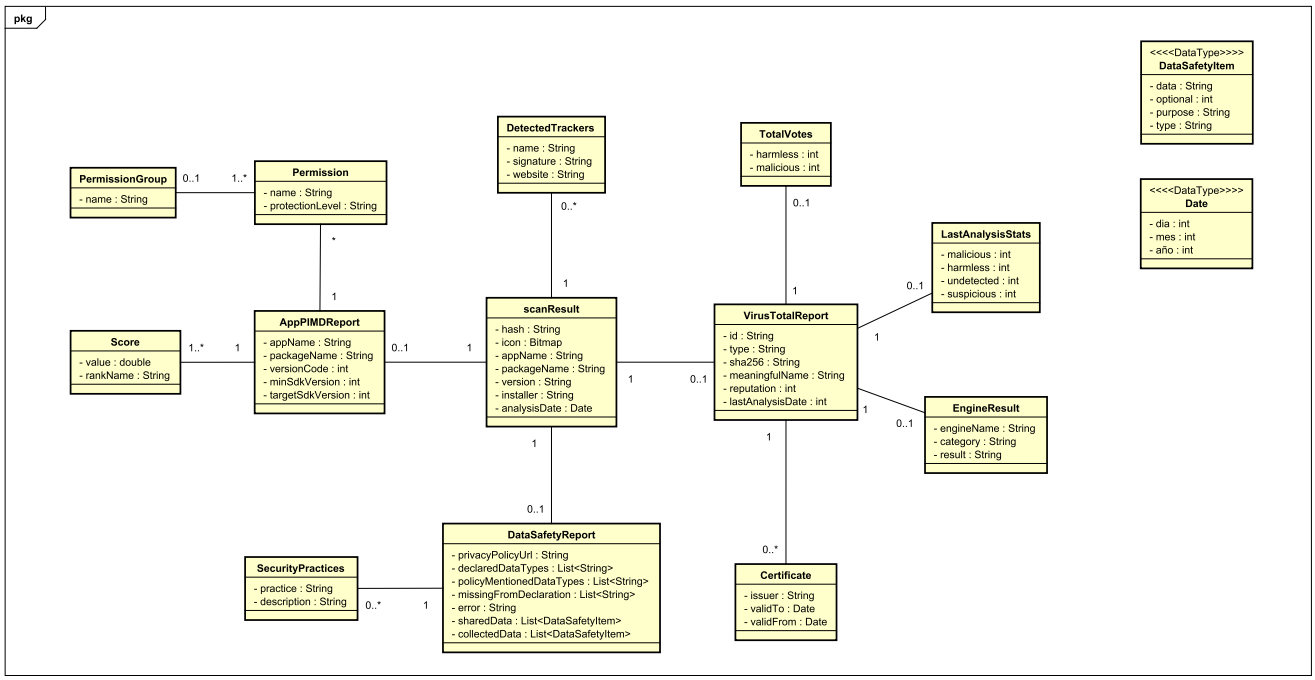


Figura 4.1: Diagrama de clases del modelo de dominio

En el modelo de dominio anterior se puede observar la clase *ScanResult*, relacionada con los distintos reportes que se obtienen de las fuentes de datos. Estas clases que representan los reportes obtenidos de las distintas fuentes de datos son *DataSafetyReport*, *VirusTotalReport*, *AppPIMDReport* y *DetectedTrackers*. Estos reportes están relacionados con un único *ScanResult*, es decir, con un único resultado de análisis de la aplicación. Este, a su vez, puede tener o no dichos reportes ya que, es posible, que no se disponga de la información en las fuentes de datos que generan los reportes.

La clase *AppPIMDReport* contiene los datos que se reciben de la consulta al repositorio *App-PIMD*. Entre estos datos se encuentran los permisos solicitados por la aplicación, los permisos declarados, nivel de afectación a la privacidad del usuario entre otros metadatos.

La clase *VirusTotalReport* contiene los datos que se reciben de la consulta a la API de *VirusTotal*. Principalmente, se obtienen los resultados del análisis realizado por los más de 70 antivirus [29] de los que dispone el escaner de *VirusTotal* indicando si la aplicación es considerado como *malware*[2] o no.

La clase *DetectedTrackers* contiene los datos recabados tras realizar el análisis estático de la aplicación. Estos son el nombre del tracker, su firma y la página web del tracker.

Por último, la clase *DataSafetyReport* contiene la información que se obtiene de la respuesta de la consulta a la API del servidor de procesamiento que analiza, mediante un modelo de IA, los datos

[2]Malware: cualquier programa informático diseñado para causar daños en un sistema o en los datos de un usuario

que supuestamente estaría recopilando la aplicación y si se declaran en el apartado del *Data Safety* de la *Google Play Store*.

La clase central y dominante del sistema es la clase *ScanResult*, esta clase actúa como punto de agregación para la información resultante del análisis de una aplicación. Esta clase contiene y controla la relación con los distintos reportes generados por las fuentes de datos, estos son *DataSafetyReport*, *VirusTotalReport*, *AppPIMDReport* y *DetectedTrackers*.

La clase *ScanResult* es la responsable de gestionar el acceso y almacenamiento a los reportes mencionados.

4.3. Casos de uso

En esta sección se recogen los casos de uso del sistema. En la figura se muestra el diagrama de casos de uso del sistema.

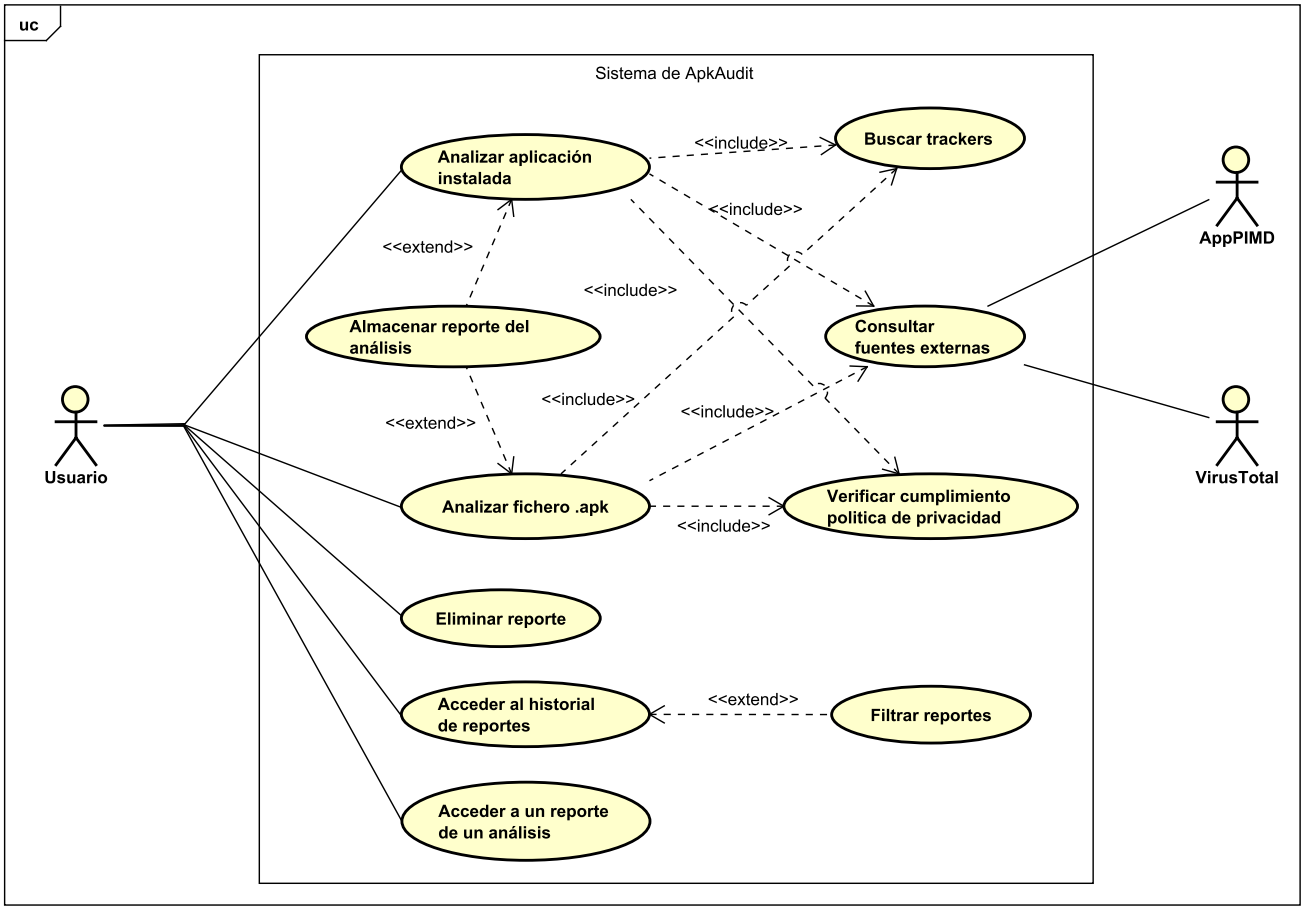


Figura 4.2: Diagrama de casos de uso

4.3.1. CU01: Analizar aplicación instalada

Descripción: Analiza la *app* instalada en el dispositivo del usuario.

Precondición: -

Secuencia normal:

1. El usuario selecciona la opción de analizar una *app* instalada.
2. El sistema carga y muestra las aplicaciones instaladas.
3. El usuario selecciona la *app* que desea analizar a través del listado de las aplicaciones instaladas de su dispositivo.
4. El sistema obtiene el *packageName* y *hash* de la *app* instalada seleccionada.

5. El sistema comprueba a través del *hash* si ya se ha analizado la *app*.
6. El sistema comprueba que el dispositivo dispone de conexión a Internet.
7. El sistema extrae el fichero *.apk* correspondiente a la *app* seleccionada.
8. El sistema realiza un análisis estático de la *app* seleccionada a partir del fichero *.apk* para encontrar firmas de trackers.
9. El sistema envía una petición al servidor de procesamiento con el *packageName* de la *app* seleccionada para comprobar el correcto cumplimiento de la política de seguridad.
10. El sistema envía una petición con el *hash* de la *app* a la API del repositorio *App-PIMD*.
11. El sistema envía una petición con el *hash* de la *app* a la API de *VirusTotal*.
12. El sistema almacena en la base de datos local la información obtenida de las fuentes de datos.

Flujos alternativos:

2a. El usuario no tiene aplicaciones instaladas.

- 2a- 1. El sistema muestra que no se encuentran aplicaciones instaladas.
- 2a- 2. El caso de uso queda sin efecto.

5a. La aplicación ya ha sido analizada por el usuario.

- 5a- 1. El sistema muestra la información correspondiente con el análisis realizado con anterioridad.
- 5a- 2. El caso de uso acaba.

6a. El dispositivo no dispone de conexión a Internet.

- 6a- 1. El sistema muestra un mensaje de error indicando que no se dispone de conexión a Internet.
- 6a- 2. El caso de uso acaba.

8a. No se encuentran firmas de trackers en el fichero *.apk*.

- 8a- 1. El sistema muestra que no se encontraron trackers en la *app*.
- 8a- 2. El caso continúa en el paso 9.

9a. No se encuentran discrepancias en la política de privacidad de la *app*.

- 9a- 1. El sistema muestra que no se encontraron discrepancias en la *app*.
- 9a- 2. El caso continúa en el paso 10.

9b. No se puede establecer conexión con el servidor de procesamiento.

9b- 1. El sistema muestra el error correspondiente.

9b- 2. El caso continúa en el paso 9.

10a. No se encuentra una coincidencia con el *hash* de la *app* en el repositorio de *App-PIMD*.

10a- 1. El sistema envía la petición con el *packageName* de la *app* al repositorio de *App-PIMD*.

10a- 2. El caso de uso continúa en el paso 11.

10b. No se encuentra una coincidencia con el *packageName* de la *app* en el repositorio de *App-PIMD*.

10b- 1. El sistema sube el fichero *.apk* al repositorio *App-PIMD*.

10b- 2. El sistema consulta el repositorio de *App-PIMD* con el *hash* de la *app*.

10b- 3. El caso de uso continúa en el paso 11.

11a. No se encuentra una coincidencia con el *hash* de la *app* en la API de *VirusTotal*.

11a- 1. El sistema sube el fichero *.apk* a la API de *VirusTotal*.

11a- 2. El sistema consulta la API de *VirusTotal* con el *hash* de la *app*.

11a- 3. El caso de uso continúa en el paso 12.

Postcondición: La información del análisis de la *app* solicitada queda almacenada en el *dispositivo* del usuario.

Diagrama de secuencia en análisis CU01

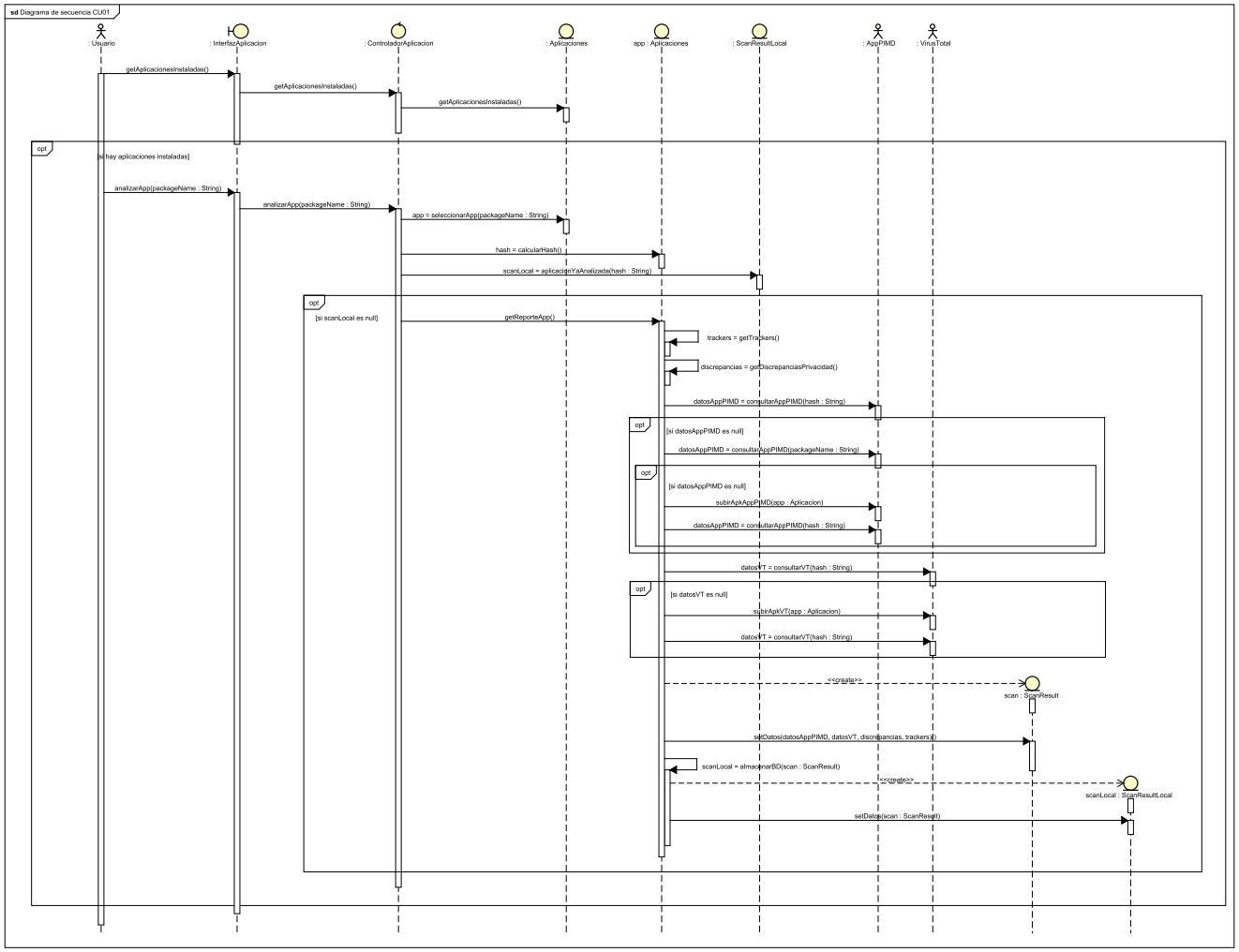


Figura 4.3: Diagrama de secuencia en análisis del caso de uso CU01

4.3.2. CU02: Analizar un archivo *apk*

Descripción: Analiza el fichero *.apk* que seleccione el usuario.

Precondición: -

Secuencia normal:

1. El usuario selecciona la opción de analizar una *apk*.
2. El sistema muestra el selector de ficheros del dispositivo del usuario.
3. El usuario selecciona el fichero *.apk* que desea analizar.
4. El sistema obtiene el *hash* del fichero *.apk* seleccionado.
5. El sistema comprueba a través del *hash* si ya se ha analizado la *app*.

6. El sistema comprueba que el dispositivo dispone de conexión a Internet.
7. El sistema obtiene el *packageName*, *version* y *nombre* de la *app*.
8. El sistema envía una petición con el *hash* de la *app* a la API del repositorio *App-PIMD*.
9. El sistema envía una petición con el *hash* de la *app* a la API de *VirusTotal*.
10. El sistema envía una petición al servidor de procesamiento con el *packageName* de la *app* seleccionada para comprobar el correcto cumplimiento de la política de seguridad.
11. El sistema decompila el fichero *.apk* correspondiente para realizar un análisis estático y extraer los trackers utilizados por la *app*.
12. El sistema almacena en la base de datos local la información obtenida de las fuentes de datos.

Flujos alternativos:

5a. La aplicación ya ha sido analizada por el usuario.

5a- 1. El sistema muestra la información correspondiente con el análisis realizado con anterioridad.

5a- 2. El caso de uso acaba.

6a. El dispositivo no dispone de conexión a Internet.

6a- 1. El sistema muestra un mensaje de error indicando que no se dispone de conexión a Internet.

6a- 2. El caso de uso acaba.

8a. No se encuentra una coincidencia con el *hash* de la *app* en el repositorio de *App-PIMD*.

8a- 1. El sistema envía la petición con el *packageName* de la *app* al repositorio de *App-PIMD*.

8a- 2. El caso de uso continúa en el paso 9.

8b. No se encuentra una coincidencia con el *packageName* de la *app* en el repositorio de *App-PIMD*.

8b- 1. El sistema extrae el fichero *.apk*.

8b- 2. El sistema sube el fichero *.apk* al repositorio *App-PIMD*.

9b- 3. El sistema consulta el repositorio de *App-PIMD* con el *hash* de la *app*.

8b- 4. El caso de uso continúa en el paso 9.

9a. No se encuentra una coincidencia con el *hash* de la *app* en la API de *VirusTotal*.

9a- 1. El sistema extrae el fichero *.apk*.

9a- 2. El sistema sube el fichero *.apk* a la API de *VirusTotal*.

9a- 3. El sistema consulta la API de *VirusTotal* con el *hash* de la *app*.

9a- 4. El caso de uso continúa en el paso 11.

11a. No se encuentran trackers en el fichero *.apk*.

11a-1. El sistema muestra al usuario que no se han encontrado trackers en la *app*.

11a-2. El caso de uso continúa en el paso 12.

Postcondición: La información del análisis de la *app* solicitada queda almacenada en el *dispositivo* del usuario.

Diagrama de secuencia en análisis CU02

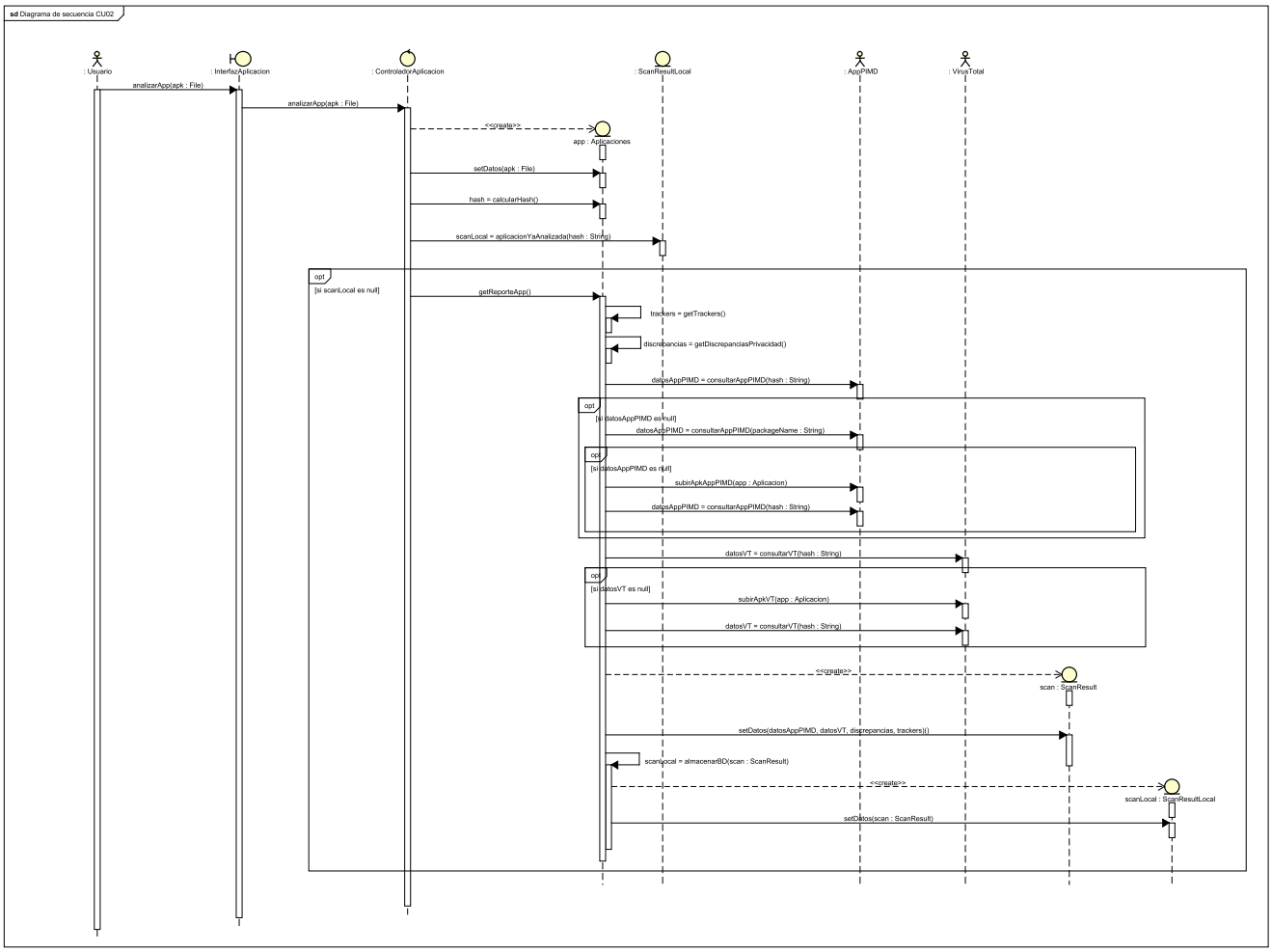


Figura 4.4: Diagrama de secuencia en análisis del caso de uso CU02

4.3.3. CU03: Acceder análisis ya realizado

Descripción: Se accede a un análisis ya realizado y almacenado en el dispositivo del usuario.

Precondición: El usuario ya ha analizado al menos una *app*.

Secuencia normal:

1. El usuario selecciona la opción de ver historial de escaneos.
2. El sistema carga y muestra los reportes de las aplicaciones analizadas con anterioridad.
3. El usuario selecciona el análisis de la *app* que desea ver a través del listado de las aplicaciones analizadas de su dispositivo.
4. El sistema busca en función del *hash* de la app la información asociada a dicho escaneo.
5. El sistema muestra el reporte al usuario.

Flujos alternativos: -

Postcondición: -

Diagrama de secuencia en análisis CU03

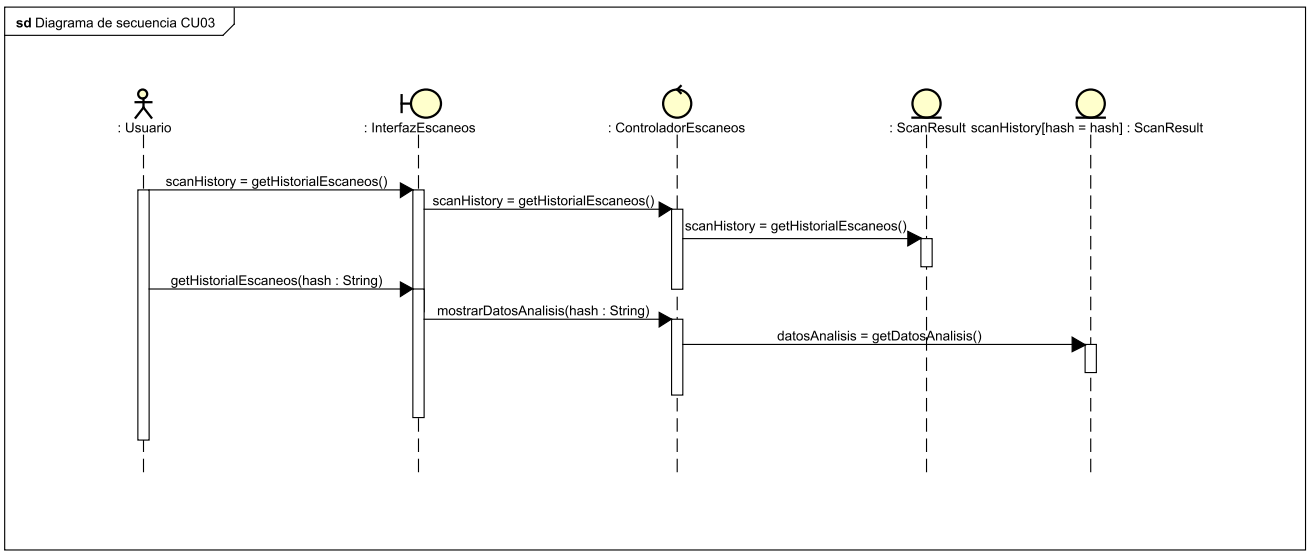


Figura 4.5: Diagrama de secuencia en análisis del caso de uso CU03

4.4. Análisis de las fuentes de datos

La aplicación móvil *ApkAudit* recoge la información de varias fuentes de datos independientes entre sí. Por tanto, actúa como un Sistema Integrador y su función principal es la de consultar, unificar y presentar los datos obtenidos de las APIs y/o fuentes de datos. A continuación, se describen las fuentes de datos que se utilizan y la información que proporcionan.

4.4.1. Repositorio *App-PIMD*

El repositorio *App-PIMD* es un sistema accesible a través de una API Restful que permite consultar información sobre las aplicaciones Android. Esta información se almacena en una base de datos relacional e incluye los permisos utilizados por las aplicaciones, el impacto de estos sobre la privacidad del usuario, los permisos que definen, la categoria a la que pertenece la aplicación (Comunicación, Compras, Finanzas, etc). Además, ofrece una métrica de privacidad global que permite saber el impacto total que tiene sobre la privacidad del usuario.

Esta fuente de datos almacena los datos estructurados de las diferentes aplicaciones analizadas. Sin embargo, existen aplicaciones que no se encuentran en el repositorio y, consecuentemente, las aplicaciones tienen que ser subidas al repositorio para poder ser analizadas y, posteriormente, extraer sus metadatos.

4.4.2. *VirusTotal*

VirusTotal es una plataforma que inspecciona ficheros mediante más de 70 antivirus con el fin de detectar si un fichero contiene malware[3]. En este caso, la aplicación *ApkAudit* utiliza la API de *VirusTotal* para verificar si una aplicación está infectada con malware y mostrar los resultados de este análisis al usuario.

No obstante, tambien cabe la posibilidad de que una aplicación no haya sido escaneada previamente en *VirusTotal*, en cuyo caso, la aplicación sube el fichero *.apk* a la plataforma para que está sea analizada y, posteriormente, se obtengan los resultados.

