

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“PRIVACY AUDITING PLATFORM FOR ANDROID APPLICATIONS THROUGH THE PERMISSION ANALYSIS”

JUAN BARROS ACUÑA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: FELIPE BEROIZA OSSES
Profesor Correferente: MARCELLO VISCONTI

Abril - 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): Memoria o trabajo de título; ☒ Tesis de Postgrado;

Título del trabajo:

Nombre del candidato(a): Juan Ignacio Barros Acuña

Carrera / Grado: Ingeniería Civil Informatica

Campus: ; Casa central **Departamento:** Informatica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Felipe Beroiza Osses, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

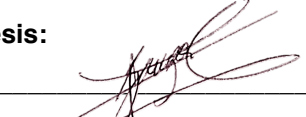
El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

6 meses; 12 meses; 2 años; 3 años; 5 años; 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: ; **Firma:** 

Estudiante o Candidato(a):

Fecha: ; **Firma:** 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

Este trabajo representa la culminación de mi estadía universitaria, es por esto que se lo quiero dedicar a mi familia ya que fueron un soporte clave en mi experiencia universitaria apoyando, a mi madre Marta, mi padre Juan, mi hermana Pamela, mis sobrinas Julieta y Abigail. También a mi pareja Sol que me ha apoyado durante los últimos 11 años de relacion.

Aprovecho también agradecer a mi profesor guía Felipe Beroiza por su dedicación durante todo el proceso, también a Pablo Cruz que junto a Felipe ayudaba con la discusión de ideas sobre la memoria. A mis compañeros de investigación Ivan, Emmanuel, Felipe, Benjamin, Maria Paz, Tomas, Diego, Carlos y Juan Nemiña.

Finalmente agradezco de corazón a mis amigos y compañeros de estudio que he conocido en este breve pero intenso camino por Ingeniería Civil e Ingeniería Civil Informática. Sin su compañía, apoyo y buena onda, esto no habría sido posible.

RESUMEN

Resumen— El uso masivo de dispositivos Android plantea riesgos para la privacidad de los usuarios debido a la compleja gestión de permisos. Aunque Android implementa el principio de mínimo privilegio, la amplia gama de aplicaciones puede derivar en la fuga de datos sensibles. Para abordar este problema, se diseñó e implementó una aplicación que audita los permisos de aplicaciones preinstaladas y de terceros, informando a los usuarios sobre posibles fugas de información sensible y evaluando el nivel de riesgo de cada aplicación. El sistema integra técnicas de Procesamiento de Lenguaje Natural (NLP) como servicio adicional para analizar descripciones de aplicaciones, evaluar los permisos peligrosos, y resumir políticas de privacidad. La solución fue validada con pruebas unitarias e integración continua, asegurando su calidad y efectividad. Su impacto reside en aumentar la conciencia y control sobre la privacidad, contribuyendo a la ciberseguridad móvil y estableciendo bases para investigaciones futuras.

Palabras Clave— Análisis de permisos; Seguridad Android; Privacidad.

ABSTRACT

Abstract— The widespread use of Android devices poses privacy risks for users due to the complex management of permissions. Although Android follows the principle of least privilege, the large number of applications can lead to the leakage of sensitive data. To address this issue, an application was designed and implemented to audit the permissions of pre-installed and third-party applications, informing users about potential sensitive data leaks and evaluating the risk level of each application. The system integrates Natural Language Processing (NLP) techniques as an additional service to analyze app descriptions, assess dangerous permissions, and summarize privacy policies. The solution was validated through unit testing and continuous integration, ensuring its quality and effectiveness. Its impact lies in raising awareness and control over privacy, contributing to mobile cybersecurity and establishing a foundation for future research.

Keywords— Permission analysis; Android Security; Privacy.

GLOSARIO

- **Play Store:** Tienda de aplicaciones de Android.
- **Android:** Sistema operativo basado en el kernel de Linux, diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas.
- **API (Application Programming Interface):** Conjunto de rutinas, protocolos y herramientas para construir software y aplicaciones. Permite que diferentes componentes de software se comuniquen entre sí.
- **AWS (Amazon Web Services):** Proveedor de servicios cloud de Amazon.
- **APK (Android Application Package):** Formato de archivo utilizado por Android para distribuir e instalar aplicaciones móviles.
- **BERT (Bidirectional Encoder Representations from Transformers):** Modelo de procesamiento de lenguaje natural preentrenado, diseñado para entender el contexto de las palabras en una oración.
- **Bluetooth:** Tecnología de comunicación inalámbrica de corto alcance para el intercambio de datos entre dispositivos electrónicos.
- **Container Docker:** Plataforma que permite empaquetar una aplicación y sus dependencias en un contenedor que puede ejecutarse en cualquier sistema Linux.
- **Flutter:** Framework de código abierto de Google para desarrollar aplicaciones nativas móviles, web y de escritorio desde una única base de código.
- **GitHub Actions:** Herramienta de automatización de software para flujos de trabajo de integración continua (CI) y despliegue continuo (CD) en GitHub.
- **GPL (General Public License):** Licencia de software libre que garantiza la libertad de usar, estudiar, compartir y modificar el software.
- **Kotlin:** Lenguaje de programación que corre sobre la JVM, enfocado en simplicidad y productividad. Soportado oficialmente para desarrollo Android.
- **Linux:** Sistema operativo tipo Unix, de código abierto y distribuido bajo licencia GPL.
- **Lorem ipsum:** Texto genérico de relleno.
- **ML (Machine Learning):** Conjunto de técnicas que permiten a las computadoras aprender tareas basadas en patrones de datos, sin ser explícitamente programadas.
- **NFC (Near Field Communication):** Tecnología de comunicación inalámbrica de corto alcance y alta frecuencia entre dispositivos cercanos.
- **PackageManager:** Clase de Android encargada de gestionar aplicaciones y paquetes instalados en el dispositivo.

- **Python:** Lenguaje de programación de alto nivel, interpretado, con énfasis en la simplicidad y legibilidad del código.
- **Room Database:** Biblioteca que abstrae la capa de acceso a datos en Android y facilita el uso de SQLite.
- **Scraper:** Programa que extrae datos de páginas web.
- **SDK (Software Development Kit):** Conjunto de herramientas para desarrollar aplicaciones en una plataforma específica.
- **SQLite:** Sistema de gestión de bases de datos relacional, ligero y autónomo.
- **WORA (Write Once, Run Anywhere):** Principio que permite que el código funcione en cualquier plataforma sin modificaciones.
- **Declarative Programming:** Método de desarrollo que especifica lo que se quiere lograr, como en SQL o HTML.
- **OEM (Original Equipment Manufacturer):** Fabricante de equipo original.
- **Bytecode:** Código intermedio generado tras compilar una aplicación Android, que es interpretado o ejecutado por la máquina virtual en el dispositivo.
- **Intent:** Objeto utilizado en Android para iniciar componentes del sistema, como actividades o servicios, y permitir la comunicación entre ellos.

ÍNDICE DE CONTENIDOS

RESUMEN	III
ABSTRACT	III
GLOSARIO	IV
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	X
CAPÍTULO 1: INTRODUCCION	1
1.1 Motivación y contexto.	1
1.2 Definición del problema.	2
1.3 Propuesta de Solución	3
1.4 Objetivo General	3
1.4.1 Objetivos específicos	4
1.5 Estructura del documento.	4
CAPÍTULO 2: MARCO CONCEPTUAL	5
2.1 ANDROID OPERATIVE SYSTEM	5
2.1.1 ANDROID PLATFORM ARCHITECTURE	5
2.2 ANDROID APPLICATIONS	8
2.2.1 APLICACIÓN NATIVA	9
2.2.2 CROSS-PLATFORM APPLICATIONS	9
2.2.3 ARQUITECTURA DE APLICACIONES ANDROID	12
2.3 PRIVACIDAD Y SEGURIDAD EN ANDROID	13
2.4 ARQUITECTURAS DE PRESENTACIÓN	15
2.4.1 MODEL VIEW INTENT	15
2.4.2 MODEL VIEW PRESENTER	15
2.4.3 MODEL-VIEW-VIEWMODEL	16
2.5 PATRONES DE DISEÑO	16
2.5.1 INYECCIÓN DE DEPENDENCIAS	16
2.5.2 SINGLETON	17
2.6 ANDROID DEVELOPMENT	17
2.6.1 ACTIVITY	17
2.6.2 JETPACK COMPOSE	18
2.6.3 ROOM DATABASE	18
2.6.4 DAGGER HILT	19
2.6.5 BROADCAST RECEIVER	19
2.7 CLOUD COMPUTING	19
2.7.1 SERVERLESS COMPUTING	20
2.7.2 FUNCTION AS A SERVICE	20
2.8 ESTADO DEL ARTE	20

2.8.1	PERMISOS DE ANDROID	20
2.8.2	Análisis de permisos	21
2.8.3	ANÁLISIS DE PERMISOS A TRAVÉS DE DESCRIPCIÓN	21
2.8.4	ANÁLISIS DE POLÍTICA DE PRIVACIDAD	22
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN		24
3.1	Características	24
3.2	Elección Tecnológica	24
3.3	Componentes	25
3.3.1	Sistema auditor de privacidad a través del análisis de permisos	25
3.3.2	Android Triggers	27
3.3.3	Servicios externos	27
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		29
4.1	Metodología propuesta	29
4.2	Definición de requerimientos	29
4.2.1	Requerimientos funcionales	29
4.2.2	Requerimientos no funcionales	30
4.2.3	Especificación de casos de usos	30
4.3	Diseño del sistema y del software.	33
4.3.1	Diagrama de contexto	33
4.3.2	Modelo de dominio	34
4.3.3	Arquitectura de alto nivel	35
4.3.4	Historias de usuario	37
4.3.5	Diagrama de despliegue	38
4.3.6	Diagramas de paquete	40
4.3.7	Base de datos	42
4.4	Implementación y prueba de unidad.	44
4.4.1	ExtractPermissionInfoService	44
4.4.2	API	44
4.4.3	Configuración de aplicación Android.	46
4.4.4	Configuración de base de datos	47
4.4.5	Flujo de datos entre paquetes	50
4.4.6	Comunicación a API	51
4.4.7	Calculadora de riesgo	51
4.4.8	Primer Análisis	52
4.4.9	Análisis a una aplicación	53
4.4.10	Análisis todas las aplicaciones	54
4.4.11	Scheduler Diario	55
4.4.12	Pruebas unitarias	56
4.5	Integración y prueba del sistema.	57
4.5.1	Integración continua	57
4.5.2	[HU-01] Listar aplicaciones instaladas	59
4.5.3	[HU-02] Abrir detalle aplicación	61

4.5.4	[HU-03] Abrir editar permisos	62
4.5.5	[HU-04] Descargar informe aplicación	66
4.5.6	[HU-05] Resumen de política de privacidad	70
4.5.7	[HU-06] Ver política de privacidad	71
4.5.8	[HU-07] Grupos de permisos según descripción	73
4.5.9	[HU-08] Ver descripción	74
4.5.10	[HU-09] Ver análisis de riesgo	76
4.5.11	[HU-10] Historial de cambios en permisos	78
4.5.12	[HU-11] Nuevo análisis	80
4.5.13	[HU-12] Nuevo análisis por aplicación	82
4.6	Operación y mantenimiento.	83
5	CONCLUSIONES	86
	REFERENCIAS BIBLIOGRÁFICAS	89

ÍNDICE DE FIGURAS

1	Arbol del problema	3
2	Pila de software de Android	6
3	Flujo de trabajo general para usar permisos en Android.	14
4	Activity Lifecycle	18
5	Categorías de datos sensibles	21
6	Grupos de permisos AC-Net	22
7	Arquitectura de componentes	25
8	Diagrama Casos de usos con análisis de riesgo.	31
9	Diagrama Casos de usos con análisis descripción.	32
10	Diagrama Casos de usos con resumen política de privacidad.	33
11	Diagrama de contexto	34
12	Modelo de dominio	35
13	Arquitectura alto nivel	36
14	Diagrama de despliegue	39
15	Diagrama de paquete: Aplicación	40
16	Diagrama de paquete: API web	41
17	Diseño de base de datos	43
18	Estructura de API	45
19	Ejecución API en GIN GONIC	46
20	Librerías usadas en android kotlin	47
21	Base de datos configurada	48
22	Base de datos configurada	49
23	Flujo de datos entre paquetes	50
24	Interfaz de API	51
25	Calculadora de riesgo	52
26	Primer Análisis	53
27	Analizar una aplicación	54
28	Análisis a todas las aplicaciones	55
29	Diagrama de clases Scheduler Service	55
30	Configuración Scheduler	56
31	Pruebas unitarias	57
32	GitHub Integración continua	58
33	Resultado GitHub Integración continua	59
34	[HU-01] - Primer inicio	60
35	[HU-01] - Segundo inicio	60
36	[HU-01] listado aplicaciones instaladas	61
37	[HU-02] Vista detalle de aplicación parte a	62
38	[HU-02] Vista detalle de aplicación parte b	62
39	[HU-03] Dialogo ir a editar permisos	63
40	[HU-03] Información de aplicación desde configuraciones	64
41	[HU-03] Vista de permisos con permisos no aceptados	65
42	[HU-03] Vista de permisos con permisos aceptados	66

43	[HU-04] Dialogo descargar informe	67
44	[HU-04] Compartir informe	68
45	[HU-04] Json exportado parte a	69
46	[HU-04] Json exportado parte b	70
47	[HU-05] Vista resumen de política de privacidad	71
48	[HU-06] Dialogo ir a política de privacidad	72
49	[HU-06] Política de privacidad Aliexpress	73
50	[HU-07] Grupos de permisos según descripción	74
51	[HU-08] Dialogo ir tienda de aplicaciones	75
52	[HU-08] Tienda de aplicaciones para Aliexpress	76
53	[HU-09] Informe de riesgo parte a	77
54	[HU-09] Informe de riesgo parte b	78
55	[HU-10] Vista sin cambios	79
56	[HU-10] Cambios en permisos	80
57	[HU-11] Nuevo análisis: Sin cambios	81
58	[HU-11] Nuevo análisis: Cambios	82
59	[HU-12] Nuevo análisis por aplicación	83

ÍNDICE DE TABLAS

1	EVALUATION OF CROSS-PLATFORM CATEGORIES	11
2	EVALUATION OF CROSS-PLATFORM CATEGORIES	12
3	Categorías y nombres para Permisos peligrosos	26
4	Risk Value and Permissions Combination for Android Apps	27
5	Listado grupo de permisos	31
6	Categorías resumen política privacidad	32
7	Historias de usuario	37
8	Historias de usuario	38
9	Pruebas de caja negra para el método calculateRisk.	57
10	Estado casos de uso	84
11	Repositorios	85

CAPÍTULO 1

INTRODUCCION

1.1. Motivación y contexto.

La sociedad actual ha crecido en la era de los smartphones. En cada bolsillo u hogar se encuentran diversos dispositivos, no solo celulares, sino también televisores y tablets, de los cuales un 72,15 % utilizan el sistema operativo Android al año 2024¹. El uso de aplicaciones es tan común que se emplean para el entretenimiento, redes sociales, operaciones financieras e incluso para trabajar². Esto convierte a los usuarios de Android como a los de otros dispositivos basados en aplicaciones en objetivos atractivos para los ciberdelincuentes.

Según la documentación de Android³ cada aplicación reside en su propia zona de pruebas de seguridad y está protegida por las siguientes características de seguridad de Android:

- El sistema operativo Android es un sistema linux multiusuario en el que cada aplicación es un usuario diferente. Lo que limita el acceso a recursos propios de cada aplicación a otras aplicaciones.
- De forma predeterminada el sistema le asigna a cada aplicación un identificador de usuario de linux único y desconocido por la aplicación. El sistema establece permisos para todos los archivos en una aplicación de modo que solo el identificador de usuario asignado a esa aplicación pueda acceder a ellos. Esto permite que cada usuario administre las actividades que a través de permisos puede acceder cada aplicación
- Cada aplicación es un proceso y cada proceso tiene su propia máquina virtual(VM), por lo que el código de una aplicación se ejecuta de forma independiente de otras aplicaciones. De esta forma se asegura que cada ejecute sin requerimientos de versiones entre aplicaciones, junto con también, aislar el comportamiento entre aplicaciones y periféricos.

De esta manera, el sistema Android implementa el principio de mínimo privilegio. Es decir, de forma predeterminada, cada aplicación tiene acceso solo a los componentes que necesita para llevar a cabo su trabajo y nada más. Esto crea un ambiente muy seguro, donde una aplicación solo puede acceder a las partes en las que tiene permiso.

¹<https://es.statista.com/grafico/29620/sistema-operativo-movil-con-la-mayor-cuota-de-mercado-por-pais/>

²<https://www.appbrain.com/stats/android-market-app-categories>

³<https://developer.android.com/guide/components/fundamentals>

1.2. Definición del problema.

A pesar de tanta seguridad en Android, existen formas en las que una aplicación puede compartir datos con otras aplicaciones y en las que también puede acceder a servicios del sistema:

- Es posible coordinar que dos aplicaciones tengan el mismo identificador de usuario linux, y con ello compartan el sistema de archivos. Las aplicaciones deben estar firmadas con el mismo certificado⁴.
- A través de la gestión de permisos, una aplicación puede solicitar acceder a los datos del dispositivo, como los contactos de un usuario, los mensajes de texto, el dispositivo de almacenamiento (tarjeta SD), la cámara y la conexión de bluetooth.
- El usuario por su parte, debe conceder de manera explícita estos permisos⁵.

Los permisos de la app ayudan a admitir la privacidad del usuario, ya que protegen el acceso a lo siguiente⁶:

- Datos restringidos: como el estado del sistema y la información de contacto del usuario
- Acciones restringidas: como conectarte a un dispositivo vinculado y grabar audio.

Los permisos permiten que las aplicaciones obtengan información sensible, conduciendo a una posible fuga de información y violación de privacidad[Faruki *et al.*, 2015]. Por ejemplo, en marzo del 2018, el escándalo de Facebook-Cambridge Analytica involucró la colección de datos de 87 millones de usuarios de Facebook con información personal de identificación a través de las apps⁷ resultando en discusiones públicas y hacer que las personas pongan más atención a la protección de su información privada, esto se puede ver resumido en la figura 1.

⁴<https://stackoverflow.com/questions/9783765/what-is-shareduserid-in-android-and-how-is-it-used>

⁵<https://www.avg.com/en/signal/guide-to-android-app-permissions-how-to-use-them-smartly>

⁶<https://developer.android.com/guide/topics/permissions/overview>

⁷<https://www.bbc.com/mundo/noticias-49093124>

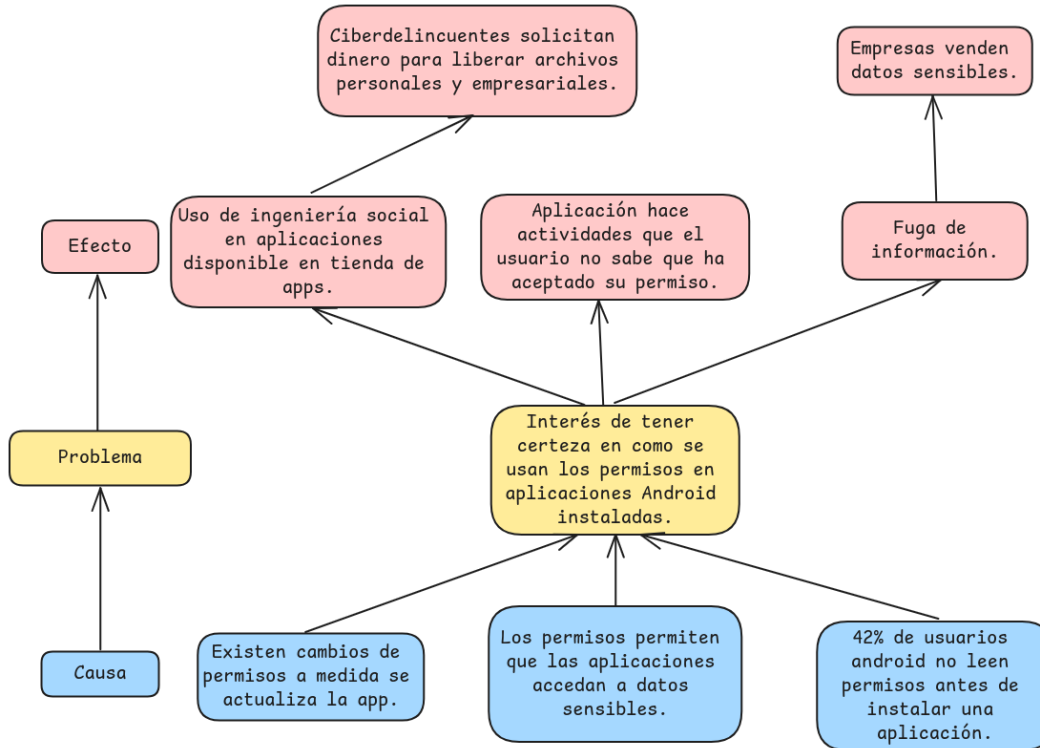


Figura 1: Arbol del problema
Fuente: Elaboración propia.

1.3. Propuesta de Solución

Se propone diseñar un sistema que analice los permisos de aplicaciones Android, así identificar y catalogar la posibilidad de fugas de información privada del usuario, esto mediante la implementación de una calculadora de riesgo alimentada por los permisos de la aplicación y la capacidad de la aplicación para exportar estos datos desde el dispositivo a un tercero.

1.4. Objetivo General

Diseñar e implementar un software que audite la privacidad de aplicaciones Android a través del análisis de permisos.

1.4.1. Objetivos específicos

- Construir aplicación que analice el dispositivo Android para que obtenga una lista de las aplicaciones instaladas y los permisos asignados.
- Hacer una evaluación de riesgo de permisos que están siendo ocupados por las aplicaciones preinstaladas y de terceros.
- Presentar reporte de análisis de descripción y resumen de política de privacidad.

1.5. Estructura del documento.

En las siguientes secciones se revisará lo siguiente:

- Marco Conceptual: Se profundizará en conceptos claves que sirven como base para comprender la la propuesta de solución.
- Propuesta de Solución: Se explicaran las características que debe tener el sistema, es el primer acercamiento a su descripción de arquitectura y el informe de riesgo propuesto.
- Validación de Solución: Capitulo en el cual se desarrolla lo propuesto y se valida mediante la metodología de cascada.
- Se finaliza con conclusiones y retroalimentación de lo desarrollado.

CAPÍTULO 2

MARCO CONCEPTUAL

A continuación se describe el marco conceptual que servirá como base para el desarrollo de la plataforma que audite privacidad a través del análisis de permisos para aplicaciones android.

2.1. ANDROID OPERATIVE SYSTEM

2.1.1. ANDROID PLATFORM ARCHITECTURE

Android es una pila de software de código abierto basada en Linux creada para una variedad de dispositivos móviles y factores de forma como relojes, televisores entre otros. En la Figura 2 se muestran los componentes principales de la plataforma Android⁸⁹.

Kernel de Linux La base de la plataforma Android es el *kernel* de Linux. Por ejemplo, el tiempo de ejecución de Android (*Android Runtime*, ART) se basa en el kernel de Linux para funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel.

El uso de un *kernel* de Linux permite que Android aproveche funciones de seguridad clave y que los fabricantes de dispositivos desarrollen controladores de hardware para un *kernel* conocido.

Capa de abstracción de hardware (HAL) La Capa de Abstracción de Hardware (*Hardware Abstraction Layer*, HAL) brinda interfaces estándar que exponen las capacidades de hardware del dispositivo al marco de trabajo de la API de Java de nivel más alto. La HAL consta de varios módulos de biblioteca, cada uno de los cuales implementa una interfaz para un tipo específico de componente de hardware, como la cámara o el módulo Bluetooth. Cuando el marco de trabajo de una API realiza una llamada para acceder al hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.

