Universidad de Valladolid

Máster universitario

Ingeniería Informática

Trabajo Fin de Máster

Selección de indicadores basada en análisis de datos: Aplicación a la gobernanza de la privacidad y la seguridad en aplicaciones móviles.

Realizado por **Víctor Diez Pérez**

✠✠✠

Universidad de Valladolid

27 de junio de 2025

Tutores: M. Mercedes Martínez González y Amador Aparicio de la Fuente

Universidad de Valladolid

Máster Universitario en Ingeniería Informática

D. M. Mercedes Martínez González y Amador Aparicio de la Fuente , profesores del departamento de Informática de la Universidad de Valladolid, área de Ingeniería de la Privacidad de la Universidad de Valladolid.

Expone:

Que el alumno D. Víctor Diez Pérez, ha realizado el Trabajo final de Máster en Ingeniería Informática titulado "Selección de indicadores basada en análisis de datos: Aplicación a la gobernanza de la privacidad y la seguridad en aplicaciones móviles.".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Valladolid, 27 de junio de 2025

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D.ª M. Mercedes Martínez González	D. Amador Aparicio de la Fuente

Agradecimientos

Una vez que hemos llegado a este punto en el que se reflejan estos años de aprendizaje, me gustaría primero agradecérselo a mis padres, por su apoyo incondicional. Su ayuda, comprensión y constante compañía han sido fundamentales en cada etapa de mi vida. Siempre presentes con una palabra de aliento y ánimo justo cuando más lo necesitaba.

También, me gustaría destacar la ayuda que he recibido y encontrado en el Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid, a Mercedes y Amador como mis tutores y Alejando por toda su atención, dedicación y ayuda durante todo el Trabajo Fin de Máster en el que me he sentido orientado, guiado y acompañado en todos y cada uno de estos momentos.

Y por último, a todas aquellas personas que me han acompañado durante mi recorrido.

Resumen

En la actualidad, el número de aplicaciones que se descargan en móviles Android y que son utilizados para distintas actividades no dejan de crecer. Cada una de estas aplicaciones, cuenta con sus propias políticas de privacidad y solicitudes de permisos. Sin embargo, muchas de estas aplicaciones solicitan un número excesivo de permisos, lo que al usuario le pueden llegar a generar ciertas dudas razonables sobre la verdadera necesidad de dichos permisos y sobretodo, plantearse acerca del uso que realmente se hace de la información recopilada por dichas aplicaciones.

En este Trabajo de Fin de Máster nos planteamos si es posible estimar el riesgo que suponen las aplicaciones para la privacidad y seguridad de sus usuarios. Nuestra propuesta es construir indicadores basados en características como el número de permisos solicitados, categoría, y otros metadatos de las aplicaciones. Entre nuestros objetivos está ofrecer a los usuarios una herramienta que les sirva de apoyo para manejar esta información y tomar mejores decisiones para autoprotegerse. Además, este proyecto se apoya en el uso de inteligencia artificial, para encontrar algoritmos que ayuden a detectar aplicaciones que puedan ser particularmente intrusivas.

Abstract

Currently, the number of applications downloaded on Android devices and used for various activities continues to grow. Each of these applications has its own privacy policies and permission requests. However, many of these apps request an excessive number of permissions, which can raise reasonable concerns for users about the true necessity of these permissions and, above all, prompt questions about how the collected information is actually being used.

In this Master's Thesis, we explore whether it is possible to estimate the risk that applications pose to the privacy and security of their users. Our proposal is to build indicators based on features such as the number of permissions requested, app category, and other metadata. One of our main goals is to provide users with a tool that helps them manage this information and make better decisions to protect themselves. Moreover, this project relies on the use of artificial intelligence to identify algorithms that can help detect particularly intrusive applications.