4.4.3. Servidor de procesamiento

Este servidor ha sido desarrollado específicamente para este proyecto junto con la aplicación *ApkAudit*. Su función principal es analizar el correcto cumplimiento de la políticas de privacidad de las aplicaciones solicitadas y proporcionar información adicional relacionada con la seguridad del usuario.

El servidor de procesamiento expone su funcionalidad a través de una API Restful que recibe como entrada el nombre de paquete de la aplicación y, consecuentemente, obtiene mediante *Google Play Scraper* la información del apartado *Data Safety* de la aplicación correspondiente en la *Google Play Store*.

Tras realizar este scraping, se obtiene los datos que el desarrollador ha indicado que se recogen en la aplicación junto con el enlace a la política de privacidad. Por tanto, mediante este servidor, se analiza la política de privacidad al completo de la aplicación y se obtiene un reporte que presenta los datos que se indican que se recogen en la política de privacidad pero no el apartado de *Data*

[3]Software malicioso que tiene como objetivo dañar o alterar el funcionamiento de un sistema informático

Safety de la *Google Play Store*.

Además, de este servidor tambien se obtiene una descripción para los permisos que utilizan las aplicaciones, una breve descripción de los tipos básicos que se declaran en el *Data Safety* y una serie de consejos de seguridad relacionados con el entorno del sistema Android. Todo esto con el fin de orientar al usuario sobre el significado de los datos que se recogen en la aplicación.

4.4.4. Recursos locales de trackers

La información relativa a los trackers utilizados por las aplicaciones se encuentra definida como arrays de cadenas de texto en los archivos de recursos (`res/values/arrays.xml`) de la aplicación.

Cada tracker está representado por su nombre, firma con la que se puede identificar en las clases *.dex* de la aplicación y la página web en la que se encuentra documentada. Esta estructura permite acceder a los recursos rápidamente desde el código sin tener que realizar consultas a bases de datos que pueden ser más lentas. Desde el código, el acceso a estos arrays se realiza a través de la función `getStringArray(R.array.trackers)`, que devuelve el array de cadenas de texto solicitado, en este caso las firmas de los trackers.

La información acerca de los trackers se ha obtenido del proyecto *OpenSource*: *Tracker Control* [1]

Capítulo 5

Diseño

En este capítulo se describe el diseño del sistema desarrollado. El objetivo de este capítulo es describir la arquitectura de la aplicación y el servidor, los patrones de diseño empleados y los distintos componentes que conforman el sistema.

5.1. Componentes del sistema

A continuación, se muestra el diagrama de componentes del sistema en el que se detallan los principales componentes que forman parte del mismo.

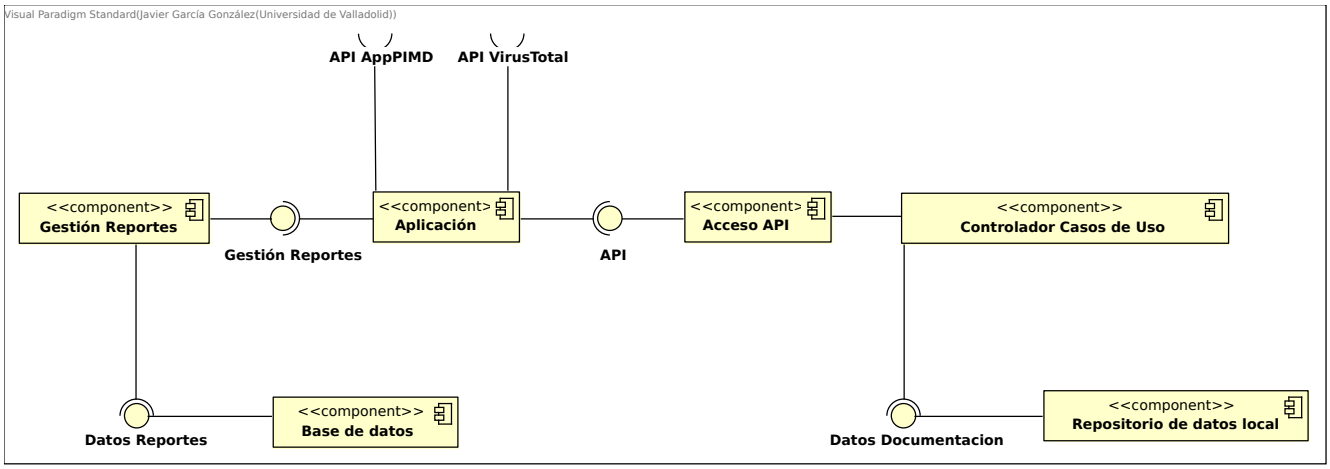


Figura 5.1: Diagrama de componentes del sistema

En la figura 5.1 se puede observar que el sistema se podría dividir en dos grandes partes:

- **Aplicación**: la aplicación móvil gestiona y almacena los reportes de las aplicaciones que se escanean. Para obtener el resultado del análisis de la política de privacidad de la aplicación analizada se solicita al servidor a través de su API.

- **Servidor**: se encarga de procesar las peticiones recibidas por la aplicación a través de la API que se expone. Además, cuenta con un repositorio de datos local que almacena información sobre permisos de Android, consejos de seguridad entre otros y, estos datos son solicitados por la aplicación para mostrar información adicional al usuario.

Además, se muestra que se hace uso de la API otras fuentes de datos pero, a diferencia de la API del servidor de procesamiento, no se detalla la posible implementación de dichas APIs debido a que se tratan de APIs externas y de terceros.

5.2. Arquitectura de la aplicación

En esta sección se presentan muchas de las decisiones que se han tomando en el diseño de la aplicación y el motivo por el cual se han optado por ellas.

5.2.1. Arquitectura MVVM

Model-View-ViewModel (MVVM) es un patrón arquitectónico que se utiliza principalmente en el desarrollo de aplicaciones móviles. El patrón MVVM ayuda a mantener separada la lógica de neogocio y la lógica de presentación de la interfaz de usuario. Esto lo hace fácil de mantener, probar y de escalar. [30]. Esta patrón de arquitectura ha ganado bastante popularidad, como ya se ha mencionado, en el desarrollo de las aplicaciones móviles principalmente desde que *Google* la conviertiera en su arquitectura oficial al lanzar la guía de arquitecturas [31].

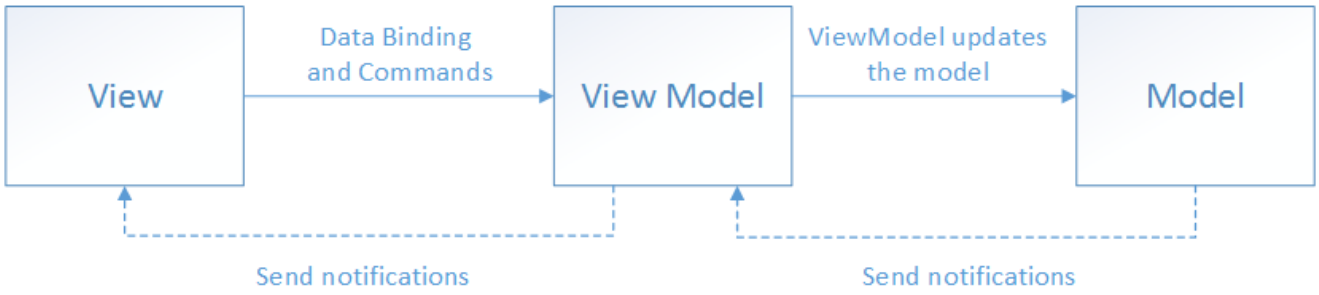


Figura 5.2: Patrón arquitectónico MVVM[1]

Como se puede observar en la figura 5.2, patrón MVVM consta de 3 componentes:

- **Model**: Representa la parte de datos, es decir, la manera en la que se almacenan, gestionan y/o recuperamos datos de una base de datos o algún otro recurso externo.
- **View**: Representa la parte de la interfaz de usuario. Es el encargado de mostrar los datos al usuario y de recibir las interacciones del usuario.
- **ViewModel**: Es el encargado de realizar la conexión entre la vista y el modelo. De modo que las vistas se subscriben a sus viewModels correspondientes y estos, a su vez, se encargan de gestionar que los cambios en el modelo se reflejen en la vista.

Gracias a aplicar este patrón, cada componente es independiente de los demás y permite que el código sea más fácil de mantener y probar. Este patrón se ha aplicado en el desarrollo de la aplicación *ApkAudit* debido a las ventajas que ya han sido nombradas anteriormente.

[1]Imagen tomada de [30].

5.2.2. Diseño de acceso a datos

Para el acceso a los datos se ha optado por el uso del patrón Repositorio el cual se utiliza para manejar la lógica de acceso a datos de una manera centralizada. De este modo, se separa la lógica que obtiene los datos y lo asigna a modelo de entidad que es usado en la lógica de negocio [32]. En el caso de la aplicación *ApkAudit*, se hace uso de dos repositorios:

- ***OnboardingRepository***: se encarga de solicitar y administrar los datos necesarios en el primer comienzo de la aplicación.
- ***ScanRepository***: se encarga de solicitar todos los datos relacionados con el escaneo de la aplicación. Desde obtener los escaneos almacenados localmente a solicitar a cada una de las fuentes de datos el correspondiente reporte de una aplicación.

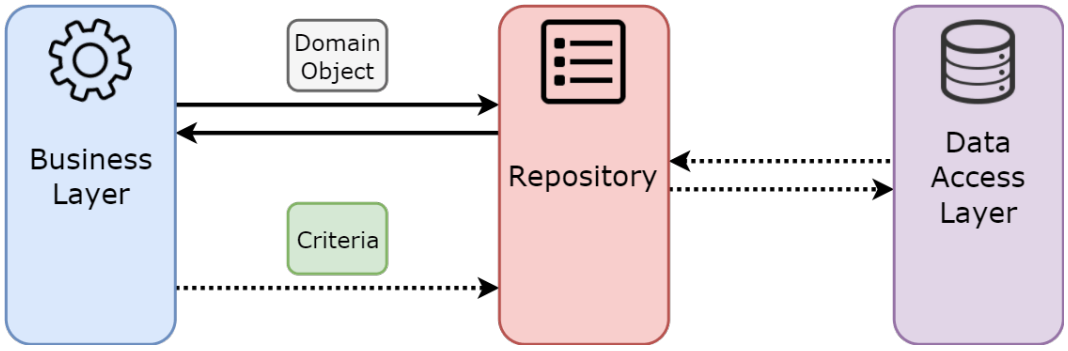


Figura 5.3: Patrón Repositorio[2]

En cuanto al almacenamiento local, se utiliza el patrón arquitectónico *Data Access Object (DAO)* con el cual se propone el uso de un objeto (DAO) el cual centraliza el acceso y almacenamiento en la base de datos. Mediante este patrón de acceso a datos se gestionan los datos de la base de datos local de *Room*.

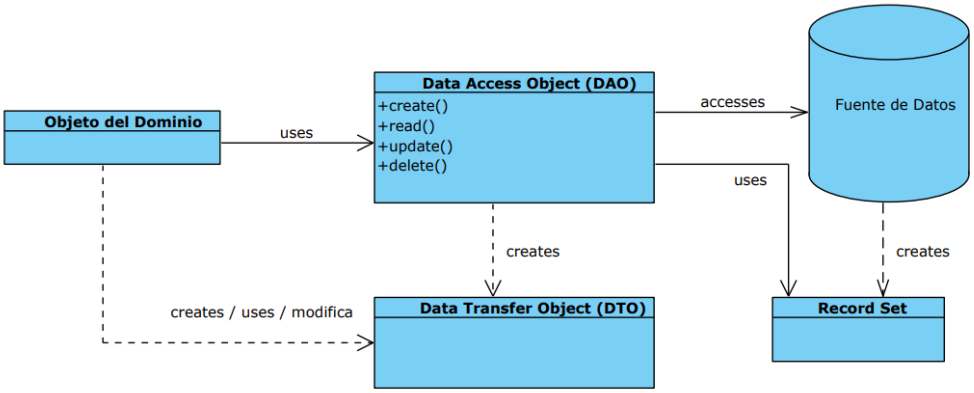


Figura 5.4: Data Access Object (DAO)[3]

[2]Imagen tomada de [33].
[3]Imagen tomada de [34].

5.2.3. Diseño de la base de datos

En cuanto a la persistencia de los datos en la aplicación se ha optado por el uso de *Room*. *Room* es una libreria que proporciona una abstracción sobre la base de datos *SQLite*. En esta base de datos se almacena lo siguiente:

- Escaneos realizados: se almacenan todos los escaneos realizados por el usuario. Estos escaneos incluyen los datos de las distintas fuentes de datos, como *App-PIMD* y el servidor de procesamiento.

- Consejos de seguridad: se almacena una serie de consejos de seguridad relacionados con el entorno del sistema Android. Esta información se obtiene al realizar una petición al endpoint */get/security-tips* del servidor de procesamiento.

- Permisos de Android: se almacena una descripción para todos los permisos definidos en la documentación de Android. Esta información se obtiene al realizar una petición al endpoint */permissions.json* del servidor de procesamiento.

- Información tipos de datos básicos del *Data Safety*: se almacena una descripción para todos los tipos de datos definidos en el *Data Safety*. Esta información se obtiene al realizar una petición al endpoint */data-safety-info.json* del servidor de procesamiento.

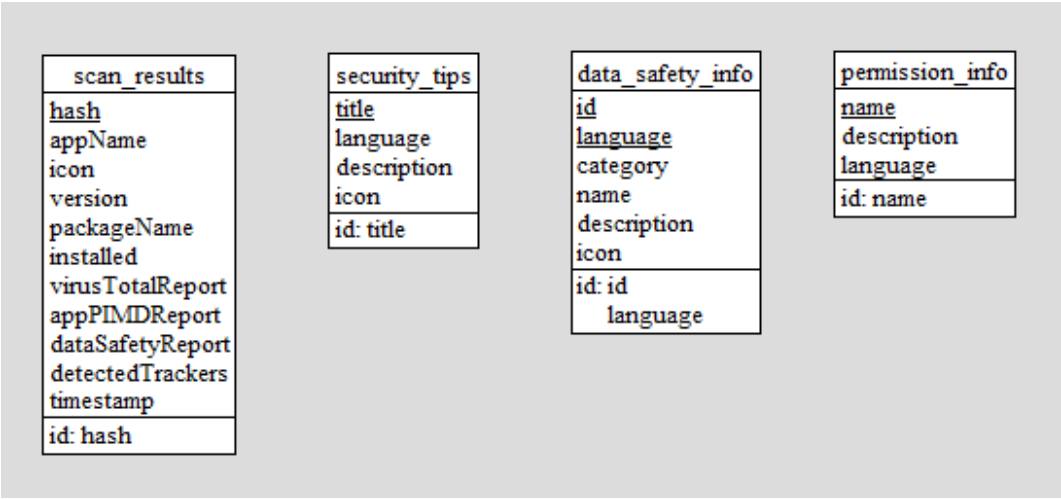


Figura 5.5: Tablas de la base de datos

En la figura 5.5 se puede apreciar el diagrama E/R de la base de datos de la aplicación *ApkAudit*. Como se puede apreciar no existe relación entre las entidades, ya que cada una de ellas se almacena de forma independiente.

A continuación, se detalla para cada entidad la información que se almacenan:

scan_results	
hash	SHA256 del *apk* extraído de la aplicación analizada
appName	Nombre de la aplicación analizada
icon	Icono de la aplicación analizada
version	Version en formato de cadena de caracteres de la aplicación analizada
installed	Instalador de la aplicación analizada
virusTotalReport	JSON serializado como cadena de caracteres de la respuesta obtenida por el servicio de *VirusTotal*
appPIMDReport	JSON serializado como cadena de caracteres de la respuesta obtenida del servicio de *App-PIMD*
dataSafetyReport	JSON serializado como cadena de caracteres de la respuesta obtenida del servidor de procesamiento tras el análisis de la política de privacidad de la aplicación analizada
detectedTrackers	JSON serializado de la lista de trackers identificados en la aplicación analizada
timestamp	Fecha en la que se realizó el escaneo

Tabla 5.1: Atributos de los elementos de la tabla *scan_results*.

En la tabla 5.1 se puede observar los atributos de los elementos de la tabla *scan_results*. Los atributos *virusTotalReport*, *appPIMDReport*, *dataSafetyReport*, *detectedTrackers* a diferencia de como se muestra en el modelo de dominio 4.1 se almacenan como cadenas de caracteres. Esto se debe a que se trata de una respuesta JSON que se serializa a una cadena de caracteres.

security_tips	
title	Título del consejo de seguridad
language	Idioma del consejo de seguridad almacenado
description	Descripción del consejo de seguridad
icon	Icono asociado al consejo de seguridad

Tabla 5.2: Atributos de los elementos de la tabla *security_tips*.

En la tabla 5.2 se puede observar los atributos de los elementos de la tabla *security_tips*. Debido a que se almacenan los mismos consejos de seguridad pero en diferentes idiomas, se almacena el atributo *language* para indicar el idioma en el que se encuentra almacenado el consejo de conseguridad en cuestión. Este atributo también se puede encontrar para los permisos de Android y los tipos de datos del *Data Safety*.

data_safety_info	
id	Nombre en inglés del tipo de dato del *Data Safety*
language	Idioma del consejo de seguridad almacenado
category	Categoría del tipo de dato del *Data Safety*
name	Nombre del tipo de dato del *Data Safety*
description	Descripción del tipo de dato
icon	Icono asociado al tipo de dato

Tabla 5.3: Atributos de los elementos de la tabla *data_safety_info*.

En la tabla 5.3 se presentan los atributos de los elementos de la tabla *data_safety_info*. A diferencia de la tabla de 5.4, en esta tabla la clave primaria está formada por el atributo *id* y el *language*. Esto se debe a que se puede encontrar la definición de un tipo de dato en varios idiomas.

permission_info	
name	Nombre en inglés del permiso
description	Descripción del permiso
language	Idioma del permiso almancenado

Tabla 5.4: Atributos de los elementos de la tabla *permission_info*.

En la tabla 5.4 se presentan los atributos de los elementos de la tabla *permission_info*. En este caso, la clave primaria de la tabla es el nombre del permiso en inglés.

5.2.4. Desarrollo nativo

Como ya se ha mencionado anteriormente, para el desarrollo de la aplicación móvil se ha optado por el desarrollo nativo mediante el uso del lenguaje de programación *Kotlin* y el framework de UI *Jetpack Compose*. Esta decisión se debe a los siguientes motivos [35]:

- **Rendimiento:** a diferencia del desarrollo con soluciones multiplataforma como *Flutter*, *React Native* entre otros, el desarrollo nativo con *Kotlin* ofrece la ventaja de que se ejecuta directamente sobre el sistema operativo, sin capas intermedias. Ofreciendo, de este modo, un acceso más directo a los recursos y hardware del dispositivo decrementando los tiempos de respuesta y disminuyendo el consumo de recursos.
- **Funciones del sistema:** se puede acceder a todas las funciones y APIs del sistema operativo sin tener que depender de plugins o bibliotecas externas. Además, se puede acceder a las nuevas funcionalidades del sistema sin tener que esperar que se desarrolle un plugin que lo incorpore.
- **Experiencia de usuario:** al ser una aplicación nativa esta sigue las guías de diseño y estándares de la plataforma en la que se ejecuta.

Además, el uso de *Kotlin* ha tomado un primer plano en el desarrollo de aplicaciones móviles debido al fuerte soporte que tiene por parte de *Google* y su amplia comunidad de desarrolladores.

5.2.5. Permisos de la aplicación

La aplicación móvil *ApkAudit* tiene uno de sus objetivos concienciar a los usuarios sobre su privacidad y seguridad. Por ello y cumplimiendo el RP2, la aplicación unicamente solicita y usa los permisos necesarios para su funcionamiento y no hace uso de ningún tracker. A continuación, se indican los permisos que se solicitan a los usuarios junto con su descripción [36] y motivo de uso:

- **Permiso de acceso a internet** (`android.permission.INTERNET`): Permite a la aplicación abrir sockets de red para establecer conexiones. El motivo de su uso es para poder realizar las peticiones y conexiones con las distintas fuentes de datos (*VirusTotal*, repositorio de *AppPIMD* y el servidor de procesamiento).
- **Permiso de consulta estado red** (`android.permission.ACCESS_NETWORK_STATE`): permite a la aplicación a consultar el estado de red del dispositivo. El motivo de su uso es, también, para poder realizar las peticiones y conexiones con las distintas fuentes de datos (*VirusTotal*, repositorio de *AppPIMD* y el servidor de procesamiento).
- **Permiso de visualización aplicaciones instaladas** (`android.permission.QUERY_ALL_PACKAGES`): se trata de un permiso que permite listar todas las apps instaladas en el dispositivo. El motivo de su uso es para ofrecer al usuario la funcionalidad de auditar las aplicaciones instaladas en su dispositivo.

5.2.6. Diseño arquitectónico de la aplicación

La arquitectura de la aplicación *ApkAudit* consiste en 5 paquetes principales que están envueltos en un paquete general denominado *com.uva.apkaudit*. A continuación, se detalla cada uno de estos paquetes:

- **core**: agrupa las clases que se utilizan de forma transversal en la aplicación. Es decir, los componentes UI que se comparten, las clases de utilidad, tema y clases de navegación de la app.
  - **navigation**: se define el *NavigationHost* con el que se establecen las rutas de navegación hacia las distintas *Screens* de la aplicación.
  - **theme**: en él se definen los colores y temas que tiene la aplicación.
  - **ui**: contiene los componentes de interfaz de usuario que se comparten entre varias de las *Screens* de la aplicación.

    - **utils**: contiene las clases de utilidad. Por ejemplo, para dar formato a las fechas, obtener la métrica de privacidad formateada, etc.

- **data**: consiste en las clases que manejan el acceso a datos y la persistencia de estos. A su vez, esta dividida en los siguientes subpaquetes:

    - **local**: contiene las clases y entidades que se utilizan para gestionar la persistencia de datos localmente. En estas clases se define la base de datos, las tablas, entidades, DTOs y DAOs.
    - **remote**: contiene las clases que realizan las peticiones a las fuentes de datos remotas y APIs. Además, contiene el modelo para recibir la respuesta correspondiente y tratarla en el dominio.

- **di**: contiene el módulo de la aplicación para gestionar la inyección de dependencias. En esta clase se define cómo se inyectan los objetos y clases como *ViewModels*, *Repositories* y otros objetos que son utilizados en la aplicación.

- **features**: contiene cada una de las clases que componen cada una de las distintas funcionalidades de la aplicación. Cada uno de los subpaquetes contiene las clases que definen su interfaz y, además, su clase *ViewModel* correspondiente con la que se maneja la lógica de negocio de ese *feature*.En concreto se tienen las siguientes *features*:

    - **history**: encargado de mostrar el historial de escaneos realizados por el usuario y almacenados localmente en el dispositivo.
    - **home**: es el punto de acceso principal a la aplicación. En él se muestra las opciones relacionadas a escanear una aplicación o ver el historial de escaneos, que redirige a la vista de historial, en el caso de que se hayan realizado.
    - **installedApps**: contiene la vista y lógica de negocio que obtiene y muestra las aplicaciones que están instaladas en el dispositivo del usuario.
    - **scan**: representa la funcionalidad principal de la aplicación, que es el reporte de la aplicación. En este subpaquete se incluye la vista que define como se muestran los resultados obtenidos de las fuentes de datos y, también, la lógica de negocio que se encarga de solicitar los datos a las fuentes de datos, solicitar su almacenaje y, posteriormente, recuperarlos.
    - **virustotal**: contiene la vista y lógica de negocio que muestra información adicional sobre el resultado obtenido de escanear una aplicación mediante la API de *VirusTotal*.

- **repository**: contiene los dos repositorios (`OnboardingRepository` y `ScanRepository`) ya mencionados anteriormente. Mediante estos se aila la lógica de acceso a datos de la lógica de negocio y se obtiene un diseño modular y escalable.

A continuación, en la figura 5.6 se muestra el diagrama de paquetes detallando la estructura que sigue la aplicación.

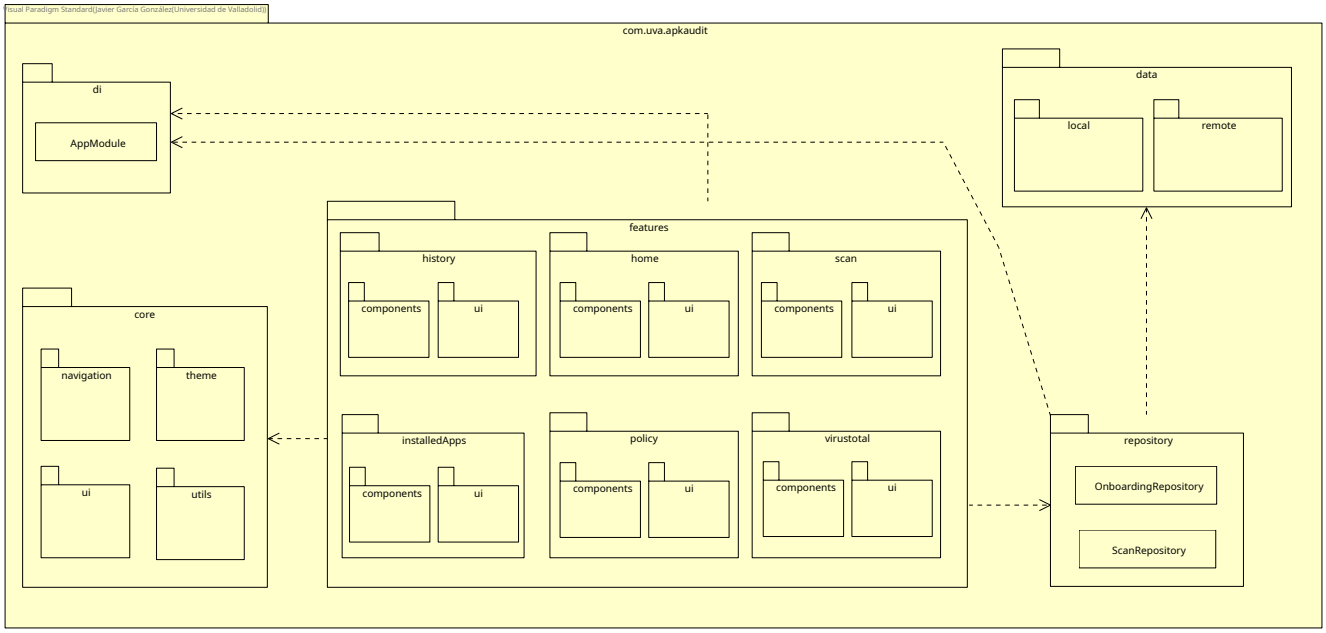


Figura 5.6: Diagrama de paquetes de la aplicación

En la figura anterior 5.6 se muestra una vista general de los paquetes que componen la aplicación y cómo se relacionan entre sí. Como se puede apreciar en dicho diagrama se aplica una modularización por cateracterísticas (Feature-based Modularization) mediante la cual se separa cada funcionalidad de la app en módulos independientes. Esto permite tener una mejor organización, escalabilidad y mejora el aislamiento de cada una de las funcionalidades.

En la figura 5.6 no se muestran la totalidad de los paquetes internos y sus clases debido a la gran cantidad de ellos que hay. En consecuencia, desde la figura 5.7 hasta la figura 5.9 se muestra un diagrama de clases para cada uno de los paquetes que no se han mostrado al completo en la figura 5.6.