Tiempo de ejecución de Android (ART) Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (*Android Runtime*, ART). El ART está diseñado para

⁸<https://developer.android.com/guide/platform?hl=es-419>

⁹<https://source.android.com/docs/core/architecture?hl=es-419>

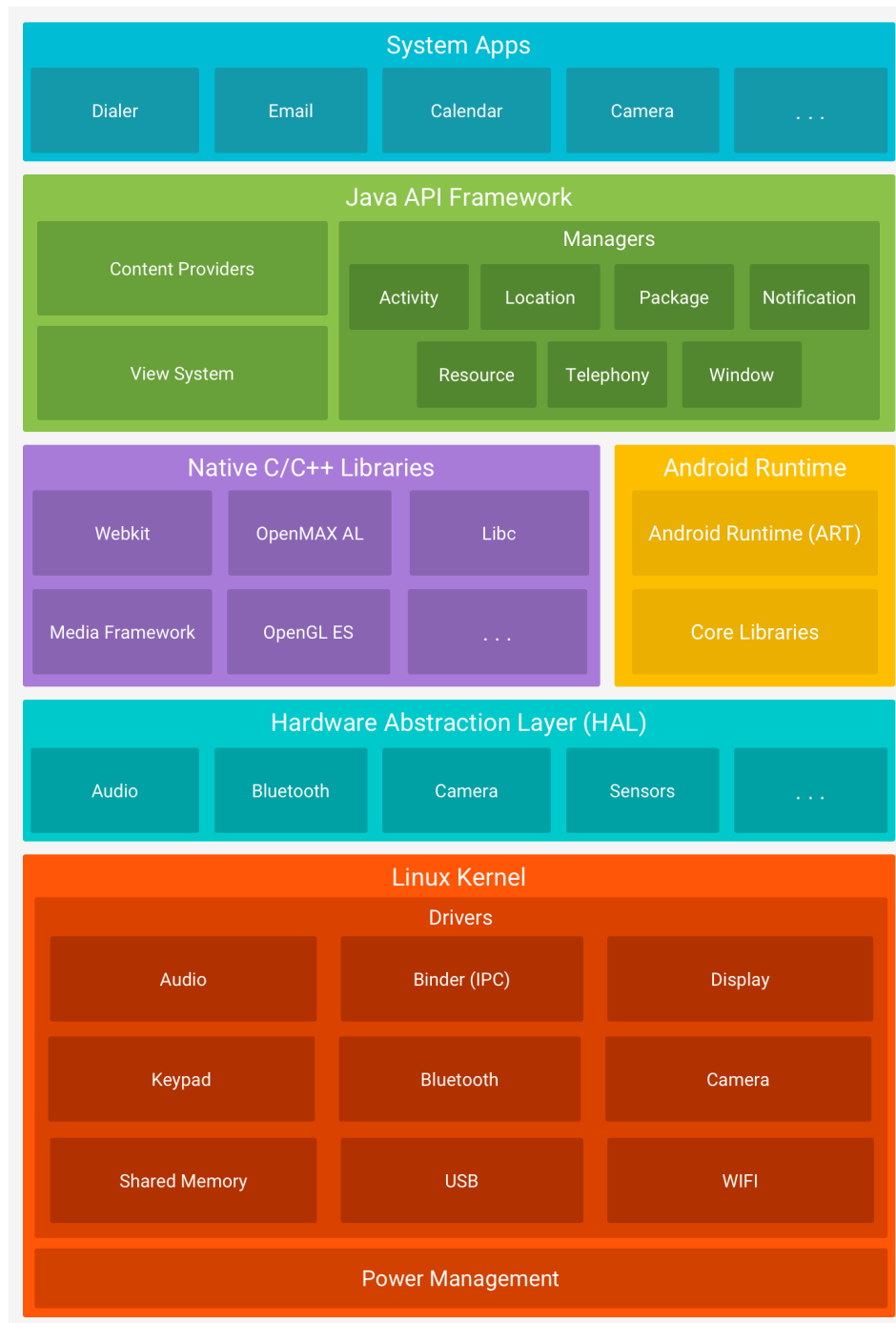


Figura 2: Pila de software de Android

ejecutar varias máquinas virtuales en dispositivos con poca memoria, utilizando archivos en formato *Dalvik Executable* (DEX), un formato de código de bytes diseñado específicamente para Android y optimizado para un uso de memoria mínimo.

Entre las funciones principales del ART se encuentran:

- Compilación *ahead-of-time* (AOT) y *just-in-time* (JIT).
- Recolección optimizada de elementos no utilizados (*garbage collection*, GC).
- Conversión de archivos DEX a código máquina más compacto en Android 9 (nivel de API 28) y versiones posteriores.

Antes de Android 5.0, Dalvik era el entorno de ejecución del sistema operativo. Si una aplicación se ejecuta bien en ART, es probable que también funcione en Dalvik, aunque no necesariamente al contrario.

Bibliotecas C/C++ nativas Muchos componentes y servicios centrales del sistema Android, como ART y HAL, se compilan a partir de código nativo que requiere bibliotecas escritas en C y C++. La plataforma Android proporciona APIs del *framework* de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las aplicaciones. Por ejemplo, se puede acceder a *OpenGL ES* a través de la API de Java de OpenGL en Android para agregar soporte de gráficos 2D y 3D.

Para aplicaciones que requieren C o C++, los desarrolladores pueden usar el NDK de Android para acceder a las bibliotecas nativas directamente.

Framework de la API de Java El conjunto completo de funciones del sistema operativo Android está disponible a través de APIs escritas en Java. Estas APIs permiten a los desarrolladores construir aplicaciones reutilizando componentes del sistema, tales como:

- Un sistema de vista extensible para construir la interfaz de usuario, como listas, cuadrículas, cuadros de texto, botones, e incluso un navegador web integrable.
- Un administrador de recursos que proporciona acceso a recursos sin código, como *strings* localizadas, gráficos y archivos de diseño.
- Un administrador de notificaciones que permite a todas las aplicaciones mostrar alertas personalizadas en la barra de estado.
- Un administrador de actividades que gestiona el ciclo de vida de las aplicaciones y proporciona una pila de actividades de navegación común.
- Proveedores de contenido que permiten a las aplicaciones acceder a datos de otras aplicaciones, como la app de contactos, o compartir sus propios datos.

Aplicaciones del sistema Android incluye un conjunto de aplicaciones principales para correo electrónico, mensajería SMS, calendarios, navegación por internet y contactos. Las aplicaciones incluidas en la plataforma no tienen un estado especial entre las aplicaciones que el usuario elige instalar. Algunas excepciones, como la app de Configuración, siguen estando presentes como parte integral del sistema.

2.2. ANDROID APPLICATIONS

Android OS representa aproximadamente un 72.15 % de la cuota del mercado mundial, lo que lo convierte en la plataforma móvil dominante¹⁰. Esto significa que, si una empresa o desarrollador quiere llegar al mayor número posible de usuarios, Android es una opción estratégica clave.

Uno de los factores que explica su popularidad, según la literatura, es que Android es un sistema operativo de código abierto (open-source). Esto implica que no existen restricciones de mercado para su distribución, lo que facilita su adopción por fabricantes y desarrolladores. Gracias a esto, los desarrolladores pueden crear y distribuir sus propias aplicaciones libremente, sin depender de una única tienda o ecosistema cerrado.

Además, la plataforma Android proporciona un amplio conjunto de herramientas que permiten acceder a las interfaces de programación de aplicaciones (APIs). Estas herramientas hacen posible que las aplicaciones interactúen con recursos del sistema operativo y el hardware del dispositivo, como la cámara, el GPS, sensores, entre otros [Almomani y Khayer, 2020].

Cada aplicación en Android es instalada a través de un archivo APK el cual es un paquete comprimido que contiene todo lo que la aplicación necesita para ser ejecutada como su bytecode, archivos de recursos y metadata. Esta APK es instalada al invocar al servicio del sistema Package Manager. La manera oficial y más usada en el mundo es a través de Google Play, una aplicación con elevados permisos que conecta con el Mercado de aplicaciones Google y descarga la aplicación solicitada, habilitando también un sistema de reputación. Los usuarios pueden invocar al Package Installer al descargar o copiar una aplicación al almacenamiento del dispositivo y solicitando a Android que lo abra. Finalmente, para usuarios avanzados, Android Debug Bridge(ADB) puede ser usado para instalar aplicaciones a través de Android Software Development Kit(SDK)[Alepis y Patsakis, 2019].

¹⁰<https://es.statista.com/grafico/29620/sistema-operativo-movil-con-la-mayor-cuota-de-mercado-por-pais/>

2.2.1. APLICACIÓN NATIVA

El desarrollo de aplicaciones nativas nace mediante la construcción de aplicaciones escritas en Java, al igual que su propio código fuente¹¹. Actualmente, Google sugiere escribir aplicaciones con Kotlin¹², y es que la interoperabilidad que ofrece para trabajar con código Java, junto con su implementación de corrutinas, lo hace bastante poderoso para entregar una aplicación de alto rendimiento[Laurence *et al.*, 2021]. Las corrutinas son un patrón de diseño de simultaneidad que se puede ocupar en Android para simplificar el código que se ejecuta de forma asíncrona (como llamadas a APIs, procesos, etc.)¹³

2.2.2. CROSS-PLATFORM APPLICATIONS

[Shah *et al.*, 2019] hace una comparación entre aplicaciones Cross-platform y nativas:

Ventajas:

- Las herramientas Cross-platform son además conocidas como WORA (Write once, run anywhere), que es la primera ventaja de dicho método. La aplicación es implementada usando un solo código fuente, pero puede ser desplegada en múltiples plataformas, esto quiere decir que no está restringida a una sola plataforma.
- Las herramientas Cross-platform usualmente usan lenguajes de programación bien conocidos, lo que lo hace una fácil y rápida manera de desarrollar.
- La mayoría de estas herramientas tienen una amplia comunidad y son de código abierto por naturaleza. Por lo que nuevas características son añadidas constantemente y no está limitado a un cerrado grupo de desarrolladores.
- Se reducen costos ya que el desarrollo es simple y un tipo de desarrollador es necesario a diferencia de varios equipos especializados en plataformas
- En consecuencia, las actualizaciones y el arreglo de errores puede ser desplegado para todas las plataformas.

Desventajas:

- Las aplicaciones nativas son diseñadas para trabajar para la plataforma en específico, existe una latencia mínima con lo rápido del tiempo de renderizado que brinda el CPU. La performance es mucho mayor en comparación a su contraparte Cross-platform

¹¹<https://source.android.com/?hl=es-419>

¹²<https://developer.android.com/kotlin?hl=es-419>

¹³<https://developer.android.com/kotlin/coroutines?hl=es-419>

- Las aplicaciones nativas tienen una profunda integración con el dispositivo. Existencia de APIs para todas las características del dispositivo como cámara, servicios basados en ubicación entre otros.
- Las aplicaciones nativas proveen una rica experiencia de usuario. Renderizando gráficos con altas terminaciones, lo que es posible con el desarrollo de aplicaciones nativas.
- El deployment a la tienda de aplicaciones de la respectiva plataforma es usualmente mal visto en el caso de enfoques Cross-platform. Ellos tienen la más baja prioridad en la lista de recomendaciones. Por ejemplo, algunas apps que simplemente inyectan código HTML y Javascript en un contenedor nativo son rechazadas de ser publicadas.
- Se vuelve algo común que se congele la aplicación o que se rompa la aplicación en ejecución. Esto se debe a que las aplicaciones Cross-platform no tienen soporte para todos los dispositivos, lo que vuelve impredecible su comportamiento.

Y además, categoriza las aplicaciones Cross-platform en [Shah *et al.*, 2019] [Dalmaso *et al.*, 2013]:

- Web Apps: Una aplicación web es una app que usa el navegador de internet para ejecutarse, además, tiene un acceso limitado al hardware del dispositivo móvil, ejemplos populares son: Angular, JQuery, Django y otros.
- Hybrid Apps: Estas apps son esencialmente desarrolladas usando HTML y Javascript pero son inyectadas dentro de software llamados contenedores. Apache cordova, previamente llamado PhoneGap es un popular contenedor usado para desarrollar aplicaciones híbridas.
- Interpreted Apps: Estas apps emulan aplicaciones nativas permitiendo al usuario interactuar con componentes de interfaz de usuarios que son específicos para una plataforma en particular. Estas aplicaciones pueden ser desarrolladas usando Ruby, XML, Javascript etc. React Native es una tecnología de código abierto desarrollada por Facebook en el 2015, en la categoría de Interpreted Apps es conocida como frontrunner.
- Widget Based Apps: Todos los componentes en estas aplicaciones son widgets que sirven como bloques de construcción para formar la interfaz de usuario. Estas apps usan su propio motor gráfico para generar código nativo. Flutter fue inicialmente lanzada en mayo del 2017 y se ha convertido en una popular alternativa en los desarrolladores de aplicaciones Cross-platform.

Luego, siguiendo la tabla que resume las evaluaciones de cada categoría de cross-platform2.

Tabla 1: EVALUATION OF CROSS-PLATFORM CATEGORIES

Fuente: [Shah *et al.*, 2019]

Parámetros	Native	Web apps	Hybrid apps	Interpreted apps	Widget Based (Flutter)
UI / UX	Excelente	Moderado	Moderado	Bastante Bueno	Muy Bueno
Usuarios Potenciales	Limitados	Máximos	Grandes	Grandes	Limitados
Costo de Desarrollo	Alto	Bajo	Bajo	Moderado	Moderado
Seguridad de la Aplicación	Muy Alta	Muy Baja	Baja	Moderada	Alta
Facilidad de Actualización	Baja	Alta	Variada	Variada	Moderada
Complejidad de Implementación	Alta	Baja	Moderada	Baja a Moderada	Moderada

Tabla 2: EVALUATION OF CROSS-PLATFORM CATEGORIES

 Fuente: [Shah *et al.*, 2019]

Parámetros	Native	Web apps	Hybrid apps	Interpreted apps	Widget Based (Flutter)
Acceso a APIs Nativas	Sí	Sí, solo con HTML5	Sí, a través de complementos	Sí	Sí
Entornos de Desarrollo Preferidos	Android Studio, XCode, Momentics	Visual Studio Code, PhpStorm, WebStorm	Eclipse Hybrid Mobile Tools (THyM), Intel XDK	Nuclide usando Atom IDE, Appcelerator Studio	Android Studio, IntelliJ
Licencia	Android: Open-source, iOS: Closed-source	Open-source	Generalmente open-source	Open-source	Open-source
Lenguaje	Java, Kotlin, Swift, Objective-C, etc.	HTML, CSS, JavaScript, TypeScript, etc.	HTML, CSS, JavaScript, Node.js, etc.	JavaScript, JSX, UX Markup, etc.	Dart
Publishing to Marketplace	Yes	No	Yes	Yes	Yes

2.2.3. ARQUITECTURA DE APLICACIONES ANDROID

La arquitectura de aplicaciones Android moderna se basa en la organización de los componentes en capas bien definidas, que separan la lógica de presentación, de datos y de dominio. Esto permite que cada parte de la aplicación sea más fácil de gestionar y modificar sin afectar otras áreas del código.¹⁴

- **UI Layer:** Interactúa con el usuario y se actualiza en función de los cambios en los datos.

¹⁴<https://developer.android.com/topic/architecture?hl=es-419>

Utiliza `ViewModel` para separar la lógica de UI del acceso a los datos.

- **Data Layer:** Gestiona el acceso a las fuentes de datos. En este nivel, se manejan los repositorios que encapsulan los detalles de implementación del almacenamiento y la red.
- **Domain Layer:** (Opcional) Contiene la lógica de negocio que puede ser utilizada en múltiples partes de la aplicación.

Este enfoque facilita la reutilización de componentes y asegura que las clases estén enfocadas en una única responsabilidad, haciendo que la aplicación sea escalable y más fácil de mantener.

2.3. PRIVACIDAD Y SEGURIDAD EN ANDROID

Los permisos de la app ayudan a mantener segura la privacidad del usuario, ya que protegen el acceso a lo siguiente:

- Datos restringidos, como el estado del sistema y la información de contacto de los usuarios.
- Acciones restringidas, como conectarte a un dispositivo vinculado y grabar audio.

Flujo de trabajo para el uso de permisos: Si una aplicación ofrece funcionalidades que pueden requerir acceso a datos restringidos o acciones restringidas, determina si puedes obtener la información o realizar las acciones sin necesidad de declarar permisos. Se pueden cumplir muchos casos de uso al construir una aplicación, como tomar fotos, pausar la reproducción de contenido multimedia y mostrar anuncios relevantes, sin necesidad de declarar ningún permiso.

Si la aplicación debe acceder a datos restringidos o realizar acciones restringidas a fin de cumplir con un caso de uso, se debe declarar los permisos adecuados. Algunos permisos, conocidos como permisos en el momento de la instalación, se otorgan automáticamente cuando se instala la aplicación. Otros permisos, conocidos como permisos de tiempo de ejecución, requieren que la aplicación vaya un paso más allá y solicite el permiso durante el tiempo de ejecución.

Tipos de permisos: Android categoriza los permisos en diferentes tipos, incluidos los del momento de la instalación, los permisos de tiempo de ejecución y los permisos especiales. Cada tipo de permiso indica el alcance de los datos restringidos a los que puede acceder la aplicación, así como el alcance de las acciones restringidas que puede realizar, cuando el sistema le otorga ese permiso. El nivel de protección de cada permiso se basa en su tipo y se muestra en la página de referencia de la API de permisos.

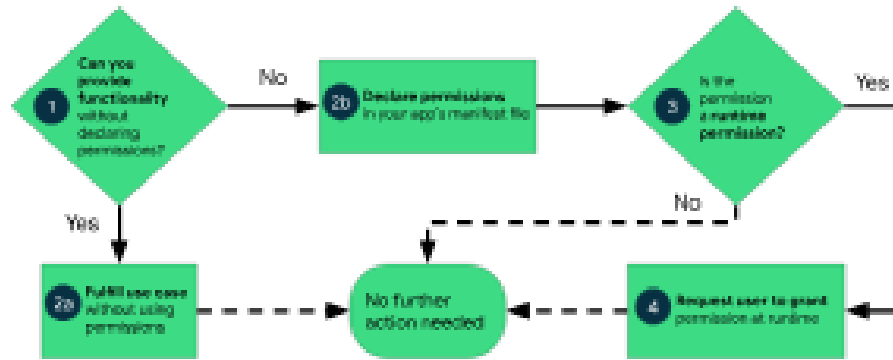


Figura 3: Flujo de trabajo general para usar permisos en Android.

- **Permisos en el momento de la instalación:** Estos permisos otorgan a la app acceso limitado a datos restringidos o permiten que realice acciones restringidas que casi no afectan al sistema ni a otras apps.
- **Permisos normales:** Permiten el acceso a datos y acciones que suponen un riesgo mínimo para la privacidad del usuario.
- **Permisos de firma:** Se otorgan solo a apps firmadas por el mismo certificado que la app o el S.O. que define el permiso.
- **Permisos de tiempo de ejecución:** También conocidos como permisos peligrosos, requieren que la app solicite el permiso durante la ejecución.
- **Permisos especiales:** Correspondientes a operaciones particulares de la app, definidos por la plataforma y los OEM.

Grupos de permisos: Los permisos pueden pertenecer a grupos de permisos, que consisten en un conjunto de permisos relacionados de forma lógica. Los grupos de permisos ayudan al sistema a minimizar la cantidad de diálogos que se presentan al usuario cuando una app solicita permisos relacionados.

Prácticas recomendadas: Los permisos de la app se basan en las funciones de seguridad del sistema y ayudan a Android a cumplir con objetivos relacionados con la privacidad del usuario, como el control de datos, la transparencia y la minimización de datos. En esta sección, se presentan algunas prácticas recomendadas para usar los permisos de manera efectiva en tu app.

2.4. ARQUITECTURAS DE PRESENTACIÓN

El patrón original para aplicaciones con interfaces de usuario fue el patrón Modelo Vista Controlador(Model View Controller). La innovación introducida por el patrón es que la vista, que es renderizada en la pantalla, sea siempre consistente. Hizo esto insistiendo en un flujo unidireccional de datos.

Primero todo parte con el usuario, el ve algo en la pantalla, y responde a lo que ve con un input o interacción con esta. Ellos pueden tocar la pantalla, escribir algo, hablar, lo que sea. Ellos hacen algo que cambia el estado de la aplicación.

El input del usuario es capturado por el controlador. El controlador tiene dos responsabilidades. La primera es accionar la interacción del usuario, esto mediante un proceso. La segunda es traducir las acciones del usuario en operaciones sobre el modelo.

El modelo es la lógica del negocio de una aplicación. Esta contiene algunos tipos de almacenamientos persistentes y conexiones a nivel de red, con reglas para combinar e interpretar las entradas del usuario. En una arquitectura MVC ideal, este es el único componente que mantiene el estado actual de la aplicación.

Si bien el patrón MVC fue revolucionario, este abrió el paso para hacer mejoras.

2.4.1. MODEL VIEW INTENT

Una de las más viejas versiones de MVC adoptado por la comunidad Android fue llamado Model-View-Intent. El patrón desacopla la Activity del modelo usando Intents dentro de sus payloads. Mientras la estructura produce excelentes componentes aislados, puede ser un poco lento y el código para construir los intents un tanto voluminoso; este fue altamente usado, nuevos patrones lo han suplantado[Laurence *et al.*, 2021].

2.4.2. MODEL VIEW PRESENTER

A diferencia del patrón anterior, este incorpora el concepto de Local Model, que se refiere a mantener aislado el estado de las variables en la vista. De esta forma la vista tiene una instancia de el local Model con una suerte de Factory Pattern, y luego la vista es registrada con este.

En este patrón la vista se responsabiliza de los inputs del usuario en su Presentador. El presentador responde a los eventos actualizando el Local Model y el Model State como sea necesario. Entonces, le avisa a la vista que necesita dibujarse a si misma. Esto fuerza a la vista a actualizar solamente las secciones afectadas.

El atributo mas importante de esta arquitectura es que el Presentador puede ser testeado unitariamente. Ya que permite hacer un mock de la interfaz que la vista provee a el presentador para aislarlo completamente de la vista. Vistas pequeñas en cuanto a código y presentadores testeables tienen como fin lograr aplicaciones mas robustas [Laurence *et al.*, 2021].

2.4.3. MODEL-VIEW-VIEWMODEL

Google con la introduccion de Jetpack, da soporte a una arquitectura llamada Model-View-ViewModel(MVVM). Esto porque es soportado internamente por el Framework de Android moderno, este es el mas común y mas discutido patrón para aplicaciones de android modernas.

En el patrón MVVM, como es usual, una Activity o un Fragment toman el rol de la vista. El código de la vista es lo mas simple como es posible, a menudo contenido como una subclase dentro de una Activity o un Fragment.

The ViewModel es responsable de unir juntos los comandos necesarios para actualizar la vista y el Model del backend. La característica novedosa de este patrón es que una sola interfaz, Observable, es usado para transmitir cambios en el estado del Local Model hacia la Vista.

En vez de usar múltiples interfaces Presenter usados en el patrón MVP, el ViewModel representa data visible como una colección de Observables. La vista se registra como un observador de esos Observables y reacciona a la notificación de cambios en los datos que ellos contienen. La librería Jetpack llama a estos Observables LiveData.

Tal como MVP, MVVM hace que el hacer Mock y las pruebas unitarias algo fácil. La nueva característica importante que introduce es el lifecycle awarenes(conciencia del ciclo de vida de la variables o los datos) [Laurence *et al.*, 2021].

En resumen es una técnica que expone el estado a la IU y encapsula la lógica empresarial relacionada. Su principal ventaja es que almacena en caché el estado y lo conserva durante los cambios de variables. Esto significa que la IU no tiene que recuperar datos cuando navegas entre actividades o si sigues cambios de configuración, como cuando rotas la pantalla[Developers, 2024].

2.5. PATRONES DE DISEÑO

2.5.1. INYECCIÓN DE DEPENDENCIAS

La inyección de dependencias (DI) es un mecanismo para mejorar la modularidad del software. La DI permite un menor acoplamiento entre módulos al evitar que estos sean conscientes

de los detalles de implementación de los demás. La DI se ha convertido en una práctica común en la industria del software, como lo demuestra la existencia de frameworks de DI y publicaciones al respecto orientadas a la industria [Laigner *et al.*, 2019]. En el caso de android, existe una librería llamada Hilt, que implementa la inyección de dependencias¹⁵.

2.5.2. SINGLETON

El patrón Singleton es un patrón de diseño creacional que tiene como objetivo garantizar que una clase tenga una única instancia durante el ciclo de vida de una aplicación, proporcionando además un punto de acceso global a dicha instancia. Este patrón fue propuesto originalmente por el grupo conocido como Gang of Four [Gamma *et al.*, 1994].

El patrón Singleton resuelve dos problemas al mismo tiempo, vulnerando el Principio de responsabilidad única [Refactoring Guru, 2023]:

- Garantizar que una clase tenga una única instancia.
- Proporcionar un punto de acceso global a dicha instancia.

2.6. ANDROID DEVELOPMENT

La siguiente sección corresponde a las tecnologías sugeridas por Android para un desarrollo moderno.

2.6.1. ACTIVITY

El componente de Activity gestiona una sola página de la interfaz gráfica de la aplicación. Este usa una librería rica de android de "widgets" para dibujar una sola e interactiva página. Widgets como botones, cajas de texto son los elementos básicos de una Activity y ellos son combinados para formar representaciones de la vista.

Es importante diferenciar un Activity de una aplicación. Las Activities son efímeras, y garantizan su existencia solo mientras la página que ella maneja es visible. Mientras la página se vuelve invisible, se preserva la instancia de activity o cualquier estado que este representa [Laurence *et al.*, 2021].

La figura, representa la máquina de estado que controla el ciclo de vida de una actividad, esto quiere decir que por cada tipo de acción, se detona un evento el cual puede ser interpretado

¹⁵<https://dagger.dev/hilt/>

a nivel de código y hacer los procedimientos necesarios para las características que se estén desarrollando.