Índice general

Índice de figuras

Índice de tablas

1: Introducción

1.1. Contexto

En la actualidad, se observa en todos los campos de la sociedad, tanto en España como a nivel mundial, cómo la tecnología ha pasado de ser una herramienta de conveniencia a convertirse en un elemento esencial y habitual en la vida cotidiana de cada persona. Cada vez es más patente cómo la sociedad se encuentra en un mundo más digitalizado en la figura 1.1 se puede observar el elevado número de descargas de diferentes aplicaciones a nivel mundial. Actividades tan comunes como realizar la compra, llevar a cabo operaciones bancarias, planificar viajes, redes sociales o gestionar tareas domésticas dependen directamente de aplicaciones en dispositivos móviles.

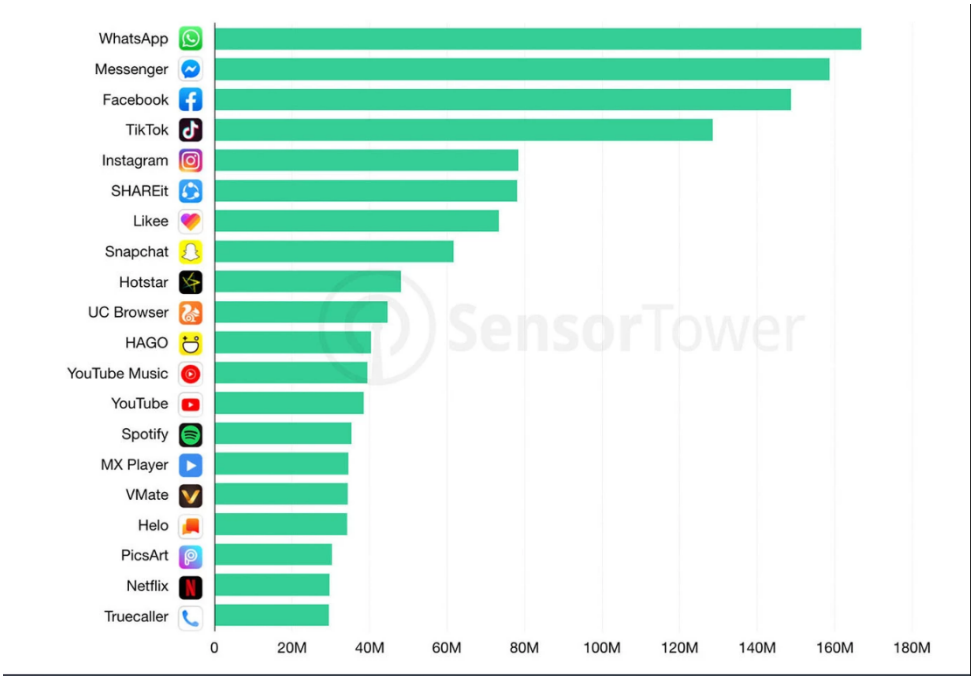

Figura 1.1: Número descargas mundiales de diferentes aplicaciones [11].

Esta creciente dependencia de aplicaciones que intentan facilitar la vida del ciudadano, plantea un interrogante crucial: ¿hasta qué punto son seguras estas aplicaciones y no invaden nuestra privacidad?

Los usuarios en su interacción constante con aplicaciones móviles se ven obligados a otorgar permisos para acceder a diversas funcionalidades, desde el uso de cámaras hasta la geolocalización o datos personales. Sin estos permisos, la mayoría de estas aplicaciones ni siquiera pueden ser utilizadas. Sin embargo, surgen preocupaciones sobre la cantidad de información solicitada, la legalidad del tratamiento de los datos recopilados y las verdaderas necesidades detrás de estas solicitudes. En este contexto, la falta de cuestionamiento por parte de muchos usuarios y el desconocimiento sobre los riesgos asociados al manejo de datos plantean una nueva pregunta ¿Es un problema de educación digital?. En el Instituto Nacional de Ciberseguridad (INCIBE), la falta de educación digital en la población general es una de sus principales preocupaciones. Por ello, dedican gran parte de sus esfuerzos a la concienciación, generando recomendaciones útiles, desarrollando actividades, talleres interesantes para abordar este problema [7].