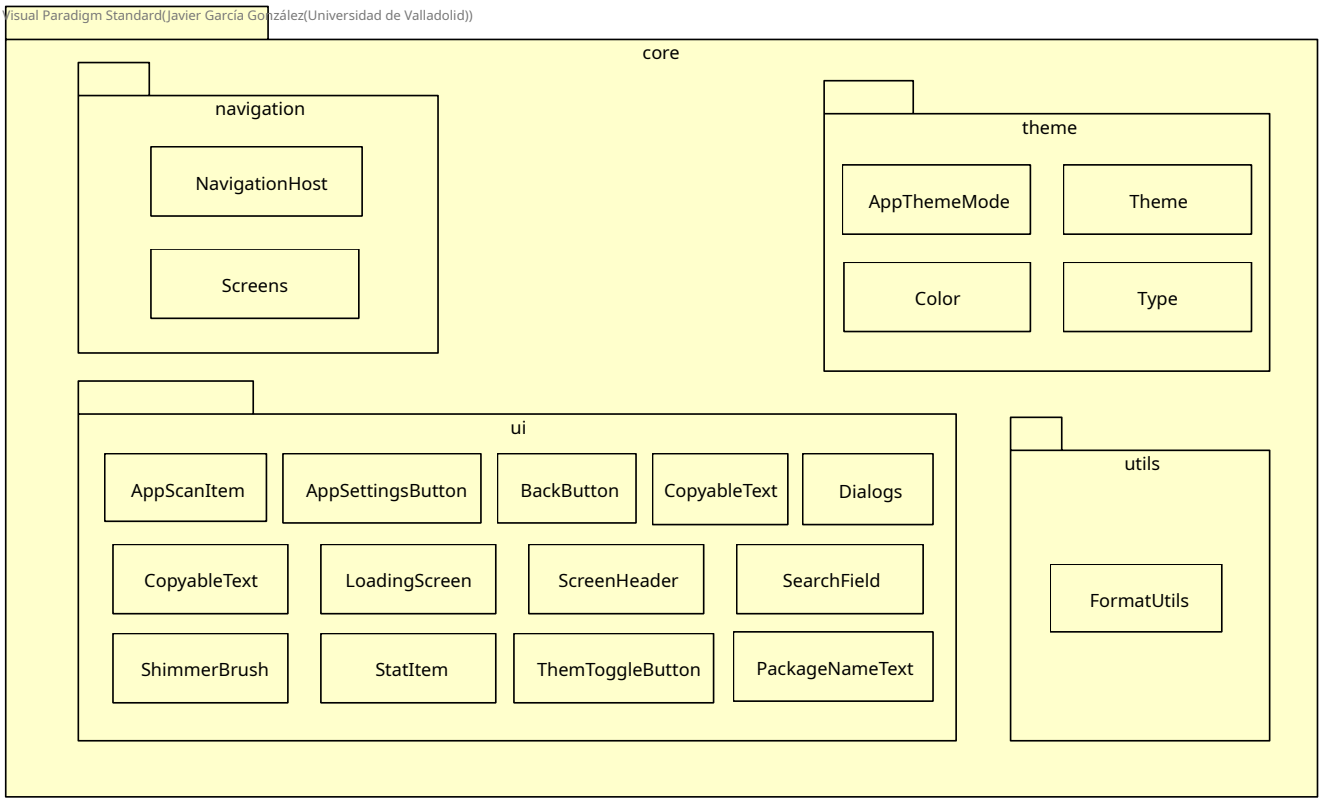


Figura 5.7: Diagrama de clases del paquete *core*

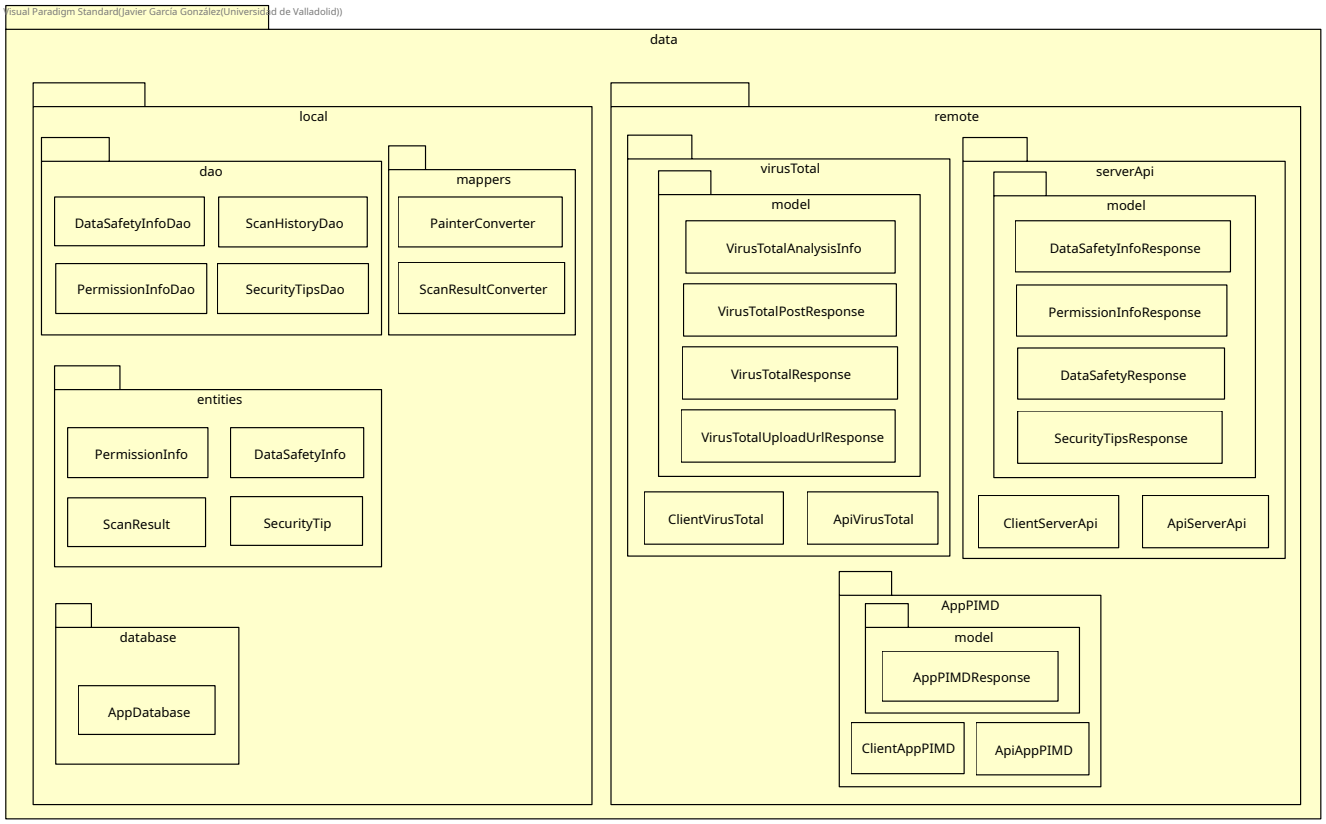


Figura 5.8: Diagrama de clases del paquete *data*

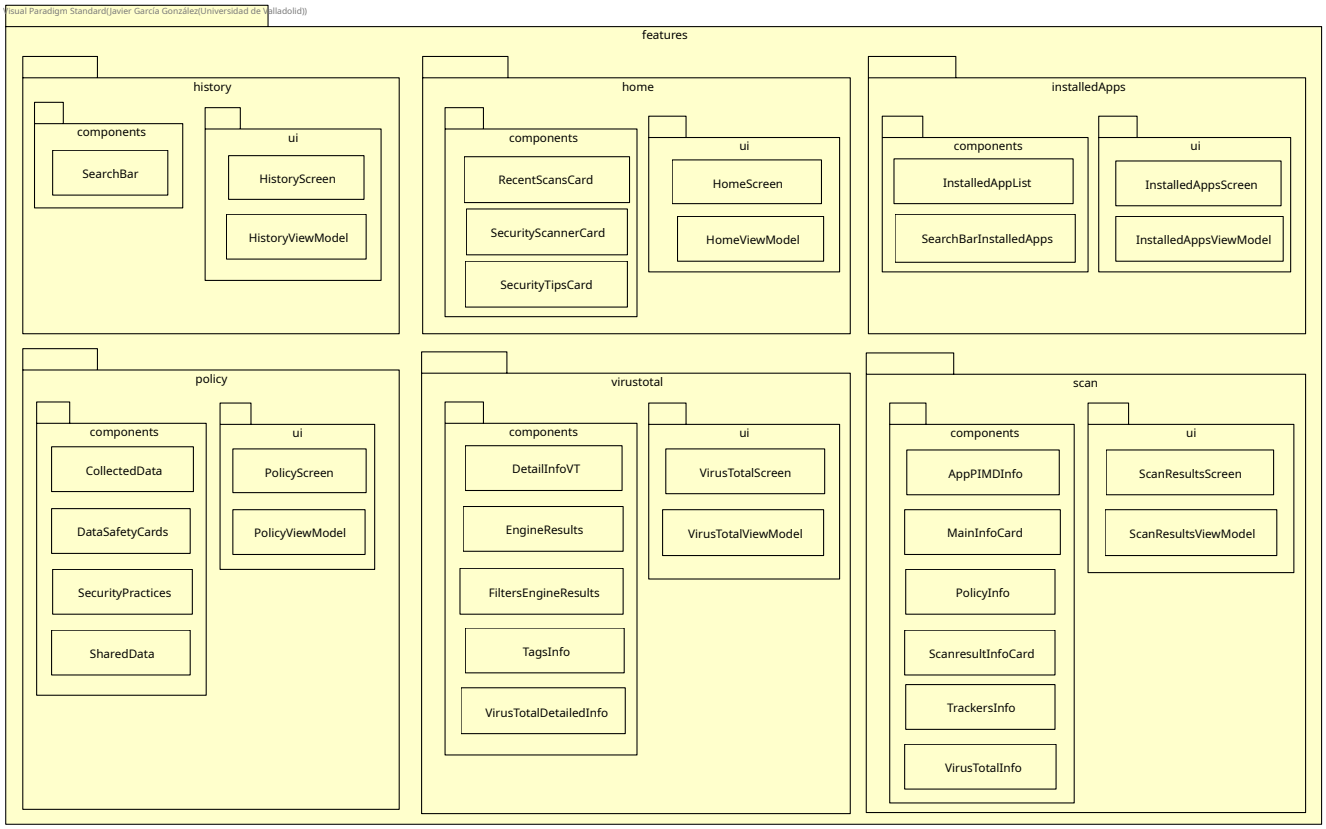


Figura 5.9: Diagrama de clases del paquete *features*

5.3. Arquitectura del servidor

En esta sección se describe la arquitectura que sigue el servidor de procesamiento y los patrones de diseño que se han utilizado y el motivo de su elección.

5.3.1. Arquitectura basada en microservicios

La arquitectura del servidor sigue un enfoque basado en microservicios con contenedores. Cada componente del sistema es un microservicio independiente que se ejecuta en su propio contenedor, lo que facilita la escalabilidad, la gestión y el aislamiento de responsabilidades.

Además, se emplea el patrón arquitectónico **API Gateway** mediante el cual se tiene un punto de entrada único para todas las peitciones entrantes al sistema. Posteriormente, redirige la petición al servicio correspondiente.

La elección del patrón *API Gateway* se fundamenta en los siguientes motivos:

- **Mantenimiento**: Mejora la mantenibilidad del sistema al proporcionar un punto centralizado de gestión de las peticiones entrantes.
- **Escalabilidad**: Permite realizar las modificaciones y evoluciones oportunas a cada uno de los microservicios sin afectar a la manera en la que es llamado.

- **Seguridad**: Permite establecer mecanismos de autenticación y autorización en un lugar único evitando, de esta manera, duplicar la lógica de seguridad en los microservicios.

En la figura 5.10 se muestra el funcionamiento del patrón *API Gateway*, empleado en el proyecto con el fin de establecer un mecanismo de gestión de las comunicaciones realizadas entre la aplicación y el servidor.

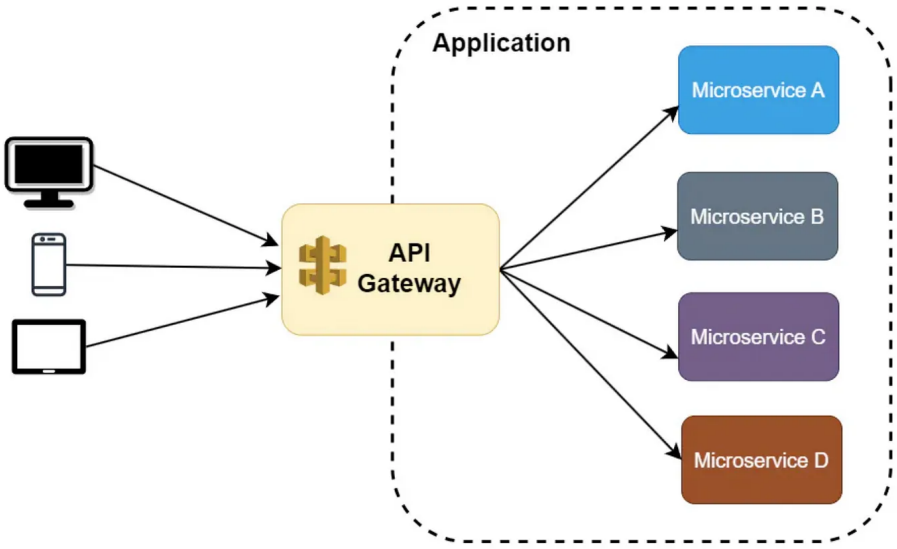


Figura 5.10: Patrón arquitectónico API Gateway[4]

5.3.2. Separación de intereses

El patrón de diseño **Separación de intereses** o, en ingles, *Separation of concerns (SoC)* es un principio fundamental que busca dividir un sistema en partes distintas, donde cada una de las partes aborda una responsabilidad o "interés"específico. Se aplica en el desarrollo del servidor para separar el sistema en las siguientes partes:

- **Routes**: Maneja el enrutamiento de las peticiones entrantes a la API del servidor.
- **Features**: Contiene la lógica de negocio correspondiente a cada una de las funcionalidades principales de la API del servidor.
- **Manejador de errores**: Se separa en un módulo aparte la lógica que se sigue para el tratamiento y manejo de los errores que se producen.
- **Logging**: Se establece un módulo para la configuración y gestión de los logs que se generan durante el funcionamiento de la API del servidor.
- **API Key**: Se separa la lógica de seguridad de acceso a los endpoints de la API.

[4]Imagen tomada de [37]

5.3.3. Data Transfer Object (DTO)

El **Data Transfer Object (DTO)** es un patrón de diseño utilizado para transferir datos entre diferentes capas de una aplicación o sistema. En el servidor de procesamiento, se utiliza para procesar y devolver el resultado del análisis de la política de privacidad de una aplicación.

Mediante el DTO se define la manera en la que los datos serán enviados por la red y representados en las respuestas de la API del servidor. El uso de DTOs aporta múltiples beneficios como:

- **Encapsulamiento de datos**: Permite definir y agrupar los datos que se quieren enviar y/o exponer a los clientes, evitando la exposición de detalles internos de la aplicación.
- **Desacoplamiento**: Favorece la separación de las distintas capas del sistema reduciendo, de esta manera, la dependencia entre las estructuras de datos internas.
- **Seguridad**: Permite omitir campos sensibles o irrelevantes en las respuestas que se devuelven hacia el cliente.

A continuación, en la figura 5.11 se muestra el funciomiento del patrón de diseño *Data Transfer Object* utilizado en el servidor de procesamiento.

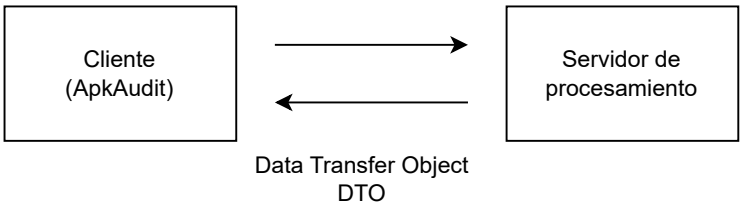


Figura 5.11: Patrón de diseño DTO

5.3.4. Arquitectura

La arquitectura del servidor se estructura en componentes claramente diferencias. Por una parte, se tiene el API Router, que es el componente encargado de gestionar el enrutamiento de las solicitudes hacia los controladores correspondientes. La autenticación se realiza mediante el componente *API Key Auth* que verifica que la clave API introducida sea la correcta. La lógica de negocio se encuentra encapsulada en los *Use Case Controllers*.

Dependiendo de la funcionalidad a realizar en los controladores, se crea un modelo que devuelve ficheros estáticos almacenados en el repositorio local del servidor o bien, se emplea el *Sistema Integrador* que realiza las llamadas a las distintas fuentes de datos externas para obtener la información necesaria para formular la respuesta al cliente.

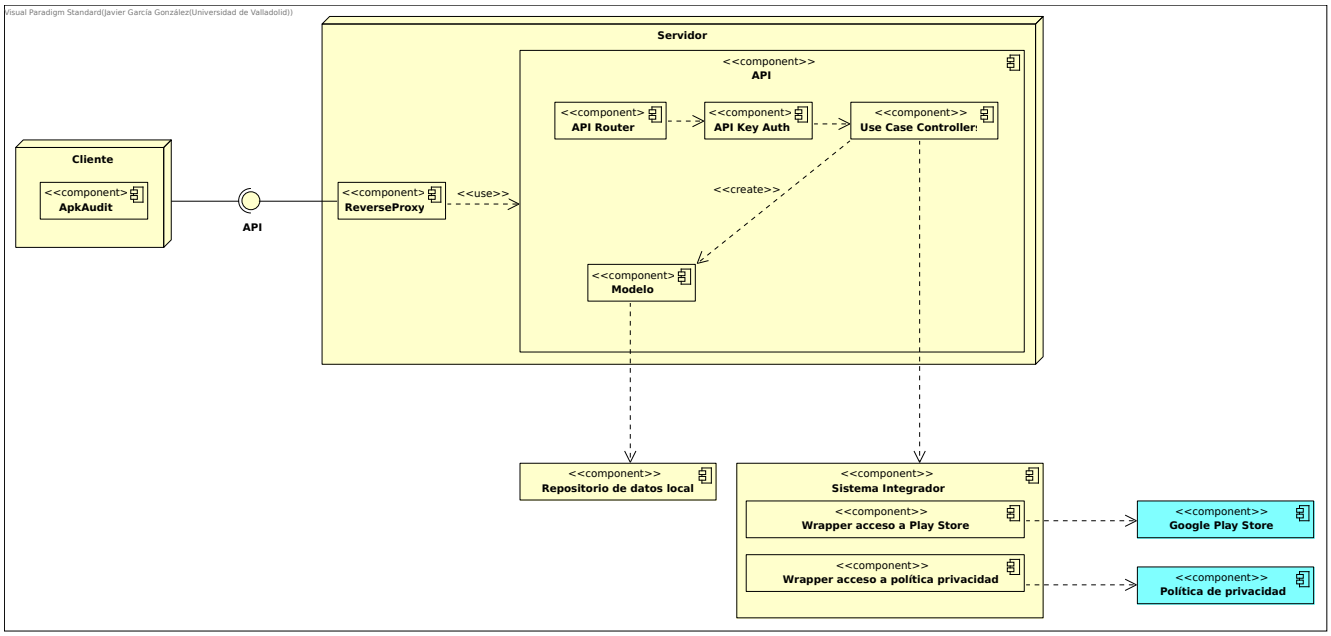


Figura 5.12: Diagrama de arquitectura del servicio del servidor de procesamiento

En la figura 5.12 se muestra el diagrama de arquitectura del servicio que realiza el servidor de procesamiento. Como se puede observar, el cliente, en este caso la aplicación *ApkAudit*, hace uso de este servidor mediante la interfaz proporcionada por el API Gateway. Además, las fuentes de datos externas se representan en otro color para poder ser diferenciadas del repositorio local del servidor.

5.4. Flujo de trabajo del servidor

En este apartado, se indicar el flujo de proceso para la obtención del análisis de la política de privacidad de una aplicación que realiza el servidor de procesamiento de los datos.

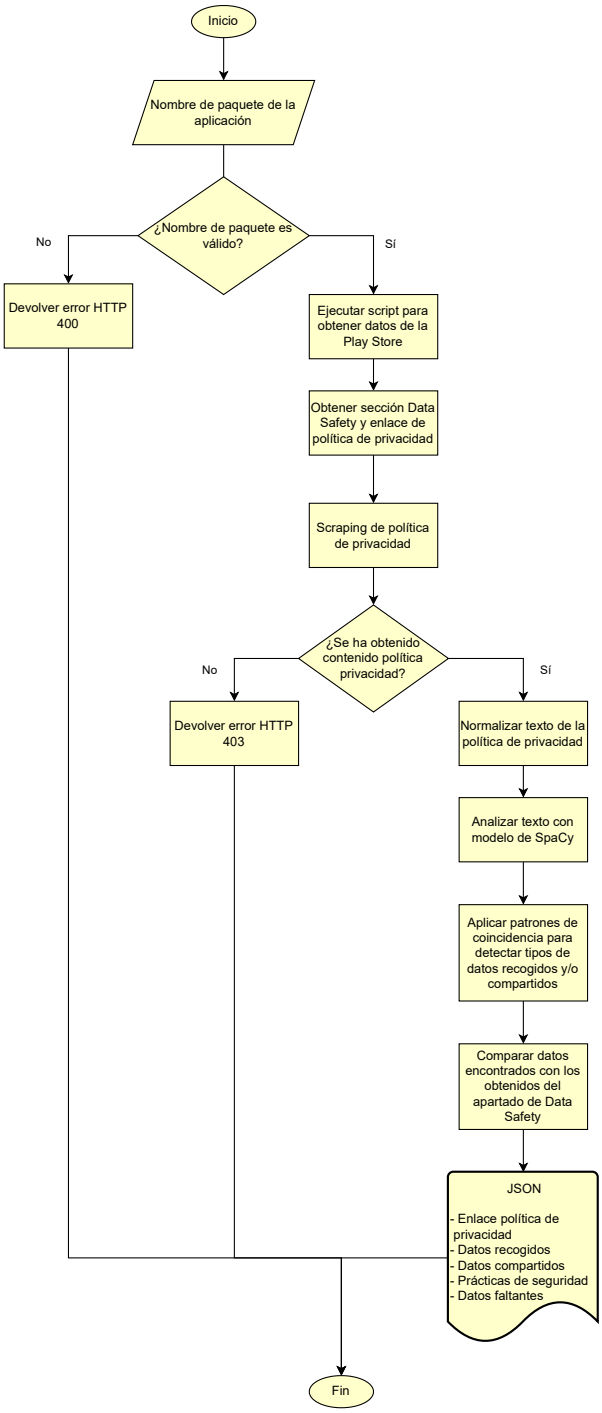


Figura 5.13: Diagrama de flujo de proceso del servidor para la obtención del análisis de la política de privacidad de una aplicación

Como se puede observar en el diagrama de flujo 5.13 se recibe como entrada el nombre de paquete de la aplicación a analizar y, posteriormente, se obtienen los datos de la Play Store los cuales dan como resultado el apartado *Data Safety* de la aplicación. En este apartado se encuentra los datos que recoge y comparte la aplicación, las prácticas de seguridad empleadas y, por último, el enlace

a la política de privacidad.

A partir del enlace de la política de privacidad se hace *scraping* para obtener el texto y así, con patrones de coincidencia, basados en los tipos de datos básicos que se pueden recoger, se obtiene los datos que se estarían recopilando a partir de los usuarios según la política de privacidad. Finalmente, se devuelve un documento JSON con el análisis completo de la política de privacidad de la aplicación.

Aunque, como se muestra en el siguiente apartado, existen otros 3 endpoints la funcionalidad de estos es básica debido a que devuelven ficheros estáticos que se encuentran en el servidor y no se necesita de ningún tipo de procesamiento.

5.5. Diseño detallado de la API del servidor

Se diseña la *api* según la funcionalidad que aporta. Por tanto, se utiliza el estándar *OpenAPI* para documentar el diseño de la misma. Se detallan cada uno de los endpoints que se exponen en la *api* junto con el tipo de petición (GET, POST, etc.) y los parámetros que se reciben en la petición y los códigos de respuesta que se devuelven en la respuesta.

Autenticación mediante cabecera `x-api`

Todas las peticiones a esta API requieren una cabecera HTTP con una clave de autenticación. La decisión de incluir esta cabecera es para evitar posibles ataques y que, de este modo, solo se puedan recibir peticiones a la API desde la aplicación.

Nombre	Tipo	Descripción
`x-apikey`	string	Clave de autenticación proporcionada por el sistema. Debe incluirse en todas las peticiones como cabecera HTTP.

Tabla 5.5: Cabecera obligatoria para autenticar peticiones

Ejemplo de uso:

```
GET /data-safety-info.json HTTP/1.1
Host: api.ejemplo.com
x-api: clave_api
```

Listing 5.1: Ejemplo de llamada al endpoint `/data-safety-info.json`.

En este caso, la clave de autenticación se ha generado mediante el siguiente *script*[5] en *Python* y es igual para todos los usuarios de la aplicación.

```
import secrets
```

[5]Script: archivo de texto que contiene un código ejecutable

```
api_key = secrets.token_urlsafe(32)
print(api_key)
```

Listing 5.2: Script en *Python* para generar la API Key.

GET */get/compare-privacy*

Descripción: petición síncrona que solicita la obtención del estudio de la política de privacidad de una aplicación dada por su *packageName*.

Parámetros:

Nombre	Tipo	Descripción	Opcional
packageName	string	*packageName* de la aplicación de la que se desea obtener el estudio de la política de privacidad	No

Tabla 5.6: Parámetros del endpoint `/get/compare-privacy`

Respuestas:

Código	Descripción
200	Se ha podido realizar el estudio de la política de privacidad y se devuelve respuesta en formato JSON
400	Error: El *packageName* es inválido
401	Error: la petición no está autorizada
403	Error: No se puede acceder a la política de privacidad
404	Error: El *packageName* no se ha encontrado

Tabla 5.7: Respuestas del endpoint `/get/compare-privacy`

GET */get/security-tips*

Descripción: petición síncrona que solicita la descarga de un fichero con varios consejos de seguridad a la hora de instalar aplicaciones en Android.

Parámetros:

Nombre	Tipo	Descripción	Opcional
language	string	Indica el idioma en el que se quiere descargar el recurso. Por defecto `"es"`. `["es", "en"]`	Sí

Tabla 5.8: Parámetros del endpoint `/get/security-tips`

Respuestas:

Código	Descripción
200	Se devuelve el fichero de respuesta en formato JSON
401	Error: la petición no está autorizada
422	Error: *language* inválido

Tabla 5.9: Respuestas del endpoint `/get/security-tips`

GET */permissions.json*

Descripción: petición síncrona que solicita la descarga de todas las descripciones de los permisos en Android en base a su documentación oficial [36] en un fichero estático.

Parámetros:

Nombre	Tipo	Descripción	Opcional
language	string	Indica el idioma en el que se quiere descargar el recurso. Por defecto `"es"`. `["es", "en"]`	Sí

Tabla 5.10: Parámetros del endpoint `/permissions.json`

Respuestas:

Código	Descripción
200	Se devuelve el fichero de respuesta en formato JSON
401	Error: la petición no está autorizada
422	Error: *language* inválido

Tabla 5.11: Respuestas del endpoint `/permissions.json`

GET */data-safety-info.json*

Descripción: petición síncrona que solicita la descarga de la descripción de los tipos de datos definidos para el apartado del *Data Safety* de la *Google Play Store* [38].