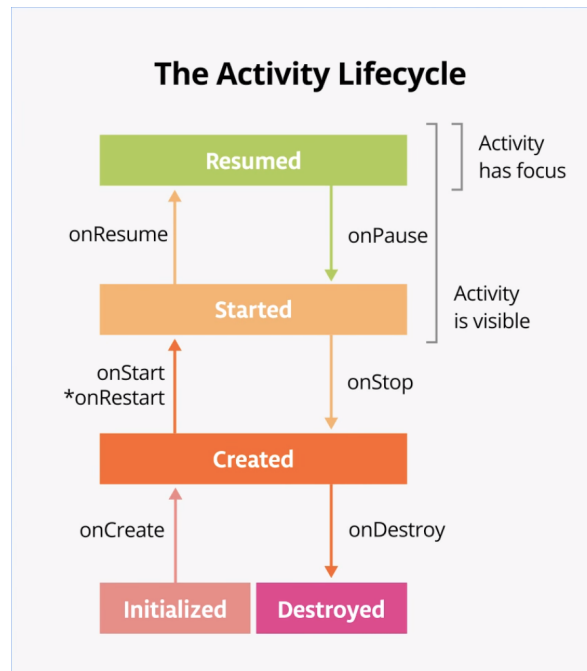


Figura 4: Activity Lifecycle

Fuente: CodeLabs Android kotlin[Android Developers, 2024]

2.6.2. JETPACK COMPOSE

Jetpack Compose es un moderno kit de herramientas declarativas de IU para Android. Compose facilita la escritura y el mantenimiento de la IU de la app, y proporciona una API declarativa que te permite renderizarla sin cambiar las vistas del frontend de manera imperativa¹⁶. Esto quiere decir que usando el paradigma de programación declarativo a través de código kotlin, es posible renderizar interfaces de usuario

2.6.3. ROOM DATABASE

Las apps que controlan grandes cantidades de datos estructurados pueden beneficiarse con la posibilidad de conservar esos datos localmente. El caso de uso más común consiste en almacenar en caché datos relevantes para que el dispositivo no pueda acceder a la red, de modo que el usuario pueda explorar ese contenido mientras está sin conexión.

¹⁶<https://developer.android.com/jetpack/compose/mental-model?hl=es-419>

La biblioteca de persistencias Room brinda una capa de abstracción para SQLite que permite acceder a la base de datos sin problemas y, al mismo tiempo, aprovechar toda la tecnología de SQLite. En particular, Room brinda los siguientes beneficios:

- Verificación del tiempo de compilación de las consultas en SQL
- Anotaciones de conveniencia que minimizan el código estándar repetitivo y propenso a errores
- Rutas de migración de bases de datos optimizadas

Se recomienda usar Room en lugar de usar las APIs de SQLite directamente.

2.6.4. DAGGER HILT

Hilt es una biblioteca de Inyección de dependencias para Android que permite reducir el trabajo repetitivo de inyectar dependencias de forma manual en el proyecto.

Hilt proporciona una forma estándar de usar la inserción de dependencias en tu aplicación, ya que proporciona contenedores para cada clase de Android en el proyecto y administra automáticamente sus ciclos de vida. Hilt se basa en la popular biblioteca de inserción de dependencias Dagger y se beneficia de la corrección en tiempo de compilación, el rendimiento del entorno de ejecución, la escalabilidad y la compatibilidad con Android Studio que proporciona.

2.6.5. BROADCAST RECEIVER

El concepto original para un BroadcastReceiver era de un tipo de data bus. Esto quiere decir que es un Listener(Escuchador) que se pueden subscribir con el fin de obtener notificaciones de un evento que sea del interés. Un ejemplo es el Intent `android.intent.action.BOOT_COMPLETED`, es publicado por el sistema operativo de Android, después de un reinicio de sistema, una aplicación podría registrar este broadcast para realizar una operación una vez se haya reiniciado el sistema.

2.7. CLOUD COMPUTING

En la siguiente sección se revisaran conceptos para el diseño de una arquitectura que pueda usar componentes en la nube.

2.7.1. SERVERLESS COMPUTING

La Standard Performance Evaluation Corporation (SPEC) Research Group define serverless como un modelo de computación impulsado por eventos y sus operaciones escalan elásticamente, los usuarios solo pagan por los recursos que necesitan, con granularidad fina y la lógica operativa se abstrae de los usuarios [van Eyk *et al.*, 2019].

2.7.2. FUNCTION AS A SERVICE

En los últimos años FaaS se ha convertido en un servicio de Cloud Computing muy popular, basando su popularidad a la experiencia que brinda sin servidor, en la que la complejidad de implementar recursos, la ejecución de aplicaciones, la administración y gestión de recursos informáticos se delega a la plataforma, lo que proporciona a los usuarios agilidad en el desarrollo y despliegue de sus aplicaciones. Esto contrastado a servicios de cloud computing convencionales como la infraestructura como servicio(IaaS) que tienen tiempos de inicio mas lentos.

Se espera que las aplicaciones estén diseñadas usando la arquitectura de microservicios, Las funciones FaaS son impulsados por eventos, que desencadenan solicitudes de ejecución de funciones. Se pueden configurar una variedad de eventos para activar funciones, por ejemplo: eventos HTTP, inserciones en la base de datos, creación/eliminación de un archivo, finalización de otra función, step functions¹⁷ entre otros[Sheshadri y Lakshmi, 2022].

En la referencia [Fox *et al.*, 2017] los autores, a través de un workshop hecho por IBM, definen serverless y las FaaS como una plataforma nativa de la nube, para la computación sin estado de corta duración, para aplicaciones impulsadas por eventos, que se escala instantánea y automáticamente hacia arriba y hacia abajo, por ultimo, cobrar por el uso real con una granularidad de milisegundos.

2.8. ESTADO DEL ARTE

2.8.1. PERMISOS DE ANDROID

Dos autores discuten los resultados de dos preguntas claves sobre el uso de permisos. ¿Usted lee los permisos de las aplicaciones al momento de ser instaladas? con un 42.9 % de personas que nunca leyeron los permisos de la aplicación. ¿Se ha negado a instalar una aplicación en vista de los permisos solicitados por la app? con un 52.12 % de encuestados que se niegan a hacerlo[Ngobeni y Mhlongo, 2019]. Además, proponen un modelo conceptual para mejorar la actual seguridad en dispositivos Android

¹⁷<https://aws.amazon.com/es/step-functions/>

Por otro lado, [Almomani y Khayer, 2020] proponen un documento que provee un comprensivo auto contenido estudio del sistema de permisos Android. Contiene la historia y una actualización del estado de los permisos Android, en adición una definición formal de sus componentes, un análisis comparativo de permisos para el top aplicaciones compartido a la comunidad de investigadores a través de resultados y dataset generados en el estudio.

2.8.2. Análisis de permisos

Un enfoque para auditar privacidad mediante el análisis de permisos, se basa en los requerimientos de seguridad de los usuarios, como el acceso seguro a datos y la protección de información sensible para diseñar un sistema de evaluación de riesgo para smartphones que categoriza el nivel de riesgo según una combinación de permisos peligrosos, identificando categorías de datos sensibles, que si son expuestas por combinaciones de salidas de datos del sistema para una aplicación, se les evalúa con un numero de riesgo de 1 a 5 [Irwan *et al.*, 2015]. En la siguiente imagen se muestran las categorías de datos sensibles.

No.	Sensitive Data Category	Permissions
1.	Account information	GET_ACCOUNTS and USE_CREDENTIALS
2.	Browser history and bookmarks	READ_HISTORY_BOOKMARKS
3.	Calendar	READ_CALENDAR
4.	Calling information	READ_CALL_LOG
5.	Contacts and profile	READ_CONTACTS and READ_PROFILE
6.	Location	ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION
7.	Media	CAMERA and RECORD_AUDIO
8.	Messages	READ_SMS, RECEIVE_SMS, READ_VOICE_MAIL, RECEIVE_MMS
9.	Network information	ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE
10.	Phone information	READ_PHONE_STATE
11.	External storage data	READ_EXTERNAL_STORAGE and MANAGE_DOCUMENTS

Figura 5: Categorías de datos sensibles

Fuente: [Irwan *et al.*, 2015]

En la siguiente se muestran las combinaciones de permisos asociados a la salida de datos del dispositivos junto con un permiso que pertenezca a la categoría de dato sensible

2.8.3. ANÁLISIS DE PERMISOS A TRAVÉS DE DESCRIPCIÓN

La tienda de aplicaciones es un foco común para el descubrimiento y descarga de nuevas aplicaciones. Analizar las descripciones de las aplicaciones es una forma útil de examinar si algunos permisos son necesarios para las aplicaciones. AC-NET, por su parte, evalúa la

coherencia entre la descripción de una aplicación en la tienda y sus permisos reales mediante procesamiento de lenguaje natural, se enfoca en los permisos de las aplicaciones desde una perspectiva más detallada y buscan predecir los permisos correspondientes a una oración de la descripción de la aplicación[Feng *et al.*, 2019]. Identificando 16 permisos pertenecientes a 11 grupos de permisos peligrosos para mantener la privacidad.

Protection Levels	Permission Groups	Permission
DANGEROUS	STORAGE	WRITE_EXTERNAL_STORAGE
	CONTACTS	GET_ACCOUNTS
		READ_CONTACTS
		WRITE_CONTACTS
	LOCATION	ACCESS_FINE_LOCATION
		ACCESS_COARSE_LOCATION
	CAMERA	CAMERA
	MICROPHONE	RECORD_AUDIO
	SMS	READ_SMS
		SEND_SMS
	CALL_LOG	READ_CALL_LOGS
SIGNATURE	PHONE	CALL_PHONE
	CALENDAR	READ_CALENDAR
NORMAL	SETTINGS ¹	WRITE_SETTINGS
	TASKS ²	GET_TASKS
		KILL_BACKGROUND_PROCESS

Figura 6: Grupos de permisos AC-Net
Fuente: [Feng *et al.*, 2019]

2.8.4. ANÁLISIS DE POLÍTICA DE PRIVACIDAD

Con el rápido desarrollo de las aplicaciones móviles y su amplia adopción en diferentes dominios, cada vez más datos personales han sido proporcionados a los proveedores de aplicaciones. La política de privacidad es un documento que establece el acuerdo legal entre los proveedores de servicios y los usuarios [Liu *et al.*, 2020], describiendo en sí a que datos e información sensible puede acceder el proveedor de aplicaciones o servicios.

Entonces, en [Liu *et al.*, 2020] proponen una tecnica para analizar automáticamente las políticas de privacidad, reduciendo la tarea de clasificación a nivel de párrafo, y comparándolo en cuanto al impacto que tiene con GDPR¹⁸. Es por ello que el autor en la tecnica clasifica cada parrafo en cuanto los siguientes elementos:

1. **Introducción a la Política (PI):** Descripciones generales del documento de política de privacidad, incluidas definiciones de pronombres referenciales.
2. **Recopilación y Uso de la Primera Parte (FPCU):** Detalles sobre qué, cuándo y cómo el controlador recopila, utiliza y procesa la información de los usuarios.
3. **Cookies y Tecnologías Similares (CT):** Cómo se recopilan y utilizan las cookies y otras tecnologías similares, con descripciones de estas técnicas.

¹⁸<https://gdpr-info.eu/>

4. **Compartir y Recopilar de Terceros (TPSC):** Cómo el controlador comparte y divulga información con terceros, incluidos afiliados corporativos, proveedores de servicios o socios publicitarios.
5. **Derecho y Control del Usuario (URC):** Derechos del usuario garantizados por el GDPR y opciones disponibles para controlar su información personal.
6. **Seguridad de los Datos (DS):** Métodos y facilidades de seguridad implementados por el controlador para proteger la información de los usuarios.
7. **Retención de Datos (DR):** Descripciones sobre el período de retención de la información de los usuarios.
8. **Transferencia Internacional de Datos (IDT):** Cómo se almacena y transfiere la información a nivel internacional.
9. **Audiencias Específicas (SA):** Términos específicos para audiencias particulares, como niños o sujetos de datos de áreas/países específicos.
10. **Cambio de Política (PC):** Información sobre cambios en las políticas de privacidad y cómo se notifican estos cambios.
11. **Información de Contacto de la Política (PCI):** Datos de contacto del controlador de datos.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

A partir del análisis realizado en el marco teórico, se propone el desarrollo de un sistema auditor de privacidad para dispositivos Android, que permita analizar las aplicaciones instaladas en un dispositivo, evaluando los permisos que solicitan y otorgando un valor de riesgo en función del uso de información sensible. Este sistema también permitirá llevar un registro histórico de los análisis realizados, notificando al usuario sobre cambios relevantes, como la instalación de nuevas aplicaciones o modificaciones en los permisos otorgados.

La solución contempla el diseño de una aplicación móvil para Android, apoyada por servicios externos que permitirán enriquecer el análisis con información adicional como políticas de privacidad y descripciones obtenidas desde las tiendas de aplicaciones. A continuación, se presentan las características principales del sistema, las tecnologías seleccionadas para su desarrollo y los componentes que lo conforman.

3.1. Características

Se considerarán las siguientes características, derivadas de los capítulos anteriores.

- Analizar el dispositivo obteniendo un listado de aplicaciones instaladas.
- Analizar la aplicación obteniendo el listado de permisos otorgados y no otorgados, también un valor de riesgo de la aplicación.
- Registrar un histórico de análisis de las aplicaciones, con la evolución de los permisos otorgados vs totales y un numero de riesgo.
- Analizar una aplicación cuando: sea el primer análisis del dispositivo, se actualice una aplicación.
- Registre permiso eliminado
- Registre aplicación eliminada.

3.2. Elección Tecnológica

Se seleccionará el uso de Android con kotlin¹⁹ dado que es lo que se sugiere para proyectos nuevos, esto para ser usado como cliente mobile, y la extracción de datos del dispositivo.

¹⁹<https://developer.android.com/kotlin?hl=es-419>

También se acompañará de apis en python para el uso de modelos de machine learning. Y también un api en golang para consumir las apis anteriores.

3.3. Componentes

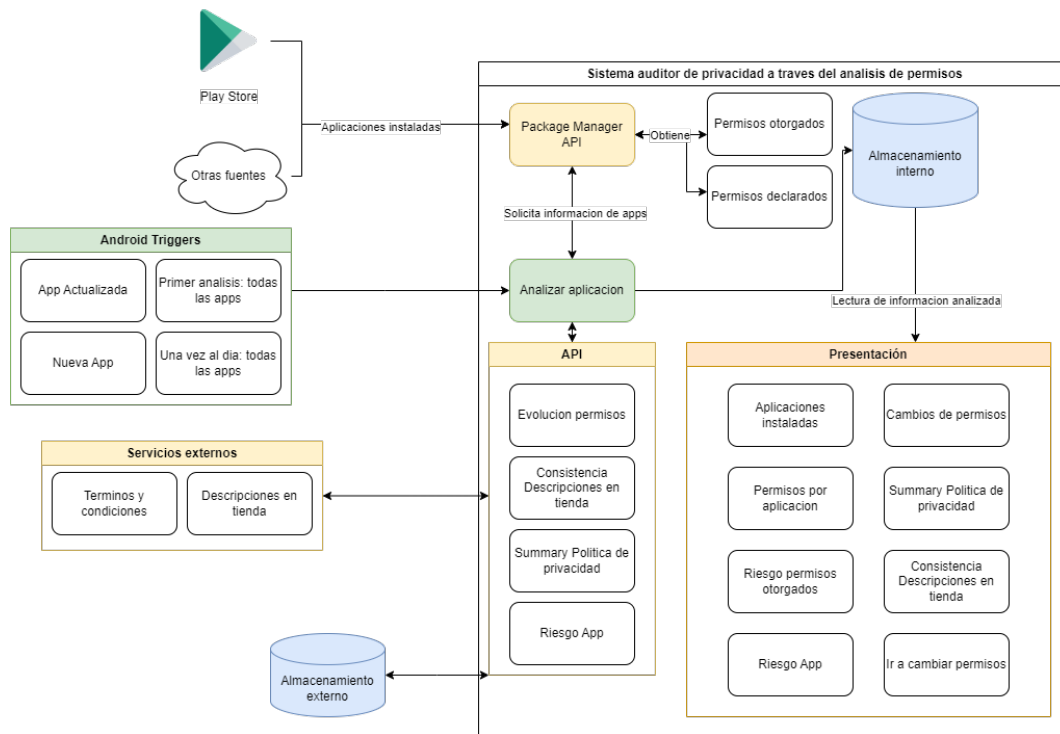


Figura 7: Arquitectura de componentes
Fuente: Elaboración propia.

La figura [7] representa la arquitectura de componentes donde **Sistema auditor de privacidad a través del análisis de permisos** es en lo que nos enfocaremos en este trabajo, se describirá cada uno de los componentes y subcomponentes.

3.3.1. Sistema auditor de privacidad a través del análisis de permisos

El sistema auditor corresponde a la solución propuesta. Primero, una aplicación puede ser instalada desde play Store y otras fuentes, cuando se analiza una aplicación, una API de android que es el Package Manager, es una fuente de información con la que se pueden obtener datos de los permisos otorgados y permisos declarados por la aplicación a analizar, tambien, se pueden obtener el listado de aplicaciones instaladas. El sistema propuesto debe implementar un sistema de calculo de riesgo, definido por una actualización a [Irwan *et al.*, 2015],

considerando los siguientes permisos como pertenecientes a permisos peligrosos para la privacidad del usuario.

PermissionInfo Category	PermissionInfo Name
Account Information	GET_ACCOUNTS
Calendar	READ_CALENDAR
Calling information	READ_CALL_LOG
Contacts and profile	READ_CONTACTS
Location	ACCESS_FINE_LOCATION
Location	ACCESS_COARSE_LOCATION
Location	ACCESS_MEDIA_LOCATION
Location	ACCESS_LOCATION_EXTRA_COMMANDS
Media	CAMERA
Media	RECORD_AUDIO
Media	READ_MEDIA_VIDEO
Media	READ_MEDIA_IMAGES
Media	READ_MEDIA_AUDIO
Messages	READ_SMS
Messages	RECEIVE_SMS
Messages	READ_VOICEMAIL
Messages	RECEIVE_MMS
Messages	RECEIVE_WAP_PUSH
Network information	ACCESS_NETWORK_STATE
Network information	ACCESS_WIFI_STATE
Phone information	READ_PHONE_STATE
External storage data	READ_EXTERNAL_STORAGE
External storage data	MANAGE_DOCUMENTS

Tabla 3: Categorías y nombres para Permisos peligrosos

Entonces, dado que cada categoría anterior corresponde a la de un permiso sensible, es que se crea la siguiente tabla 4:

Donde, sensitive data permission corresponde a tener al menos un permiso de la aplicación correspondiente a una categoría de permiso peligrosa. Por lo tanto, si la aplicación tiene al menos un permiso en esta categoría, ya es de riesgo 1(Very Low), por otro lado, el nivel de riesgo 5(very high) es cuando aparte de la aplicación tener capturado un dato proveniente de una categoría sensible, entonces comparte esta categoría por internet o un sms, e incluso por bluetooth o NFC.

Por otro lado. El sistema tendrá un almacenamiento interno para ir guardando información como si fuera una memoria de caché, donde se guardará el listado de aplicaciones, el registro de cambios, los permisos otorgados, declarados y la información consultada a los servicios externos.

Risk Value	Permissions Combination
5 (Very High)	Sensitive data permission and (INTERNET or SEND_SMS) and (BLUETOOTH or NFC)
4 (High)	Sensitive data permission and (INTERNET or SEND_SMS)
3 (Medium)	Sensitive data permission and (BLUETOOTH or NFC)
2 (Low)	Sensitive data permission (more than 1 permission)
1 (Very Low)	Sensitive data permission

Tabla 4: Risk Value and Permissions Combination for Android Apps

En el almacenamiento externo, nos interesa guardar el resultado por cada análisis de aplicación en el tiempo.

Un API Web para comunicaciones con los servicios externos y el almacenamiento externo.

También una capa de presentación donde poder visibilizar las aplicaciones instaladas, los permisos por aplicación, la evolución del riesgo de la aplicación y de los permisos otorgados

3.3.2. Android Triggers

Los Android triggers corresponden a una funcionalidad en android, para el que se ejecuta una acción dado un evento, aquí debemos capturar cuando:

- Cuando se ejecute por primera vez el sistema auditor.
- Una aplicación es actualizada.
- Una aplicación es instalada.
- Una vez al día a todas las aplicaciones.

En todos los casos anteriores de guardará un registro si los permisos presentan un cambio.

3.3.3. Servicios externos

1. Políticas de privacidad: Este servicio consta del uso de un sistema de NLP²⁰ que usando bert²⁰ evalúa el contenido de las políticas de privacidad, generalmente publicadas también en la tienda de aplicaciones a través de un enlace externo.

²⁰<https://blog.research.google/2018/11/open-sourcing-bert-state-of-art-pre.html>

2. Descripciones en tienda de aplicaciones: Este servicio consta del uso de un sistema de NLP que usando bert, evalúa el contenido de las descripciones de la tienda de aplicaciones haciendo también una extracción de esta.

Finalmente en el siguiente capítulo se especificarán los alcances de la etapa de validación de solución, donde se acotará el proyecto acorde a poder validar el núcleo de la aplicación.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Para validar la solución propuesta se explicará la metodología propuesta para luego aplicarla en base al modelo cascada, validando finalmente la implementación del software propuesto.

4.1. Metodología propuesta

La metodología propuesta se basa en aplicar el modelo cascada que dice “Este toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y, luego, los representa como fases separadas del proceso, tal como especificación de requerimientos, diseño de software, implementación, pruebas, etcetera.” [Sommerville, 2011]. Entonces, en el resto del capítulo se harán los siguientes pasos:

- Definición de requerimientos.
- Diseño del sistema y del software.
- Implementación y prueba de unidad.
- Integración y prueba del sistema.
- Operación y mantenimiento.

Se espera en el último paso al menos dejar recomendaciones para su uso en un ambiente productivo.

4.2. Definición de requerimientos

4.2.1. Requerimientos funcionales

- Listar las aplicaciones instaladas.
- Listar los permisos de cada aplicación.
- Listar los permisos solicitados por la aplicación.
- Listar los permisos aceptados por el usuario.
- Calcular un nivel de riesgo según análisis de permisos propuesto.

- Registrar cuando se haya hecho un análisis.
- Actualizar análisis de la aplicación cuando se haya actualizado.
- Actualizar análisis de la aplicación una vez al día.
- Conectarse a sistema ML resumen política de privacidad.
- Conectarse a sistema ML consistencia descripción tienda de aplicaciones con permisos.

4.2.2. Requerimientos no funcionales

- Seguridad: El sistema debe hacer conexiones seguras a sus fuentes de datos.
- Confidencialidad: Un externo no debería poder identificar que dispositivo tienen las aplicaciones y permisos registrados.
- Rendimiento: La aplicación debe ser eficiente y no ralentizar el dispositivo del usuario.

4.2.3. Especificación de casos de usos

Para complementar los requerimientos funcionales y no funcionales, se especificaran dos tipos de casos de uso, primero aquel propuesto en el documento, y segundo, la integración de fuentes externas.

El primer diagrama 8 muestra como el usuario android usaría el sistema auditor de privacidad para ver las aplicaciones que tiene instaladas, junto con sus permisos otorgados. Guardará un historial de los permisos otorgados, registrado por cada análisis de las aplicaciones. Entregará una nota del 1 al 5 para el análisis de riesgo, junto con los permisos que son de información sensible. También, se hará un enlace a lo que es el cambiar los permisos en el sistema android.

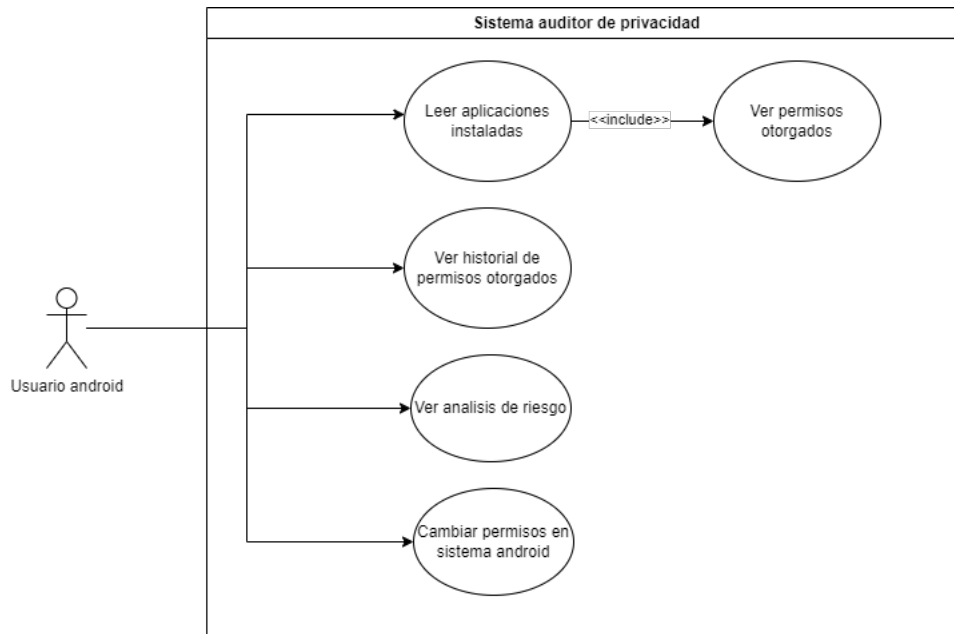


Figura 8: Diagrama Casos de usos con análisis de riesgo.