Con este aumento exponencial de las aplicaciones que invaden nuestros dispositivos móviles y la necesidad de integrarlas en casi todos los ámbitos de la vida social, laboral y personal, resulta imprescindible prestar atención al tema de la privacidad y la seguridad. Algunos estudios como el desarrollado por NordVPN [12] muestran como muchas aplicaciones solicitan más permisos de los necesarios.

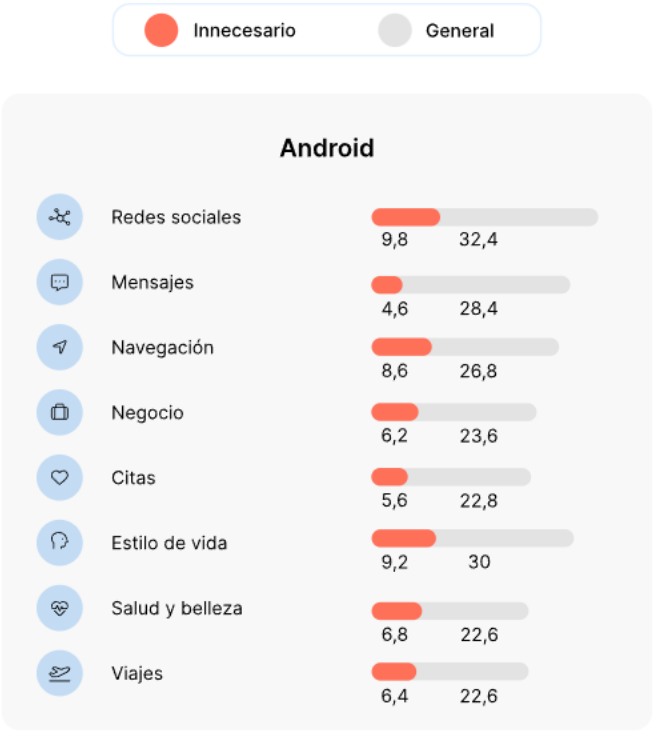

Figura 1.2: Solicitudes innecesarias por categoría [12].

1.2. Motivación

Ante esta problemática, se ha identificado un aumento de la cantidad de información personal que los usuarios comparten con las aplicaciones móviles, lo que incrementa sus riesgos de privacidad y seguridad. A esto se suma que muchos usuarios no comprenden con claridad la información que están proporcionando a las aplicaciones, ni con qué fines será utilizada. Esta falta de concienciación despierta preocupaciones e interrogantes sobre la privacidad de los datos del usuario.

De ahí que, este Trabajo de Fin de Máster haya surgido con un enfoque hacia un análisis comparativo de diversos indicadores de privacidad en dispositivos Android, sistema operativo móvil más habitual en el mercado, evaluando tanto aplicaciones individuales como categorías específicas. Además de observar la evolución de los permisos solicitados entre las versiones más recientes. Con el objetivo de aportar una herramienta con la que proporcionar mayor información al usuario y mejorar la concienciación digital de los ciudadanos hacia su privacidad en la descarga de aplicaciones.

Para este análisis, se interactua con App-PIMD [4], un repositorio que aloja los metadatos necesarios para evaluar indicadores de privacidad en sistemas Android. Con el uso de esta API se desarrollan comparativas no solo a nivel de aplicaciones individuales, sino también considerando categorías y otros criterios relevantes.

Dentro de este análisis se emplea inteligencia artificial, la cuál, se ha convertido en una herramienta muy interesante con la que detectar y encontrar patrones ocultos que esconden los datos y que a simple vista no son tan claros. Es por ello que, se ha decidido investigar el desarrollo de un algoritmo de inteligencia artificial que a partir de los datos obtenidos del repositorio de App-PIMD, agrupe los permisos más comunes de las aplicaciones con las que se trabaja en el repositorio en categorías específicas mediante técnicas de clustering. También, se emplea este modelo para identificar permisos inusuales o atípicos, con el fin de detectar posibles aplicaciones que hagan un uso indebido de los permisos solicitados, fortaleciendo así la protección del usuario.