Parámetros:

Nombre	Tipo	Descripción	Opcional
language	string	Indica el idioma en el que se quiere descargar el recurso. Por defecto `"es"`. `["es", "en"]`	Sí

Tabla 5.12: Parámetros del endpoint `/data-safety.json`

Respuestas:

Código	Descripción
200	Se devuelve el fichero de respuesta en formato JSON
401	Error: la petición no está autorizada
422	Error: *language* inválido

Tabla 5.13: Respuestas del endpoint `data-safety-info.json`

5.6. Tipos de datos básicos del *Data Safety*

La sección *Data Safety* o sección de seguridad de los datos de *Google Play* ofrece a los desarrolladores una forma transparente de mostrar a los usuarios si recogen, comparten y protegen los datos de los usuarios [17]. Para ello, en esta sección *Google* define una serie de tipos de datos básicos los cuales son aquellos que pueden estar involucrados en el tratamiento de información dentro de una aplicación. A continuación, se detallan algunos de estos tipos de datos básicos que se pueden encontrar en [19]:

Categoría	Tipo de datos
Ubicación	Ubicación aproximada
	Ubicación precisa
Información personal	Nombre
	Dirección de correo
	IDs de usuario
	Dirección
	Número de teléfono
	Raza y etnia
	Opiniones políticas o creencias
	Orientación sexual
	Otra información
Información financiera	Información de pago del usuario
	Historial de compras
	Calificación crediticia
	Otra información financiera
Contacto	Contactos
Fotos y vídeos	Fotos
	Vídeos

Tabla 5.14: Algunos de los tipos de datos básicos del *Data Safety* de *Google Play*.

Ante este punto, es el desarrollador quien debe establecer cuales son de estos tipos de datos básicos los que recoge, comparte o protege durante el uso de su aplicación.

En este proyecto, estos tipos de datos básicos se utilizan como referencia para realizar el análisis

del correcto cumplimiento de las políticas de privacidad de las aplicaciones. Para ello, se emplea un modelo preentrenado de procesamiento de lenguaje natural (NLP) proporcionado por *SpaCy*, junto con una serie de patrones léxicos definidos manualmente para detectar cualquier mención relevante de estos tipos de datos en el texto de las políticas de privacidad de las aplicaciones.

Una vez realizado este análisis de la política de privacidad se obtiene un informe con el listado de tipos de datos básicos detectados en la política de privacidad de la aplicación. A continuación, se compara con el informe declarado en el apartado del *Data Safety* de *Google Play* para determinar la existencia de posibles discrepancias entre los datos declarados en la política de privacidad y los datos declarados en el *Data Safety* de *Google Play*.

5.7. Diagramas de secuencia en diseño

A continuación, se muestran los diagramas de secuencia a nivel de diseño para los casos de uso *CU01*, *CU02* y *CU03*. Estos diagramas se emplean para mostrar como se relacionan los componentes del sistema y los patrones de diseño utilizados en el sistema.

5.7.1. CU01: Analizar aplicación instalada

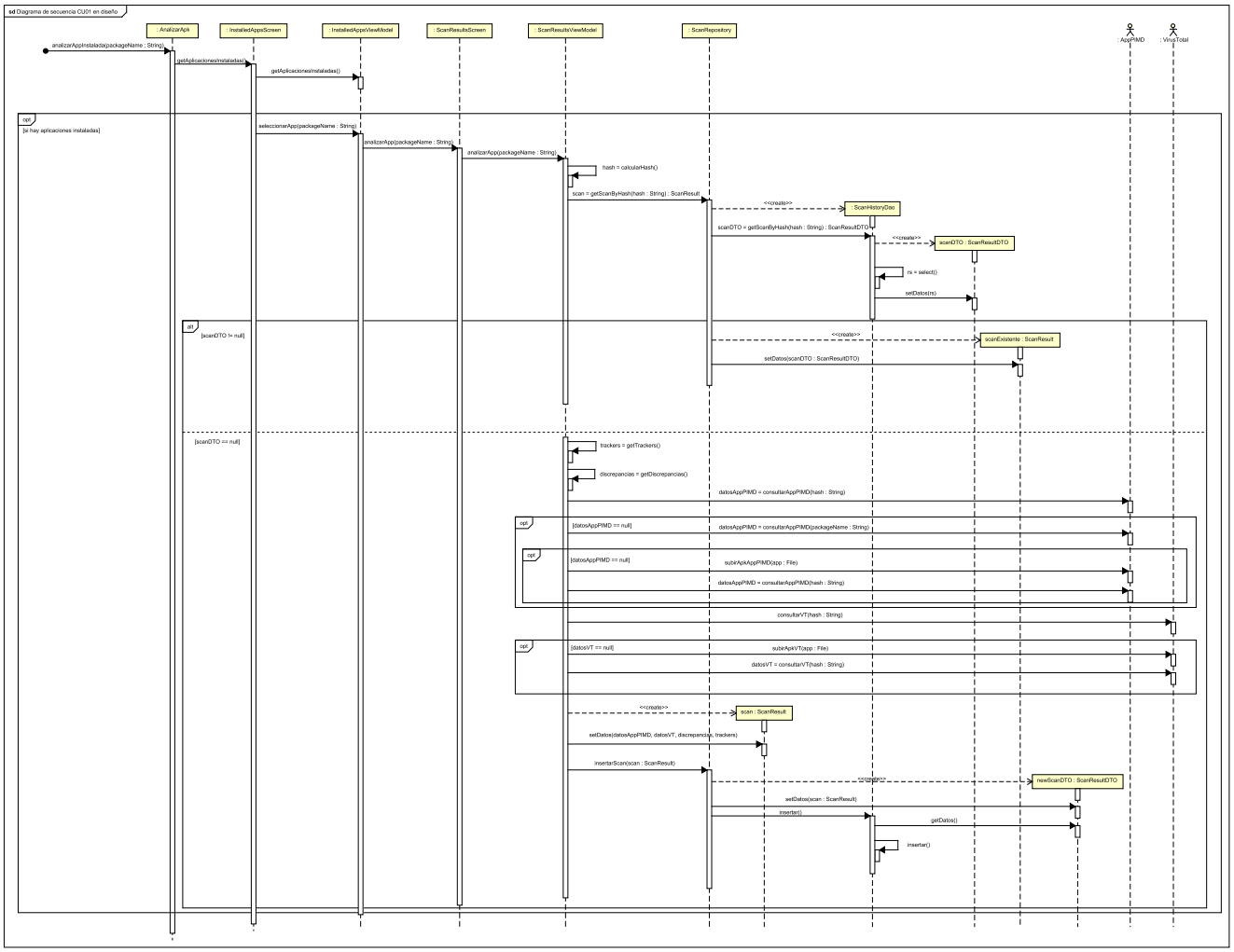


Figura 5.14: Diagrama de secuencia a nivel de diseño del caso de uso *CU01*

En la figura 5.14 se muestra el diagrama de secuencia a nivel de diseño del caso de uso *CU01*. En este diagrama se puede ver que, en primer lugar, se muestran las aplicaciones instaladas en el dispositivo para ello, primero se accede al `InstalledAppsScreen` y este, a su vez, solicita al `InstalledAppsViewModel` la carga de la lista de aplicaciones instaladas.

Posteriormente, se selecciona una aplicación y se comunica al `ScanResultsViewModel`. Este *ViewModel* solicita al `ScanRepository` los datos del análisis de la aplicación. El *ScanRepository* mediante el `ScanHistoryDao` solicita el análisis en base al hash del fichero de la aplicación correspondiente.

Si se obtiene un resultado se devuelve al `ScanResultsViewModel` y este, a su vez, se comunica al `ScanResultsScreen` para que se muestre el resultado del análisis. De lo contrario, se debe elaborar un reporte nuevo (obtención trackers, resultado del análisis de política de privacidad, respuesta del repositorio de *App-PIMD*, etc) y crear un nuevo registro en la base de datos.

5.7.2. CU02: Analizar un archivo *apk*

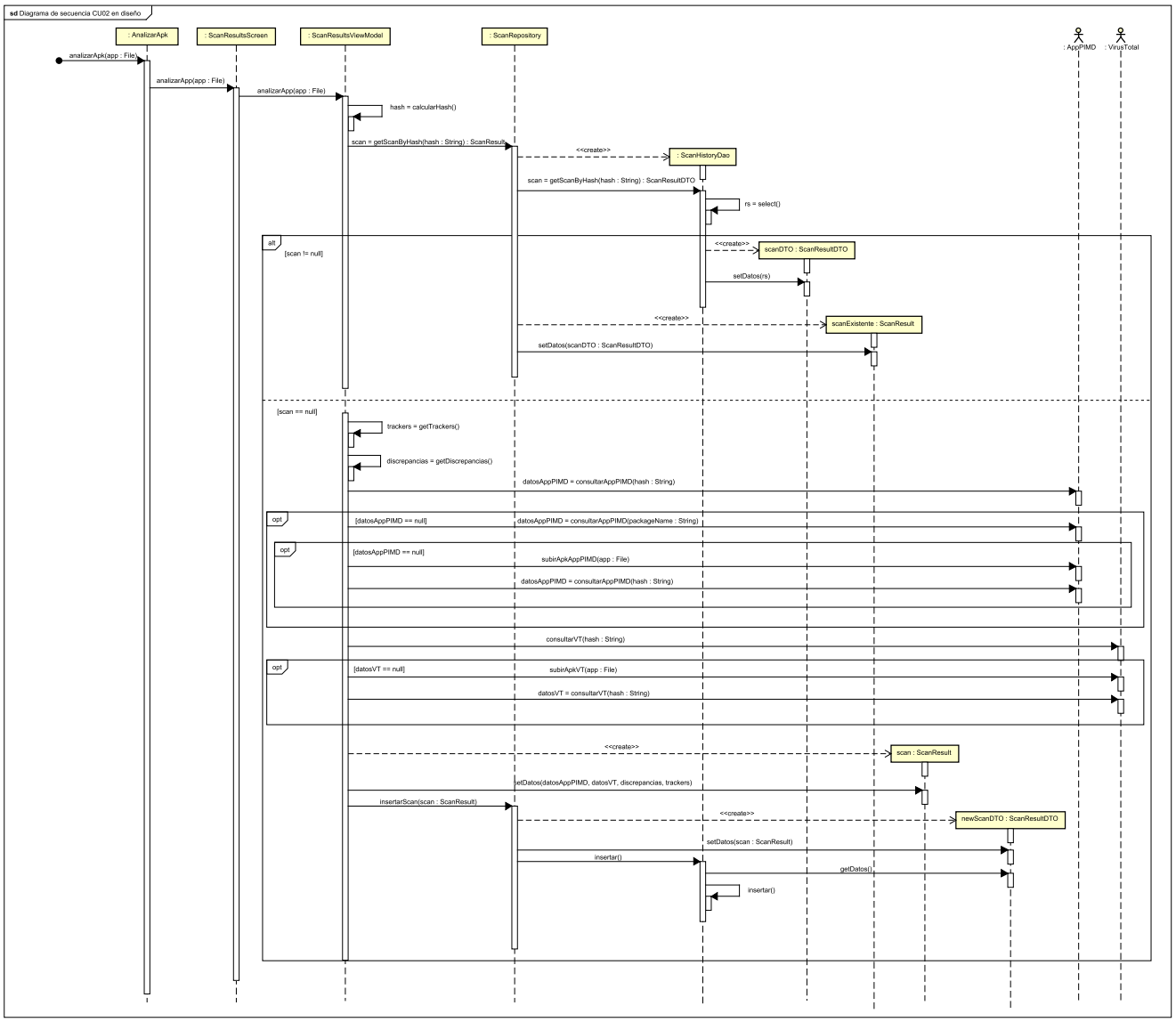


Figura 5.15: Diagrama de secuencia a nivel de diseño del caso de uso *CU02*

En el diagrama de la figura 5.15 se muestra el diagrama de secuencia a nivel de diseño del caso de uso *CU02*. Este diagrama es similar al del caso de uso *CU01* debido a que, en ambos casos, se realiza un análisis de una aplicación. Sin embargo, en este caso, se origina desde un fichero *apk* y no de una aplicación instalada en el dispositivo.

En consecuencia, no se accede al `InstalledAppsScreen` ni se solicitan las aplicaciones instaladas en el dispositivo al `ScreenViewModel`.

5.7.3. CU03: Acceder análisis ya realizado

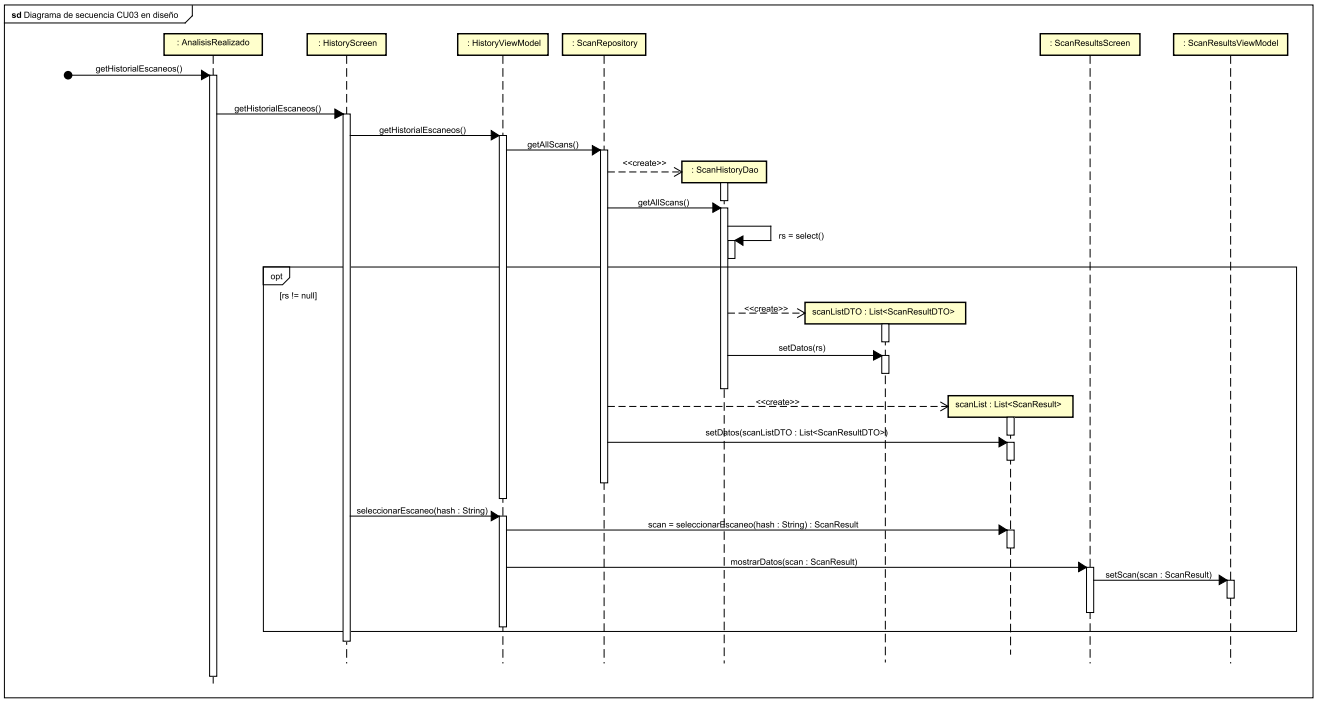


Figura 5.16: Diagrama de secuencia a nivel de diseño del caso de uso *CU03*

En la figura 5.16 se muestra el diagrama de secuencia a nivel de diseño del caso de uso *CU03*. En este diagrama se muestra como se accede a un análisis ya realizado. Para ello, se accede a `HistoryScreen` quien mostrará la lista de análisis realizados en la aplicación. En primer lugar, se comunica al `HistoryViewModel` que solicite la lista de análisis realizados al `ScanRepository` quien, a su vez, solicita al `ScanHistoryDao` la lista de análisis realizados.

El `ScanRepository` obtiene la lista de `ScanResultsDTO`, estos son los objetos entidad de la base de datos (En el diagrama se omite el *loop* para crear cada DTO de la lista por motivos de simplicidad y claridad en el diagrama). Tras obtener la lista de DTO, el `ScanRepository` los convierte a objetos de tipo `ScanResults` y se devuelven al `HistoryViewModel`. Este *ViewModel* los comunica al `HistoryScreen` para que se muestre la lista de análisis realizados. Finalmente, se selecciona un análisis de todos los mostrados y se comunica al `ScanResultsViewModel` para que se muestre el resultado del análisis.

5.8. Diagrama de despliegue

El despliegue de la aplicación está formado por dos componentes principales: el servidor de procesamiento y la aplicación móvil *ApkAudit*. Sin embargo, también se realizan consultas a dos APIs externas: la API del repositorio de *App-PIMD* y la API de *VirusTotal*.

Respecto a la aplicación se ejecuta exclusivamente en dispositivos con sistema operativo Android y

utiliza almacenamiento local a través de la libreria *Room*, que proporciona una capa abstracción sobre la base de datos *SQLite*. La aplicación *ApkAudit* realiza llamadas a las diversas APIs externas, representadas en la figura 5.17 mediante una interfaces debido a que su implementación no se detalla en este contexto. Además, se comunica con la API que se expone en el servidor de procesamiento, donde se muestran los distintos componentes que la conforman.

El servidor de procesamiento tiene el sistema operativo *Ubuntu Server*. La comunicación con la API que expone el servidor de procesamiento se realiza mediante el protocolo *HTTP* y la petición la recibe un contenedor *Docker* que ejecuta un *Reverse proxy* quien redirige la petición a la API ejecutandose con un contenedor *Docker* ejecutando el servicio *Uvicorn*.

En la figura 5.17 se muestran los distintos componentes que conforman el despliegue de la aplicación.

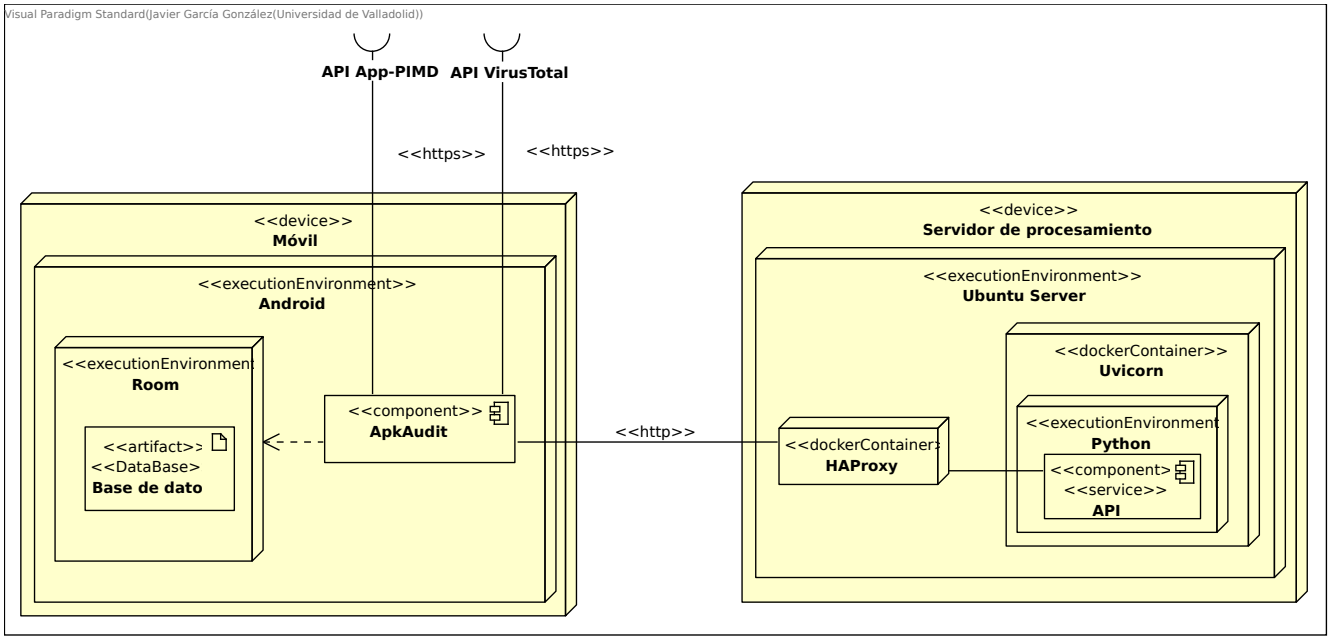


Figura 5.17: Diagrama de despliegue

5.9. *Mock-ups* de la interfaz de usuario

En este apartado se incluyen los *mock-ups* de la interfaz de usuario de la aplicación *ApkAudit* realizados con la herramienta *Figma*. Para ello se muestran distintas pantallas cada una con una breve descripción de su funcionalidad.

5.9.1. Pantalla principal

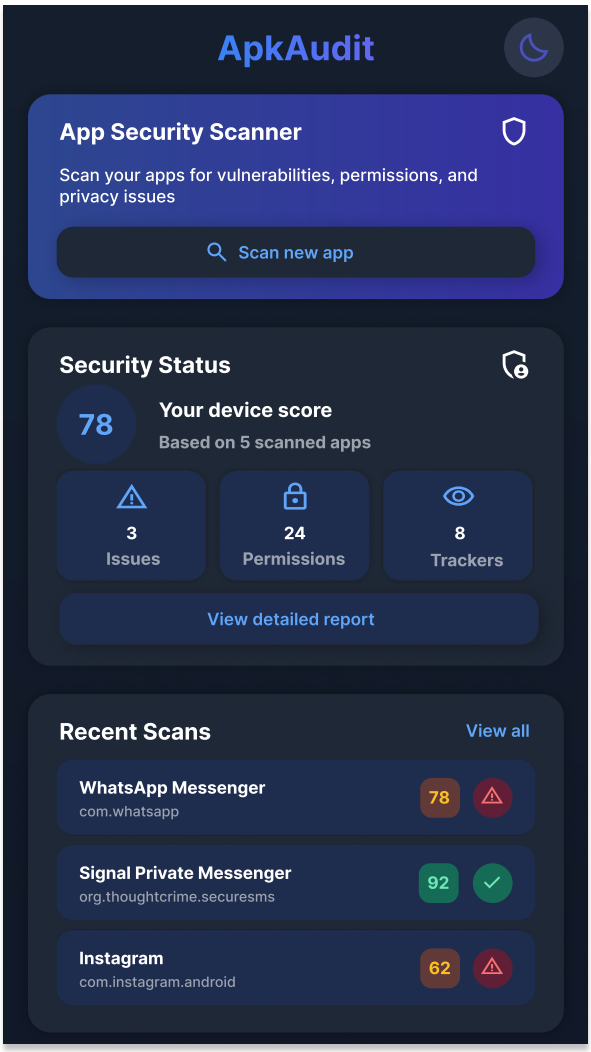


Figura 5.18: Pantalla principal de la aplicación

En la figura 5.18 se muestra la pantalla principal de la aplicación. En esta pantalla se da la opción al usuario de realizar un nuevo escaneo mediante el botón *Scan new app*. Además, se muestra el estado de la seguridad del móvil según las aplicaciones que se han escaneado desde *ApkAudit*. Por último, se muestra el historial de escaneos realizados anteriormente.

5.9.2. Pantalla de historial de escaneos

Figura 5.19: Pantalla de historial de escaneos

En la figura 5.19 se muestra la pantalla de historial de escaneos. En esta pantalla se muestra un listado con las aplicaciones que han sido escaneadas usando *ApkAudit*. Para cada aplicación se muestra su nombre, fecha de escaneo y estado de seguridad. El usuario puede seleccionar una aplicación para ver más detalles.

Además, se ofrece la posibilidad de buscar entre las aplicaciones escaneadas según distintos filtros y por nombre.

5.9.3. Pantalla de resultado de un escaneo

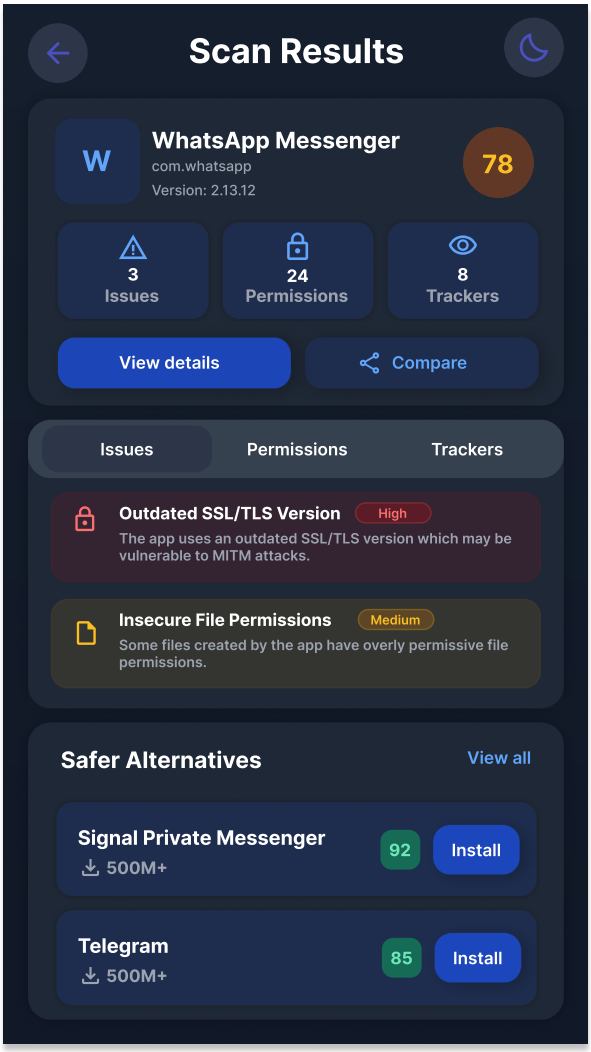


Figura 5.20: Pantalla de resultado de un escaneo

En la figura 5.20 se muestra la pantalla de resultado de un escaneo. Esta pantalla se muestra una vez se ha elegido la aplicación a escanear y se ha terminado de procesar la información relativa a su privacidad, permisos y seguridad. Finalmente, se muestra el resultado de este procesamiento y se ofrece alternativas si las hubiera a esta aplicación.

Capítulo 6

Implementación

En este capítulo se abordan detalles y aspectos técnicos relacionados con la implementación del proyecto. Desde las tecnologías y herramientas utilizadas, hasta aspectos como los principales retos y dificultados que se han encontrado durante su desarrollo.

Todo el código correspondiente al proyecto se encuentra en los siguientes repositorios:

- **Servidor de procesamiento**: `https://gitlab.inf.uva.es/jagarci/apkaudit-server`
- **Aplicación móvil *ApkAudit***: `https://gitlab.inf.uva.es/jagarci/apkaudit`

6.1. Entorno de desarrollo

Las herramientas que se han utilizado para el desarrollo de la aplicación móvil ha sido el IDE *Android Studio* en el cual se ha programada con el lenguaje de programación *Kotlin*. Mediante este IDE se pueden poner a prueba las funcionalidades que se desarrollan gracias a la emulación de dispositivos móviles.

Al emular dispositivos móviles se ha podido comprobar que la aplicación funciona correctamente en diferentes dispositivos y versiones de Android. Además, esta funcionalidad es especialmente útil para garantizar el *Responsive Design*[1] de la aplicación.

Se ha utilizado el siguiente plugin en *Android Studio*:

- *kdoc-generator*: Es un plugin que facilita la creación de documentación *Kdoc* para las funciones y clases en *Kotlin* [39].

Por otro lado, para el desarrollo del servidor de procesamiento se ha empleado el IDE *Visual Studio Code* con el plugin siguiente:

- *Remote - SSH*: Permite abrir una carpeta remota a través de SSH. Un plugin que es muy útil para desarrollar a través del IDE en un servidor remoto.

[1]Responsive Design: formato de programación que permite ajustar una aplicación automáticamente al tamaño y disposición de los dispositivos de sus usuario.

Se ha desarrollado en el sistema operativo *Ubuntu 24.04.2 LTS*. Además, para comprobar el correcto funcionamiento de las APIs de las que se hace uso en el proyecto y el formato de los datos que se devuelven, se ha utilizado la herramienta *Postman*. Esta herramienta permite crear peticiones de cualquier tipo (GET, POST, PUT, DELETE, etc.) y enviar dichas peticiones a los distintos *endpoints* de las APIs que se quieran probar [40].

6.2. Entorno tecnológico

En este apartado se listan las tecnologías y herramientas que se utilizan para la implementación del proyecto en su totalidad.