El segundo diagrama 9 se enfoca en el análisis de descripción a través de machine learning con una técnica de procesamiento de lenguaje natural llamado BERT. En este el usuario mediante un enlace puede ir a la tienda de aplicaciones(si la aplicación es publicada) y revisar su descripción. También, puede revisar que grupos de permisos predominan en la descripción, extraídos con BERT y explicados al usuario.

Los grupos son los siguientes:

Grupos de permisos
STORAGE
CONTACTS
LOCATION
CAMERA
MICROPHONE
SMS
CALL_LOG
PHONE
CALENDAR
SETTINGS
TASKS

Tabla 5: Listado grupo de permisos

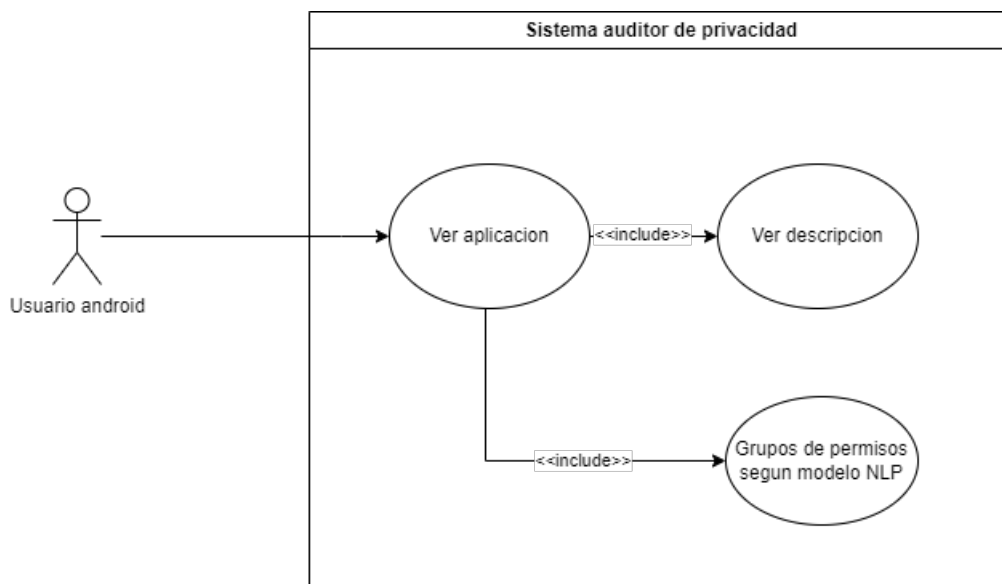


Figura 9: Diagrama Casos de usos con análisis descripción.

El tercer diagrama 10 se enfoca en el resumen de la política de privacidad a través de machine learning con una técnica de procesamiento de lenguaje natural llamado BERT. En este el usuario mediante un enlace puede ir a la web de la aplicación con la política de privacidad (si es publicada debería tenerlo) y revisar su política de privacidad. También, puede ver un resumen hecho con bert, enfocado en 11 categorías de información sobre privacidad.

Las categorías de información de política de privacidad son las siguientes:

Categorías
Introducción a la política
Recolección y uso por el sistema
Cookies y tecnologías similares
Retención de datos
Compartir y recolectar con terceros
Seguridad de los datos
Transferencia internacional de datos
Derechos del usuario y control
Audiencias específicas
Cambios en la política
Información de contacto de la política

Tabla 6: Categorías resumen política privacidad

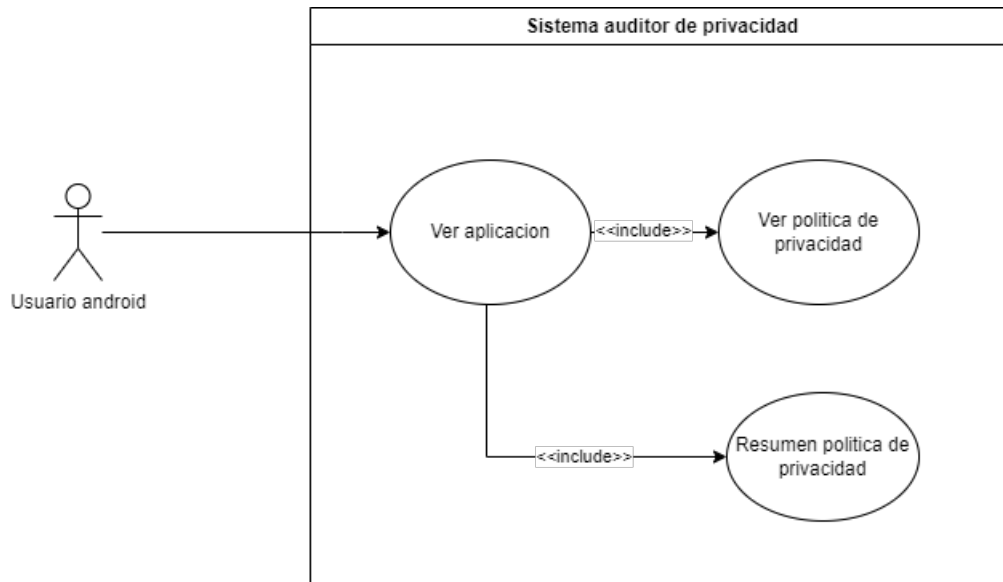


Figura 10: Diagrama Casos de usos con resumen política de privacidad.

4.3. Diseño del sistema y del software.

Para el diseño del sistema y del software se usaran las siguientes herramientas:

- Diagrama de contexto
- Modelo de dominio
- Arquitectura de alto nivel
- Diagrama de secuencia
- Interfaz usuaria de vistas claves

4.3.1. Diagrama de contexto

El siguiente diagrama de contexto 11 considera al usuario como una parte que interactúa directamente con nuestro sistema propuesto. Mientras que el sistema, trabaja con el sistema “Fuente política de privacidad” para obtener el resumen y el enlace a la política de privacidad original. Con la “Fuente descripciones” obtiene que grupos de permisos predominan en la descripción de la aplicación, también, el enlace a la tienda de aplicaciones. Finalmente con “Fuente información permisos” se obtienen los nombre de los permisos, junto con su descripción, esto para ser mostrado en las vistas.

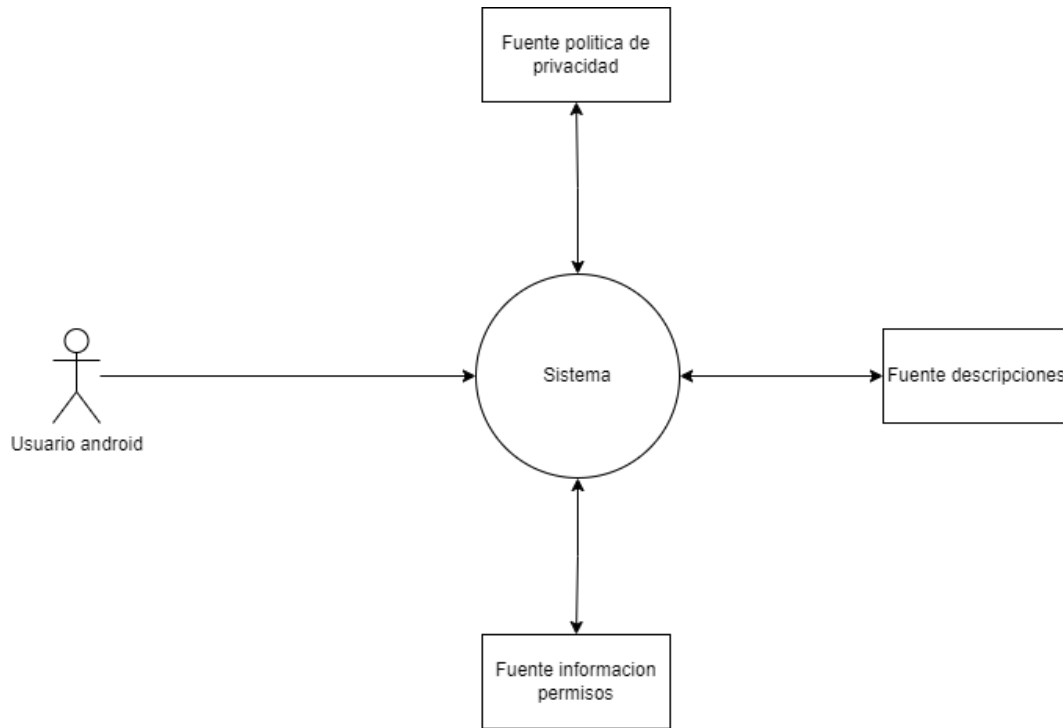


Figura 11: Diagrama de contexto
Fuente: Elaboración propia.

4.3.2. Modelo de dominio

El siguiente modelo de dominio se hace con una finalidad de entender como se comportan las entidades a implementar, con la finalidad de lograr un mejor entendimiento al escribir las historias de usuario y hacer las interfaces usuarias.

Por la parte izquierda se puede identificar una parte orientada a lo que es el usuario android que mediante nuestro sistema analiza los paquetes que instalan aplicaciones, las mismas que el usuario instaló en su dispositivo. Si la aplicación es una publicada, entonces, detrás hay una empresa que tiene una política de privacidad. También, si es publicada en la tienda de aplicaciones tiene una descripción, de la cual se derivan grupos de permisos peligrosos.

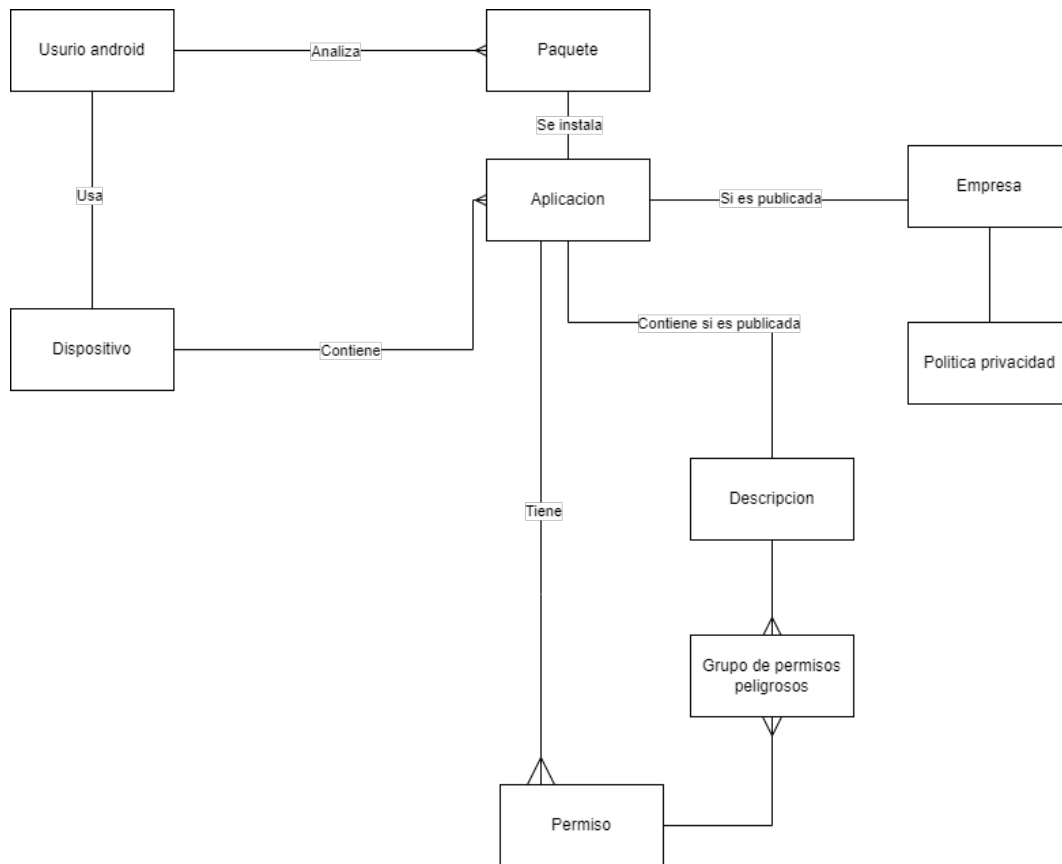


Figura 12: Modelo de dominio
Fuente: Elaboración propia.

4.3.3. Arquitectura de alto nivel

Para esta propuesta de arquitectura es considerada la solución a un alto nivel, esto quiere decir que considera las interacciones con las APIs a integrar, junto con sus fuentes de datos, abstrayéndonos de como las APIs consiguen la información transmitida.

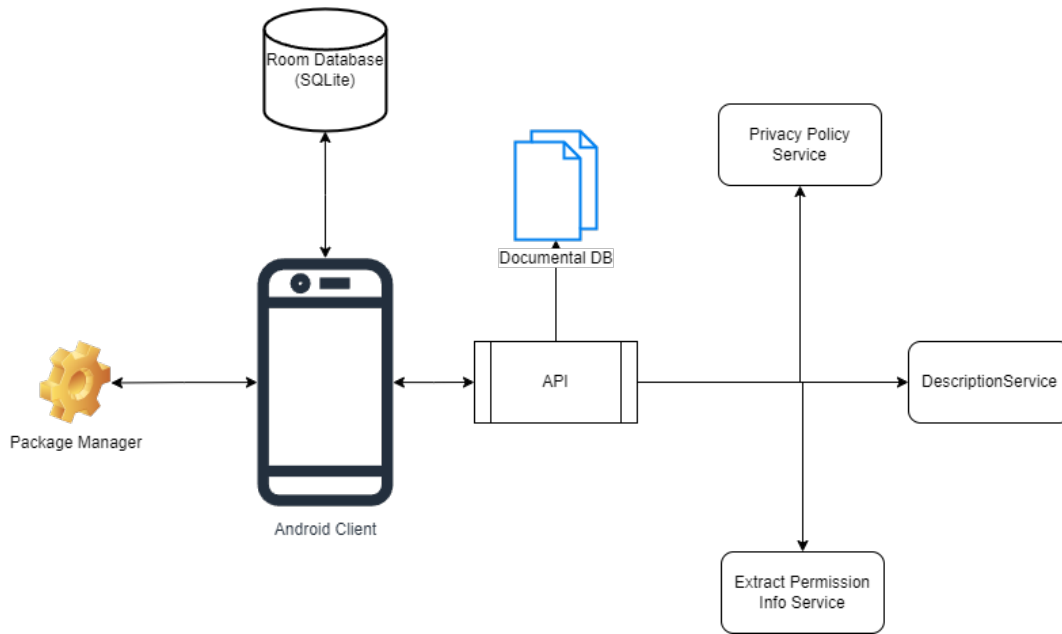


Figura 13: Arquitectura alto nivel
Fuente: Elaboración propia.

Donde:

- **Android Client:** Corresponde a una aplicación - cliente que será desarrollada con kotlin(Android Nativo).
- **Room Database:** Corresponde a una base de datos ligera que sirve para cachear los datos tanto del análisis interno al dispositivo, como para datos extraídos de las fuentes de datos externas.
- **Package Manager:** Corresponde a una clase de Android, introducida en la API Level 1, que sirve para entregar información de las aplicaciones y paquetes instalados.
- **API:** Se usará para ser un intermediario entre el Android client y los servicios correspondientes a las fuentes de datos.
- **Privacy Policy Service:** Corresponde al servicio que para una aplicación seleccionada, el resumen de la política de privacidad.
- **Description Service:** Corresponde al servicio que para una aplicación seleccionada, este entrega los grupos de permisos que debería tener según su descripción.
- **Extract Permission Info Service:** Se ocupa para obtener información actualizada de la documentación de Android sobre los permisos y sus descripciones.

- Document DB: Es una base de datos a ser usada para guardar en el servidor datos como: Resumen política privacidad, grupos por descripción aplicación, descripciones permisos Android.

4.3.4. Historias de usuario

Tabla 7: Historias de usuario
Fuente: Elaboración Propia.

Identificador	Título	Descripción
HU-01	Listar aplicaciones instaladas	Como usuario quiero que ver un listado de aplicaciones instaladas para saber cuales reconoce el sistema.
HU-02	Abrir detalle aplicación	Como usuario quiero abrir detalle de aplicación para saber sus permisos instalados y acceder a otras características.
HU-03	Abrir editar permisos	Como usuario quiero acceder a aplicación que edita permisos por aplicación para aumentar la seguridad en mi dispositivo.
HU-04	Descargar informe aplicación	Como usuario quiero descargar informe aplicación para extraer y compartir el informe de seguridad.
HU-05	Resumen política de privacidad	Como usuario quiero ver el resumen de la política de privacidad para ahorrar tiempo al informarme sobre que hacen con mis datos.
HU-06	Ver política de privacidad	Como usuario quiero acceder a la política de privacidad de la aplicación para saber cual es su fuente original.
HU-07	Grupos de permisos según descripción	Como usuario quiero ver los grupos de permisos según descripción para saber que grupos de permisos predominan en la descripción de la aplicación.
HU-08	Ver descripción	Como usuario quiero ver la descripción de la aplicación para saber cual es su fuente original.

Tabla 8: Historias de usuario

Fuente: Elaboración Propia.

HU-09	Ver análisis de riesgo	Como usuario quiero ver el análisis de riesgo para saber en que información sensible estoy entregando a las aplicaciones.
HU-10	Historial de cambios en permisos	Como usuario quiero por cada aplicación ver el historial de cambios en permisos, para saber en caso de auditoría.
HU-11	Nuevo análisis	Como usuario quiero hacer un nuevo análisis, para actualizar el listado de aplicaciones instaladas.
HU-12	Nuevo análisis por aplicación	Como usuario quiero hacer un nuevo análisis a la aplicación, para actualizar los datos de la aplicación seleccionada.

4.3.5. Diagrama de despliegue

En el diagrama de despliegue se explica como interactúan de forma separada los distintos componentes del sistema.

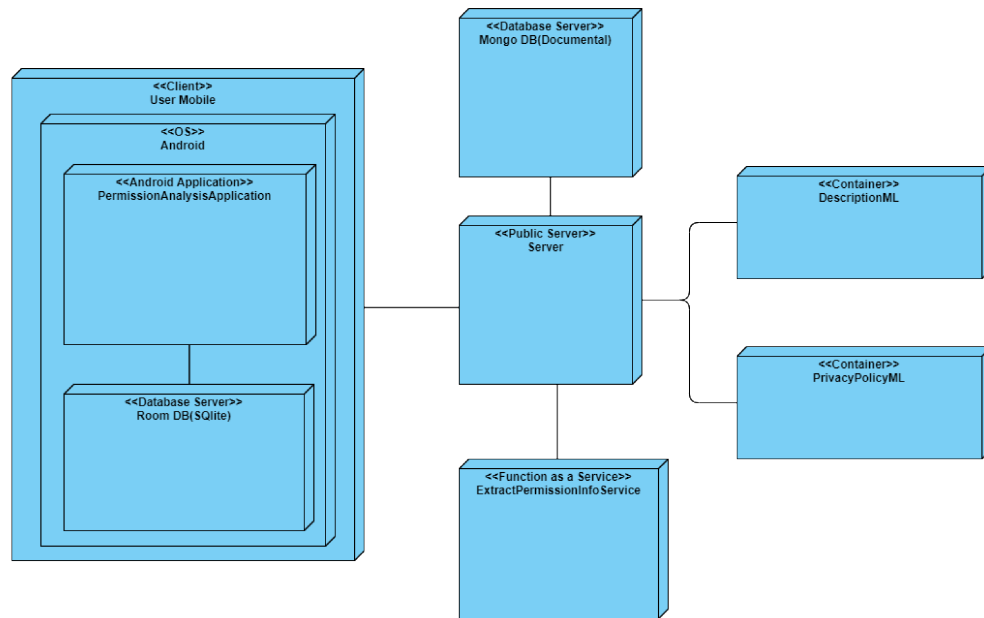


Figura 14: Diagrama de despliegue
Fuente: Elaboración propia.

Al igual que en la arquitectura de alto nivel, se repiten algunos componentes, salvo, que aquí se explica la base tecnológica para el despliegue.

- Client: Dispositivo móvil Marca Xiaomi Redmi Note 9, de 128 GB de almacenamiento y 6 GB de ram
- Sistema operativo: Es un Android 11 RP1A.200720.011
- Android Application: Corresponde al apk desarrollado en este modulo.
- Database Server(Interno): Base de datos interna con la tecnología Room DB, por lo tanto, se despliega junto a la instalación de la aplicación.
- Database Server(Externo): Base de datos de tipo MongoDB, debe ser configurada fuera de la aplicación interna, pero, en una red hacia el servidor Publico.
- Public Server: Servidor hecho con golang, permite comunicarse con los servicios externos de ML, y un servicio de extracción de datos, debe ser de acceso publico.

- Function as a service: Desplegado en plataforma de Nube AWS, que permite servicios sin servidor.
- Container: corresponden a los servicios de ML, no abordados en esta memoria, se sugiere ser desplegado cada uno en un contenedor Docker.

4.3.6. Diagramas de paquete

Considerando que desarrollaremos solo 3 artefactos, que serian:

- Aplicación Cliente Android
- Aplicación de Servidor
- ExtractPermissionInfoService como function as a service

De la Aplicación Cliente Android requiere una estructura de paquete, aquí explicaremos solo su funcionalidad por paquete.

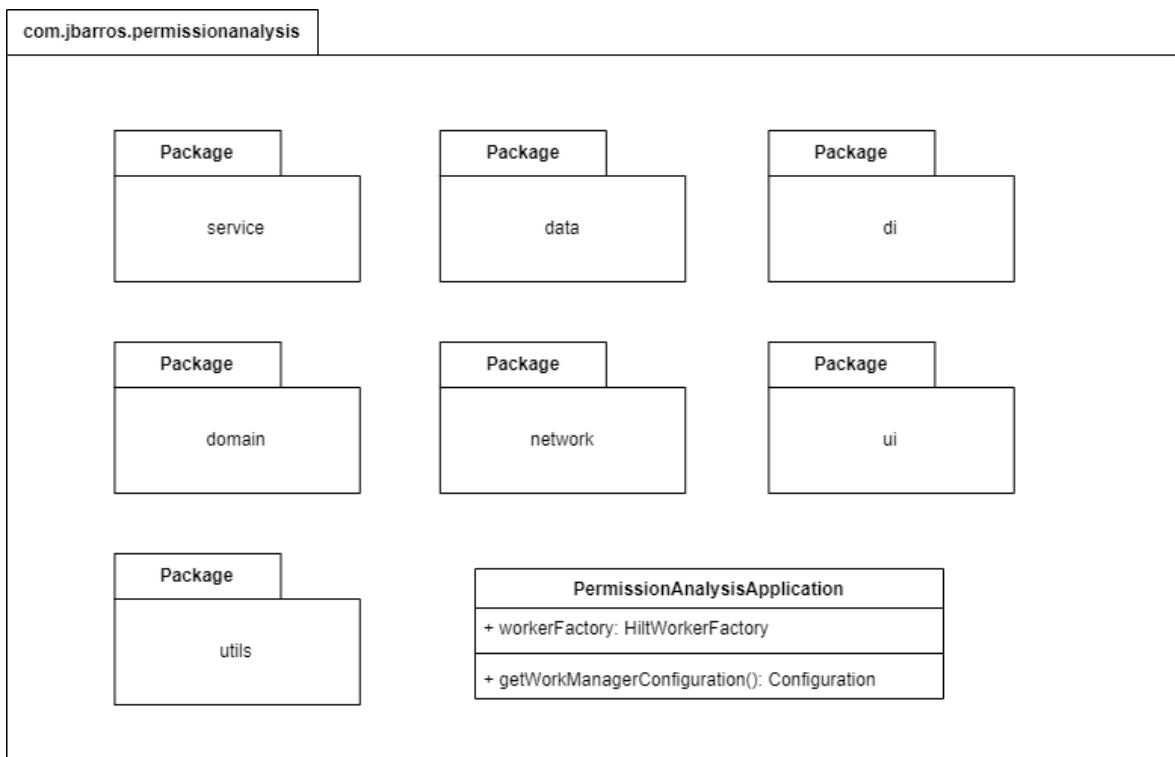


Figura 15: Diagrama de paquete: Aplicación
Fuente: Elaboración propia.