En un mundo en el que la tecnología cada vez avanza más rápido, nos planteamos aportar información sobre los riesgos asociados a la privacidad y la seguridad de nuestras interacciones digitales cotidianas. Con el objetivo de proporcionar una herramienta al usuario que le sea de utilidad para conocer los permisos y riesgos de privacidad de aplicaciones, ser conscientes de ellos y en última medida que tenga toda la información necesaria por si desea eliminar una aplicación, elegir cuál es más conveniente descargar en función de sus necesidades. En conclusión, dar el poder al usuario de conocer los permisos solicitados por una aplicación y tomar una decisión con ellos.

1.3. Marco referencial

Para desarrollar y llevar a cabo esta investigación, aparte de la información mencionada anteriormente, se han utilizado una serie de trabajos y estudios científicos con el objetivo

de profundizar en el ámbito de la privacidad, la gestión de permisos y el desarrollo de métricas con las que poder conocer y evaluar el nivel de privacidad de una aplicación. La lectura de estos trabajos ha permitido interiorizar y comprender más de cerca el tema de como medir el nivel de intromisión de las aplicaciones en la privacidad de sus usuarios. Concretamente, estos estudios han sido:

- **Un Data Warehouse para el estudio de la privacidad y seguridad de aplicaciones móviles** (A. Pérez Fuente, M.M. Martínez González, A.A. Aparicio y Q.I. Moro) [4]. Este estudio ha sido muy útil para comprender cómo acceder a datos del repositorio App-PIMD y las distintas funcionalidades que ofrece para el análisis de aplicaciones.
- **Android Malware Detection using Feature Ranking of Permissions** (Muhammad Suleman Saleem) [1]. Ha permitido conocer como ciertos permisos están relacionados con ciertos tipos de malware o comportamientos que pueden considerarse malware o extraños.
- **Quantitative Security Risk Assessment of Android Permissions and Applications** (Yang Wang, Jun Zheng, Chen Sun y Srinivas Mukkamala) [2]. Se ha tomado como referencia metodológica para realizar una evaluación cuantitativa del riesgo en función de los permisos solicitados por las aplicaciones.
- **The Effects of Integrating Mobile Devices with Teaching and Learning on Students' Learning Performance: A Meta-analysis and Research Synthesis** (Yao-Ting Sung, Kuo-En Chang y Tzu-Chien Liu) [3]. Aunque no está centrado directamente en privacidad, aporta un contexto sobre el impacto del uso de aplicaciones móviles, lo que contribuye a justificar su análisis desde una perspectiva ética y técnica.
- **Mejorando App-PIMD, un repositorio para el estudio de la privacidad de aplicaciones móviles** (A. Pérez Fuente) [5]. Ha resultado especialmente relevante para conocer la arquitectura y el funcionamiento del repositorio App-PIMD, así como, para obtener ideas útiles para la estructuración del propio trabajo.
- **Documentación oficial de Android sobre `AndroidManifest.xml`** [6]. Este recurso ha permitido profundizar en la estructura del manifiesto de las aplicaciones Android y en los distintos tipos de permisos y categorías existentes, aspectos clave para el desarrollo del presente trabajo.

1.4. Objetivos

En este Trabajo de Fin de Máster se pretende contribuir al análisis de solicitudes de permisos por parte de diferentes aplicaciones para estimar el riesgo que suponen estas para la privacidad y seguridad de sus usuarios. Nuestra propuesta se basa en construir

indicadores de diferentes métricas y proporcionar una visualización y una herramienta de la que poder extraer información de consultas al repositorio App-PIMD.

Además de, investigar sobre el uso de inteligencia artificial para detectar patrones de permisos que puedan suponer un riesgo de privacidad. Su finalidad se basa en proporcionar la mayor cantidad de información posible al usuario y mejorar la gobernanza de la privacidad y la seguridad en aplicaciones móviles.