- **Servidor de procesamiento**:

  1. *Python*: Se utiliza Python como lenguaje de programación para implementar la API y sus endpoints. Se ha optado por este lenguaje debido a su simplicidad, el gran ecosistema de librerías disponibles y la comunidad activa que la soporta. Versión: 3.11
  2. *FastAPI* y *Uvicorn*: *FastAPI* es un framework que permite la creación de APIs Restful de manera rápida y sencilla. *Uvicorn* es un servidor web ASGI (Asynchronous Server Gateway Interface) que permite la ejecución de aplicaciones web en Python de manera asíncrona [41].
  3. *JavaScript* y *Node.js*: Para uno de los endpoints se ejecuta un script de *JavaScript* que se encarga de realizar *scraping*[2] de datos a la aplicación solicitada de la *Google Play Store*. Para ello, se utiliza *Node.js* como entorno de ejecución y *JavaScript* como lenguaje de programación.
  4. *Google Play Scraper*: Se utiliza esta libreria para realizar el *scraping* de datos de la *Google Play Store* y obtener, en este caso, la política de privacidad de la aplicación.
  5. *HAProxy*: Es un *Reverse Proxy* y *Load Balancer* que, en este caso, se utiliza para enrutar el tráfico de la aplicación móvil hacia la API desplegada con *Uvicorn*. De este modo, los usuarios no necesitan conocer la dirección IP y el puerto en el que se encuentra la API para poder acceder a ella. Con esto se añade una mayor seguridad, control y permite establecer SSL de una manera sencilla. Versión: 3.2.0
  6. *Docker* y *Docker compose*: Se emplean para contenerizar los distintos servicios que componen el servidor con tal de facilitar el despliegue y tener cierto aislamiento. Versión: 28.1.1 y Versión: 2.35.1

- **Aplicación móvil *ApkAudit***:

  1. *Kotlin*: Es un lenguaje de programación moderno utilizado para desarrollar aplicaciones nativas para Android. Utilizado por el 60 % de los desarrolladores de aplicaciones

[2]Proceso de extraer datos de sitios web de forma automatizada

Android [35]. Versión: 2.0.21

2. *Jetpack Compose*: Es un framework declarativo de UI que permite construir interfaces de usuario reactivas y modernas de forma sencilla y eficiente.
3. *Dagger Hilt*: Es una biblioteca para la inyección de dependencias que facilita la gestión de los componentes y mejorar sustancialmente la mantenibilidad y calidad del código.
4. *Room*: Es una bibliotecaa que proprociona una abstracción de una base de datos SQLite para almacenar datos localmente en un dispositivo Android.

- **Control de versiones**:

1. *Git*: Se utiliza Git como sistema de control de versiones tanto para el desarrollo de la aplicación móvil como para el desarrollo del servidor de procesamiento.
2. *Gitlab*: Se utiliza para alojar el repositorio de código.

6.3. Documentación

Todas las clases y funciones de la aplicación *ApkAudit* están documentadas utilizando *Kdoc*, el lenguaje usado para documentar código *Kotlin*. Este lenguaje utiliza una sintaxis similar a la de *JavaDoc* que al compilarse genera diversos ficheros HTML que documentan el código. A continuación se muestra un ejemplo de la sintaxis de *Kdoc* en la siguiente función de llamada a la API de *App-PIMD*:

```
/**
 * Realiza una peticion GET al repositorio App-PIMD solicitando el
    reporte
 * de una aplicacion a traves de su packageName
 *
 * @param packageName El packageName de la aplicacion a consultar
 * @return Una respuesta con el reporte de la aplicacion en formato JSON
 */
@GET("/get/app/package")
suspend fun getReportPackageName(
    @Query("package") packageName: String
): Response<AppPIMDResponse>
```

Listing 6.1: Función getReportPackageName

Este código HTML es generado mediante la API *Dokka*, que, además, puede generar dicha documentación en otros formatos como *Markdown* o *PDF*. En la figura 6.1 se muestra un ejemplo de la documentación que genera *Dokka* a partir de este código.

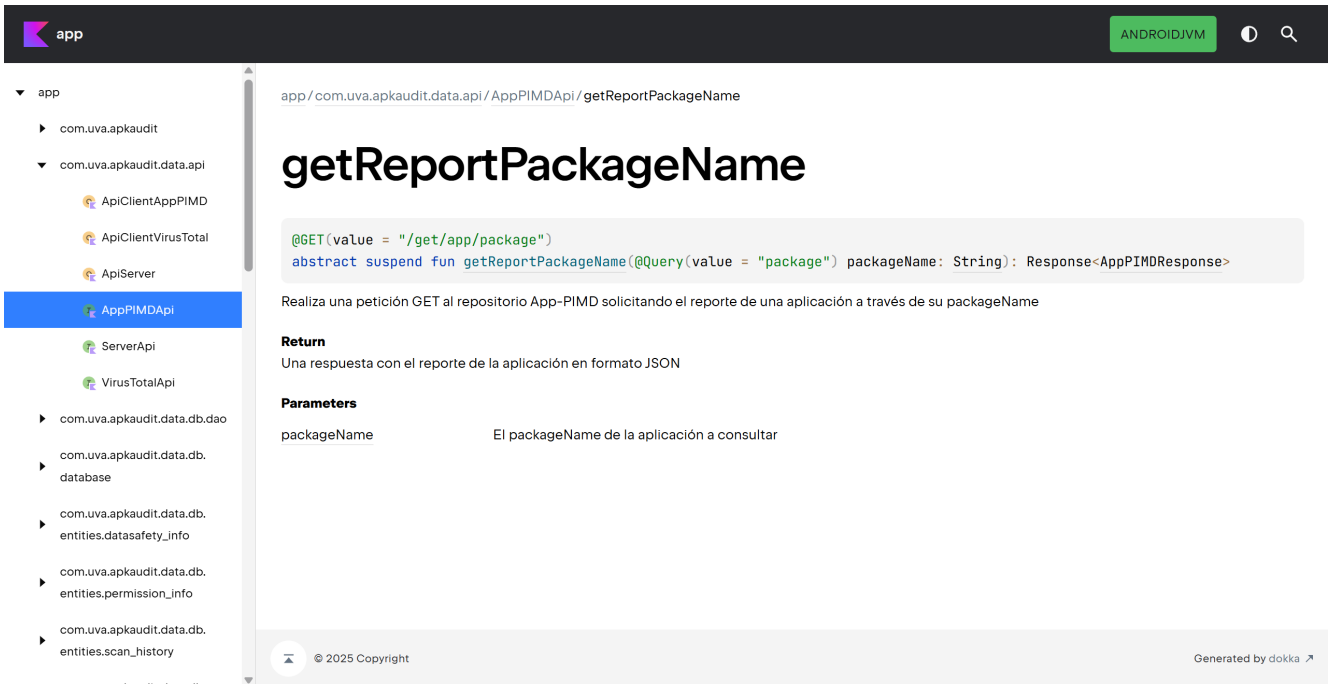


Figura 6.1: Documentación generada por Dokka

A esta documentación se puede acceder en el directorio `app/build/dokka/html` en el repositorio de la aplicación (Véase el Apéndice D).

6.4. Principales dificultades y retos

Durante el periodo de implementación del proyecto se ha encontrado con varios retos y problemas. A continuación se muestran algunos de ellos:

- **Subida de ficheros *APK* a las APIs**: En los casos en los que no se encuentre ningún análisis ya realizado anteriormente en las distintas APIs utilizadas (*App-PIMD* y *VirusTotal*) se procede a subir el fichero al correspondiente endpoint para solicitar el análisis. La dificultad radica en el tratamiento del proceso asíncrono de la subida del fichero y la espera de la respuesta. Además, en la diferente forma que tiene cada API de devolver la información debido a que por ejemplo, en la API de *VirusTotal* en primer lugar tienes que solicitar un enlace de subida, subir el fichero a ese enlace y, posteriormente, comprobar el estado del análisis haciendo *polling*[3] durante un intervalo de tiempo hasta que se obtenga la información.

- **Obtención del reporte de las discrepancias en la política de privacidad**: En el servidor de procesamiento, en el endpoint `/get/compare-privacy` se procesa la política de privacidad con un modelo para obtener los datos que se recopilan del usuario. Se me plantearon numerosos problemas debido a que muchos modelos de los que usaba no eran capaces de obtener los datos que se recopilan con relativa precisión. En primer lugar, probé

[3]Polling: una aplicación envía repetidamente peticiones contra una API hasta que encuentra la información deseada.

con el modelo *all-MiniLM-L6-v2* el cual convierte frases en vectores numéricos para entender su significado sin embargo, los resultados obtenidos no eran precisos. Finalmente, logré adaptar el procesamiento para el uso con el modelo *en_core_web_lg* de *spaCy* con el que comprobé manualmente su correcto funcionamiento.

- **Tratamiento de errores**: Considero también que ha sido un reto el hecho de implementar el tratamiento de errores en la aplicación debido a que he querido tomar una aproximación en la que se avise al usuario de que ha ocurrido un error y que se le pueda mostrar una explicación de lo que ha ocurrido pero, que esto no afecte a la usabilidad y funcionalidad de la misma.
- **Alternativas a una aplicación**: Se propuso, en un principio, ofrecer al usuario la funcionalidad de que pudiese ver alternativas a la aplicación que ha analizado pero enfocadas en una mayor privacidad. Sin embargo, no se pudo extraer dicha información de páginas como `https://alternativeto.net/` debido a su uso de protección *Cloudflare* que evita el scraping. Se trató de usar el módulo de *Python*, *Cloudscraper*[4] pero no se logró obtener la información deseada.

6.5. Privacidad del usuario

En el apartado 2.6 se exponen una serie de riesgos de privacidad que se han encontrado para este proyecto y que puede afectar al usuario. En este apartado, se aborda como se ha mitigado cada uno de ellos.

6.5.1. RP1: Almacenamiento de datos personales

Para mitigar este posible riesgo se minimiza al máximo los datos que se recaban del usuario y, a su vez, en el sistema no existe persistencia más allá de los datos que se almacenan en la memoria del dispositivo del usuario.

Además, los datos que se extraen del usuario y pueden quedar almacenados en servicios de terceros como el repositorio *App-PIMD* y *VirusTotal* solo son los siguientes:

- **Metadatos de la aplicación**: los metadatos de la aplicación que el usuario solicita analizar. Estos son datos como el *packageName*, version, nombre, etc.
- **Fichero *apk***: En los casos de que no haya ninguna aplicación coincidente ya analizada en las fuentes de datos, se procede a subir el fichero *apk* desde el equipo del usuario a la API de *VirusTotal* y/o de *App-PIMD*.

No se usa ningún tipo de tracker ni cookies que puedan rastrear la actividad del usuario en la aplicación.

[4] `https://github.com/venomous/cloudscraper`

6.5.2. RP2: Uso de datos personales

Para abordar este riesgo, como ya se comentó anteriormente en el apartado 5.2.5, unicamente se solicitan al usuario los permisos estrictamente necesarios para el correcto funcionamiento de la aplicación.

- `android.permission.INTERNET`
- `android.permission.ACCESS_NETWORK_STATE`
- `android.permission.QUERY_ALL_PACKAGES`

6.5.3. RP3: Transmisión cifrada de la información

Respecto a la comunicación con las distintas fuentes de información, el protocolo de comunicación que se utiliza es HTTPS debido a que es un protocolo que permite tener un transmisión cifrada de la información que se intercambia entre el cliente y el servidor.

No obstante, este no es el caso en cuanto a la comunicación entre la aplicación móvil y el servidor de procesamiento como se puede observar en la figura 5.17. Esto se debe a que en el servidor se trató de implementar un certificado SSL autofirmado pero Android bloquea la conexión con servidores que utilicen este tipo de certificados por seguridad [42]. Aunque, esto no supone un grave riesgo para la privacidad del usuario debido a que, en la comunicación con el servidor de procesamiento, no se envía ningún tipo de información sensible o dato que pueda identificar al usuario.

Por tanto, la comunicación con el servidor de procesamiento se realiza mediante el protocolo HTTP que envía la información en texto plano.

```
android:usesCleartextTraffic="true"
```

Listing 6.2: Parámetro para activar el uso de HTTP en una aplicación Android

6.5.4. RP4: Procesamiento de datos no consentido

El plan de mitigación que se ha realizado para abordar este riesgo de privacidad ha sido que solo se envíen y analicen las aplicaciones que seleccione manualmente el usuario. Es decir, que no se envíe automáticamente ningún dato sin la previa autorización del usuario.

Capítulo 7

Pruebas

En este capítulo, se muestran los resultados obtenidos de las distintas pruebas realizadas para verificar que la calidad y funcionamiento del proyecto sea el planificado. Para ello se llevan a cabos tres tipos de pruebas:

- Pruebas de aceptación de usuario.
- Pruebas de los endpoints
- Pruebas de usabilidad.

7.1. Pruebas de aceptación de usuario

En este apartado, se describen las pruebas de aceptación de usuario (UAT) realizadas para comprobar el correcto funcionamiento de la aplicación. Además, tienen como objetivo comprobar que el software cumple con los requisitos planteados.

Casos de pruebas

Prueba 01	
Título	Analizar un fichero *.apk*
Descripción	Se selecciona el botón de analizar un fichero *.apk*, se elige el fichero y el sistema analiza el fichero y muestra automáticamente el reporte.
Resultados esperados	Se muestra el reporte correspondiente al fichero *.apk*
Resultado obtenido	Correcto
Comprobación	Se accede al historial de escaneos y se observa que se ha registrado correctamente el análisis del fichero *.apk*
Observaciones	Pueden faltar datos de algunas fuentes y provoquen que se tenga que subir el fichero *.apk* y tarde en procesar.

Tabla 7.1: Prueba UAT 01

Prueba 02	
Título	Seleccionar un fichero que no sea *.apk*
Descripción	Al seleccionar la opción de analizar un fichero *.apk* se abre el panel de selección de archivos del dispositivo y se selecciona un fichero distinto a *.apk*. Por ejemplo, un fichero *.pdf*
Resultados esperados	No permite seleccionar el archivo
Resultado obtenido	Correcto
Comprobación	El panel de selección de archivos sigue activo hasta que se elija un fichero *.apk*
Observaciones	

Tabla 7.2: Prueba UAT 02

Prueba 03	
Título	Listado de aplicaciones instaladas
Descripción	Mostrar las aplicaciones instaladas en el dispositivo.
Resultados esperados	Se muestra un listado de las aplicaciones instaladas.
Resultado obtenido	Correcto
Comprobación	Se muestran las aplicaciones instaladas
Observaciones	Hasta que carga la lista al completo se muestra una animación tipo *shimmer* [1]

Tabla 7.3: Prueba UAT 03

Prueba 04	
Título	Buscar aplicaciones instaladas por nombre
Descripción	Se busca entre el listado de aplicaciones instaladas según su nombre.
Resultados esperados	Muestra la aplicación coincidente con el nombre introducido en la barra de búsqueda.
Resultado obtenido	Correcto
Comprobación	Se muestra la aplicación con un nombre igual o similar al introducido en la barra de búsqueda.
Observaciones	Es posible que tarde si aún no se había procesado la lista de aplicaciones al completo.

Tabla 7.4: Prueba UAT 04

[1] Es una animación que muestra que una pantalla se encuentra cargando mejorando la percepción de fluidez en una aplicación.

Prueba 05	
Título	Buscar aplicaciones por nombre del paquete
Descripción	Se busca entre el listado de aplicaciones instaladas según su nombre de paquete (*packageName*).
Resultados esperados	Muestra la aplicación coincidente con el *packageName* introducido en la barra de búsqueda.
Resultado obtenido	Correcto
Comprobación	Se muestra la aplicación con un *packageName* igual o similar al introducido en la barra de búsqueda.
Observaciones	Es posible que tarde si aún no se había procesado la lista de aplicaciones al completo.

Tabla 7.5: Prueba UAT 05

Prueba 06	
Título	Cambiar a modo oscuro
Descripción	Se da al botón correspondiente y cambia el tema completo de la aplicación para mostrar el modo oscuro.
Resultados esperados	Cambio de tema de la aplicación pasando de modo claro a modo oscuro.
Resultado obtenido	Correcto
Comprobación	El tema de la aplicación ha cambiado al modo oscuro.
Observaciones	

Tabla 7.6: Prueba UAT 06

Prueba 07	
Título	Cambio a modo claro
Descripción	Se da al botón correspondiente y cambia el tema completo de la aplicación para mostrar el modo claro.
Resultados esperados	Cambio de tema de la aplicación pasando de modo oscuro a modo claro.
Resultado obtenido	Correcto
Comprobación	El tema de la aplicación ha cambiado al modo oscuro.
Observaciones	

Tabla 7.7: Prueba UAT 07

Prueba 08	
Título	Analizar aplicación instalada
Descripción	Se analiza una de las aplicaciones del listado de aplicaciones instaladas del dispositivo.
Resultados esperados	Muestra el reporte de la aplicación instalada.
Resultado obtenido	Correcto
Comprobación	Se accede al historial de escaneos y se observa que se ha registrado correctamente el análisis de la aplicación instalada.
Observaciones	Pueden faltar datos de algunas fuentes y provoquen que se tenga que subir el fichero *.apk* y tarde en procesar.

Tabla 7.8: Prueba UAT 08

Prueba 09	
Título	Eliminar un reporte
Descripción	Se deja presionado un item del historial de reportes y, posteriormente, se acepta el cuadro de diálogo.
Resultados esperados	Se ha eliminado la información del reporte del almacenamiento del dispositivo.
Resultado obtenido	Correcto
Comprobación	El reporte ya no aparece en el historial de reportes.
Observaciones	Se muestra un *toast*[2] indicando que se ha eliminado correctamente.

Tabla 7.9: Prueba UAT 09

[2]Toast: aviso proporciona información simple sobre una acción en una pequeña ventana emergente

Prueba 10	
Título	No se elimina el reporte
Descripción	Se deja presionado un item del historial de reportes y, posteriormente, se cancela la eliminación del reporte en cuestión.
Resultados esperados	El reporte no se elimina del almacenamiento del dispositivo.
Resultado obtenido	Correcto
Comprobación	El reporte de la aplicación sigue apareciendo en el historial de reportes.
Observaciones	

Tabla 7.10: Prueba UAT 10

Prueba 11	
Título	Filtrar historial de reportes (seguras)
Descripción	Filtrar historial de reportes de aplicaciones para comprobar que sean seguras.
Resultados esperados	Se muestran los reportes de aplicaciones que son seguras.
Resultado obtenido	Correcto
Comprobación	Solo se muestran reportes de aplicaciones seguras.
Observaciones	

Tabla 7.11: Prueba UAT 11

Prueba 12	
Título	Filtrar historial de reportes (riesgo)
Descripción	Filtrar historial de reportes de aplicaciones para comprobar que sean peligrosas o de riesgo.
Resultados esperados	Se muestran los reportes de aplicaciones que son peligrosas o de riesgo.
Resultado obtenido	Correcto
Comprobación	Solo se muestran reportes de aplicaciones peligrosas o de riesgo.
Observaciones	

Tabla 7.12: Prueba UAT 12

Prueba 13	
Título	Ordenar historial de reportes
Descripción	Ordenar el historial de reportes realizados por fecha de análisis.
Resultados esperados	Se muestra el historial de escaneos ordenado según su fecha de análisis en orden creciente o decreciente.
Resultado obtenido	Correcto
Comprobación	Se muestra los reportes realizados recientemente antes y viceversa.
Observaciones	

Tabla 7.13: Prueba UAT 13

Prueba 14	
Título	Buscar reporte
Descripción	Se busca un reporte en el historial de reportes.
Resultados esperados	Muestra el reporte cuya aplicación tenga un nombre igual o similar al introducido en la barra de búsqueda.
Resultado obtenido	Correcto
Comprobación	Se muestra el reporte con la aplicación con el mismo o similar nombre al introducido
Observaciones	

Tabla 7.14: Prueba UAT 14

Prueba 15	
Título	Permisos de una aplicación
Descripción	Se deben mostrar los permisos que utiliza la aplicación.
Resultados esperados	Se muestran los permisos que utiliza la aplicación, junto con su descripción y nivel de criticidad para la privacidad del usuario.
Resultado obtenido	Correcto
Comprobación	Se muestran los permisos que utiliza la aplicación.
Observaciones	

Tabla 7.15: Prueba UAT 15

Prueba 16	
Título	Trackers de una aplicación
Descripción	Se muestran los trackers presentes en una aplicación.
Resultados esperados	Se muestran, si los hay, los trackers presentes en una aplicación junto con el número total de trackers y una descripción de qué es un tracker.
Resultado obtenido	Correcto
Comprobación	Se muestran los trackers que tiene y/o usa la aplicación.
Observaciones	

Tabla 7.16: Prueba UAT 16

Prueba 17	
Título	Política de privacidad
Descripción	Muestra las discrepancias que hay entre la política de privacidad y el apartado de *Data Safety* de la *Google Play Store*.
Resultados esperados	Se muestran las discrepancias entre la política de privacidad y el apartado de *Data Safety* de la *Google Play Store* si las hay. Además, se muestra el enlace a la política de privacidad.
Resultado obtenido	Correcto
Comprobación	Muestra que datos no son coincidentes o no están reportados en el *Data Safety* pero si en la política de privacidad.
Observaciones	

Tabla 7.17: Prueba UAT 17

Prueba 18	
Título	Vista previa de *VirusTotal*
Descripción	Muestra una vista previa del reporte de *VirusTotal*
Resultados esperados	Se debe mostrar la vista previa de los datos generados por *VirusTotal*
Resultado obtenido	Correcto
Comprobación	Se muestran los antivirus que han detectado como limpio, sospechoso y peligroso a través del reporte generado por la API de *VirusTotal*
Observaciones	

Tabla 7.18: Prueba UAT 18

Prueba 19	
Título	Información completa de *VirusTotal*
Descripción	Se muestran los detalles de la aplicación como nombre de archivo, hashes, fechas de análisis en la plataforma entre otros metadatos.
Resultados esperados	Se muestra la información disponible para cada campo.
Resultado obtenido	Correcto
Comprobación	Se muestra toda la información disponible en los apartados correspondientes.
Observaciones	

Tabla 7.19: Prueba UAT 19

Prueba 20	
Título	Redirección a *Google Play Store*
Descripción	Redirige al usuario a la *Google Play Store* para la aplicación analizada.
Resultados esperados	Se redirige al apartado de la aplicación analizada en la *Google Play Store* si el usuario lo desea.
Resultado obtenido	Correcto
Comprobación	Se abre la *Google Play Store* con la aplicación analizada.
Observaciones	Si la aplicación no está presente en la *Google Play Store* redirige igualmente pero muestra que no se ha encontrado la aplicación.

Tabla 7.20: Prueba UAT 20

Prueba 21	
Título	Ajustes aplicación analizada.
Descripción	Redirige al panel de la aplicación de los ajustes del dispositivo.
Resultados esperados	Se redirige al usuario a la aplicación de ajustes del dispositivo con la aplicación analizada e instalada.
Resultado obtenido	Correcto
Comprobación	Se abre la aplicación de ajustes del dispositivo con la aplicación analizada.
Observaciones	

Tabla 7.21: Prueba UAT 21

Prueba 22	
Título	Reanalizar aplicación
Descripción	Se vuelve a analizar la aplicación.
Resultados esperados	Se actualizan los campos que tengan información nueva de las fuentes de datos.
Resultado obtenido	Correcto
Comprobación	Cambia la información o aparece información nueva si no había previamente. Además, cambia el *timestamp*[3] del reporte.
Observaciones	

Tabla 7.22: Prueba UAT 22

Prueba 23	
Título	Detalles aplicación
Descripción	Muestra los detalles de la aplicación.
Resultados esperados	Se muestran los detalles de una aplicación analizada Estos son: la categoría, versión, nombre de paquete, instalador, nombre e icono.
Resultado obtenido	Correcto
Comprobación	Se muestran los detalles correspondientes a la aplicación analizada.
Observaciones	

Tabla 7.23: Prueba UAT 23

Prueba 24	
Título	Cambio de idioma
Descripción	Se puede cambiar el idioma de la aplicación de inglés a español
Resultados esperados	El idioma del dispositivo debe cambiar al español.
Resultado obtenido	Correcto
Comprobación	Se cambia el idioma del dispositivo y se comprueba que los distintos textos de la aplicación ahora están en español en lugar de, en inglés.
Observaciones	

Tabla 7.24: Prueba UAT 24

[3]Timestamp: secuencia de caracteres que registra la fecha y hora en la que ocurrió un evento específico.

7.2. Prueba de los endpoints

A continuación, se muestran las pruebas que se han realizado a los endpoints que se exponen con la API del servidor de procesamiento para comprobar que devuelven el resultado esperado.

Prueba	Descripción	Resultado esperado	Resultado obtenido
1	*packageName* válido	200	200
2	*packageName* inválido	400 Bad Request	400 Bad Request
3	*packageName* inexistente en la *Play Store*	404 Not found	404 Not found
4	No se puede acceder a política de privacidad	403 Forbidden	403 Forbidden
5	Sin API Key	401 Unauthorized	401 Unauthorized

Tabla 7.25: Pruebas realizadas al endpoint `/get/compare-privacy`

Prueba	Descripción	Resultado esperado	Resultado obtenido
1	Sin indicar *language*	200	200
2	Indicando como *language* “es”o “en”	200	200
3	*language* inválido	422 Unprocessable Content	422 Unprocessable Content
4	Sin API Key	401 Unauthorized	401 Unauthorized

Tabla 7.26: Pruebas realizadas al endpoint `/get/security-tips`

Prueba	Descripción	Resultado esperado	Resultado obtenido
1	Sin indicar *language*	200	200
2	Indicando como *language* “es”o “en”	200	200
3	*language* inválido	422 Unprocessable Content	422 Unprocessable Content
4	Sin API Key	401 Unauthorized	401 Unauthorized

Tabla 7.27: Pruebas realizadas al endpoint `/data-safety.json`

Prueba	Descripción	Resultado esperado	Resultado obtenido
1	Sin indicar *language*	200	200
2	Indicando como *language* “es”o “en”	200	200
3	*language* inválido	422 Unprocessable Content	422 Unprocessable Content
4	Sin API Key	401 Unauthorized	401 Unauthorized

Tabla 7.28: Pruebas realizadas al endpoint `/permissions.info`

7.3. Prueba de usabilidad

Las pruebas de usabilidad se realizan para comprobar qué tan fácil, intuitiva y eficiente es la interfaz y experiencia del usuario usando la aplicación. Se tratan de unas pruebas esenciales para este proyecto para determinar si incluso los usuarios que no tienen conocimiento técnico pueden utilizar y entender la aplicación.

Las pruebas de usabilidad se han realizado con 15 usuarios de diferentes edades y niveles de conocimiento técnico. Además, las pruebas se han realizado en varios móviles modelo *Samsung Galaxy A05s*, este es un modelo de gama baja y, por tanto, ha sido útil para comprobar también el correcto funcionamiento de la aplicación en dispositivos con menos recursos y con una versión más antigua de Android.

Tras obtener los resultados de las pruebas de usabilidad, se van a analizar los datos y realizar los cambios o mejoras que sean oportunas. Estos cambios se aplicarán al finalizar las pruebas de usabilidad para que todos los usuarios se enfrenten a la misma versión de la aplicación.

7.3.1. Tareas de la prueba de usabilidad

En primer lugar, se enumeran las tareas que se van a realizar en la prueba de usabilidad con cada uno de los usuarios.

1. Analizar un fichero .apk
2. Analizar una aplicación instalada en el dispositivo cuyo nombre de paquete es *com.whatsapp*
    2.1. Accede a la pestaña de “Política”del reporte creado.
3. Acceder al primer reporte realizado en la prueba de usabilidad.
4. Volver al inicio y borrar el primer reporte realizado.
5. Analizar la aplicación instalada en el dispositivo cuyo nombre es *AntiSplit M*
6. Filtrar por aplicaciones peligrosas en el historial de reportes.

7.3.2. Documento de recopilación de respuestas

Para recoger los datos resultantes de realizar las pruebas de usabilidad se usa para cada uno de los participantes el siguiente documento. En este documento se muestran las tareas a realizar, el tiempo que se ha tardado en completar cada una de ellas y, por último, un apartado para que el usuario dé su opinión global sobre la aplicación.