De aquí los paquetes son:

- Service: Capa usada para configurar servicios de tipo JOB para configurar ejecuciones diarias.
- Data: Aquí se escriben los modelos de datos usados en la BD room, también, sus accesos.
- DI: Capa para configurar la inyección de dependencias, en ella se hacen instancias singleton de los servicios requieren una única instancia.
- Domain: En esta capa se escriben los casos de uso, también los modelos de casos de uso.
- Network: En este lugar se hacen las conexiones hacia fuera de la aplicación mediante HTTP
- UI: Corresponde donde se dan inicio a las actividades de android
- Utils: Corresponde al lugar donde existen herramientas que apoyan a los otros paquetes.
- PermissionAnalysisApplication: Representa el punto de entrada de la aplicación

También la Aplicación de Servidor requiere una estructura de paquete, aquí explicaremos solo su funcionalidad por paquete.

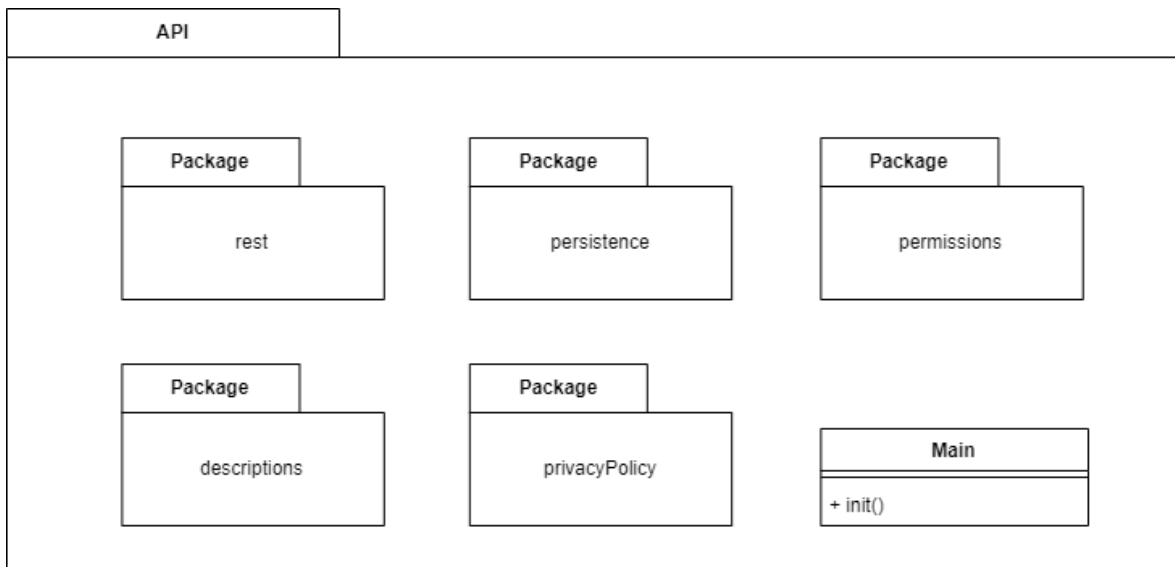


Figura 16: Diagrama de paquete: API web
Fuente: Elaboración propia.

Donde:

- Rest: Capa dirigida por librería GinGonic usada para disponibilizar endpoints con conexiones HTTP.
- Persistence: Capa usada para hacer conexiones con una base de datos, con la finalidad de hacer respaldos.
- Permissions: Capa usada para trabajar el contenido de los permisos, también, trabajar las conexiones con el servicio function as a service de ExtractPermissionInfoService.
- Descriptions: Capa usada para trabajar el contenido de las descripciones, también, trabajar las conexiones con el servicio externo de Descriptions ML.
- PrivacyPolicy: Capa usada para trabajar el contenido de política de privacidad, también, trabajar las conexiones con el servicio externo de PrivacyPolicy ML
- Main: Punto de entrada del API web.

4.3.7. Base de datos

El diseño de la base de datos, está centrado en los datos que se pueden obtener mediante la API PackageManager de Android, lo que resulta en lo siguiente:

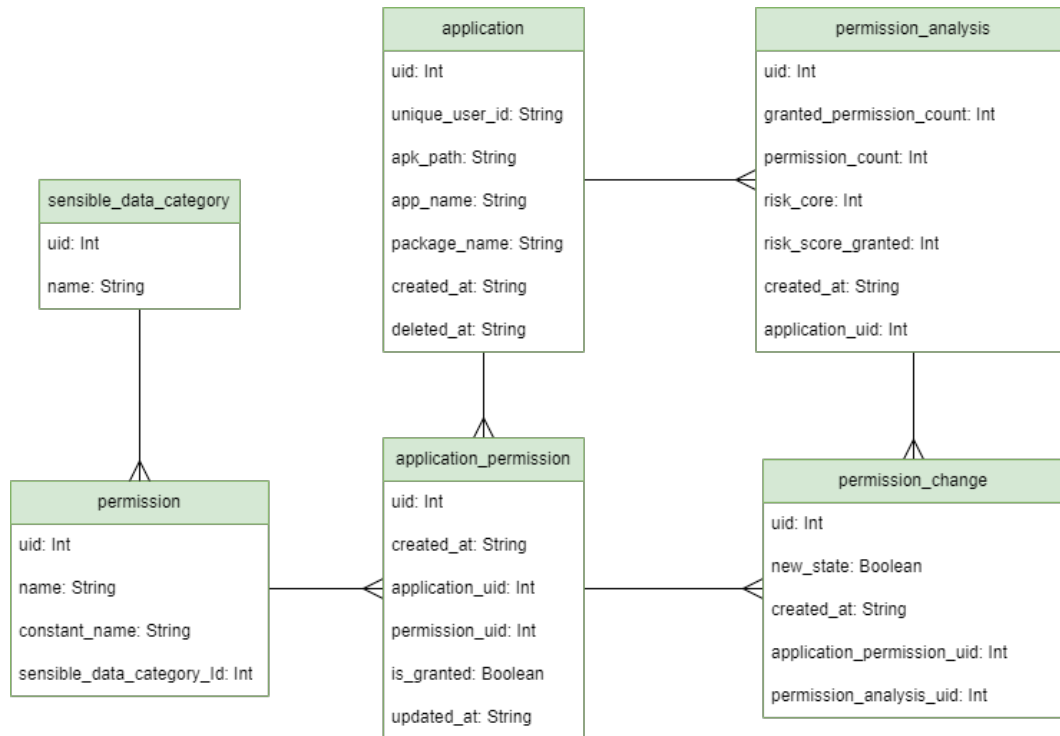


Figura 17: Diseño de base de datos
Fuente: Elaboración propia.

Donde:

- **application:** Esta tabla guarda datos de la aplicación, se incluye el dato de la ubicación, abriendo posibilidades para un análisis estático.
- **application_permission:** El valor que tiene esta tabla es registrar si el permiso es aceptado o no.
- **permission:** Guarda las definiciones de los permisos.
- **sensitive_data_category:** Si la categoría corresponde a dato de tipo sensible, esto es parte de el análisis de riesgo.
- **permission_analysis:** Corresponde al análisis de riesgo hecho, en él, se consideran las variaciones de permisos aceptados y solicitados por la aplicación, también el registro de el índice de riesgo que varía de 1 a 5.
- **permission_change:** Registra los cambios en los permisos por cada aplicación, guarda en que versión del análisis de permisos fue, y su nuevo estado.

4.4. Implementación y prueba de unidad.

Para iniciar la implementación, se comenzará con el desarrollo de la obtención de todos los nombres de los permisos, esto se hará con un scrapper hecho con una función como servicio en aws lambda, luego, se hará la API que consume la función, y serviría como intermediario con los otros servicios externos, por ultimo, se construirá la aplicación mobile.

4.4.1. ExtractPermissionInfoService

Este servicio mediante el lenguaje de programación python se construye un servicio de extracción de datos a la documentación de android.²¹ Usando la biblioteca request²² y el método GET se guarda el contenido de la pagina, esta es extraída con la biblioteca²³, que luego es tratada con JSON para ser enviado como respuesta.

4.4.2. API

La siguiente API presenta configuraciones para un proyecto completo, este quiere decir para una integración con los servicios de ML, por ello, tiene una conexión a BD, aun así, este se dedica solo a la conexión con el servicio de extracción de datos, y también a la generación de datos dummies.

²¹<https://developer.android.com/reference/android/Manifest.permission>

²²<https://pypi.org/project/requests/>

²³<https://pypi.org/project/beautifulsoup4/>

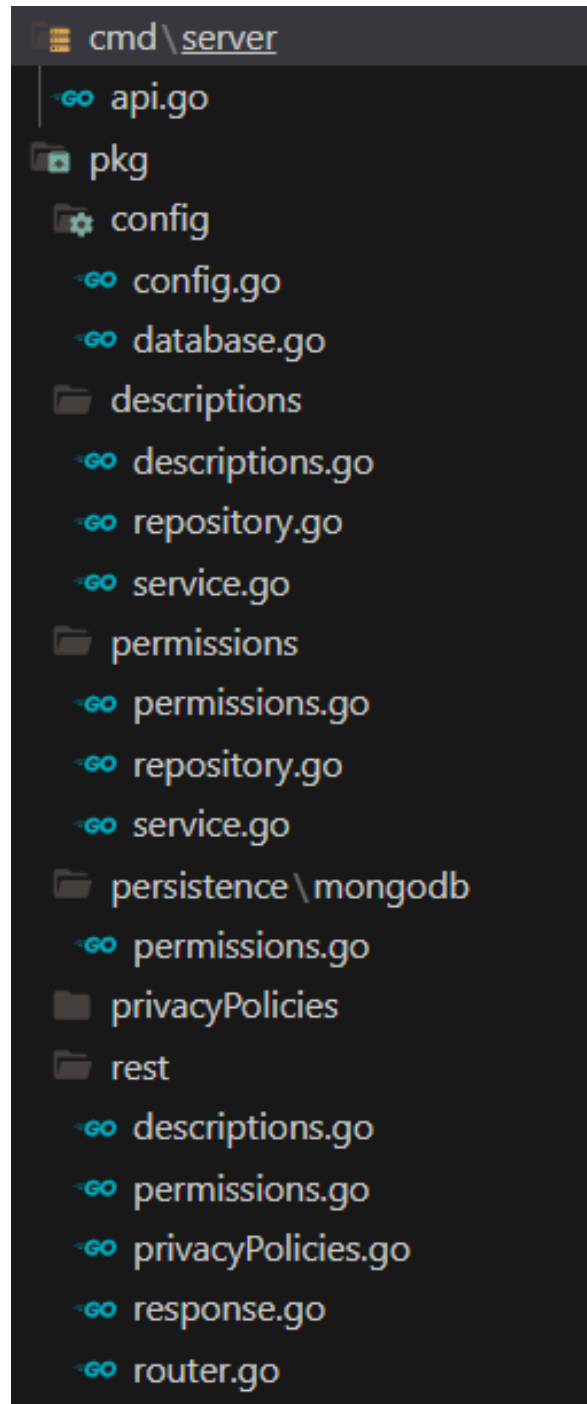


Figura 18: Estructura de API
Fuente: Elaboración propia.

Cuando la API se ejecuta, el paquete GIN GONIC²⁴ de GO entrega el listado de endpoints

²⁴<https://gin-gonic.com/>

disponibles:

```
- using env:  export GIN_MODE=release
- using code: gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /ping                --> permission-analysis/pkg/rest.Ping (4 handlers)
[GIN-debug] GET    /api/permissions     --> permission-analysis/pkg/rest.PermissionsHandler.Permi
rs)
[GIN-debug] GET    /api/descriptions/:id --> permission-analysis/pkg/rest.DescriptionsHandler.GetD
lers)
[GIN-debug] GET    /api/privacyPolicies/:id --> permission-analysis/pkg/rest.PrivacyPoliciesHandler.G
s)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8081
```

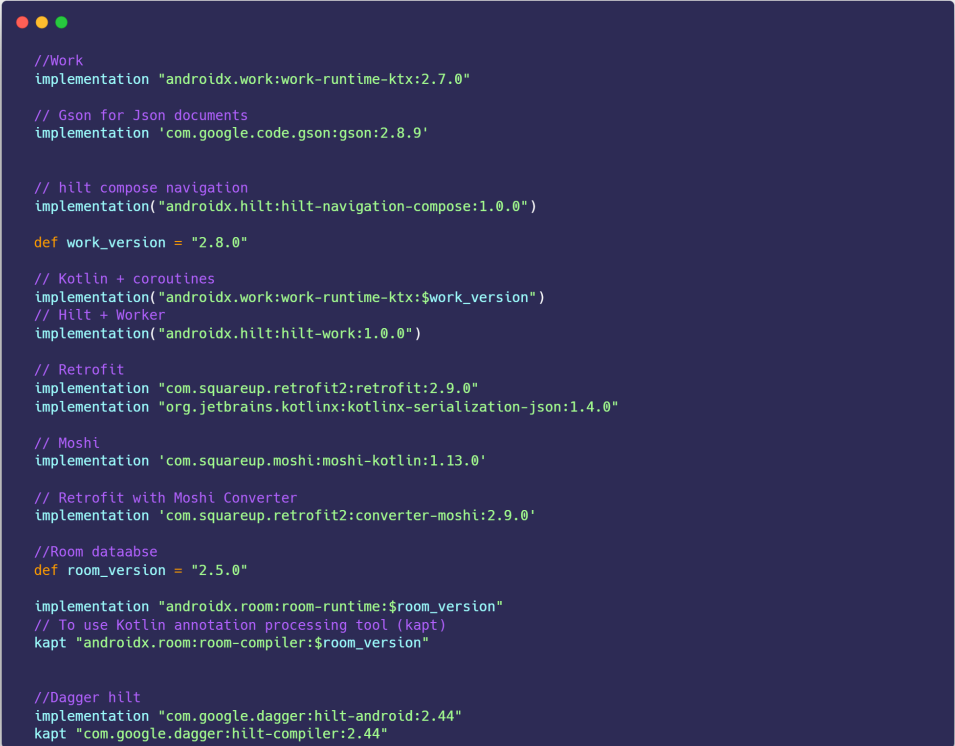
Figura 19: Ejecución API en GIN GONIC

Fuente: Elaboración propia.

- **GET /ping** Es usado a nivel de configuración con el motivo
- **GET /api/permissions** Usado para obtener el listado de permisos
- **GET /api/descriptions/:id** Usado para obtener un documento con el resultado del análisis a la descripción de la tienda de aplicaciones, dado que no existe aun un endpoint para el servicio de ML, los resultados se entregan de forma aleatoria.
- **GET /api/privacyPolicies/:id** Usado para obtener un documento con el resumen resultado de un análisis a la política de privacidad de la aplicación, también dado que no existe aun un endpoint para el servicio de ML es que como resultado el resumen es un lorent ipsum.

4.4.3. Configuración de aplicación Android.

Para iniciar su configuración es que en el archivo **build.gradle** se añadirán las siguientes librerías y sus versiones.

A screenshot of a code editor with a dark blue background and light-colored text. The code is a list of Android dependencies for a Kotlin project. It includes comments in Spanish and English, and version numbers for various libraries like Work, Gson, Hilt, Retrofit, Moshi, Room, and Dagger. The code is as follows:

```
//Work
implementation "androidx.work:work-runtime-ktx:2.7.0"

// Gson for Json documents
implementation 'com.google.code.gson:gson:2.8.9'

// hilt compose navigation
implementation("androidx.hilt:hilt-navigation-compose:1.0.0")

def work_version = "2.8.0"

// Kotlin + coroutines
implementation("androidx.work:work-runtime-ktx:$work_version")
// Hilt + Worker
implementation("androidx.hilt:hilt-work:1.0.0")

// Retrofit
implementation "com.squareup.retrofit2:retrofit:2.9.0"
implementation "org.jetbrains.kotlinx:kotlinx-serialization-json:1.4.0"

// Moshi
implementation 'com.squareup.moshi:moshi-kotlin:1.13.0'

// Retrofit with Moshi Converter
implementation 'com.squareup.retrofit2:converter-moshi:2.9.0'

//Room dataabase
def room_version = "2.5.0"

implementation "androidx.room:room-runtime:$room_version"
// To use Kotlin annotation processing tool (kapt)
kapt "androidx.room:room-compiler:$room_version"

//Dagger hilt
implementation "com.google.dagger:hilt-android:2.44"
kapt "com.google.dagger:hilt-compiler:2.44"
```

Figura 20: Librerías usadas en android kotlin
Fuente: Elaboración propia.

4.4.4. Configuración de base de datos

Para configurar la Base de datos, fue necesario ocupar el paquete de room database, se llevó el modelo de datos a través de la anotación **@Entity**, todo con la sintaxis de SQLite, cuando se instala la aplicación, se ve de la siguiente manera desde el apartado de app inspection en android studio.

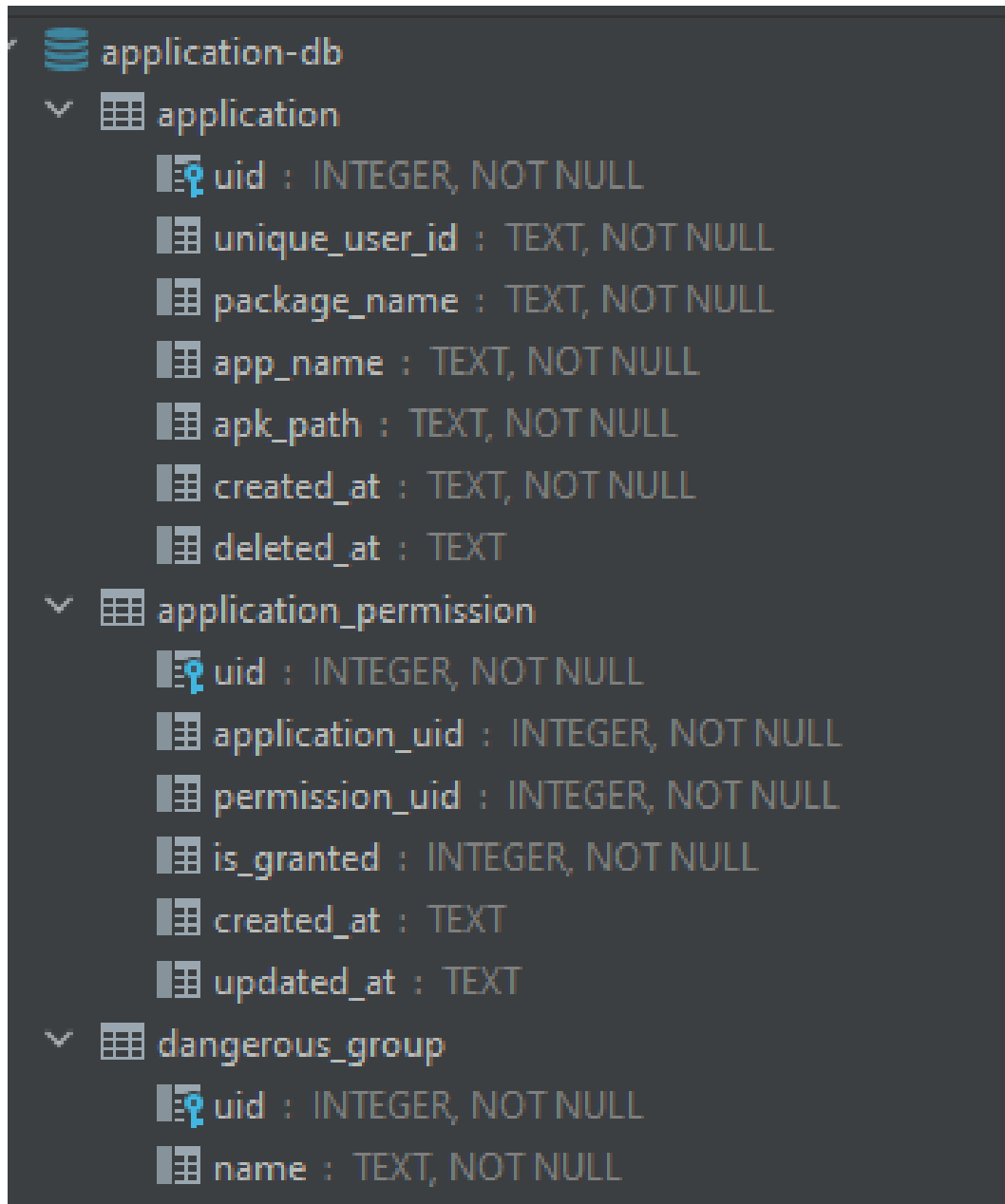


Figura 21: Base de datos configurada
Fuente: Android Studio.

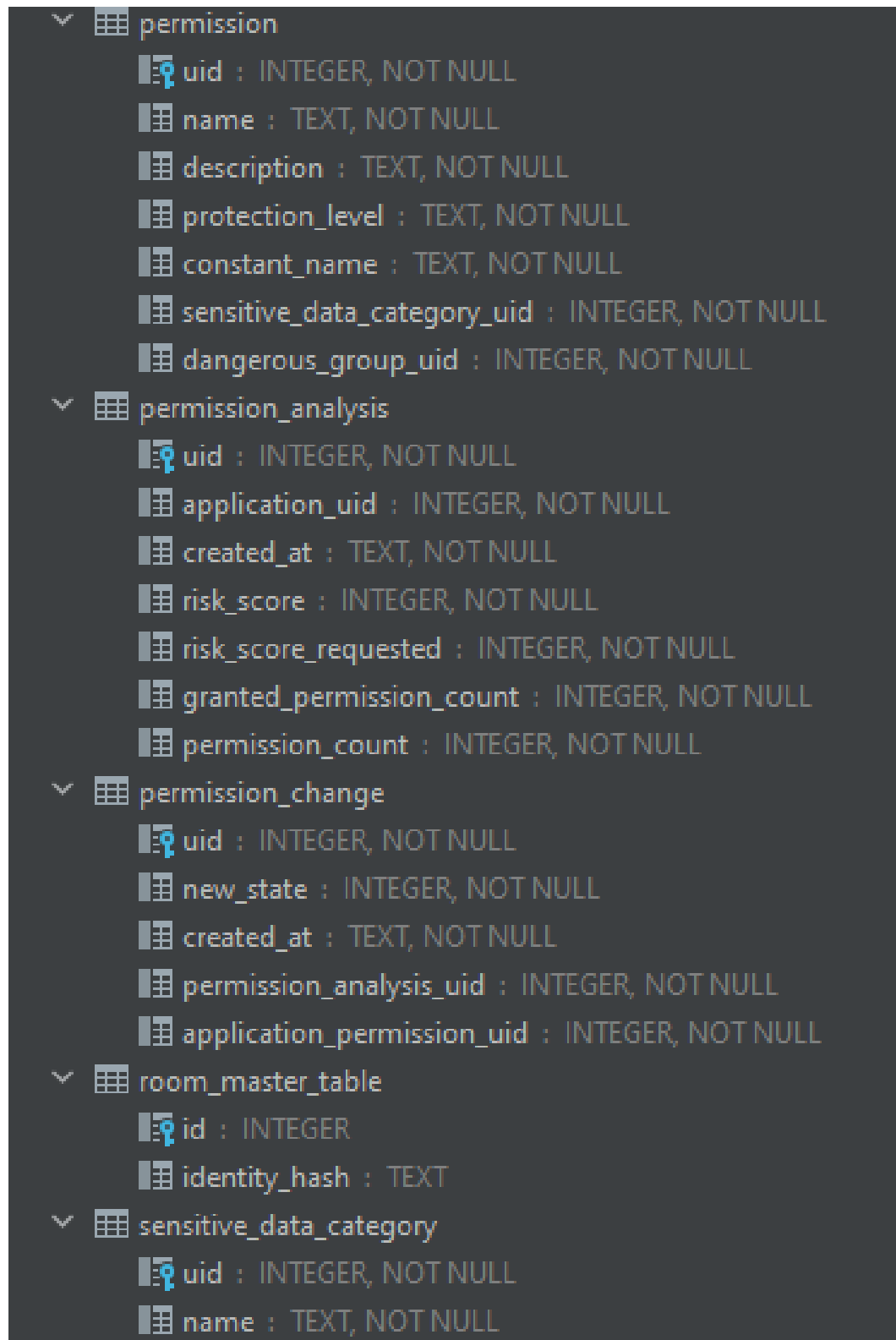


Figura 22: Base de datos configurada

Fuente: Android Studio

4.4.5. Flujo de datos entre paquetes

Para lograr las comunicaciones entre los paquetes o componentes de la aplicación, se propone la siguiente estructura.