Objetivo General

Teniendo en cuenta el contexto y motivación de nuestro Trabajo de Fin de Máster, se plantea como **objetivo general** *seleccionar un conjunto de indicadores que permitan comparar y analizar los riesgos de privacidad en aplicaciones Android* facilitando la toma de decisiones por parte de los usuarios en función de sus necesidades.

A continuación, se muestran los objetivos específicos que conforman la línea de trabajo del proyecto de Fin de Máster.

Objetivos Específicos

Para la realización del Trabajo de Fin de Máster y poder cumplir con el objetivo general, se establecen los siguientes **objetivos específicos**:

1. Obtener una herramienta que facilita el análisis y comparación de aplicaciones Android en términos de privacidad, mediante consultas de indicadores del repositorio App-PIMD con su correspondiente visualización.
2. Seleccionar algoritmos de inteligencia artificial que agrupen los tipos de aplicaciones en función de patrones que detecten algoritmos de clustering.
3. Identificar permisos inusuales o atípicos, con el fin de detectar posibles aplicaciones que hagan un uso indebido de los permisos solicitados.

1.5. Estructura de la memoria

En este apartado se muestran los capítulos y el contenido que se desarrolla en cada uno de ellos para redactar la memoria de este Trabajo de Fin de Máster.

Capítulo 1: Introducción. Se presenta el contexto y motivación de este proyecto relacionado con la privacidad en dispositivos Android y los objetivos establecidos para su desarrollo.

Capítulo 2: Metodología y planificación seguida. Se describe la metodología para desarrollar el proyecto. Así como, la planificación, fases que se van a seguir en el

proyecto y posibles riesgos que puedan surgir.

Capítulo 3: Técnicas y herramientas. Se especifican las técnicas y herramientas que se emplean para el desarrollo del presente trabajo. Herramientas para la obtención de datos, su visualización y desarrollo de la API en el backend.

Capítulo 4: Análisis. Se desarrolla las etapas de análisis del proyecto que se va a seguir, la elicitación de requisitos y la definición de las consultas que se realizan sobre el repositorio App-PIMD. Y un análisis teórico de los modelos de IA sobre los que se realiza la exploración.

Capítulo 5: Diseño. Se presentan las decisiones relacionadas con los modelos de IA que se desarrollan, así como, el diseño y la arquitectura implementados para la creación de una herramienta. Además se detalla la visualización que facilita la interpretación de los resultados que se han obtenido.

Capítulo 6: Resultados obtenidos. Se presenta la herramienta que se ha obtenido a partir de las consultas implementadas y su visualización a través de diferentes dashboards. Se detalla por medio de un pequeño ejemplo cómo el usuario puede utilizar el sistema desarrollado, para comparar los permisos solicitados entre aplicaciones en las que este interesado.

Capítulo 7: Conclusiones y líneas futuras. Se muestran las conclusiones a las que se ha llegado tras la realización del proyecto, sus aspectos positivos y negativos, así como, las lineas futuras a desarrollar.

2: Metodología y planificación seguida

En este Trabajo de Fin de Máster se ha utilizado una *metodología de cascada incremental*. Sus características de desarrollo en fases, la hacen adecuada para este trabajo de investigación.

La *metodología en cascada incremental* se caracteriza por ser una metodología de desarrollo de software que combina por un lado la secuencia del modelo en cascada y por otro lado la flexibilidad del enfoque incremental [14]. Es decir, en lugar de desarrollar todo el sistema en un único ciclo lineal, esta metodología divide el proyecto en incrementos. Cada incremento pasa por todas las fases tradicionales del ciclo de metodología en cascada (análisis, diseño, implementación, pruebas y mantenimiento), esto permite ir construyendo de forma incremental el producto.

Otra de las ventajas es que permite ir haciendo entregas tempranas de partes del sistema gracias a su enfoque incremental. Así, se puede hacer una retroalimentación, reduciendo los errores. También, permite planificar y controlar el proyecto al no ser algo cerrado, pudiendo adaptarlos a cambios que sean necesarios si surgen durante el proyecto.