Evaluación de usabilidad: ApkAudit

ID usuario	**Rango de edad**			**Conocimientos previos**		
	18-28 años	28-42 años	+42 años	Bajos	Medios	Avanzado

Nº	**Descripción de la tarea**	**Completada (Sí/No)**	**Tiempo empleado**
Tarea 1	Analizar un fichero .apk		
Observaciones:			
Tarea 2	Analizar una aplicación instalada cuyo nombre de paquete es *com. whatsapp*		
Observaciones:			
Tarea 2.1	Accede a la pestaña de "Política"del reporte creado		
Observaciones:			
Tarea 3	Acceder al primer reporte realizado en la prueba de usabilidad		
Observaciones:			
Tarea 4	Volver al inicio y borrar el primer reporte realizado		
Observaciones:			
Tarea 5	Analizar la aplicación instalada en el dispositivo cuyo nombre es *AEMET*		
Observaciones:			
Tarea 6	Filtrar por aplicaciones peligrosas en el historial de reportes		
Observaciones:			

Valoración del sistema	1	2	3	4	5
Facilidad de uso					
Diseño visual					
Comprensibilidad del informe					
Volvería a usar la aplicación					

Opinión y aspectos de mejora

7.3.3. Resultados

Una vez realizadas las pruebas de usabilidad [43] se procede a analizar los resultados obtenidos, para ello, se utiliza la librería *Matplotlib* de *Python*. En primer lugar, se analiza la distribución de los usuarios según sus conocimientos previos en el sistema de Android y su edad.

Distribución de los usuarios

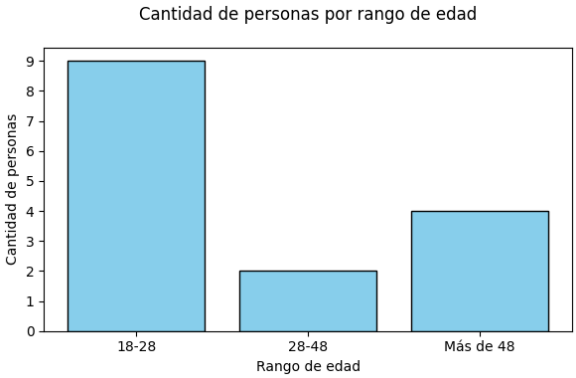


Figura 7.1: Distribución de los usuarios según su edad

En la figura 7.1 se puede observar como la mayoría de los usuarios tienen entre 18 y 28 años aunque, también se cuenta con varios usuarios mayores de 48 años que pueden aportar información sobre como se comportan ante la aplicación en distintos rangos de edad.

Cabe destacar que los intervalos utilizados para establecer los rangos de edad son cerrados por la izquierda y abiertos por la derecha. Por ejemplo, un usuario de 28 años se encuentra en el rango de 28 a 42 años.

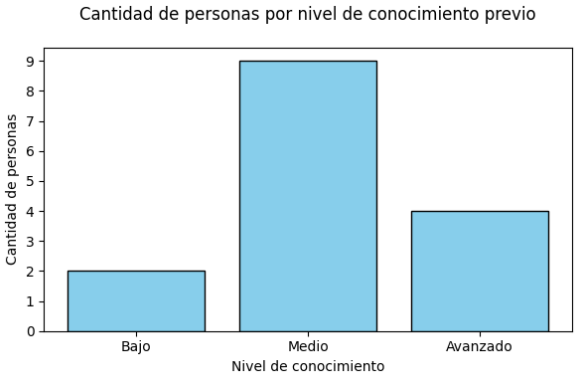


Figura 7.2: Distribución de los usuarios según sus conocimientos previos en el sistema de Android

En la figura 7.2 se puede observar que la gran mayoría de los usuarios con los que se ha realizado la prueba de usabilidad cuentan con conocimientos previos. La explicación de este hecho es que la prueba se realizó con un grupo de usuarios que tenían ya conocimientos de informática por ser estudiantes y/o profesores en el grado de Ingeniería Informática.

Duración de las tareas

A continuación, se muestra un digrama de dispersión para cada una de las tareas realizadas por los usuarios en el test de usabilidad. Además, en este diagrama se muestra el rango de edad al que pertenece el usuario mediante el color y su nivel de conocimientos previos mediante la forma del punto en el diagrama (Véase la figura 7.3).

Figura 7.3: Leyenda para el diagrama de dispersión

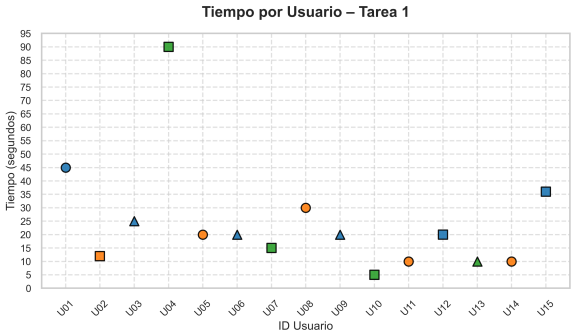


Figura 7.4: Diagrama de dispersión para la tarea 1

En la figura 7.4 se puede observar que la mayoría de los usuarios han tardado entre 11 y 27.5 segundos en completar la tarea. Algunos usuarios señalaron que tuvieron dificultades para encontrar los ficheros *.apk* en el dispositivo lo que puede dar explicación a un mayor tiempo para completar la correspondiente tarea. Esta es la tarea con mayor tiempo promedio de todas las tareas teniendo un valor de 24.53 segundos este hecho se puede deber, entre otros factores, a que es la primera tarea que se realiza de la aplicación y los usuarios no están familiarizados con la interfaz y/o funcionamiento.

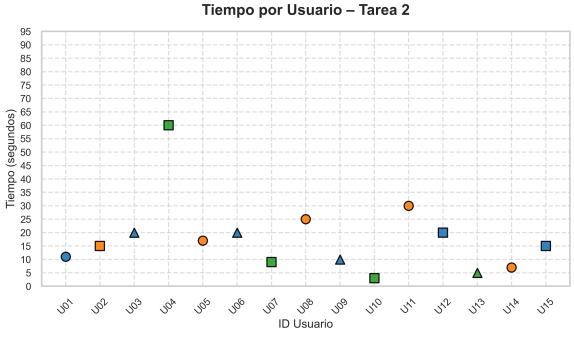


Figura 7.5: Diagrama de dispersión para la tarea 2

En la figura 7.5 se puede observar que la mayoría de los usuarios han tardado entre 9.5 y 20 segundos en completar la tarea. El valor máximo es de 60 segundos lo que sugiere la presencia de un *outlier* en los datos.

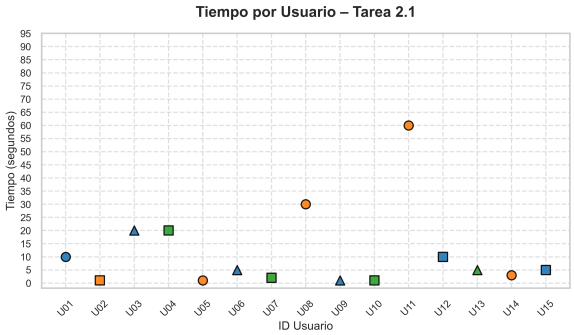


Figura 7.6: Diagrama de dispersión para la tarea 2.1

En la figura 7.6, se observa el diagrama de dispersión para la sub-tarea 2.1 de la tarea 2. En este caso, se puede observar que la mayoría de los usuarios han tardado entre 1.5 y 15 segundos en completar la tarea. Se trata de una tarea relativamente rápida en completar ya que se cuenta con un promedio de 11.6 segundos.

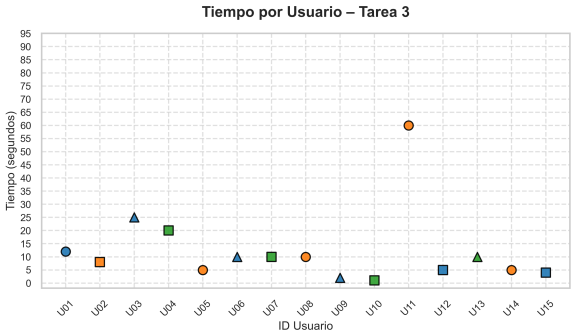


Figura 7.7: Diagrama de dispersión para la tarea 3

En la figura 7.7 se puede observar que la mayoría de los usuarios han tardado entre 5 y 11 segundos en completar la tarea. Una vez más, al igual que en la tarea anterior, se trata de una tarea que ha resultado intuitiva y rápida para la mayoría de los usuarios.

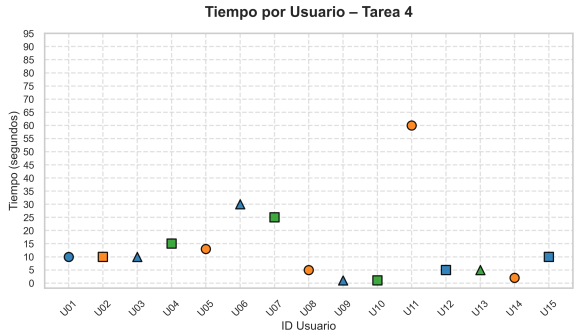


Figura 7.8: Diagrama de dispersión para la tarea 4

En la figura 7.8 la mayor parte de los usuarios han tardado entre 5 y 14 segundos. Aunque, muchos indican que les resultó poco intuitivo el hecho de tener que mantener pulsado para borrar el reporte realizado.

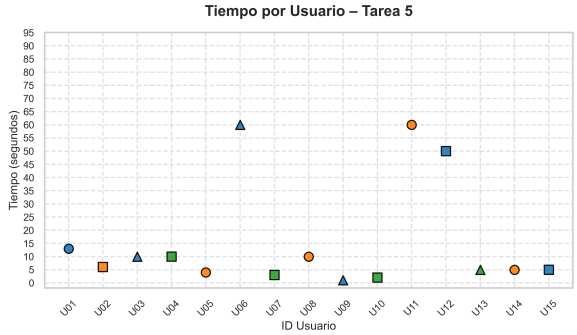


Figura 7.9: Diagrama de dispersión para la tarea 5

En la figura 7.9 se muestra el diagrama de dispersión para la tarea 5. En este caso, la mayoría de los usuarios han tardado entre 4.5 y 11.5 segundos en completar la tarea, con un promedio de 16.26 segundos. Esta es una tarea similar a la tarea 2, ya que se solicita analizar una aplicación instalada en el dispositivo. Tras comprobar los resultados obtenidos se observa que se detectan menores tiempos de ejecución en comparación con la tarea 2, esto se puede deber a que el usuario va aprendiendo a usar la aplicación a medida que la usa.

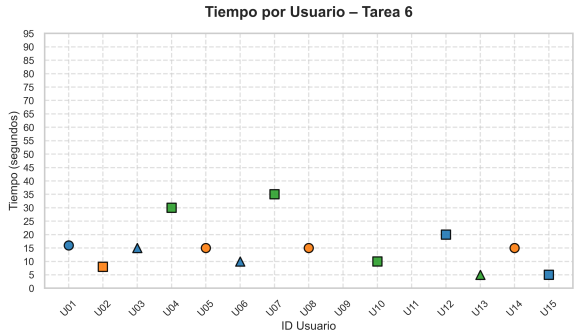


Figura 7.10: Diagrama de dispersión para la tarea 6

Finalmente, en la figura 7.10 se muestra el diagrama de dispersión para la tarea 6. En esta tarea se solicita filtrar por aplicaciones peligrosas en el historial de reportes. Sin embargo, algunos usuarios encontraron confuso el funcionamiento debido a que en la interfaz de la aplicación aparecía como aplicaciones de riesgo, en lugar de peligrosas. Este hecho provocó que dos usuarios no supieran completar la tarea.

Valoración global

En este apartado se muestra la valoración global dada por los usuarios a la aplicación. Para evaluar este aspecto se han definido 4 aspectos (facilidad de uso, diseño visual, comprensibilidad del informe y si volvería a usar la aplicación). Para cada uno de estos aspectos se ha definido una escala de 1 a 5, siendo 1 la puntuación más baja y 5 la puntuación más alta.

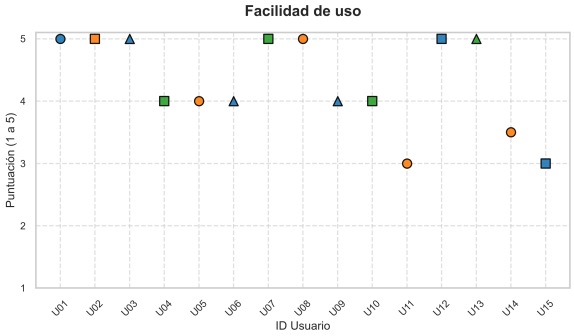


Figura 7.11: Diagrama de dispersión para la puntuación de facilidad de uso

En el diagrama de dispersión 7.11 se puede ver que, aunque los datos son positivos, algunos usuarios dan una puntuación cercana al 3. Una explicación a esto puede ser la falta de conocimientos previos en el sistema de Android y/o una falta de claridad en la interfaz de la aplicación.

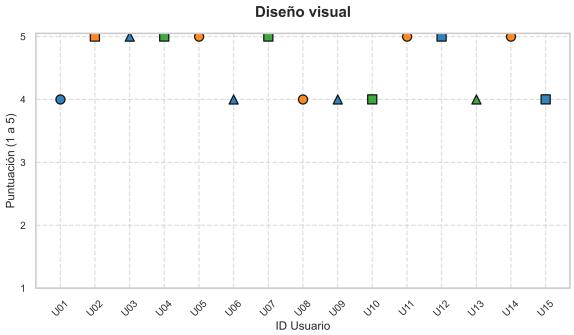


Figura 7.12: Diagrama de dispersión para la puntuación de diseño visual

En cuanto al diseño visual, todos los usuarios han dado una puntuación entre 4 y 5. Por tanto, consideran que la aplicación tiene un diseño visual atractivo.

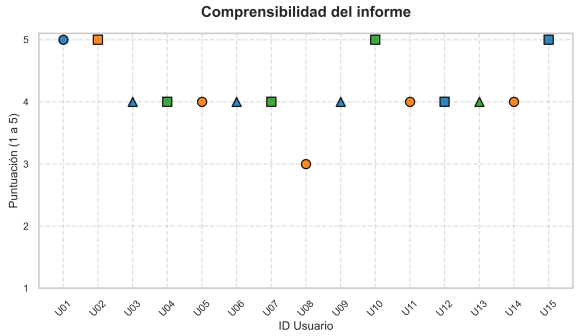


Figura 7.13: Diagrama de dispersión para la puntuación de comprensibilidad del informe

Un aspecto de gran relevancia es la capacidad de comprensión del informe generado tras el análisis de la aplicación. Tras observar la figura 7.13 se puede observar que, salvo un usuario, todos los usuarios han dado una puntuación entre 4 y 5, incluso aquellos usuarios con menores conocimientos previos en el sistema de Android.

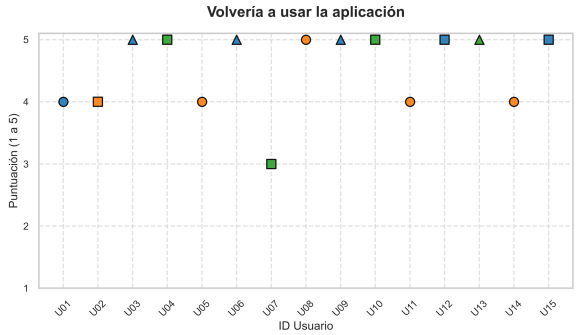


Figura 7.14: Diagrama de dispersión para la puntuación de volvería a usar la aplicación

Por último, en el diagrama de dispersión 7.14 se puede observar que la mayoría de los usuarios volverían a usar la aplicación.

Fallos identificados

Gracias a los datos obtenidos a partir de las pruebas de usabilidad se han podido identificar fallos y/o posibles mejoras en la aplicación. A continuación, se muestran dichos fallos etiquetado con un código de identificación de la forma *IXX* para, posteriormente, referenciar a ellos en el apartado de soluciones implementadas.

- **I01**: en la tarea 4, muchos usuarios señalan que aunque ellos encontraron la opción de eliminar el reporte, algunos usuarios podrían no saberlo o tardar tiempo en encontrar dicha opción. Actualmente, para borrar un reporte se debe mantener presionado el reporte correspondiente para que aparezca la opción de eliminar.
- **I02**: se ha identificado que si no se tiene conexión a Internet el reporte se genera pero

con información limitada y se tendría que analizar de nuevo manualmente para mostrar la información al completo.

- **I03**: en lugar de tener un botón con el texto “Aplicaciones instaladas”sería más apropiado un botón con el texto “Escanear apps instaladas”.
- **I04**: en el apartado de los permisos de la aplicación se listan también aquellos que define la aplicación pero no se muestra un distintivo para diferenciarlo de aquellos propios del sistema Android.
- **I05**: algunos usuarios indican que encuentran confuso el funcionamiento de algunos filtros. Principalmente el filtro de “Recientemente escaneado”que al pulsarlo se mantiene el orden de la misma manera y se tiene que pulsar nuevamente.
- **I06**: algunos usuarios indican la necesidad de tener una manera de volver directamente a la pantalla principal debido a que actualmente solo se vuelve a la pantalla anterior.
- **I07**: en la tarea 6 algunos usuarios lo han encontrado confuso al poner “Peligrosas”pero en el filtro de la aplicación tener puesto “Riesgo”.

Soluciones implementadas

A partir de los fallos que se han identificado en el apartado anterior, se han implementado las siguientes soluciones:

- **I01**: Se ha implementado un botón con el icono de una papelera para eliminar el reporte.
- **I02**: Se ha implementado un mensaje de error que se muestra cuando no se tiene conexión a Internet. Entonces, en caso de no disponer de conexión a Internet, se muestra el mensaje y no se genera el reporte.
- **I03**: Se cambia el texto del botón de “Aplicaciones instaladas”a “Escanear apps instaladas”para que sea más clara su función y se vea la diferencia con el botón de “Escanear un archivo.apk”.
- **I04**: Se añade una etiqueta para los permisos definidos por la aplicación.
- **I05**: Se cambia el filtro y el texto correspondiente para que sea más claro su función y que al pulsarlo cambie el orden.
- **I06**: Se ha implementado en el texto de título de la pantalla un botón que redirige a la pantalla principal. Además, al estar en el reporte siempre se redirige a la pantalla principal al volver atrás.

- **I07**: Se añade una descripción para cada uno de los filtros con el fin de saber cual es su funcionalidad y saber qué se considera de peligroso o de riesgo.

Capítulo 8

Seguimiento del proyecto

En este capítulo se detallan los distintos incrementos realizados durante el desarrollo del proyecto, así como el avance que se ha alcanzado en cada uno de ellos y la fecha de su finalización. De este modo, se puede observar si se han sufrido retrasos y qué riesgos se han podido materializar.

8.1. Desarrollo de los incrementos

A continuación, se indica el avance realizado para cada uno de los incrementos y sus fechas de finalización.

8.1.1. Incremento 1: Análisis y planificación inicial

Fecha: 28/03/2025

Descripción: Durante este primer incremento se ha realizado un primer análisis del proyecto, lo que incluye la recogida e identificación de los requisitos, objetivos y riesgos del proyecto. Además, se realiza la planificación inicial del proyecto y el estudio de las fuentes de datos que se emplearán en las próximas fases del proyecto.

8.1.2. Incremento 2: Desarrollo de la funcionalidad básica

Fecha: 10/04/2025

Descripción: Se ha logrado implementar la funcionalidad básica de la aplicación *ApkAudit* antes de lo planificado. Esta funcionalidad incluye la capacidad para analizar un archivo *.apk* o de las aplicaciones instaladas y generar un reporte sobre los datos obtenidos de la fuente de datos de *VirusTotal* y del repositorio de datos *App-PIMD*.

8.1.3. Incremento 3: Integración con el servidor de procesamiento

Fecha: 02/05/2025

Descripción: Se realiza el servicio básico de análisis de políticas de privacidad en el servidor de

procesamiento y se integra con la aplicación *ApkAudit*. No ha habido ningún problema durante el desarrollo de esta funcionalidad y se ha logrado alcanzar el objetivo antes de lo esperado.

8.1.4. Incremento 4: Implementación avanzada del servidor de procesamiento

Fecha: 24/05/2025

Descripción: Se ha realizado una refactorización considerable del código del servidor y se implementa mejor tratamiento de errores para que se puedan manejar correctamente los errores desde el cliente. Se ha retrasado ligeramente la finalización de este incremento debido a que se han implementado algunos endpoints y funcionalidades que no estaban previstas inicialmente.

8.1.5. Incremento 5: Refactorización y documentación final

Fecha: 05/06/2025

Descripción: Se realiza una refactorización del código de la aplicación, se adapta la integración con el servidor de procesamiento y con el resto de fuentes de datos externas para tratar de una mejor manera los errores. Además, se realiza una documentación del código al igual que se realiza el manual de usuario de la aplicación.

8.1.6. Incremento 6: Pruebas de usabilidad

Fecha: 11/06/2025

Descripción: No se ha logrado finalizar las pruebas de usabilidad en la fecha prevista (05/06/2025). Por tanto, se ha empleado una semana adicional para completar las pruebas de usabilidad, realizar el estudio de los resultados obtenidos y efectuar las soluciones a los fallos identificados en las pruebas de usabilidad. No ha supuesto un problema debido a que se había dejado un margen de tiempo por si surgía cualquier inconveniente durante el desarrollo.

8.2. Evaluación general del desarrollo del proyecto

En términos generales, se ha logrado cumplir con todos los objetivos establecidos. Sin embargo, algunos de los requisitos identificados inicialmente no pudieron implementarse tal como se habían establecido, o bien tuvieron que ser ajustados conforme avanzaba el desarrollo.

Si bien la mayoría de los incrementos se han completado dentro del plazo establecido o incluso antes de lo planificado, es importante señalar que el diseño detallado y análisis más profundo del sistema se han realizado en fases más avanzadas del proyecto. Esta situación se debió, en parte, a

la falta de claridad inicial en los requisitos y a ciertas imprecisiones en la planificación temprana.

A pesar de ello, el desarrollo del proyecto ha logrado superar ampliamente las expectativas iniciales e incluso se ha podido añadir funcionalidades adicionales que no se habían previsto inicialmente pero,que otorgan mayor calidad final y usabilidad a la aplicación.

8.3. Riesgos materializados

En este apartado se indican los riesgos que han ocurrido durante el desarrollo del proyecto y cómo se ha gestionado para poder mitigarlos.

8.3.1. R1: Estimaciones optimistas en la planificación

Durante la planificación inicial del proyecto se establecieron una serie de estimaciones ligeramente optimistas. Principalmente en las etapas de Análisis y Diseño que, llevaron más tiempo del que se había previsto. Sin embargo, se logro ajustar la planificación y que no afectase significativamente al correcto desarrollo del proyecto.

8.3.2. R8: Problemas de rendimiento

Durante el *Incremento 4 - Implementación avanzada del servidor de procesamiento* se identificaron problemas de rendimiento durante el proceso de análisis de las políticas de privacidad por parte del servidor de procesamiento. Estos problemas se manifestaban en tiempos de respuesta bastante altos causados por un uso ineficiente de la memoria disponible.

Este riesgo se solucionó identificando la parte del código responsable de una incorrecta liberación de memoria. Adicionalmente, se aumentaron los recursos del servidor, pasando de 4 GB a 8 GB de memoria RAM y de 2 a 4 núcleos de CPU, con el objetivo de prevenir la recurrencia de este tipo de incidencias.

8.3.3. RP3: Transmisión no cifrada de la información

Como ya se ha comentado anteriormente, durante el desarrollo del *Incremento 3 - Integración con el servidor de procesamiento* no se ha podido establecer que la comunicación entre el servidor de procesamiento y la aplicación *ApkAudit* se realizara de forma cifrada debido a que el sistema Android no permite el uso de certificados autofirmados para este tipo de comunicaciones.

Este riesgo, aunque se materializase no tuvo gran impacto debido a que la información que se transmite en esta comunicación no es sensible. De todos modos, se establece como una línea de trabajo futuro implementar conexiones a través de HTTPS con este servidor.

8.3.4. R10: Falta de tiempo para realizar las pruebas de usabilidad

En el *Incremento 6 - Pruebas de usabilidad* no se pudo completar a tiempo debido a la falta de disponibilidad de los usuarios con conocimientos técnicos del sistema Android. Por tanto, la fecha de finalización de este incremento se tuvo que retrasar hasta el día 11 de junio de 2025.

Capítulo 9

Conclusiones y líneas de trabajo futuro

En este capítulo, se indican las conclusiones que se obtienen tras finalizar el proyecto. Además, se tratan los objetivos que se han logrado y las posibles líneas de trabajo futuro para continuar desarrollando y ampliando la funcionalidad de la aplicación.

9.1. Conclusiones

El desarrollo del proyecto *ApkAudit* ha sido bastante interesante y me ha permitido aprender y poner en práctica los conocimientos adquiridos durante la carrera. Tras finalizar la implementación puedo asegurar que se han cumplido todos los objetivos planteados inicialmente en el proyecto. Algunos de estos objetivos en un principio parecían difíciles de implementar pero al final tras investigar y estudiar los diferentes recursos disponibles, se han podido conseguir.

Las pruebas realizadas en la aplicación han permitido cambiar algunos aspectos de la aplicación con el fin de solucionar algunos errores que se habían producido durante el desarrollo y otros, detectados durante las pruebas con los usuarios. Logrando también mejorar la usabilidad y, en consecuencia, la experiencia del usuario. Aunque se hayan conseguido la totalidad de los objetivos, quedan algunas líneas de trabajo abiertas debido a algunas limitaciones técnicas y lograr, de este modo, publicar la aplicación en la *Google Play Store* para que pueda ser utilizada por otros usuarios.

Considero que *ApkAudit* es una herramienta bastante útil para ayudar al usuario promedio de sistemas Android a elegir qué aplicaciones forman parte de su dispositivo y preocuparse por su privacidad.

Durante el análisis de las aplicaciones, se ha podido observar como en muchos casos existen discrepancias entre la información que se declara en las tiendas oficiales de descarga de aplicaciones y la política de privacidad que se emplea. Por ejemplo, se ha detectado que, aunque no se recojan datos del usuario directamente mediante los permisos solicitados por la aplicación, estos pueden ser obtenidos a través de rastreadores u otros mecanismos capaces de identificar la localización, la identidad u otros datos personales del usuario. Esta situación puede reflejar una falta de transparencia en la información que se proporciona públicamente y pone de manifiesto la necesidad de declarar de forma más explícita qué datos se recogen y con qué finalidad.

9.2. Líneas de trabajo futuro

A continuación se muestran algunas líneas de trabajo futuras para mejorar tanto la aplicación *ApkAudit* como el servidor de procesamiento de los datos.

9.2.1. Funcionalidades a añadir en la aplicación

- **Analizar automáticamente aplicaciones que desee el usuario cada cierto tiempo.** Agregar la funcionalidad de que el usuario pueda seleccionar, entre las aplicaciones instaladas en su dispositivo, que sean analizadas en segundo plano cada cierto tiempo. La finalidad de esta funcionalidad es la de controlar que las aplicaciones instaladas en sus respectivas actualizaciones conserven un comportamiento seguro y respetando la privacidad, alertando al usuario en caso de que se detecten aplicaciones sospechosas.

- **Notificaciones push.** Algunos reportes sobre algunas aplicaciones pueden tardar tiempo en generarse al completo debido a que puede ser lenta su carga y análisis en las fuentes de datos. Por tanto, se puede agregar la funcionalidad de que el sistema continúe analizando en segundo plano y al terminar al completo informe al usuario a través de una notificación push.

- **Alternativas a aplicaciones.** Implementar un apartado que incluya alternativas a las aplicaciones que se analizan. Esto resulta útil para buscar aplicaciones que respeten más la privacidad y seguridad del usuario que las que ya tiene el usuario instaladas.