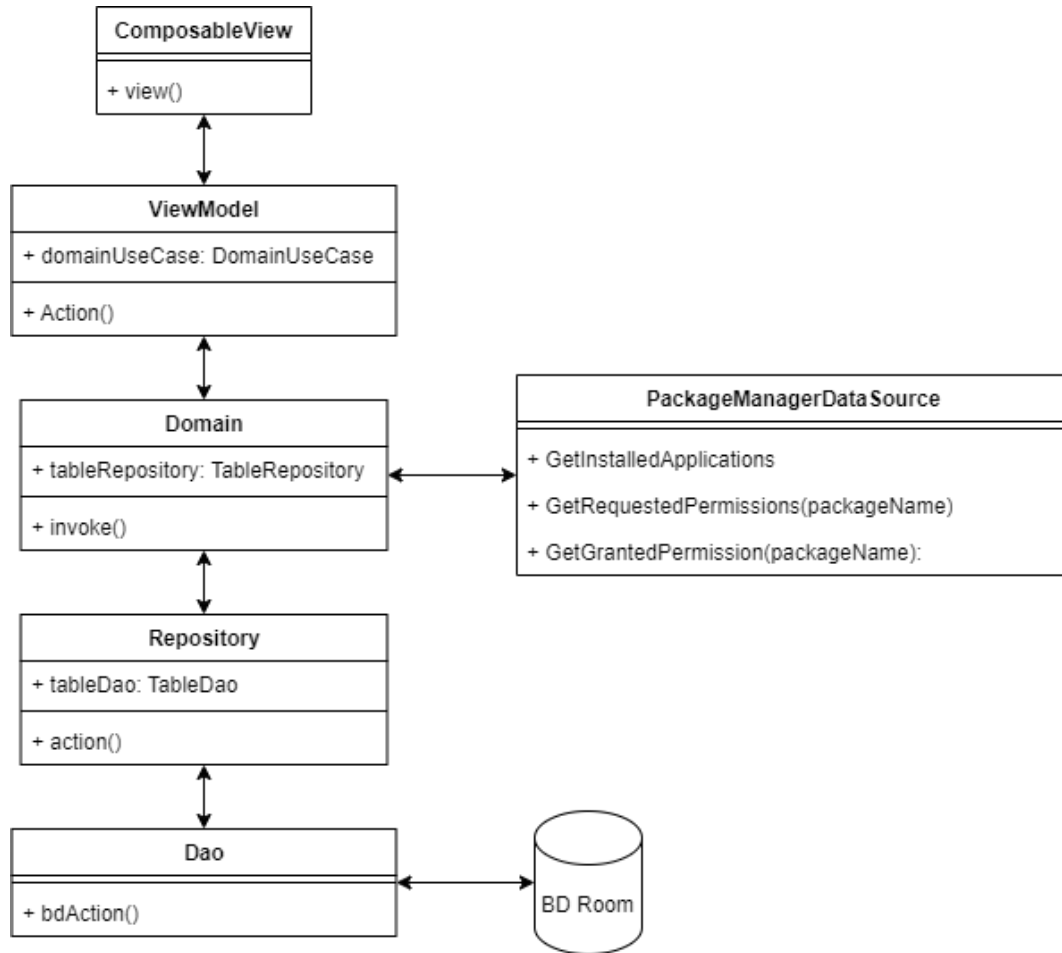


Figura 23: Flujo de datos entre paquetes
Fuente: Android Studio

Donde:

- ComposableView: Corresponde a una vista de jetpack compose con la anotación @Composable
- ViewModel: Corresponde al lugar donde se manejan las variables y eventos entre vistas, también donde se pueden usar las corutinas para ejecuciones en segundo plano, tiene la anotación @HiltViewModel.

- Domain: Capa donde están los casos de uso, se comunica con la capa repository y otras fuentes de datos.
- Repository: Capa de comunicación intermedia entre el dominio y las consultas a la base de datos.
- Dao: Capa usada para hacer las consultas a la BD y mantener el modelo de la BD.
- BD Room: Se refiere a la Base de datos room interna del dispositivo android.
- PackageManagerSource: Corresponde a un intermediario entre la capa de dominio y el paquete PackageManager de Android.

4.4.6. Comunicación a API

Para hacer la comunicación a la API, se usa la librería Retrofit²⁵ la que mediante la inyección de dependencia es instanciado solo 1 vez. La siguiente imagen corresponde a la interfaz de retrofit que permite configurar la conexión a la BD junto a sus receptores de datos.

```
interface
PermissionsAnalysisAPIService {

    @GET("permissions/")
    suspend fun getListPermission(): PermissionExtractInfoResponse

    @GET("descriptions/{packageName}")
    suspend fun getDescriptions(@Path("packageName") packageName: String): DescriptionsResponse

    @GET("privacyPolicies/{packageName}")
    suspend fun getPrivacyPolicies(@Path("packageName") packageName: String): PrivacyPoliciesResponse
}
```

Figura 24: Interfaz de API
Fuente: Elaboración propia

4.4.7. Calculadora de riesgo

La calculadora de riesgo es una parte fundamental dentro de la implementación, en 25 en ella recibe como parámetro un listado de permisos para ser consultados y entregar un numero de riesgo del 0 al 5, en ella, verifica que dentro del listado existan permisos para:

- Obtener información sensible.

²⁵<https://square.github.io/retrofit/>

- Compartir información sensible.

```
fun calculateRisk(permissionList: List<String>): Int {
    val hasInternet = permissionList.contains(android.Manifest.permission.INTERNET)

    val hasSendSms = permissionList.contains(android.Manifest.permission.SEND_SMS)
    val hasBluetooth = permissionList.contains(android.Manifest.permission.BLUETOOTH)
    val hasNfc = permissionList.contains(android.Manifest.permission.NFC)

    var hasDangerousPermission = false
    var dangerousPermissionCount = 0

    for (permissionElement in permissionList) {
        val permissionInfo = getPermissionInfo(permissionElement)
        if (permissionInfo != null) {
            hasDangerousPermission = true
            dangerousPermissionCount += 1
        }
    }

    var riskValue = 0

    when {
        hasDangerousPermission && ((hasInternet || hasSendSms) && (hasBluetooth || hasNfc)) -> riskValue = 5
        hasDangerousPermission && (hasInternet || hasSendSms) -> riskValue = 4
        hasDangerousPermission && (hasBluetooth || hasNfc) -> riskValue = 3
        hasDangerousPermission && dangerousPermissionCount > 1 -> riskValue = 2
        hasDangerousPermission -> riskValue = 1
    }

    return riskValue
}
```

Figura 25: Calculadora de riesgo
Fuente: Elaboración propia.

4.4.8. Primer Análisis

Cuando la aplicación inicia por primera vez debe hacer un primer análisis para extraer el listado de aplicaciones, sus permisos y hacer el primer calculo de riesgo, en el siguiente diagrama de flujo se esbozan las actividades implementadas.

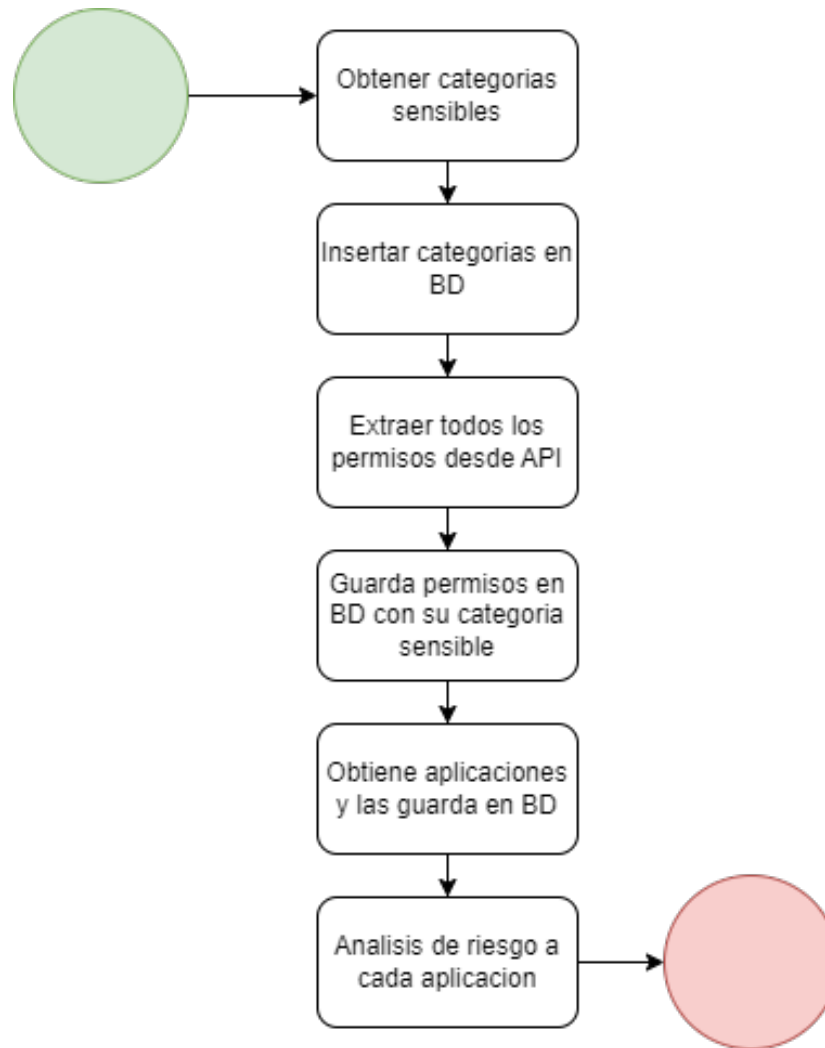


Figura 26: Primer Análisis
Fuente: Elaboración propia.

4.4.9. Análisis a una aplicación

Cuando se hace el análisis a una aplicación, se hace un extenso flujo en la búsqueda de cambios en permisos, permisos nuevos y cálculos de riesgo. Esto se ve en el siguiente diagrama de flujo.

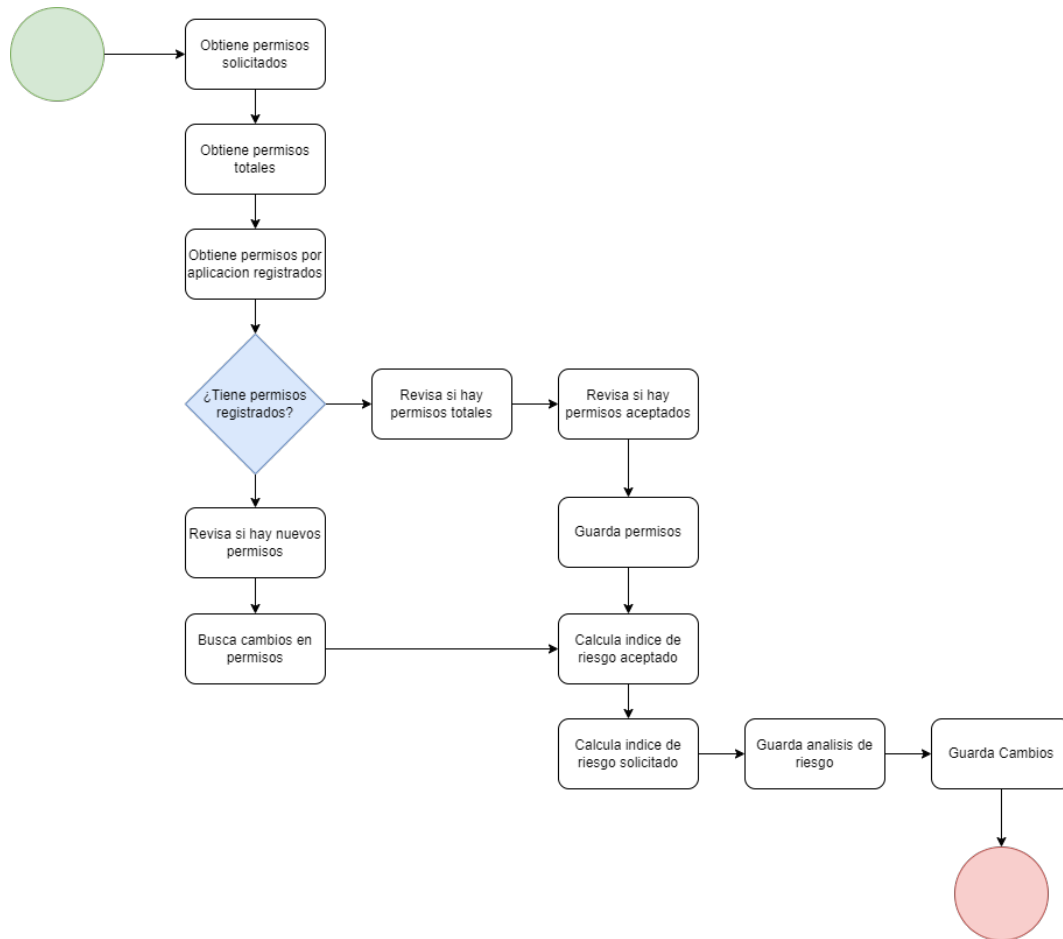


Figura 27: Analizar una aplicación
Fuente: Elaboración propia.

4.4.10. Análisis todas las aplicaciones

Cuando se hace el análisis a todas las aplicaciones, se revisan si se han añadido o eliminado aplicaciones, y se suma al flujo de análisis por aplicación, ya que es necesario para actualizar todas las aplicaciones. Se puede ver en el siguiente diagrama de flujo

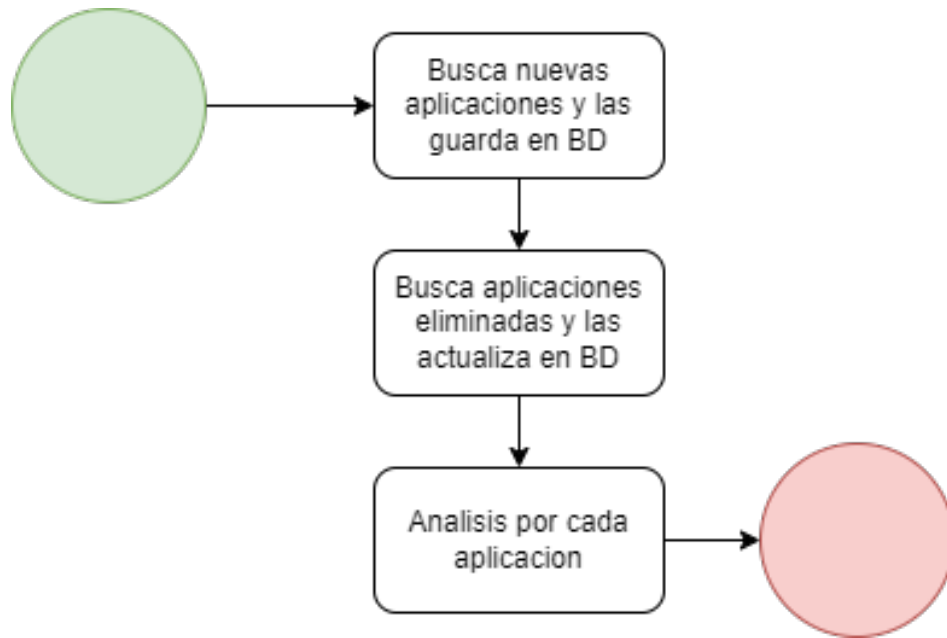


Figura 28: Análisis a todas las aplicaciones
Fuente: Elaboración propia.

4.4.11. Scheduler Diario

<https://fonts.cdnfonts.com/css/georama-2> Para poder llevar un registro diario de la variación de permisos es que se configura un servicio de kotlin para que sea ejecutado de forma diaria a las 3 am. En la siguiente figura 29 se muestran las relaciones entre el SchedulerRiskAnalysisService y RiskAnalysisService, donde el primero es usado para hacer la configuración, mientras que el segundo es para analizar todos los permisos en segundo plano.

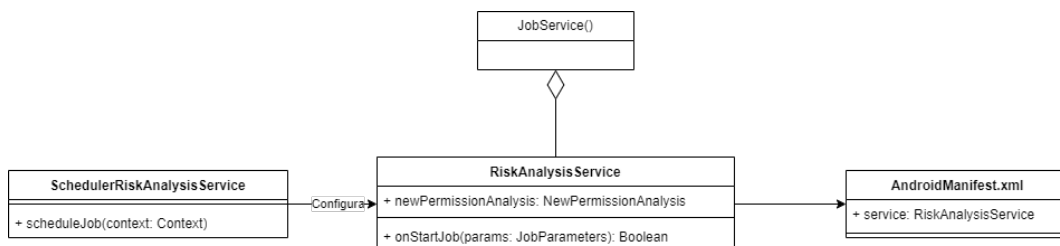


Figura 29: Diagrama de clases Scheduler Service
Fuente: Elaboración propia.

En la siguiente figura 30 se puede ver la correcta configuración a las 3 am.

```

object SchedulerRiskAnalysisService {

    private const val JOB_ID = 123

    fun scheduleJob(context: Context) {
        val serviceComponent = ComponentName(context, RiskAnalysisService::class.java)
        val builder = JobInfo.Builder(JOB_ID, serviceComponent)

        // Establece la hora para las 3 a.m.
        builder.setPersisted(true)
            .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)
            .setRequiresCharging(false)
            .setRequiresDeviceIdle(false)
            .setMinimumLatency(getTimeUntil3AM())
            .setOverrideDeadline(getTimeUntil3AM() + 1000) // Le da un margen de 1 segundo

        val jobScheduler = context.getSystemService(Context.JOB_SCHEDULER_SERVICE) as JobScheduler
        jobScheduler.schedule(builder.build())
    }

    private fun getTimeUntil3AM(): Long {
        val currentTime = System.currentTimeMillis()
        val calendar = java.util.Calendar.getInstance()
        calendar.timeInMillis = currentTime
        calendar.set(java.util.Calendar.HOUR_OF_DAY, 3)
        calendar.set(java.util.Calendar.MINUTE, 0)
        calendar.set(java.util.Calendar.SECOND, 0)

        if (calendar.timeInMillis <= currentTime) {
            calendar.add(java.util.Calendar.DAY_OF_MONTH, 1)
        }

        return calendar.timeInMillis - currentTime
    }
}
    
```

Figura 30: Configuración Scheduler
Fuente: Elaboración propia.

4.4.12. Pruebas unitarias

Se deciden hacer pruebas unitarias solo a aquello esencial dentro del proyecto que requiera ser probado, por un lado, se considera importante probar un utilitario proveedor de fechas, por otro lado, la calculadora de riesgo, y sus 6 niveles de respuesta, aquello queda reflejado en la siguiente imagen:

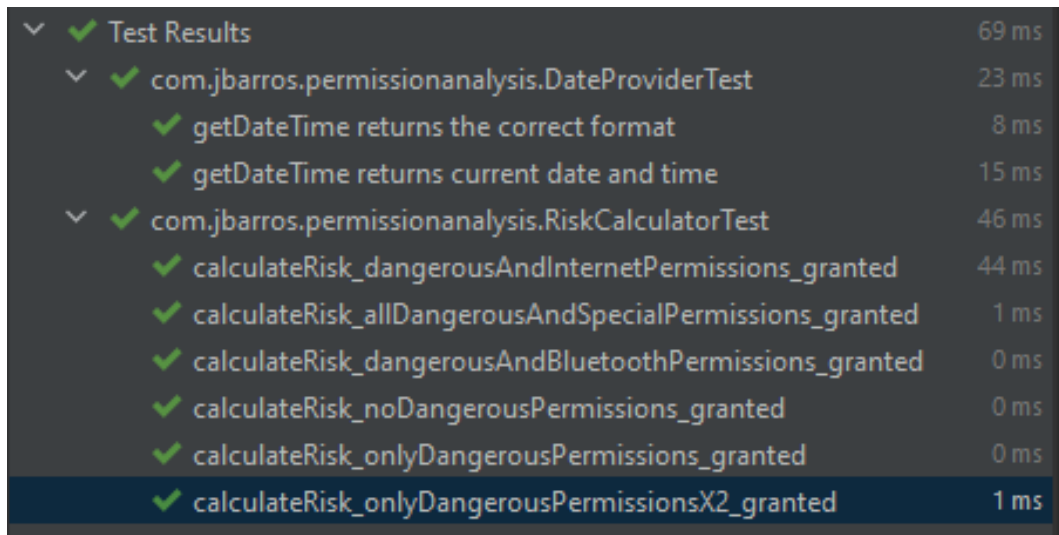


Figura 31: Pruebas unitarias

Fuente: Elaboración propia.

Donde en la siguiente tabla se especifican las pruebas para la calculadora de riesgo

Prueba	Descripción de Entrada	Resultado Esperado
1	Permisos sensibles y todos los de comunicación otorgados	Riesgo esperado: 5
2	Permisos sensibles e Internet/SMS otorgados.	Riesgo esperado: 4
3	Permisos sensibles y Bluetooth/NFC otorgados.	Riesgo esperado: 3
4	Mas de 1 permiso sensible otorgado.	Riesgo esperado: 2
5	Solo un permiso sensible otorgado.	Riesgo esperado: 1
6	Ningún permiso sensible otorgado.	Riesgo esperado: 0

Tabla 9: Pruebas de caja negra para el método calculateRisk.

4.5. Integración y prueba del sistema.

4.5.1. Integración continua

Se configura un paso de integración continua, donde mediante para la ejecución de “pruebas de unidad local”²⁶ a través de un workflow de github actions²⁷ ver 32 que corresponde al

²⁶<https://developer.android.com/studio/test/command-line?hl=es-419>

²⁷<https://github.com/features/actions>

código fuente de la integración continua.

```

name: Android Build and Test

on:
  push:
    branches:
      - master

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Set up JDK
        uses: actions/setup-java@v2
        with:
          distribution: 'adopt'
          java-version: '11'

      - name: Set execute permissions for gradlew
        run: chmod +x ./gradlew

      - name: Cache Gradle dependencies
        uses: actions/cache@v2
        with:
          path: |
            ~/.gradle/caches
            ~/.gradle/wrapper
          key: ${{ runner.os }}-gradle-${{ hashFiles('**/*.gradle*', '**/gradle-wrapper.properties') }}
          restore-keys: |
            ${{ runner.os }}-gradle-

      - name: Run specific unit tests
        run: ./gradlew test
  
```

Figura 32: GitHub Integración continua
Fuente: Elaboración propia.

Lo que se ejecuta exitosamente en 33:

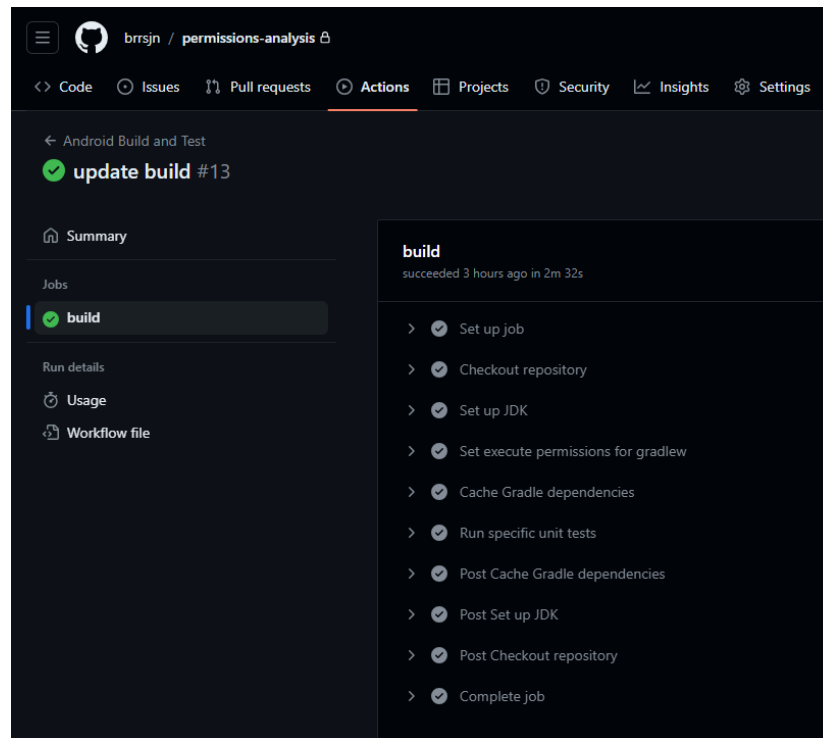


Figura 33: Resultado GitHub Integración continua
Fuente: Elaboración propia.

4.5.2. [HU-01] Listar aplicaciones instaladas

Para este caso se consideraron solo las aplicaciones que no pertenecen al sistema. Cuando se entra por primera vez a la aplicación, se muestra una ventana de carga que significa que se está analizando el dispositivo, ver fig 34.

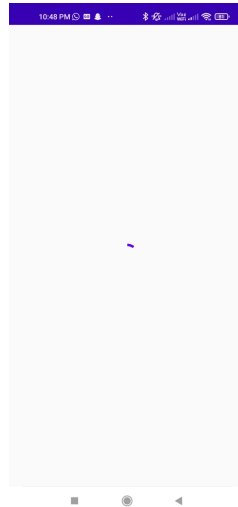


Figura 34: [HU-01] - Primer inicio
Fuente: Elaboración propia.