Figura 2.3: Metodología modelo cascada incremental [13].

En nuestro proyecto debido a sus características relativas a su duración (150 horas) se va a ver reducida esta metodología a sus tres primeras fases: análisis, diseño e implementación. Pero de igual manera, se va a realizar un trabajo de forma estructurada y planificada sabiendo cuál es el siguiente paso que hay que seguir, con iteraciones en cada etapa con el objetivo de detectar posibles fallos y mejorar progresivamente el análisis y resultados obtenidos. De esta manera, se ha buscado una mayor flexibilidad y reutilización de los resultados intermedios que sirven para ajustar la metodología sobre la marcha.

2.1. Metodología del proyecto

Una vez que se conocen los conceptos básicos sobre la metodología en cascada incremental y teniendo en cuenta la adapatación que se ha realizado. El Trabajo de Fin de Máster cuenta con las siguientes fases:

Análisis

En esta primera fase se lleva a cabo un análisis de los requisitos a tratar y cuyo objetivo es seleccionar los indicadores de privacidad a estudiar y ofrecer una herramienta que facilita la obtención de información sobre diferentes permisos. Para ello, se realizan las siguientes tareas:

- Estudio de artículos de privacidad mencionados en la sección anterior para comprender y conocer más acerca del tema de privacidad de permisos y sus características.
- Elicitación de los requisitos que se deben de cumplir en nuestro proyecto de forma que se puedan alcanzar los objetivos definidos en el apartado anterior.
- Análisis y diseño de consultas que se realizan al repositorio de App-PIMD, con los que obtener información útil para una comparativa de privacidad entre aplicaciones o categorías.
- Identificar a partir de la información proporcionada por las consultas cuál es relevante y mostrarlo en una visualización a través de diferentes dashboard.
- Establecer los conceptos teóricos sobre los que se lleva a cabo una iniciación a la exploración de modelos de inteligencia artificial para la detección de riesgos de privacidad en aplicaciones móviles.

Si durante la realización del proyecto, al ser un enfoque iterativo, se descubren nuevos indicadores, métricas o herramientas para nuestro estudio, se ajustará de nuevo esta fase.

Diseño

La segunda fase de la metodología consiste en establecer las tecnologías que se utilizan durante el proyecto y diseñar la arquitecturas que se lleva a cabo para construir una API que ayude a realizar consultas sobre App-PIMD. Además de, obtener métricas de privacidad con las que realizar comparaciones entre ellas, con el objetivo de realizar el estudio comparativo. Se detallará las decisiones tomadas en cada una de las etapas sobre las que se desarrolla los algoritmos y técnicas de clustering que se han pensado en investigar. Para lograrlo, se llevan a cabo las siguientes tareas.

- Justificar la toma de decisiones tomadas durante las diferentes etapas en las que se desarrolla los distintos modelos de inteligencia artificial y seleccionar los modelos más apropiados para alcanzar los objetivos definidos en nuestro Trabajo de Fin de Máster.
- Describir los datos de los que se parte en el repositorio App-PIMD, con los que se realiza el análisis comparativo que se plantea.
- Definir la arquitectura que sigue nuestro backend dónde se realizan tareas como: extracción de datos, procesamiento de los mismos y generar métricas para posteriormente desarrollar una visualización que permita llevar a cabo una comparación sobre riesgos de privacidad en distintas aplicaciones Android.
- Describir las diferentes métricas que se utilizan, que se pretende conocer con ellas y las visualizaciones que se emplean para mostrar la información de manera clara y comprensible.

De nuevo, se lleva a cabo un enfoque iterativo en esta fase, con el objetivo de que, en el caso de descubrir nuevas métricas, indicadores o herramientas de implementación, se definirá la nueva forma en la que se realiza el proceso.