- **Comparación de reportes.** Implementar la funcionalidad de poder comparar dos reportes entre sí para ver las diferencias entre ellos.

- **Preparación para la publicación en la *Google Play Store*.** Preparar la aplicación para ser lanzada en la *Google Play Store*. Esto implica la implementación, entre otras funcionalidades, de la solicitud al usuario de introducir su API Key de *VirusTotal* o en caso de no tenerlo, eliminar para este usuario la funcionalidad de análisis con el servicio de *VirusTotal*.

9.2.2. Funcionalidades a añadir en el servidor

- **Política de privacidad en páginas protegidas por servicios antiscraping.** Algunas de las páginas que se consultan para obtener la política de privacidad de las aplicaciones pueden estar protegidas contra el scraping con servicios como *Cloudflare* o *Akamai*. En estos casos, actualmente se informa al usuario de que no se ha podido obtener la polítca de privacidad de la aplicación. No obstante, podría ser interesante obtener la política de privacidad mediante otros servicios que no incluyan realizar el scraping.

- **Conexión mediante HTTP.** El servidor actualmente está desplegado en una máquina virtual proporcionada por la *Escuela de Ingeniería Informática* de la *Universidad de Valladolid*

y se ha intentado establecer comunicaciones cifradas con TLS. Sin embargo, no se pueden usar certificados autofirmados en Android para establecer este tipo de comunicaciones . Por tanto, las comunicaciones entre la aplicación *ApkAudit* y el servidor se realizan mediante el protocolo HTTP, que no son seguras al no estar cifradas. Por tanto, queda pendiente desplegar el servicio en un servidor que admita HTTPS.

Bibliografía

[1] TrackerControl, "Base de datos de rastreadores de Android," https://github.com/TrackerControl/tracker-control-android/blob/master/app/src/main/res/values/arrays.xml, Accedido: 2025-05-13.

[2] "Informe sobre políticas de privacidad en Internet," https://www.aepd.es/sites/default/files/2019-09/informe-politicas-de-privacidad-adaptacion-RGPD.pdf, Accedido: 2025-05-17.

[3] e. Project Management Institute, *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK)*, 5th ed. Newtown Square, PA: Project Management Institute, 2017.

[4] "App-PIMD (App Privacy Impact MetaData) – Repositorio de Metadatos de Aplicaciones – Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid," https://ingpriv.uva.es/repositorio-app-pimd/, Accedido: 2025-03-16.

[5] Grupo de Investigación en Ingeniería de la Privacidad, "Documentación de la API App-PIMD," https://app-pi.infor.uva.es/docs, Accedido: 2025-06-04.

[6] A. Pérez-Fuente, M. M. Martínez-González, A. Aparicio, and Q. I. Moro, "Un Data Warehouse para el Estudio de la Privacidad y Seguridad de Aplicaciones Móviles," Actas de las IX Jornadas Nacionales de Investigación en Ciberseguridad, JNIC 2024, Sevilla, pp. 624–631, Mayo 2024. [Online]. Available: https://idus.us.es/items/508e612b-3df5-458a-98be-2d5f7c3fb7d2

[7] B. S. s.r.o, "GanttProject," https://www.ganttproject.biz/download/free, Accedido: 2025-03-16.

[8] "Salario para Ingeniero De Software Junior en España," https://es.talent.com/salary?job=ingeniero+de+software+junior, Accedido: 2025-03-16.

[9] A. W. Services, "Estimación servicio EC2 AWS," https://calculator.aws/#/estimate?id=c05f3fa483d53e717d3d7989df22ca60891eacfa, Accedido: 2025-03-16.

[10] C. Research, "Global Smartphone Sales Share by Operating System," https://www.counterpointresearch.com/insights/global-smartphone-os-market-share/, Accedido: 2025-05-09.

[11] "Apps de Android en Google Play," https://play.google.com/store/apps?hl=es_419, Accedido: 2025-05-09.

[12] "Amazon AppStore," https://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011, Accedido: 2025-05-09.

[13] "HUAWEI AppGallery - HUAWEI España," https://consumer.huawei.com/es/mobileservices/appgallery/, Accedido: 2025-05-09.

[14] "Permisos en Android | Android Developers," https://developer.android.com/guide/topics/permissions/overview?hl=es-419, Accedido: 2025-05-09.

[15] J. O'Claire, "Nearly 50 % of Android Apps use Mobile Trackers," https://jamesoclaire.com/2025/03/11/nearly-50-of-android-apps-use-mobile-trackers/, 3 2025, Accedido: 2025-05-09.

[16] J. A. Vázquez, "Casi nadie entiende las políticas de privacidad de las redes sociales - Dosdoce.com," https://www.dosdoce.com/2020/11/30/casi-nadie-entiende-las-politicas-de-privacidad-de-las-redes-sociales/, 11 2020, Accedido: 2025-06-17.

[17] "Proporcionar información sobre la sección Seguridad de los datos de Google Play," https://support.google.com/googleplay/android-developer/answer/10787469?hl=es, Accedido: 2025-05-09.

[18] ZDNET, "14 % of Android app privacy policies contain contradictions about data collection," https://www.zdnet.com/article/14-of-android-app-privacy-policies-contain-contradictions-about-data-collection/, Accedido: 2025-05-09.

[19] "Tipos de datos de la sección de Seguridad de Google Play," https://support.google.com/googleplay/android-developer/answer/10787469?hl=es#zippy=%2Ctipos-de-datos, Accedido: 2025-06-17.

[20] D. CSV, "INTRO al Natural Language Processing (NLP) #1 - ¡De PALABRAS a VECTORES! - YouTube," https://www.youtube.com/watch?v=Tg1MjMIVArc, Accedido: 2025-05-18.

[21] ——, "¿Qué es un EMBEDDING? - YouTube," https://www.youtube.com/watch?v=RkYuH_K7Fx4, Accedido: 2025-05-18.

[22] "HTTP messages | MDN," https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Messages, 5 2025, Accedido: 2025-06-03.

[23] MDN, "Métodos de petición HTTP," https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Methods, 3 2025, Accedido: 2025-05-10.

[24] "What is a TCP 3-way handshake process?" https://afteracademy.com/blog/what-is-a-tcp-3-way-handshake-process/, Accedido: 2025-06-03.

[25] Cloudflare, "¿Qué es HTTPS?" https://www.cloudflare.com/es-es/learning/ssl/what-is-https/, Accedido: 2025-05-10.

[26] H. Bhatt, “What is TLS/SSL and How It Works,” https://www.encryptionconsulting.com/education-center/what-is-tls-ssl/, 4 2024, Accedido: 2025-06-03.

[27] “REST API components & How to read them,” https://www.skiplevel.co/blog/part-2-rest-api-components-how-to-read-them, Accedido: 2025-06-03.

[28] O. Initiative, “OpenAPI Initiative,” https://www.openapis.org/, Accedido: 2025-05-10.

[29] VirusTotal, “Guía de uso de API de VirusTotal,” https://docs.virustotal.com/docs/how-it-works, Accedido: 2025-05-17.

[30] michaelstonis, “Model-View-ViewModel - Microsoft Learn,” https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm, Accedido: 2025-05-17.

[31] “Guía de arquitectura de apps - Android Developers,” https://developer.android.com/topic/architecture?hl=es-419, Accedido: 2025-05-17.

[32] “What is Repository Pattern?” https://www.linkedin.com/pulse/what-repository-pattern-alper-sara%C3%A7/, Accedido: 2025-05-19.

[33] “Patrón Repository,” https://www.linkedin.com/pulse/patr%C3%B3n-repository-angel-hermozo-glibe/, Accedido: 2025-05-20.

[34] Universidad de Valladolid, Escuela de Ingeniería Informática, “DIAS - Diseño, Integración y Adaptación del Software. Tema 4: Diseño de software. Métodos y técnicas. Parte III: Patrones de Acceso a Datos.”

[35] A. Developers, “Kotlin y Android,” https://developer.android.com/kotlin?hl=es-419, Accedido: 2025-05-11.

[36] “Manifest.permission,” https://developer.android.com/reference/android/Manifest.permission, Accedido: 2025-05-19.

[37] A. Singh, “API Gateway Design Pattern: Centralizing Control for Distributed Systems,” https://blog.stackademic.com/understanding-the-api-gateway-design-pattern-3e5d226bb16a, 10 2024, Accedido: 2025-06-03.

[38] “Understand app privacy & security practices with Google Play’s Data safety section - Computer - Google Play Help,” https://support.google.com/googleplay/answer/11416267?hl=en#zippy=%2Cdata-types, Accedido: 2025-05-20.

[39] “Kdoc-Generator Plugin for JetBrains IDEs | JetBrains Marketplace,” https://plugins.jetbrains.com/plugin/10389-kdoc-generator, Accedido: 2025-05-22.

[40] “Postman: The World’s Leading API Platform,” https://www.postman.com/, Accedido: 2025-05-22.

[41] “Uvicorn,” https://www.uvicorn.org/, Accedido: 2025-05-11.

[42] “Seguridad con protocolos de red | Android Developers,” https://developer.android.com/privacy-and-security/security-ssl?hl=es-419#SelfSigned, Accedido: 2025-05-23.

[43] J. García González, “Pruebas de Usabilidad del Estudio de Usuarios en ApkAudit,” https://doi.org/10.5281/zenodo.15652323, jun 2025, Accedido: 2025-06-12.

[44] Docker, “Docker Docs,” https://docs.docker.com/engine/install/, 11 2024, Accedido: 2025-05-10.

Apéndice A

Manual de usuario

La primera vez que se accede a la aplicación, no se muestra ninguna opción en el apartado de escaneos recientes. Sin embargo, se puedeb ver dos consejos de seguridad que a medida que se usa la aplicación van cambiando y, se muestran los dos botones para realizar las acciones principales de la aplicación. En la figura A.1 se muestra la pantalla de inicio de la aplicación.

1. **Analizar un fichero APK**: Permite analizar un fichero APK almacenado en el dispositivo.
2. **Aplicaciones instaladas**: Muestra las aplicaciones instaladas en el dispositivo. Permite analizar las aplicaciones que el usuario tiene instaladas en su dispositivo.

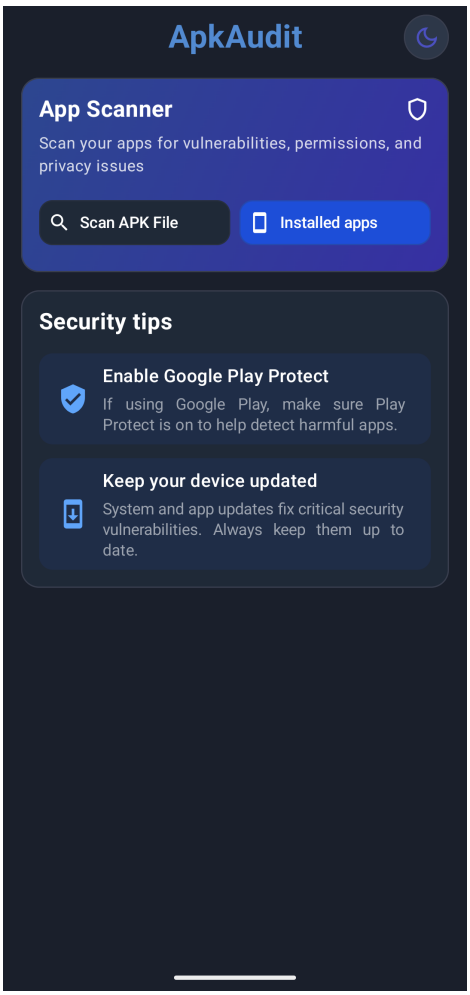


Figura A.1: Pantalla de inicio de la aplicación

En la figura A.1, además de los botones con las funcionalidades principales, se puede dar en el botón situado en la esquina superior derecha para poder cambiar el tema de la aplicación, es decir, entre claro u oscuro. El modo claro se ve como se muestra en la figura A.2.

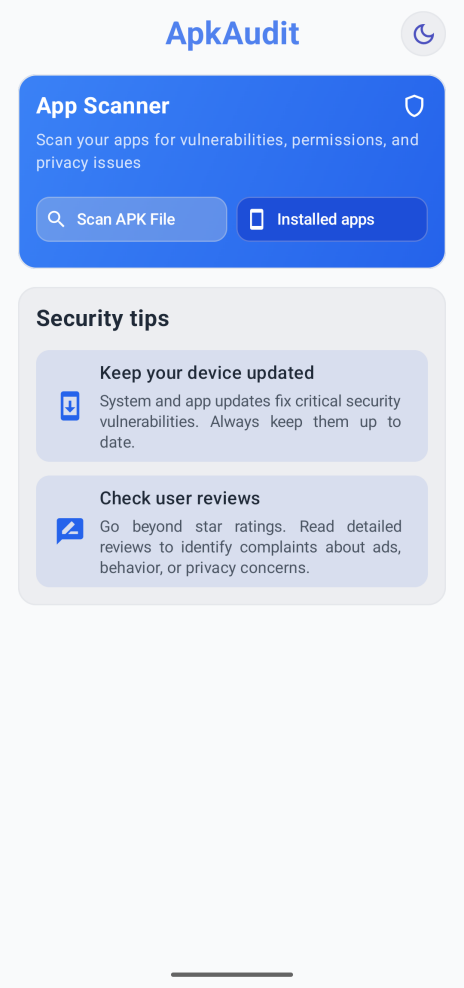


Figura A.2: Pantalla de inicio de la aplicación en modo claro

A.1. Analizar un fichero APK

Al seleccionar el botón de **Analizar un fichero APK**, se muestra la pantalla de selección de fichero del sistema Android. A través de esta pantalla, el usuario podrá seleccionar el fichero APK que quiera analizar navegando por su sistema de archivos, si es necesario. A continuación, en la figura A.3 se muestra como se ve dicha pantalla de elección.

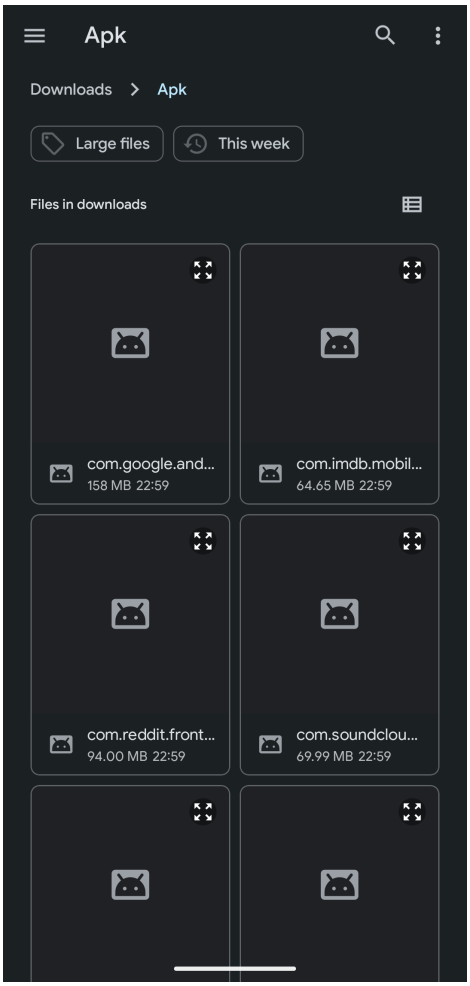


Figura A.3: Pantalla de selección de fichero APK

A.2. Aplicaciones instaladas

Al pulsar en el botón de **Aplicaciones instaladas**, se carga el listado de aplicaciones instaladas en el equipo del usuario. Estas aplicaciones se muestran en forma de tarjetas y solo se muestran aquellas aplicaciones del usuario, es decir, se excluyen las aplicaciones del sistema. En la figura A.4 se muestra la pantalla de aplicaciones instaladas.

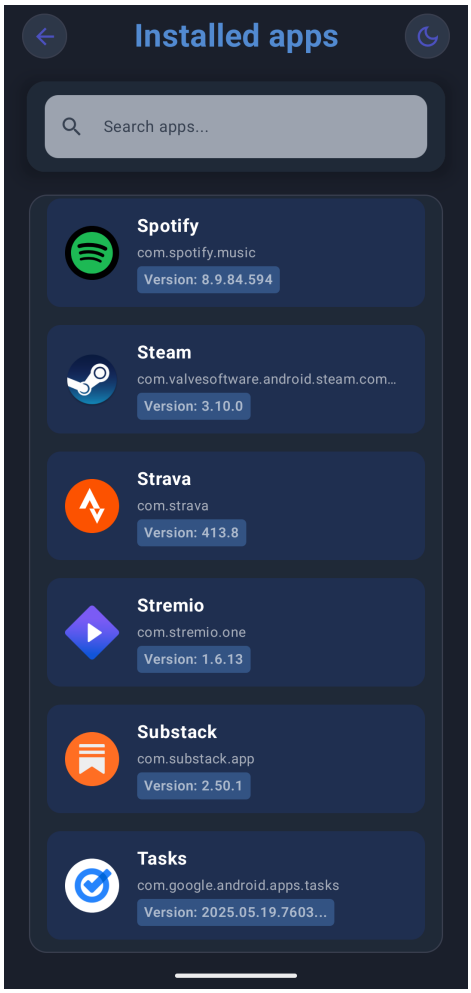


Figura A.4: Pantalla de aplicaciones instaladas

A.3. Reporte de una aplicación

Una vez se selecciona una aplicación, bien desde su fichero APK o desde la lista de aplicaciones instaladas, se comienza a procesar el reporte de dicha aplicación. Esto es un proceso que puede tardar unos pocos segundos, dependiendo de la aplicación a analizar y de la conexión a Internet. Una vez se obtiene el reporte y los resultados de todas las fuentes de datos, se muestra la pantalla de reporte de la aplicación como se puede ver en la figura A.5.

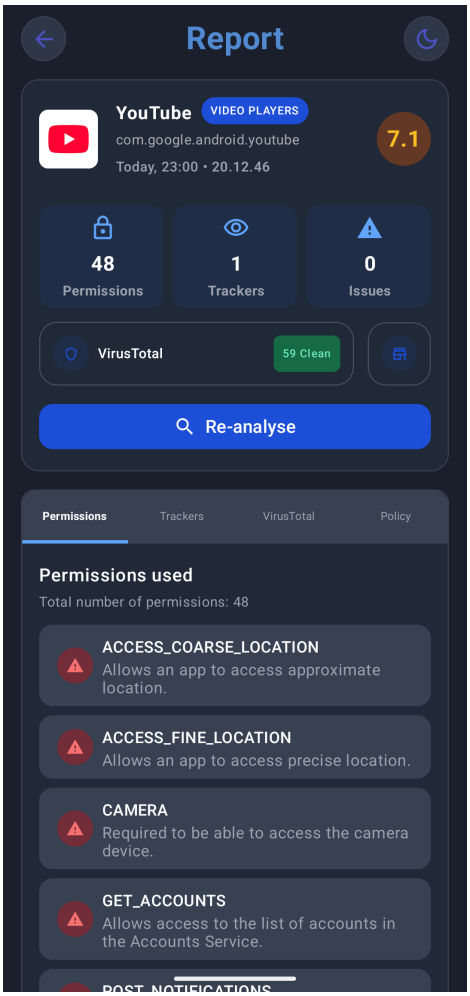


Figura A.5: Pantalla de reporte de una aplicación

En la tarjeta superior se muestra información general sobre el reporte obtenido, como los permisos que utiliza en total, los trackers y/o rastreados empleados y las incidencias que se han detectado. En cuanto a las incidencias, se tiene en cuenta las discrepancias en la política de privacidad y los motores de antivirus que lo han detectado como *malware*. Además, se muestra una puntuación que representa la métrica de privacidad obtenida como resultado del análisis en el repositorio *App-PIMD*.

En la parte inferior, se muestra una serie de pestañas, en concreto, se muestran las pestañas siguientes:

- **Permisos**: Muestra los permisos que utiliza la aplicación junto con una breve descripción de cada uno de ellos y si es un permiso categorizado como *dangerous*.
- **Trackers**: Muestra una lista con los trackers y rastreadores que utiliza la aplicación.
- **VirusTotal**: Muestra el resultado de analizar la aplicación en el servicio de *VirusTotal*. Esto es, entre otros datos, si la aplicación ha sido reconocida como *malware* o no.
- **Política de privacidad**: Muestra el resultado de analizar la política de privacidad y

compararla con el *Data Safety* de la *Google Play Store*.

En el caso, de que no se hayan podido obtener los resultados de alguna de las fuentes de datos, no se muestra la pestaña correspondiente.

A.3.1. Permisos

La pestaña de *Permisos* muestra todos los permisos que utiliza la aplicación junto con la descripción de estos permisos correspondiente en la documentación oficial de Android. Además, se muestra el nivel de protección del permiso a través de un icono que indica si es un permiso *dangerous*, *normal* o *signature*. En la figura A.6 se muestra la pestaña de *Permisos* para el reporte de la aplicación *YouTube*.

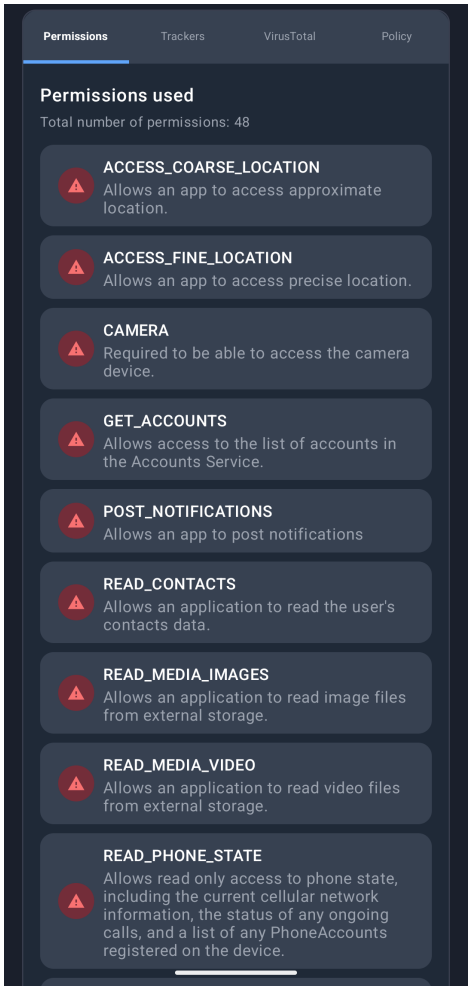


Figura A.6: Pestaña de permisos

Como se puede ver en la figura A.6, la aplicación *YouTube* utiliza 48 permisos y, entre otros, utiliza los permisos `ACCESS_COARSE_LOCATION` y `ACCESS_FINE_LOCATION`.

A.3.2. Trackers

Como se ha mencionado anteriormente, la pestaña de *Trackers* muestra los trackers que se han detectado en la aplicación escaneada tras realizar un análisis estático a sus clases. Se muestra una definición de lo que es un tracker, el número de trackers detectados y la lista de ellos. En la figura A.7 se muestra la pestaña de *Trackers* para el reporte de la aplicación *YouTube*.

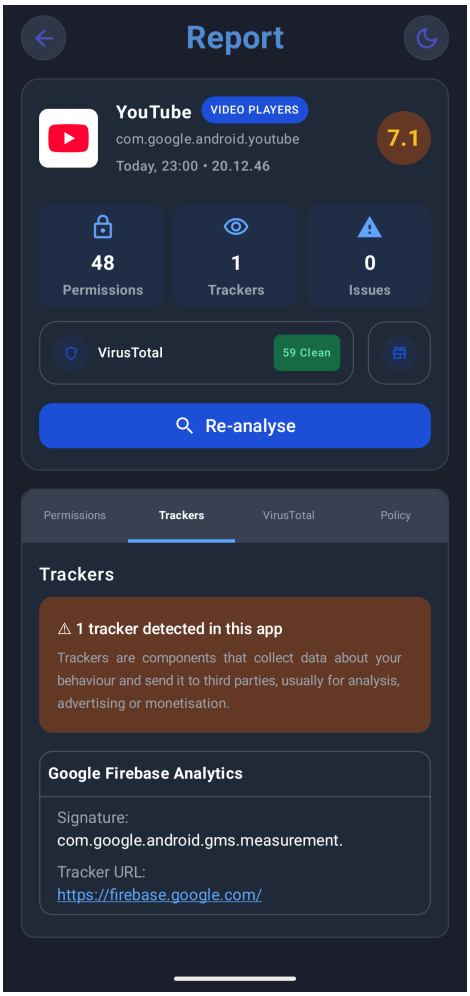


Figura A.7: Pestaña de trackers

En cada una de las tarjetas de los tracker se muestra el nombre del tracker detectado, su firma y la página web asociada, pudiendo acceder a ella pulsando sobre el link correspondiente.

A.3.3. VirusTotal

En la pestaña de *VirusTotal* se muestran los resultados del análisis de la aplicación por parte de numerosos motores de antivirus. Estos indican si la aplicación es considerada *malware*, *sospechosa*, *limpia* o, en el caso de que no se haya podido analizar, *desconocido*. En la figura A.8 se muestra la pestaña de *VirusTotal* para el reporte de la aplicación *YouTube*. Solo se muestra el resultado de 4 de los motores de antivirus priorizando aquellos que han detectado la aplicación como *malware*.

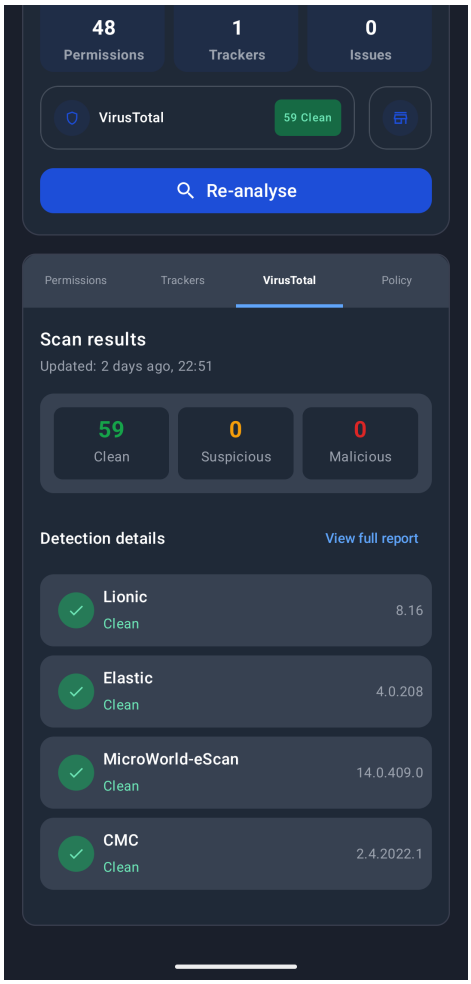


Figura A.8: Pestaña de *VirusTotal*

Se puede pulsar sobre el botón de *Ver reporte completo* para acceder a más información obtenida por el servicio de *VirusTotal*. En la figura A.9 se muestra la pantalla a la que se redirige al pulsar sobre dicho botón.