Luego, una vez analizado, se muestra la pantalla de 35 que representa a que está buscando ya en memoria el listado de aplicaciones para mostrar.



Figura 35: [HU-01] - Segundo inicio
Fuente: Elaboración propia.

Una vez cargado de la memoria el listado de aplicaciones instaladas, se presenta en un formato de Card, junto con su icono, nombre y nombre del paquete cada , ver 36.

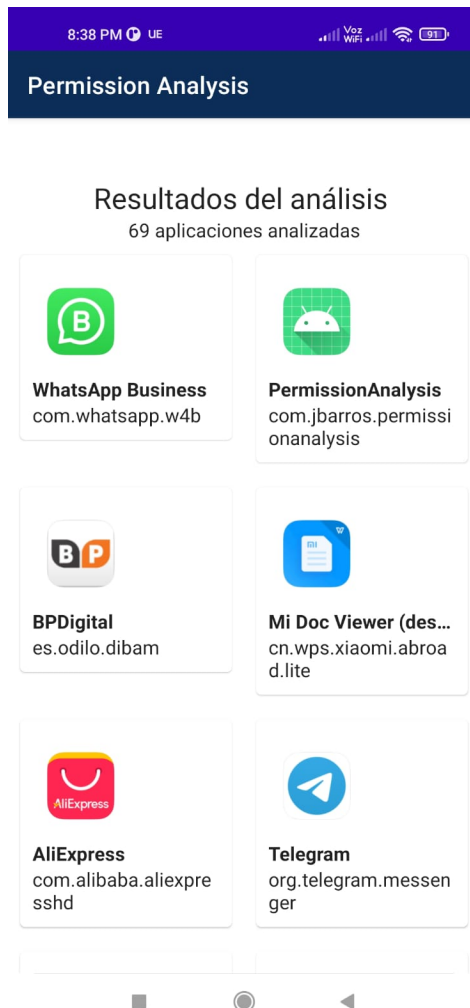


Figura 36: [HU-01] listado aplicaciones instaladas
Fuente: Elaboración propia.

4.5.3. [HU-02] Abrir detalle aplicación

Una vez desplegado el listado de aplicaciones, cuando se le pulsa la Card de una aplicación, entonces, se muestra el detalle de la aplicación incluyendo datos del análisis de riesgo, también, botones para acceder a acciones de otras Historias de usuario. Esto se puede ver en 37 y 38

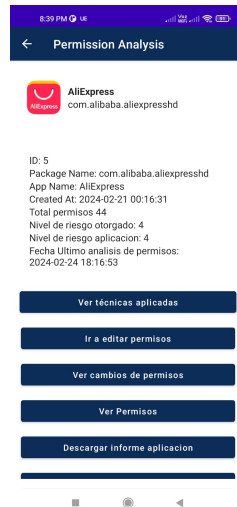


Figura 37: [HU-02] Vista detalle de aplicación parte a
Fuente: Elaboración propia.



Figura 38: [HU-02] Vista detalle de aplicación parte b
Fuente: Elaboración propia.

4.5.4. [HU-03] Abrir editar permisos

Dentro de la vista de detalle de la aplicación, está el botón de editar permisos, esta acción requiere acceder al navegador, considerando que con ello el usuario sale de la aplicación es que se añade un cuadro de dialogo para confirmar que el usuario quiere salir de la aplicación e ir a configuraciones para hacer su edición de permisos.

En la siguiente figura se puede ser el cuadro de imagen abierto

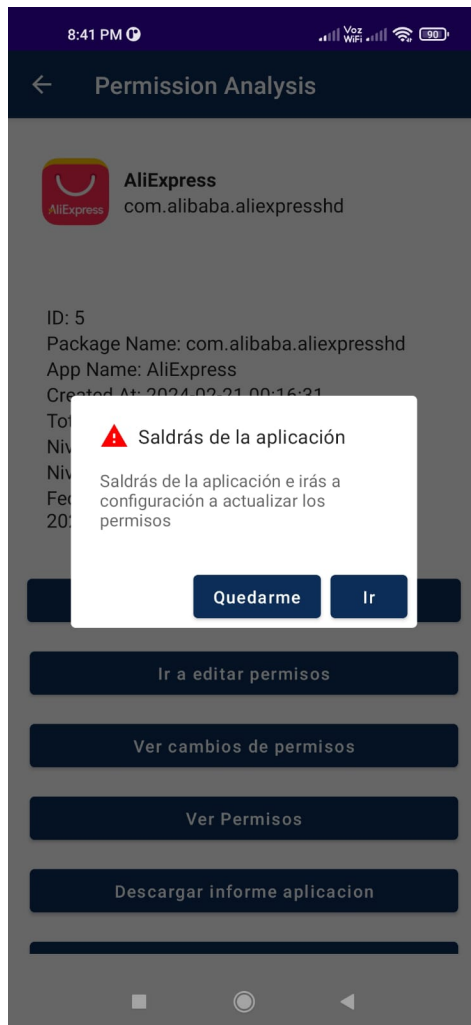


Figura 39: [HU-03] Dialogo ir a editar permisos
Fuente: Elaboración propia.

Una vez se selecciona ir, entonces, se abre una actividad de Android que lleva a configuraciones de la aplicación, dejando visible la actualización de permisos.

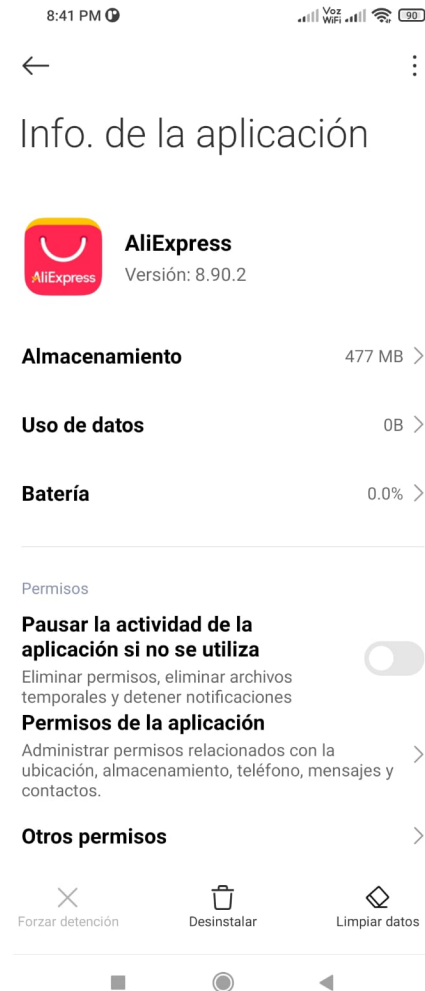


Figura 40: [HU-03] Información de aplicación desde configuraciones
Fuente: Elaboración propia.

Luego cuando se selecciona permisos, se pueden ver a nivel de permisos peligrosos los aceptados y no aceptados, en la siguiente imagen se pueden ver permisos peligrosos no aceptados.

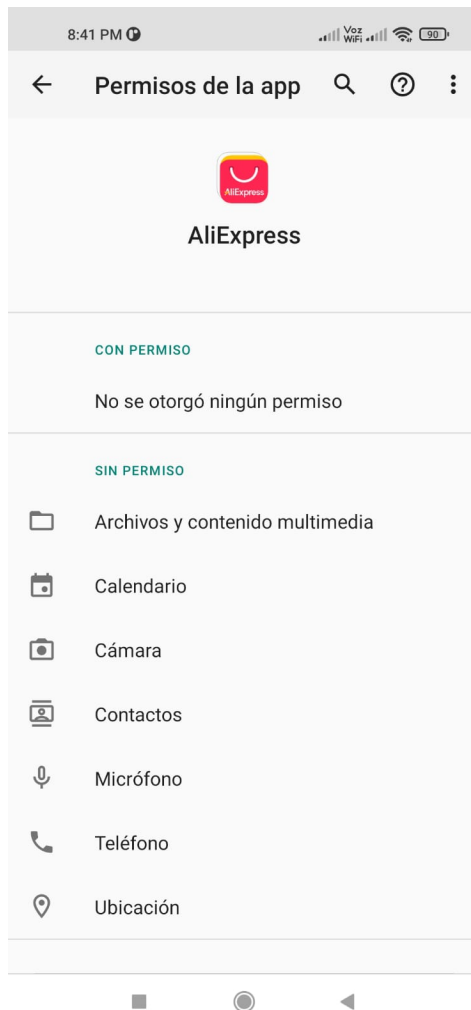


Figura 41: [HU-03] Vista de permisos con permisos no aceptados
Fuente: Elaboración propia.

Después en la siguiente imagen, se pueden ver un par de permisos peligrosos aceptados.

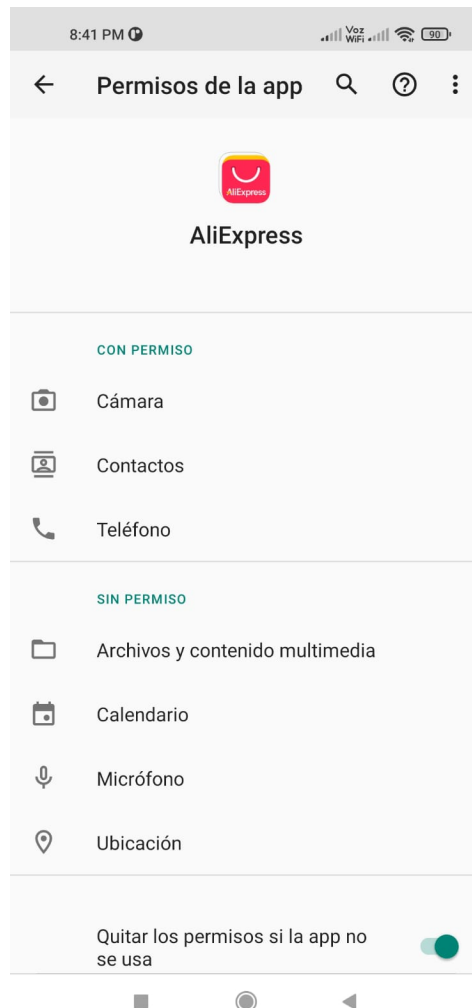


Figura 42: [HU-03] Vista de permisos con permisos aceptados
Fuente: Elaboración propia.

4.5.5. [HU-04] Descargar informe aplicación

Desde la vista del detalle de la aplicación, al pulsar en el botón “Descargar informe aplicación” entonces, se descarga un archivo json que inmediatamente puede ser compartido, su nombre está dado por el nombre del paquete de la aplicación y la fecha actual.

En la siguiente imagen se ve que una vez presionado el botón, primero muestra un mensaje de que ha sido guardado en el dispositivo en la carpeta de descargas.

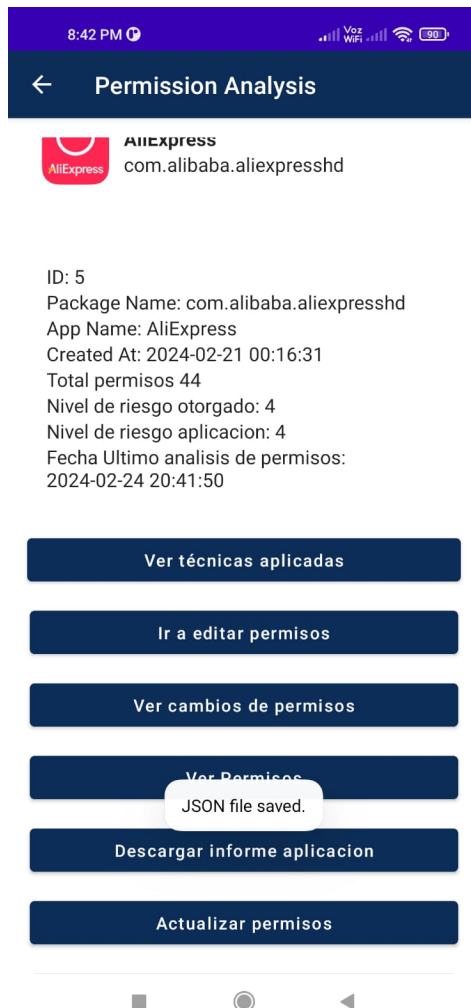


Figura 43: [HU-04] Dialogo descargar informe
Fuente: Elaboración propia.

Luego aparece la opción de compartir el informe, el cual puede ser enviado por alguna herramienta de mensajería como el correo electrónico.

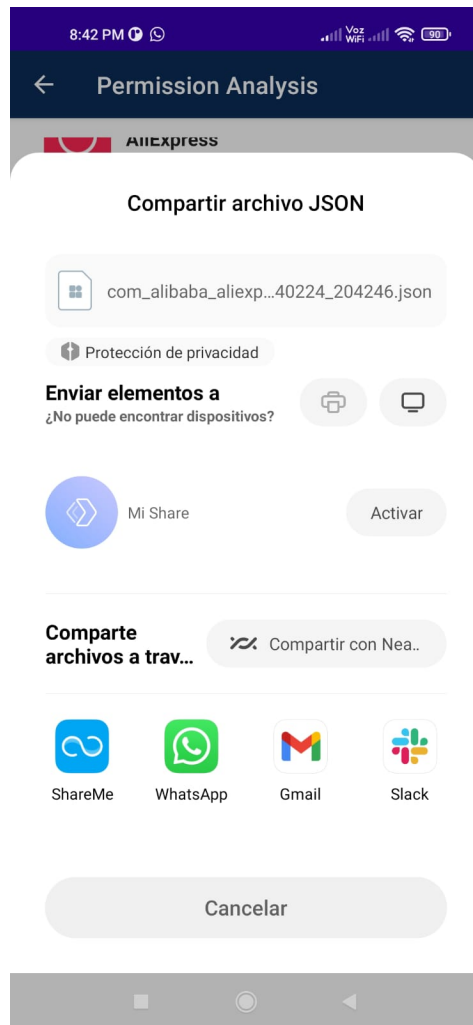


Figura 44: [HU-04] Compartir informe
Fuente: Elaboración propia.

Luego, digamos que se abre el informe desde un computador con un editor de código, entonces, se muestra de la siguiente forma:

```

{
  "application": {
    "apkPath": "/data/app/~~gCoyus6q6rtDQY_e8UYRvw\u003d\u003d/com.jbarros.permissionanalysis-
    pXI17bRbPa5mwJ91S-qaNQ\u003d\u003d/base.apk",
    "appIcon": {
      "mPaint": {
        "mNativePaint": -5476376604075997696
      },
      "mSrcDensityOverride": 0
    },
    "appName": "PermissionAnalysis",
    "createdAt": "2024-02-24 22:48:59",
    "id": 2,
    "packageName": "com.jbarros.permissionanalysis",
    "uniqueUserId": "10378"
  },
  "permissionChangeStrings": [],
  "permissionsName": [
    {
      "constantName": "android.permission.ACCESS_NETWORK_STATE",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "ACCESS_NETWORK_STATE",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.FOREGROUND_SERVICE",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "FOREGROUND_SERVICE",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.INTERNET",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "INTERNET",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.QUERY_ALL_PACKAGES",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "QUERY_ALL_PACKAGES",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.READ_EXTERNAL_STORAGE",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": false,
      "name": "READ_EXTERNAL_STORAGE",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.RECEIVE_BOOT_COMPLETED",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "RECEIVE_BOOT_COMPLETED",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.WAKE_LOCK",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "WAKE_LOCK",
      "updatedAt": ""
    },
    {
      "constantName": "android.permission.WRITE_EXTERNAL_STORAGE",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": false,
      "name": "WRITE_EXTERNAL_STORAGE",
      "updatedAt": ""
    },
    {
      "constantName": "com.jbarros.permissionanalysis.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION",
      "createdAt": "2024-02-24 22:49:03",
      "isGranted": true,
      "name": "com.jbarros.permissionanalysis.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION",
      "updatedAt": ""
    }
  ]
}
    
```

Figura 45: [HU-04] Json exportado parte a
Fuente: Elaboración propia.

Donde se muestran datos de la aplicación, también un listado de los permisos cambiados desde que la aplicación fue instalada, los permisos aceptados y rechazados, el análisis de riesgo, y los permisos de categorías sensibles a los que pertenece.

```
{
  "riskAnalysis": {
    "applicationUid": 2,
    "createdAt": "2024-02-24 22:49:03",
    "id": 2,
    "riskScore": 4,
    "riskScoreRequested": 4
  },
  "riskAnalysisInfo": [
    {
      "permissionName": "QUERY_ALL_PACKAGES",
      "sensitiveDataCategoryName": "no-category"
    },
    {
      "permissionName": "INTERNET",
      "sensitiveDataCategoryName": "no-category"
    },
    {
      "permissionName": "WRITE_EXTERNAL_STORAGE",
      "sensitiveDataCategoryName": "no-category"
    },
    {
      "permissionName": "READ_EXTERNAL_STORAGE",
      "sensitiveDataCategoryName": "EXTERNAL_STORAGE_DATA"
    },
    {
      "permissionName": "WAKE_LOCK",
      "sensitiveDataCategoryName": "no-category"
    },
    {
      "permissionName": "ACCESS_NETWORK_STATE",
      "sensitiveDataCategoryName": "NETWORK_INFORMATION"
    },
    {
      "permissionName": "RECEIVE_BOOT_COMPLETED",
      "sensitiveDataCategoryName": "no-category"
    },
    {
      "permissionName": "FOREGROUND_SERVICE",
      "sensitiveDataCategoryName": "no-category"
    },
    {
      "permissionName": "com.jbarros.permissionanalysis.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION",
      "sensitiveDataCategoryName": "no-category"
    }
  ]
}
```

Figura 46: [HU-04] Json exportado parte b
Fuente: Elaboración propia.

4.5.6. [HU-05] Resumen de política de privacidad

Para poder ver el resumen de la política de privacidad, desde la vista de detalle de aplicación, se debe ir al botón “Ver técnicas aplicadas”, en él se debe pulsar “Ver resumen política de privacidad” entonces, se mostrara la siguiente imagen con el resumen de la política de

privacidad:

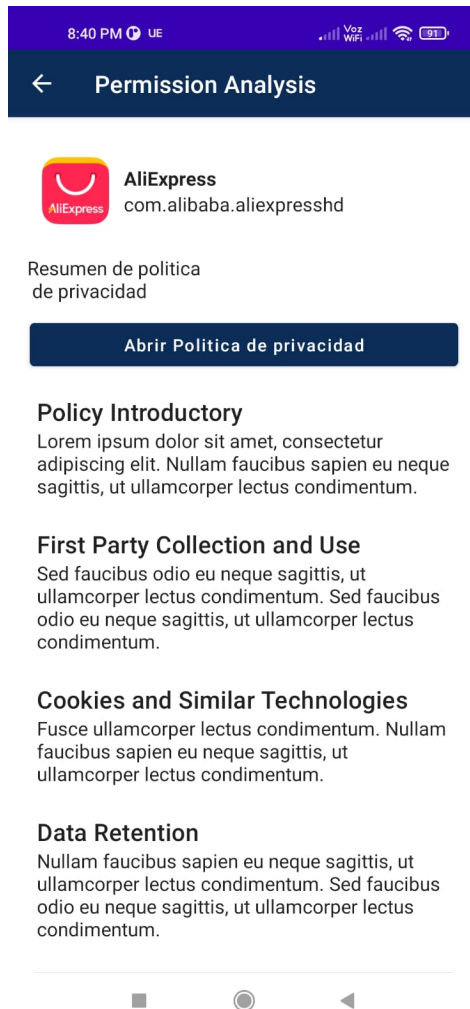


Figura 47: [HU-05] Vista resumen de política de privacidad
Fuente: Elaboración propia.

Los datos del resumen de la política de privacidad es mostrado con datos dummy.

4.5.7. [HU-06] Ver política de privacidad

Dentro de la vista del resumen de la política de privacidad, se mostrará el botón “Abrir política de privacidad” solo si la aplicación tiene y fue descargada previamente desde el Play Store. Una vez pulsado se mostrará el siguiente cuadro de dialogo:

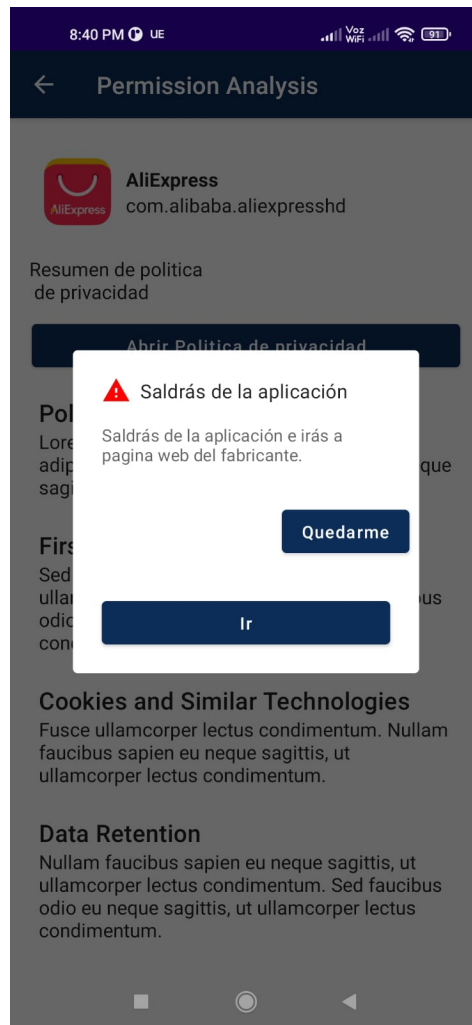


Figura 48: [HU-06 Dialogo ir a política de privacidad
Fuente: Elaboración propia.

El que al aceptarlo, entonces se abre el navegador junto con la política de privacidad dentro de la pagina web del fabricante, en la siguiente imagen se puede ver el caso de Aliexpress

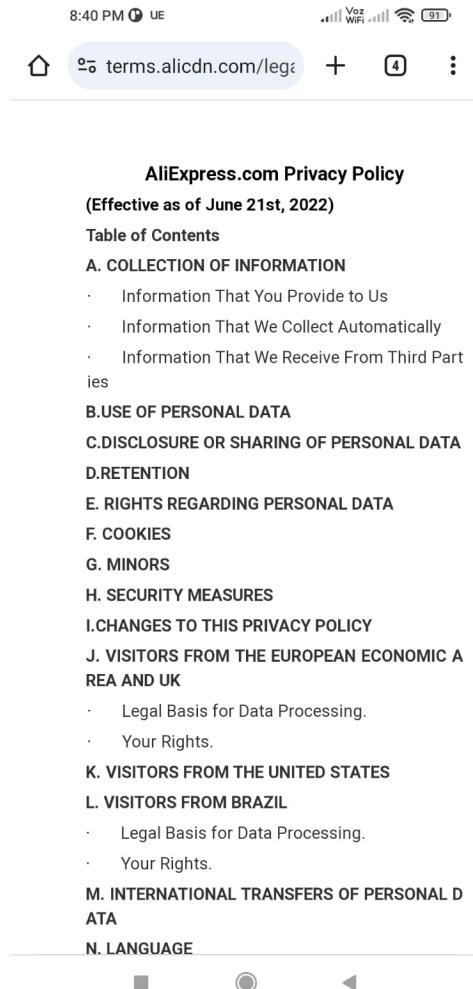


Figura 49: [HU-06] Política de privacidad Aliexpress
Fuente: Elaboración propia.

4.5.8. [HU-07] Grupos de permisos según descripción

Para poder ver los grupos de permisos según descripción, desde la vista de detalle de aplicación, se debe ir al botón “Ver técnicas aplicadas” , en él se debe pulsar “Ver análisis descripción” entonces, se mostrara la siguiente imagen con los grupos de permisos que podrían pertenecer a la aplicación según la descripción, en el se mancan con un check los grupos que si podrían pertenecer:

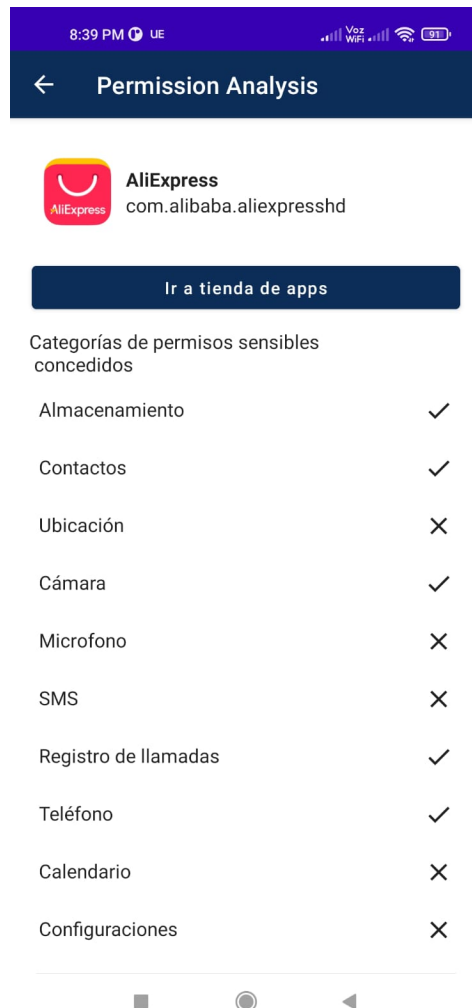


Figura 50: [HU-07] Grupos de permisos según descripción
Fuente: Elaboración propia.