Implementación

En esta tercera fase, se aplica y lleva a la práctica toda la metodología propuesta en las fases anteriores de análisis y diseño. Es decir, se lleva a cabo las consultas y arquitectura diseñadas para extraer información del repositorio App-PIMD. Con tareas cómo:

- Crear una API con diferentes endpoints con los que se tenga acceso a una variedad de consultas en formato JSON. A partir de cada una de las consultas, se extrae información de privacidad sobre la aplicación o categoría en la que el usuario se encuentre interesado.
- Evaluar los resultados obtenidos de la investigación sobre el uso de algoritmos de inteligencia artificial para el análisis de patrones y detección de posibles aplicaciones con riesgos para la privacidad.

- Desarrollar una visualización por medio de distintos dashboards con las que realizar un análisis comparativo entre aplicaciones Android o categorías con las que observar los riesgos de privacidad y seguridad que llevan asociados.

En esta fase, el enfoque iterativo se llevará a cabo a través de los resultados obtenidos. Se realizarán ajustes de las métricas y visualizaciones que se utilizan para el estudio en el caso de que estas no sean del todo claras y no se puedan sacar conclusiones del estudio comparativo.

2.2. Planificación

Una vez que se ha mostrado la metodología que se ha llevado a cabo en este proyecto de investigación, las distintas fases de las que se compone y las tareas que se realizan en cada una de ellas, se debe tener en cuenta su temporalización. Teniendo en cuenta que en la guía docente se indica que la duración del Trabajo de Fin de Máster es de 150 horas, se ha establecido como fecha de inicio del Trabajo de Fin de Máster el 26 de mayo de 2025 y se finaliza el 19 de Junio de 2025, trabajando una media de 8 horas diarias de lunes a viernes.

En la siguiente tabla, se muestra detalladamente las actividades que se llevan a cabo en este periodo de tiempo.

Semana	**Fechas**	**Actividades a realizar**
1	26/05 - 02/06	Recopilación de estudio y documentación de aspectos relacionados con la privacidad y los metadatos del repositorio App-PIMD e investigación sobre modelos de IA. Desarrollo de consultas sobre el repositorio. Redacción de memoria.
2	02/06 - 09/06	Diseño arquitectura del proyecto y bocetos de visualización. Comienzo de implementación de la API y desarrollo modelos de inteligencia artificial. Redacción de memoria.
3	09/06 - 16/06	Finalización implementación API, creación de visualización y evaluación de modelos de inteligencia artificial. Redacción de memoria.
4	16/06 - 19/06	Evaluación resultados obtenidos y revisión final de la memoria de TFM.

Tabla 2.1: Planificación Trabajo Fin Master

Riesgos

Se debe tener en cuenta que a la hora de planificar e iniciar cualquier proyecto es importante definir una serie de posibles **riesgos** que puedan surgir durante su desarrollo. Se pretende crear un plan en el que definir diferentes medidas con las que poder paliar los diferentes riesgo que se pueden encontrar durante su desarrollo. En las siguientes tablas, se destacan algunos posibles riesgos:

Riesgo 01	**Fallos en los componentes de los equipos de trabajo.**
Descripción	Los equipos de trabajo con los que se realiza este TFM, no están exentos de posibles problemas o fallos, por lo que se puede perder parte del trabajo realizado. Es importante estar protegido frente a este tipo de posibles riesgos.
Probabilidad	Bajo.
Impacto	Alto.
Riesgo	Medio
Plan de mitigación	Programar copias de seguridad periódicas en las que almacenar la información que se va generando. Guardar código en github para que pueda ser mas fácil acceder a ellos desde diferentes equipos.
Plan de contingencia	Con las copias de seguridad realizadas actualizar periodicamente los datos e información que se dispone y actualizar el código de los repositorios periódicamente.