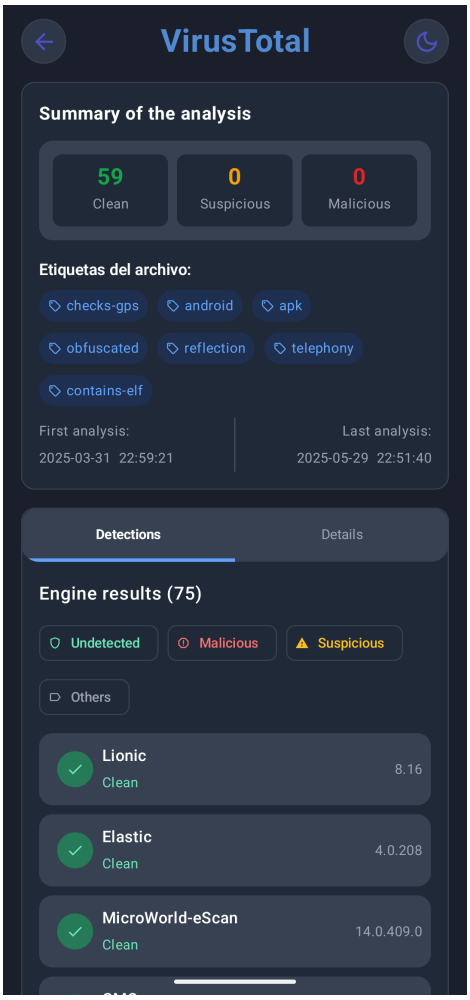


Figura A.9: Pantalla de informe completo de *VirusTotal*

En la figura A.9 se muestra el informe completo de *VirusTotal*, donde se puede acceder al resultado de todos los motores de antivirus, etiquetas asociadas a la aplicación, fechas de análisis y, por último, en la pestaña de *Detalles* se muestran los metadatos de la aplicación. En la figura A.10 se muestra la pestaña de *Detalles* para el reporte de la aplicación *YouTube*.

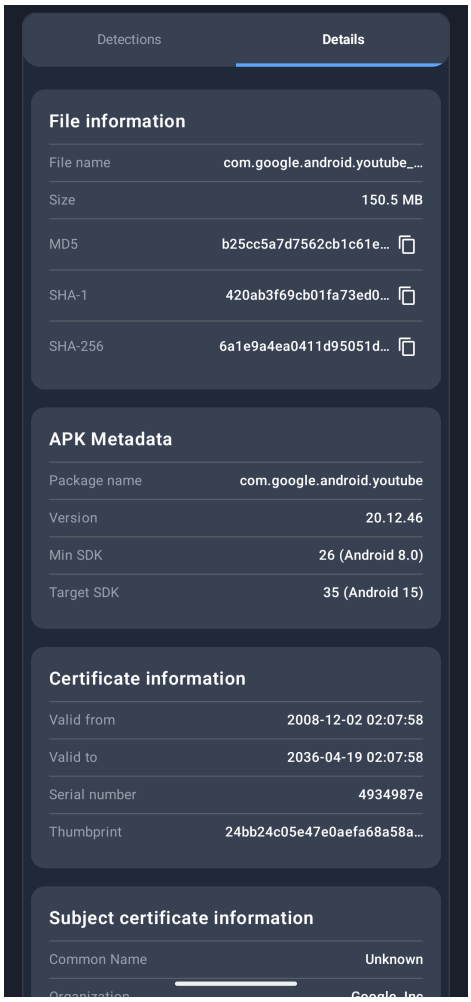


Figura A.10: Pantalla de detalles de *VirusTotal*

Entre la información que se muestra en la pestaña de *Detalles* se dividen en diferentes secciones:

- **Información del fichero**: Se muestra información como el nombre del fichero, el tamaño y los hashes[1] del fichero (SHA-1, SHA-256, MD5).
- **Metadatos del apk**: Muestra información como el nombre del paquete, la versión de la aplicación, la versión mínima de Android requerida y la versión objetivo de Android.
- **Información de certificado**: Se muestra información relacionada con el certificado empleado por la aplicación. Esto incluye las fechas de validez, número de serie y thumbprint (hash del certificado).
- **Información del certificado del emisor/sujeto**: Muestra la información del certificado del emisor/sujeto. Esto incluye el nombre del emisor, el nombre del sujeto, el nombre de la organización, la ciudad, el estado y el país.

[1]Hash: es un valor alfanumérico (una cadena de texto) generado por una función hash criptográfica que resume el contenido del archivo.

A.3.4. Política de privacidad

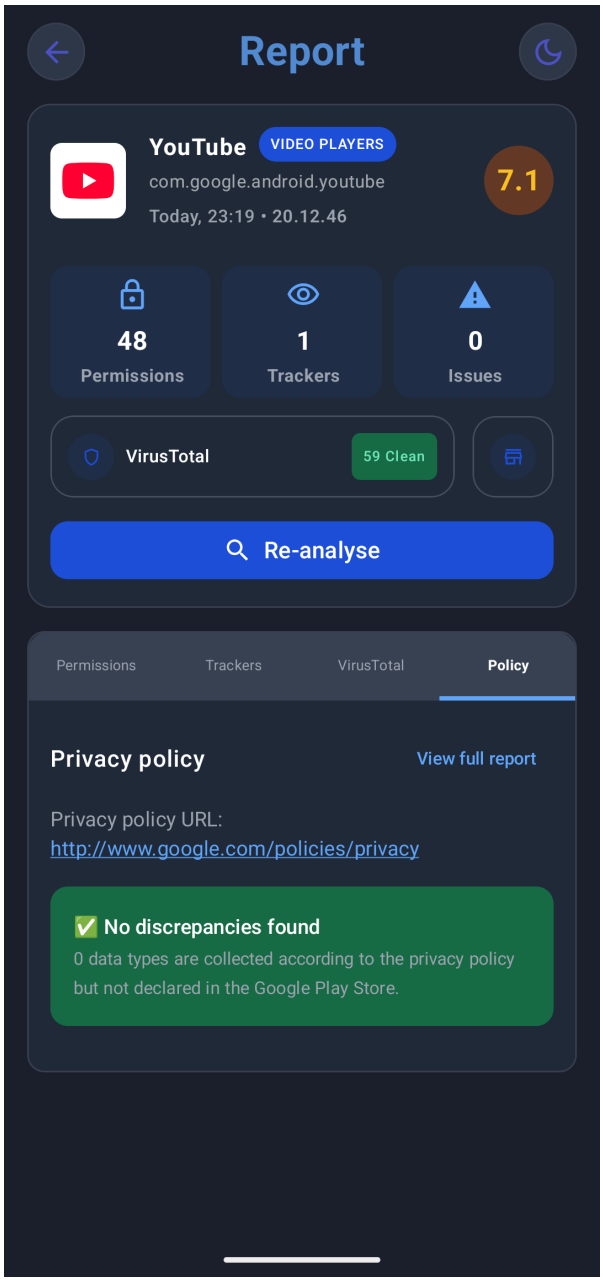
En la pestaña de *Política de privacidad* se muestra la definición de lo que es una discrepancia, la url de la política de privacidad de la aplicación, número de discrepancias detectadas, los datos declarados correctamente y los recopilados pero no declarados.

Los datos recopilados pero no declarados son datos que tras analizar el texto de la política de privacidad de la aplicación se ha detectado que se recopilan pero, que estos datos no están declarados explicitamente en el apartado de *Data Safety* de la *Google Play Store*.

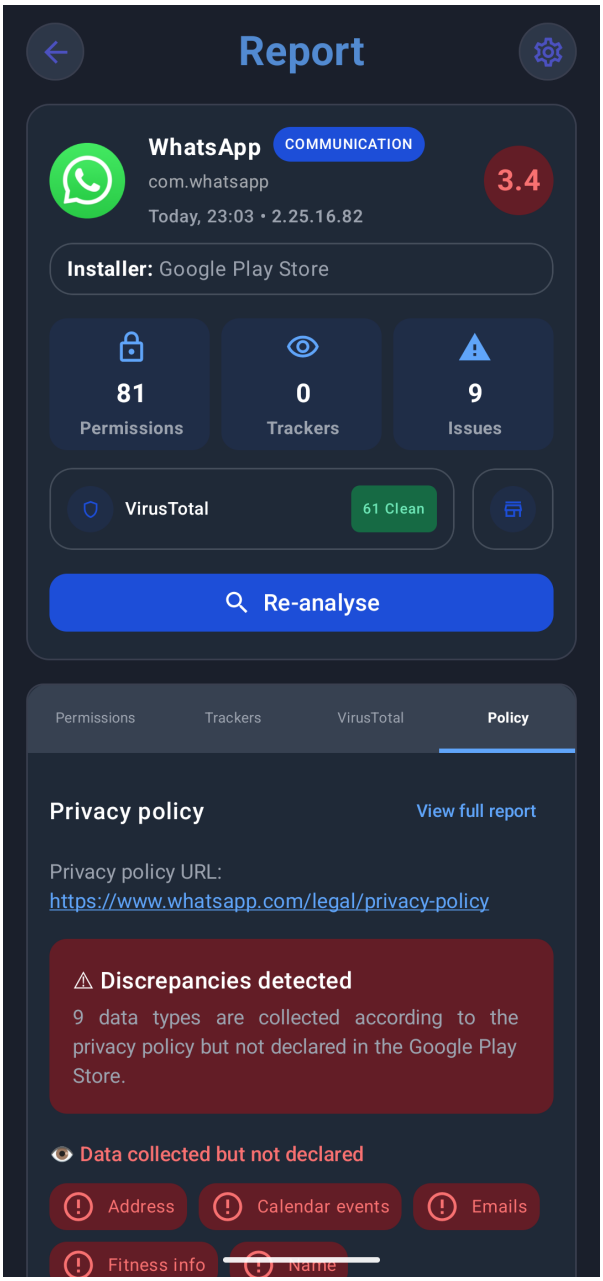
Los datos declarados correctamente son aquellos datos que se han detectado tras analizar el texto de la política de privacidad pero que, en este caso, si que se han declarado en el apartado de *Data Safety* de la *Google Play Store*.

Para obtener más información sobre cada uno de los tipos básicos de datos se puede pulsar sobre cada uno de ellos, esto abrirá un cuadro de diálogo con la descripción detallada de este tipo de dato.

En la figura de la izquierda, A.11a se muestra la pestaña de *Política de privacidad* para la aplicación de *YouTube*. Mientas que, en la figura de la derecha, A.11b para el reporte de la aplicación *WhatsApp*.



(a) Resultado del análisis de la política de privacidad de *YouTube*.



(b) Resultado del análisis de la política de privacidad de *WhatsApp*.

Figura A.11: Comparativa del análisis de políticas de privacidad de dos aplicaciones.

A continuación, en la figura A.12 se ofrece la pestaña completa del resultado de la política de privacida de *WhatsApp*

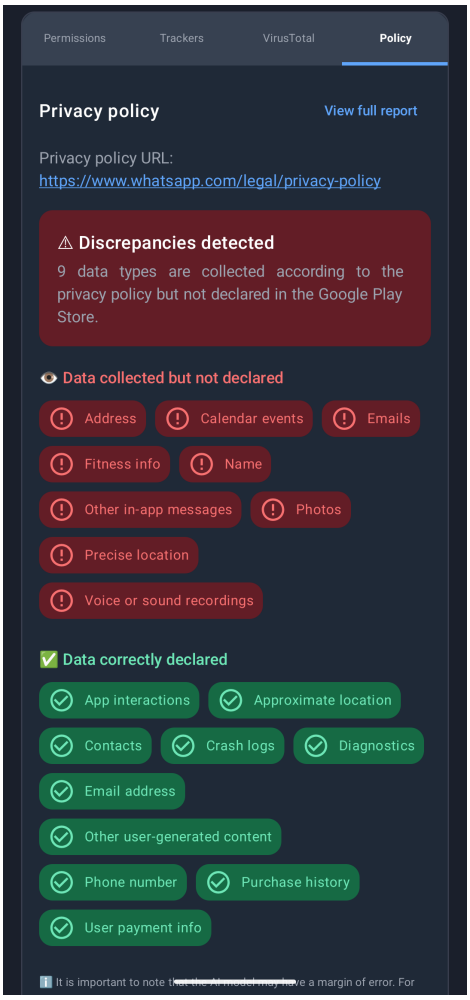


Figura A.12: Resultado completo del análisis de la política de privacidad de *WhatsApp*.

Como se puede ver en la figura A.12, también hay un botón de *Ver reporte completo* que permite una vista completa de los datos recogidos, compartidos y prácticas de seguridad de la aplicación.

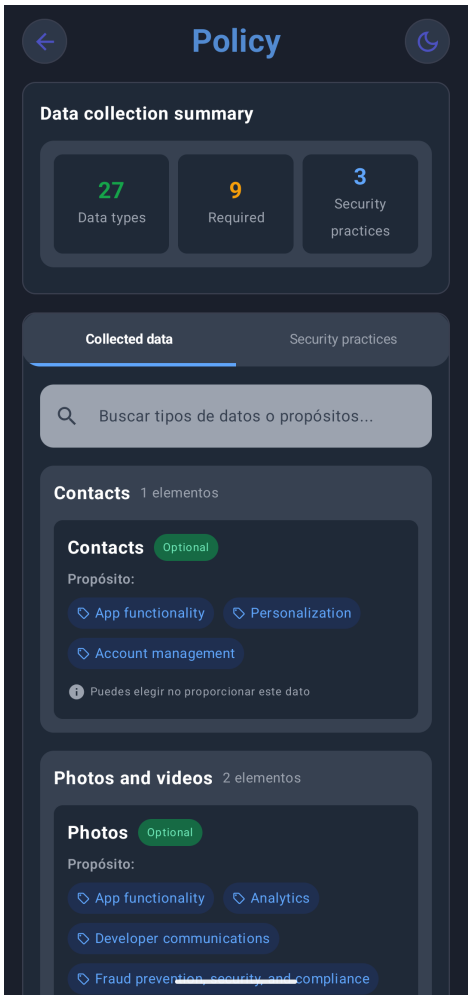


Figura A.13: Pantalla de informe completo de los datos recogidos para *WhatsApp*.

En la figura A.13 se muestra la pantalla de informe completo de los datos recogidos para *WhatsApp*. En esta pantalla se pueden ver la cantidad de datos que se recogen, si son obligatorios o no para el uso de la app, datos compartidos a terceros (si existen) y las prácticas de seguridad empleadas por la aplicación. A esta información se accede a través de las distintas pestañas mostradas en la figura. A continuación, se muestra la pestaña de *Prácticas de seguridad*.

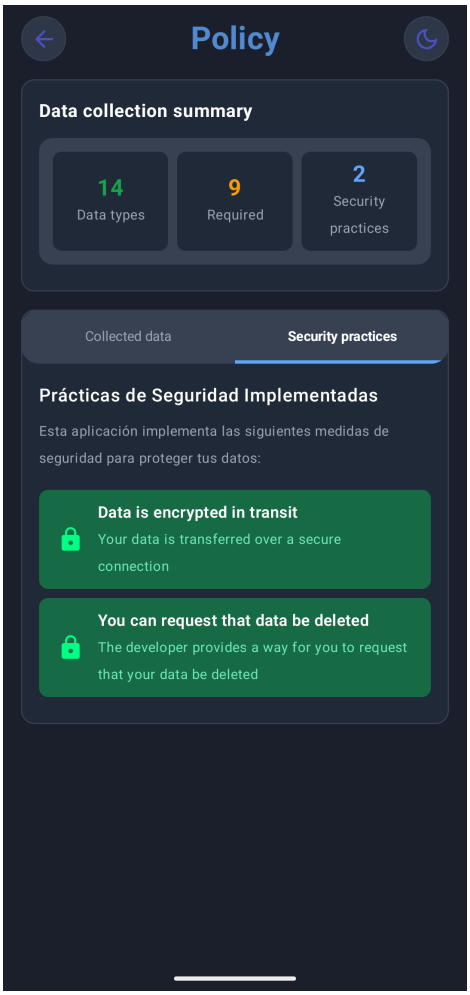


Figura A.14: Pestaña de prácticas de seguridad de *WhatsApp*.

A.4. Historial

Desde la pantalla principal de la aplicación y una vez se ha realizado como mínimo un análisis, se desbloquea el acceso al historial de análisis o escaneos. Esta es una pantalla que permite buscar, filtrar, ordenar y acceder a los análisis realizados con anterioridad y, que se encuentran almacenados localmente. Esto hará posible su acceso sin conexión a Internet. En la figura A.15 se muestra la pantalla de historial.

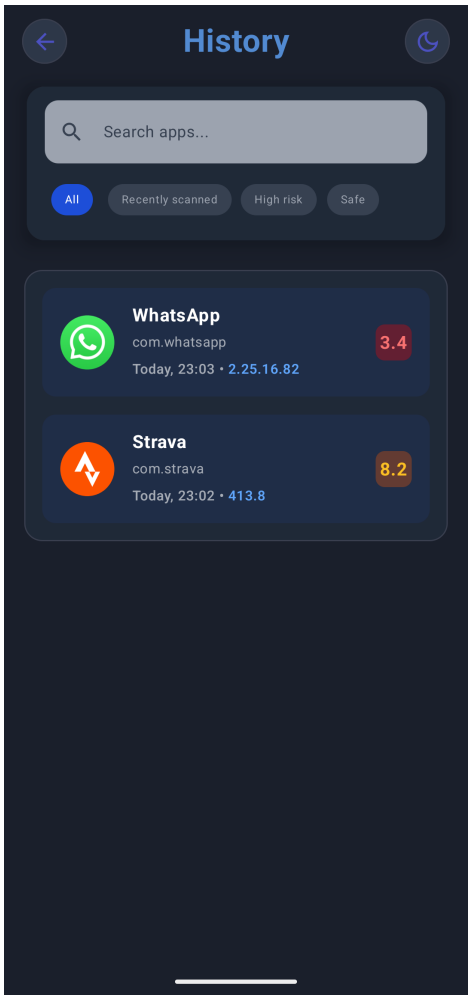
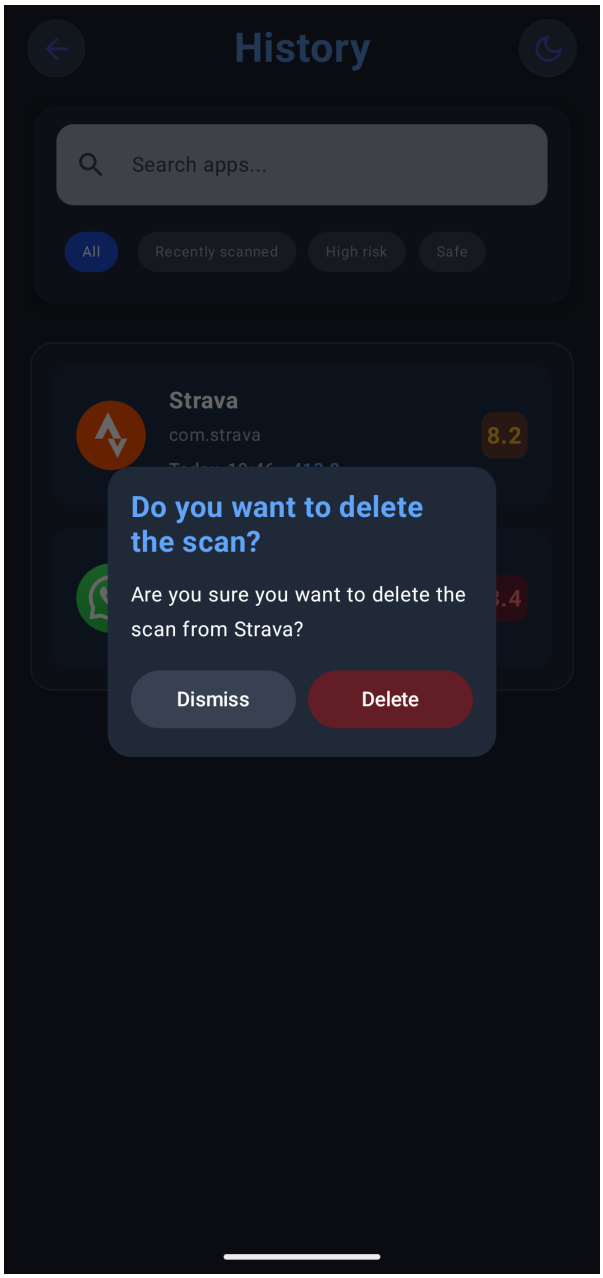
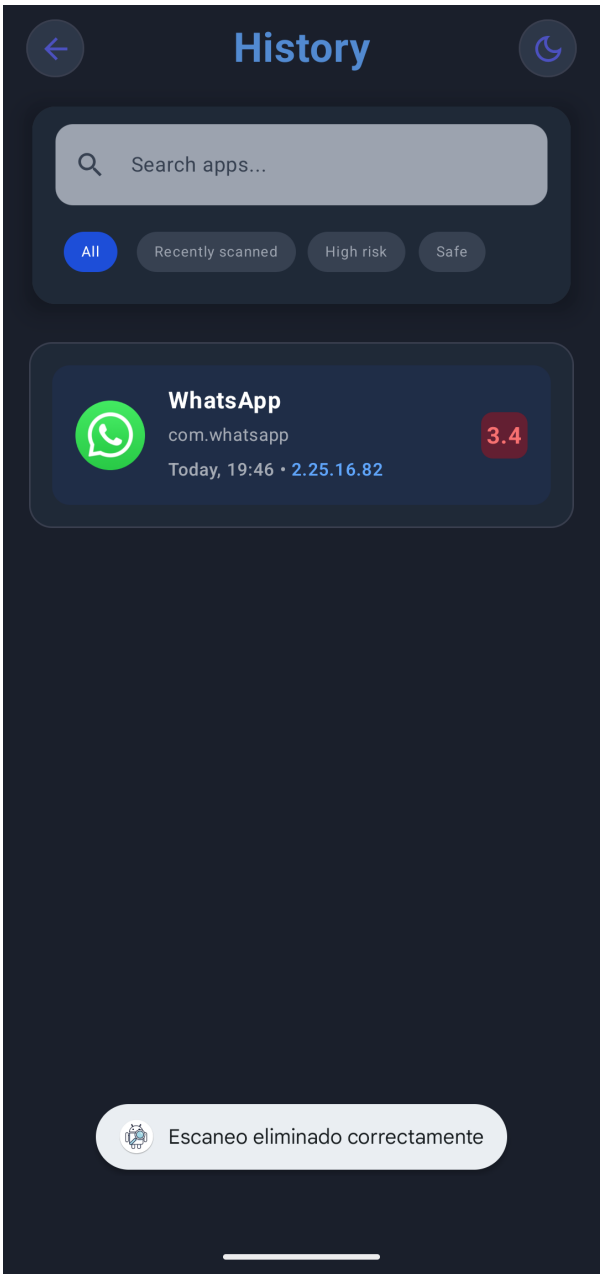


Figura A.15: Pantalla de historial.

En la figura A.15 se puede ver como se tiene almacenado el reporte de las aplicaciones *WhatsApp* y *Strava* pudiendo acceder a estos pulsando sobre las tarjetas correspondientes. Como se ha mencionado anteriormente, también se da la funcionalidad de buscar, filtrar y ordenar los reportes. Además, si se quiere eliminar alguno se puede dejar pulsado sobre la tarjeta correspondiente lo que desplegará un cuadro de diálogo con la opción de eliminar el reporte. Como se puede ver en las figuras A.16a y A.16b se ha eliminado el reporte de *Strava*.



(a) Cuadro de diálogo para confirmar el borrado del reporte de *Strava*.



(b) Resultado del análisis de la política de privacidad de *WhatsApp*.

Figura A.16: Borrado del reporte de *Strava*.

Apéndice B

Manual de instalación y despliegue

En este apartado se detalla el proceso para desplegar el entorno del sistema completo

- Aplicación *ApkAudit*: se indican los pasos necesarios para descargar el IDE *Android Studio* con el repositorio de la aplicación Android.
- Servidor de procesamiento: se indican los pasos necesarios para descargar, instalar y desplegar el servidor de procesamiento al completo.

B.1. Aplicación *ApkAudit*

1. El primer paso es descargar el IDE *Android Studio* para el desarrollo e instalación de la aplicación *ApkAudit*. Para realizar esto, descargamos el IDE desde el siguiente enlace:

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

2. Una vez descargado el IDE e instalados los controladores, seleccionamos la opción "Get from VCS"para clonar el repositorio de *ApkAudit* e introducimos la siguiente URL:

   `https://gitlab.inf.uva.es/jagarci/apkaudit`

3. Tras clonar el repositorio, abrimos el proyecto en Android Studio e instalamos todas las dependencias necesarias. Mediante el botón "Sync now".

4. A continuación, se debe establecer las APIs de Keys de *VirusTotal* y del servidor de procesamiento.

   En el caso, de *VirusTotal*, se debe crear una cuenta en este servicio y obtener la clave correspondiente.

   En el caso del servidor de procesamiento, la clave es la siguiente:

   `Smdx00VcgDyf68etb7J6aQfLl0V0NkfFWb4DQGcLVbQ`

   Por último, las claves se introducen en el fichero *local.properties* del proyecto.

   ```
   VIRUSTOTAL_API_KEY=<VIRUSTOTAL_API_KEY>
   SERVER_API_KEY=Smdx00VcgDyf68etb7J6aQfLl0V0NkfFWb4DQGcLVbQ
   ```

5. Una vez finalizado, podemos iniciar la app clicando el botón de "Run".

B.2. Servidor de procesamiento

1. El primer paso instalar las dependencias necesarias en el equipo que actuará de servidor. En este caso, instalamos todas las dependencias de *Docker* y de *Docker compose* mediante la guía oficial [44].

2. Tras instalar las dependencias, clonamos el repositorio del código del servidor de procesamiento. Para ello, se ejecuta el comando siguiente:

   ```
   git clone https://gitlab.inf.uva.es/jagarci/apkaudit-server
   ```

3. El comando anterior descargará el contenido del repositorio en una carpeta llamada "apkaudit-server". Nos movemos al directorio:

   ```
   cd apkaudit-server/
   ```

4. Desplegamos todos los contenedores definidos en el fichero *docker-compose.yml*, es decir, el correspondiente al Reverse Proxy (*HAProxy*) y a *FastAPI*. Para realizar esto, ejecutamos el comando siguiente:

   ```
   sudo docker compose up -d -build
   ```

5. Por último, para comprobar que todo se ha desplegado correctamente realizamos una petición al servidor ejecutando el siguiente comando:

   ```
   curl http://localhost:8000/data-safety-info.json
   ```

   Si se ha desplegado correctamente deberiamos recibir un json con información acerca de los tipos básicos del *Data Safety* de Google y con un código HTTP 200.

Apéndice C

Ejemplo de JSON devuelto en el endpoint de /get/compare-privacy


```
{
    "privacy_policy_url": "https://www.whatsapp.com/legal/privacy-policy
       ",
    "declared_data_types": [
        "App interactions",
        "Approximate location",
        "Contacts",
        "Crash logs",
        "Device or other IDs",
        "Diagnostics",
        ...
    ],
    "policy_mentioned_data_types": [
        "Address",
        "App interactions",
        "Approximate location",
        "Calendar events",
        "Contacts",
        "Crash logs",
        "Diagnostics",
        ...
    ],
    "missing_from_declaration": [
        "Address",
        "Calendar events",
        "Emails",
        "Fitness info",
        "Name",
        ...
    ],
    "shared_data": [],
```

```
    "collected_data": [
        {
            "data": "Email address",
            "optional": 1,
            "purpose": "App functionality, Fraud prevention, security,
               and compliance",
            "type": "Personal info"
        },
        {
            "data": "User IDs",
            "optional": 0,
            "purpose": "App functionality, Analytics, Fraud prevention,
               security, and compliance, Account management",
            "type": "Personal info"
        },
        {
            "data": "Phone number",
            "optional": 0,
            "purpose": "App functionality, Analytics, Fraud prevention,
               security, and compliance",
            "type": "Personal info"
        },
        ...
    ],
    "security_practices": [
        {
            "practice": "Data is encrypted in transit",
            "description": "Your data is transferred over a secure
               connection"
        },
        {
            "practice": "You can request that data be deleted",
            "description": "The developer provides a way for you to
               request that your data be deleted"
        }
    ],
    "error": null
}
```

Apéndice D

Enlaces adicionales

- **Enlace al repositorio de *GitLab* con el código fuente de la aplicación móvil**: `https://gitlab.inf.uva.es/jagarci/apkaudit`
- **Enlace al repositorio de *GitLab* con el código fuente del servidor**: `https://gitlab.inf.uva.es/jagarci/apkaudit-server`
- **Enlace a los prototipos de la interfaz creados en *Figma***: https://www.figma.com/design/ApkAudit
- **Enlace a los resultados de la prueba de usabilidad escaneados**: `https://doi.org/10.5281/zenodo.15652323`