Los datos de si pertenece o no son creados de forma aleatoria, lo que permite que para una próxima integración, se haga a través de la API hacia el sistema de ML que implemente la solución.

4.5.9. [HU-08] Ver descripción

Una vez abierta la vista de análisis de descripción, aparece el botón “Ir a tienda de apps”, el que al pulsar aparece el siguiente cuadro de dialogo:

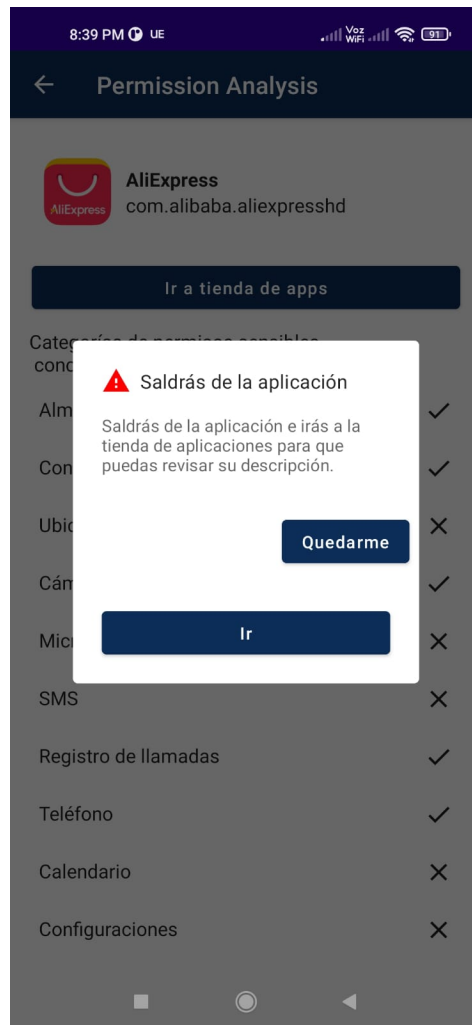


Figura 51: [HU-08] Dialogo ir tienda de aplicaciones
Fuente: Elaboración propia.

Luego, al pulsar en ir, entonces, se abrirá la tienda de aplicaciones donde se puede ver la descripción de la aplicación.



Figura 52: [HU-08] Tienda de aplicaciones para Aliexpress
Fuente: Elaboración propia.

4.5.10. [HU-09] Ver análisis de riesgo

Para poder ver los grupos de permisos según descripción, desde la vista de detalle de aplicación, se debe ir al botón “Ver técnicas aplicadas”, en él se debe pulsar a “Ver análisis de riesgo” entonces, se mostrara la siguiente vista con un listado de Categoría - Permiso.

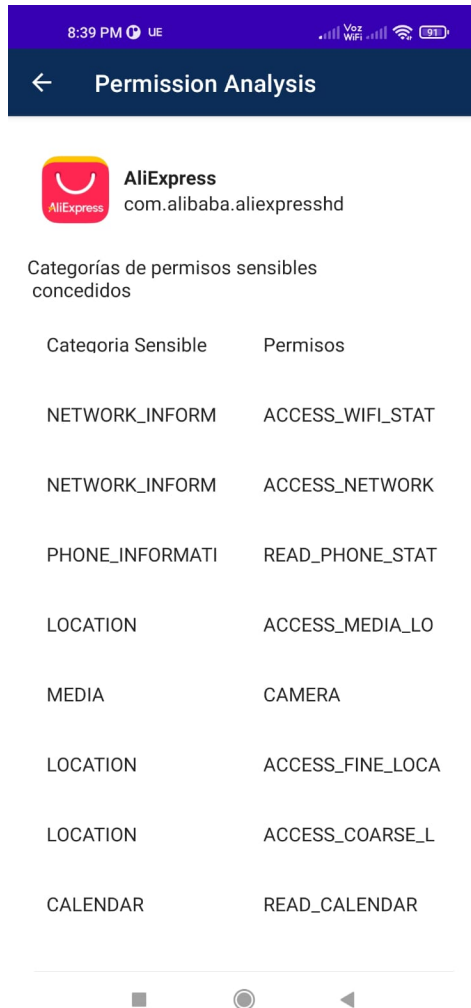


Figura 53: [HU-09] Informe de riesgo parte a
Fuente: Elaboración propia.

Al final de la pantalla, se muestra un numero del 1 al 5 correspondiente a su valor de riesgo solicitado por la aplicación.

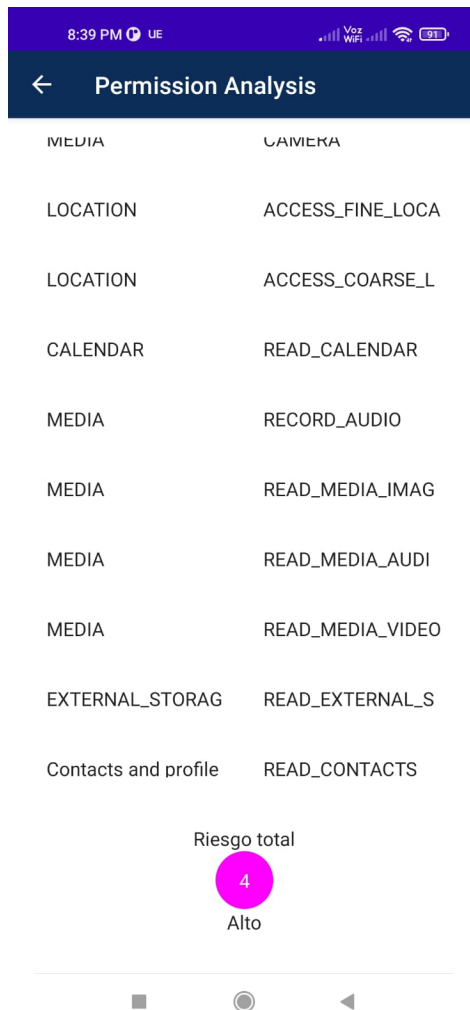


Figura 54: [HU-09] Informe de riesgo parte b
Fuente: Elaboración propia.

4.5.11. [HU-10] Historial de cambios en permisos

Cuando se acepta o rechaza algún permiso, este se mostrará en la vista “Ver cambios de permisos” desde la vista del detalle de la aplicación, en ella se abrirá la siguiente vista con el nombre del permiso, su nuevo estado y la fecha en que fue detectado el cambio.

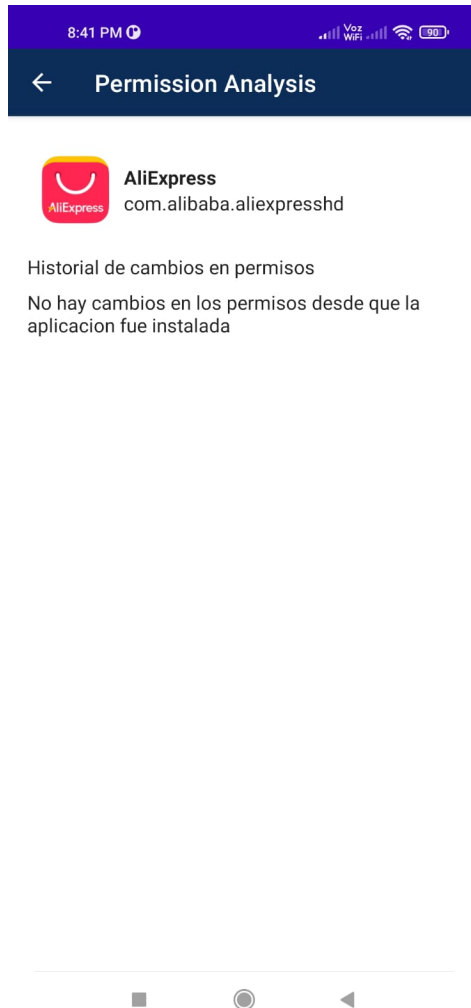


Figura 55: [HU-10] Vista sin cambios
Fuente: Elaboración propia.

Si la aplicación no presenta cambios en los permisos, entonces, se mostrará la siguiente vista:



Figura 56: [HU-10] Cambios en permisos
Fuente: Elaboración propia.

4.5.12. [HU-11] Nuevo análisis

Una vez ya se ha usado la aplicación, y se han instalado y eliminado otras aplicaciones, entonces, es momento de actualizar el listado, para ello, se dispone en la vista un botón cuyo texto es: "Actualizar análisis", que tras pulsarlo, revisará si hay aplicaciones nuevas, aplicaciones eliminadas, calcula análisis de riesgo, actualiza el listado de permisos y actualiza los cambios de permisos.

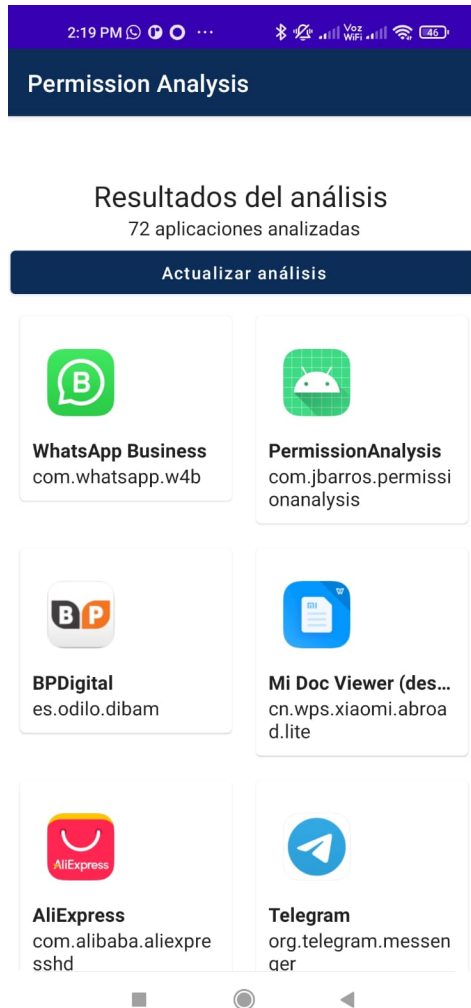


Figura 57: [HU-11] Nuevo análisis: Sin cambios
Fuente: Elaboración propia.

Cuando ha terminado de procesar, se muestra el siguiente mensaje:

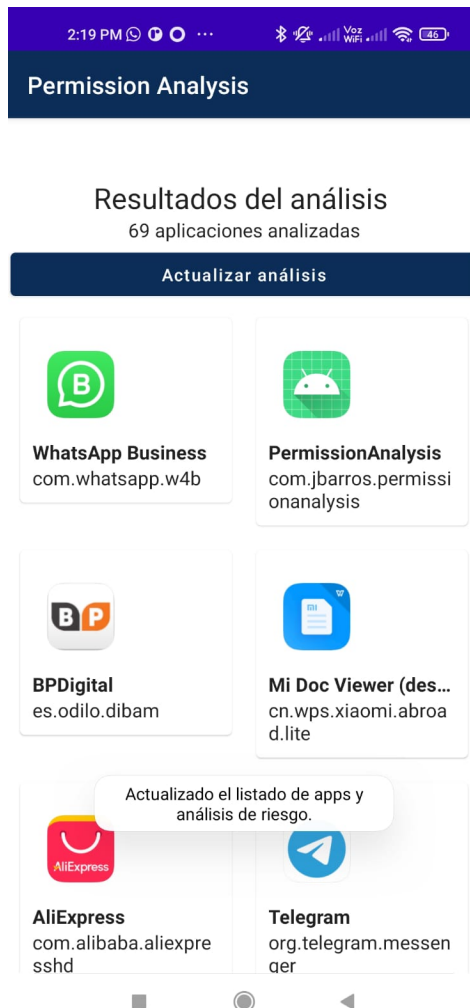


Figura 58: [HU-11] Nuevo análisis: Cambios
Fuente: Elaboración propia.

4.5.13. [HU-12] Nuevo análisis por aplicación

En el detalle de la aplicación, cuando se selecciona el botón de “Actualizar permisos” , entonces, se re calcula el análisis de riesgo, actualiza el listado de permisos y actualiza los cambios de permisos. También muestra el siguiente mensaje:

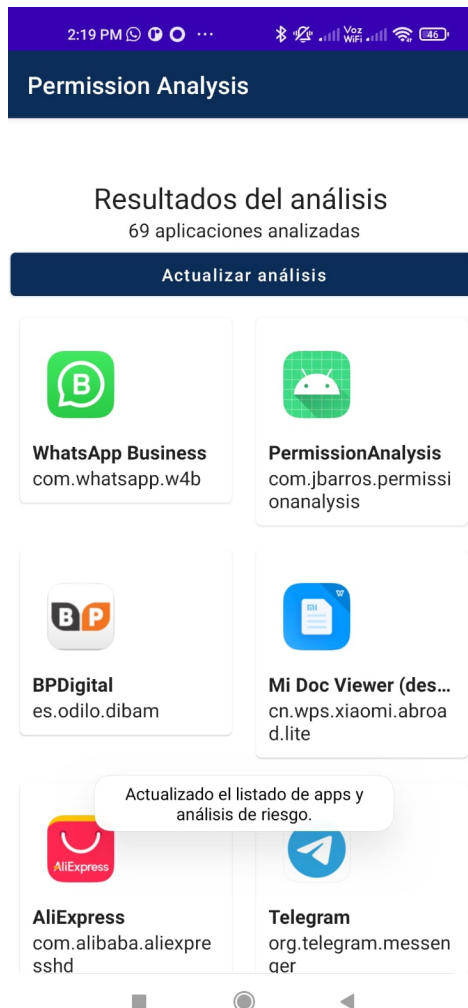


Figura 59: [HU-12] Nuevo análisis por aplicación
Fuente: Elaboración propia.

4.6. Operación y mantenimiento.

Pensando en su operación y mantenimiento, es correcto verificar el estado de los casos de uso de la aplicación y su completitud, esto dado a que existen dos técnicas las cuales fueron diseñadas pero no implementadas.

Tabla 10: Estado casos de uso

Caso de Uso	Cliente Android	Servidor	Comentario
Análisis de riesgo	Completo	No aplica	No se guardan datos de usuario en servidor
Análisis descripción	Completo	Incompleto	Falta implementación de modelo de NLP, conectar desde servidor
Resumen Política privacidad	Completo	Incompleto	Falta implementación de modelo de NLP, conectar desde servidor
Exportar Reporte	Completo	No aplica	No se guardan datos de usuario en servidor
Historial de cambios	Completo	No aplica	No se guardan datos de usuario en servidor

Además, considerar que los artefactos a mantener y operar para hacer funcionar la aplicación son los siguientes:

Tabla 11: Repositorios

Nombre	Descripción	Repositorio
Auditor privacidad	Aplicación Android, auditor de privacidad	https://github.com/brrsjn/permissions-analysis
Servidor	Es usado como API web, se conecta al servicio de extracción, y tiene interfaces hechas para la implementación de los servicios de NLP	https://github.com/brrsjn/permissions-analysis-api
Servicio extracción información permisos	Function as a service que se usa para extraer datos sobre los nombres de los permisos	https://github.com/brrsjn/permission-extract-info

CONCLUSIONES

En esta memoria, se propuso e implementó una solución para abordar los desafíos de privacidad en el entorno Android, donde la proliferación de aplicaciones y el extenso despliegue de permisos comprometen la seguridad de los usuarios. La construcción y validación de un software que audita la privacidad de aplicaciones Android mediante el análisis de permisos se reveló como una herramienta fundamental frente a la creciente preocupación por la ciberseguridad y la protección de información personal sensible.

La era digital actual ha dejado una huella significativa en la historia de la humanidad, no solo por la transformación tecnológica y digital que ha supuesto, sino también por los desafíos que presenta en términos de privacidad y seguridad informática. Los dispositivos móviles, especialmente aquellos que operan bajo el sistema operativo Android debido a su amplia adopción, son elementos cruciales en este complejo escenario, convirtiéndose en fuentes de datos personales y sensibles de sus usuarios.

Este proyecto se centró en proponer, diseñar e implementar un sistema de auditoría de privacidad para aplicaciones Android, basado en el principio de mínimo privilegio y en un análisis exhaustivo de permisos. A lo largo de la memoria, se presentaron los fundamentos teóricos y conceptuales que respaldan el sistema operativo Android, así como las características específicas de seguridad y permisos que constituyen su arquitectura. Estos fundamentos fueron esenciales para comprender la magnitud del problema y para desarrollar una solución efectiva que responda a la necesidad de proteger la privacidad de los usuarios.

El software desarrollado emerge como una solución innovadora que permite a los usuarios obtener una comprensión y control mayores sobre los permisos solicitados y obtenidos por las aplicaciones instaladas en sus dispositivos. A través del análisis de permisos, el software audita aplicaciones preinstaladas y de terceros, proporcionando una evaluación de riesgo basada en la combinación y el tipo de permisos solicitados por cada aplicación. Esta evaluación brinda a los usuarios una herramienta valiosa para tomar decisiones informadas sobre la gestión de permisos y la instalación de aplicaciones, fortaleciendo así su autonomía sobre la privacidad de sus datos.

La metodología de desarrollo empleada, centrada en la aplicación de la metodología cascada y en un enfoque iterativo para la validación de la solución, ha permitido alcanzar los objetivos específicos planteados y abordar con éxito los desafíos inherentes al desarrollo de software en el ámbito de la ciberseguridad. La implementación de pruebas unitarias, la integración continua y los procesos de validación han sido esenciales para asegurar la calidad del software desarrollado y su eficacia para cumplir con su propósito.

Se construyó una aplicación que analiza un dispositivo Android para obtener un listado de aplicaciones instaladas, sus permisos asignados y solicitados utilizando la API de Android PackageManager, una herramienta útil para extraer esta información clave para evaluar el riesgo de la aplicación.

Se configuró una arquitectura MVVM en Android, lo que permite aprovechar un patrón para gestionar el estado de las variables entre las vistas. Además, se implementó la inyección de dependencias, un patrón útil que, junto con el patrón singleton, permite configurar en un solo lugar las instancias de cada servicio o conexión, como la base de datos, el Data Access Object o la conexión a la API web. La base de datos utilizada es Room, una extensión de SQLite que permite almacenar datos en el dispositivo Android. También se configuraron conexiones a la API mediante el paquete Retrofit.

Se llevó a cabo una evaluación de riesgo, implementando una calculadora de riesgo junto con la identificación de si un permiso perteneciente a la aplicación accede a información sensible. En la calculadora de riesgo se considera si una aplicación accede a información sensible a través de uno o más permisos, combinado con permisos asociados a la comunicación, como "Acceso a internet", "Bluetooth", "Envío de SMS" y "NFC", ya que ponen en riesgo la fuga de información sensible. Los permisos que acceden a información sensible fueron actualizados teniendo en cuenta las nuevas actualizaciones de permisos, como agregar los diferentes niveles de precisión de ubicación.

Se presenta un informe de análisis de permisos junto con un resumen de la política de privacidad. Para ambos casos, se preparó una vista a nivel de Android junto con su respectiva conexión a la API web. También se añadió un enlace para ver la descripción original y otro para acceder a la política de privacidad desde la web del productor de la aplicación. Desde la API web, se dejaron configurados los servicios para integrar los servicios de ML para el análisis de descripciones junto con el resumen de privacidad, sugiriendo el uso de inferencia serverless a través de un framework de MLOPS como AWS SageMaker.

A pesar de haber cumplido los objetivos, se deja abierto un amplio espacio para añadir mejoras. En el campo de la experiencia de usuario, se puede llevar a cabo un estudio de usabilidad para obtener retroalimentación de los usuarios de Android. Esto permitiría un rediseño de la presentación de la información y la comunicación del sistema auditor de privacidad con el usuario. También se podría evaluar el uso de notificaciones push cuando se identifiquen cambios de permisos que afecten la información sensible en las aplicaciones.

El impacto potencial de este proyecto en la comunidad de usuarios de Android es significativo. Al ofrecer una herramienta capaz de auditar la privacidad de las aplicaciones instaladas, se contribuye a elevar la conciencia sobre la importancia de la seguridad de los datos personales y se promueve una cultura de privacidad y seguridad informática. Además, el software propuesto podría servir como base para futuras investigaciones y desarrollos en el campo de la ciberseguridad móvil, abriendo nuevas vías para la exploración de soluciones avanzadas que protejan la privacidad de los usuarios en un entorno digital cada vez más complejo y desafiante.

En conclusión, este proyecto de memoria ha alcanzado su objetivo de diseñar e implementar un sistema de auditoría de privacidad para aplicaciones Android, marcando un hito en la búsqueda de soluciones que aborden la creciente preocupación por la seguridad de la información personal en dispositivos móviles. Las bases establecidas en este trabajo sientan las

premisas para futuras investigaciones y desarrollos en la materia, destacando la importancia crítica de proteger la privacidad en la era digital.

REFERENCIAS BIBLIOGRÁFICAS

- [Alepis y Patsakis, 2019] Alepis, E. y Patsakis, C. (2019). Unravelling security issues of runtime permissions in android. *Journal of Hardware and Systems Security*, 3(1):45–63.
- [Almomani y Khayer, 2020] Almomani, I. M. y Khayer, A. A. (2020). A comprehensive analysis of the android permissions system. *IEEE Access*, 8:216671–216688.
- [Android Developers, 2024] Android Developers (2024). Codelab: Activity Lifecycle in Jetpack Compose. Accedido el 12 de octubre de 2024.
- [Dalmasso et al., 2013] Dalmasso, I., Datta, S. K., Bonnet, C., y Nikaein, N. (2013). Survey, comparison and evaluation of cross platform mobile application development tools. En *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 323–328.
- [Developers, 2024] Developers, A. (2024). Viewmodel | android developers. Accedido el 26 de febrero de 2024.
- [Faruki et al., 2015] Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., y Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys Tutorials*, 17(2):998–1022.
- [Feng et al., 2019] Feng, Y., Chen, L., Zheng, A., Gao, C., y Zheng, Z. (2019). Ac-net: Assessing the consistency of description and permission in android apps. *IEEE Access*, 7:57829–57842.
- [Fox et al., 2017] Fox, G. C., Isahagian, V., Muthusamy, V., y Slominski, A. (2017). Status of serverless computing and function-as-a-service(faas) in industry and research. *ArXiv*, abs/1708.08028.
- [Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., y Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.
- [Irwan et al., 2015] Irwan, Asnar, Y., y Hendradjaya, B. (2015). Confidentiality and privacy information security risk assessment for android-based mobile devices. En *2015 International Conference on Data and Software Engineering (ICoDSE)*, pp. 1–6.
- [Laigner et al., 2019] Laigner, R., Kalinowski, M., Carvalho, L., Mendonça, D., y Garcia, A. (2019). Towards a catalog of java dependency injection anti-patterns. En *Proceedings of the XXXIII Brazilian Symposium on Software Engineering, SBES '19*, p. 104–113, New York, NY, USA. Association for Computing Machinery.
- [Laurence et al., 2021] Laurence, P.-O., Hinchman-Dominguez, A., Dunn, M., y Meike, G. B. (2021). *Programming Android with Kotlin*. O'Reilly Media, Inc.

- [Liu *et al.*, 2020] Liu, S., Guo, R., Zhao, B., Chen, T., y Zhang, M. (2020). Appcorp: A corpus for android privacy policy document structure analysis. *CoRR*, abs/2005.06945.
- [Ngobeni y Mhlongo, 2019] Ngobeni, A. y Mhlongo, S. (2019). Towards enhancing security in android operating systems – android permissions user unawareness. En *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, pp. 1–6.
- [Refactoring Guru, 2023] Refactoring Guru (2023). Singleton. <https://refactoring.guru/es/design-patterns/singleton>. Último acceso: enero 2024.
- [Shah *et al.*, 2019] Shah, K., Sinha, H., y Mishra, P. (2019). Analysis of cross-platform mobile app development tools. En *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pp. 1–7.
- [Sheshadri y Lakshmi, 2022] Sheshadri, K. R. y Lakshmi, J. (2022). Qos aware faas for heterogeneous edge-cloud continuum. En *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pp. 70–80.
- [Sommerville, 2011] Sommerville, I. (2011). *Ingeniería de software*. Pearson, 9 edición.
- [van Eyk *et al.*, 2019] van Eyk, E., Grohmann, J., Eismann, S., Bauer, A., Versluis, L., Toader, L., Schmitt, N., Herbst, N., Abad, C. L., e Iosup, A. (2019). The spec-rg reference architecture for faas: From microservices and containers to serverless platforms. *IEEE Internet Computing*, 23(6):7–18.