Tabla 2.2: Riesgo 01

Riesgo 02	**Fallos temporalización de programación.**
Descripción	Mala planificación a la hora de estimar el tiempo que se va a tardar en realizar cada una de las tareas.
Probabilidad	Bajo.
Impacto	Alto.
Riesgo	Medio
Plan de mitigación	Análisis de actividades a desarrollar con la que obtener una estimación de tiempo precisa. Analizar las partes críticas del proyecto.
Plan de contingencia	Horas extras en las que completar las partes atrasadas del proyecto y reducir el alcance del proyecto para que sea posible su realización.

Tabla 2.3: Riesgo 02

Riesgo 03	**Situaciones personales.**
Descripción	Problemas personales del tipo: enfermedad, problema familiar, entre otros, que imposibilite poder avanzar en el desarrollo del proyecto.
Probabilidad	Bajo.
Impacto	Alto.
Riesgo	Medio.
Plan de mitigación	Atender y compatibilizar situaciones personales.
Plan de contingencia	Distribuir adecuadamente las actividades de cada fase y contar con flexibilidad a la hora de realizar la planificación.

Tabla 2.4: Riesgo 03

Riesgo 04	**Modificación de requisitos y funcionalidades requeridas.**
Descripción	Al iniciar el proyecto se definen una serie de requistos y funcionalidades que durante el proceso pueden verse afectados por el desarrollo del mismo. Estos cambios pueden variar la funcionalidad trabajo, o modificar el proyecto con respecto a la idea original
Probabilidad	Medio.
Impacto	Alto.
Riesgo	Alto.
Plan de mitigación	Al ser un proyecto con metodología en cascada incremental, aprovechar cada incremento para revisar las características y funcionalidades que se están desarrollando y comprobar si es necesario alguna modificación.
Plan de contingencia	Los objetivos del proyectos deben ser definidos de forma clara y asegurarse en cada incremento de que el proyecto va encaminado a lo que se desea conseguir.

Tabla 2.5: Riesgo 04

Durante el desarrollo del Trabajo de Fin de Máster se ha seguido la planificación establecida en el apartado 2.2 (Planificación). Como resultado, el trabajo se ha llevado a cabo sin incidencias, permitiendo el cumplimiento de los hitos y plazos previstos sin necesidad de activar los planes de contingencia definidos en los riesgos.

3: Técnicas y herramientas

Para la realización del Trabajo de Fin de Máster en el que se realiza una selección de indicadores de análisis de datos, se requiere contar con una serie de técnicas y herramientas para poder llevarlo a cabo. Para ello, en este capítulo se definen las técnicas y herramientas que se han tenido en cuenta y la forma en la que se han utilizado.

3.1. Técnicas

Como se comenta al inicio del capítulo el objetivo del Trabajo de Fin de Máster reside en obtener una serie de indicadores que faciliten el análisis de datos con los que poder determinar el nivel de privacidad e intromisión por parte de las aplicaciones. Para ello, se utiliza el repositorio App-PIMD, a partir de él, se dispone de toda la información con la que se elabora la investigación.

La técnica que ha seguido en este trabajo, sigue las reglas de una metodología ETL (Extracción, Transformación y Carga) [15] pero con una variación en la etapa de carga adaptada para este trabajo. Este proceso integra los datos que recopila desde múltiples fuentes que luego lo transforma a las necesidades del sistema y en lugar de cargar los datos en un sistema de almacenamiento (como sería la forma clásica de ETL), esta parte sufre una modificación en nuestro proyecto para dar respuesta en formato JSON a las consultas realizadas.

A continuación, se muestra la técnica *ETL adaptada* a nuestro proyecto.

ETL adaptada: Pipeline de datos en tiempo real

El proyecto se puede categorizar como una *ETL adaptada* ya que sí cumple con las características de las partes de *extracción de datos* y de *transformación*. La parte de la *carga* varía, es la parte que se adapta al proyecto, debido a que los datos no son almacenados en una base de datos o warehouse, sino que, se ofrecen como respuesta de un endpoint de una API propia que se crea para el desarrollo del proyecto y del que poder extraer datos

relevantes del repositorio App-PIMD. Por lo tanto, se puede decir que se crea un *Pipeline de datos en tiempo real* [44].

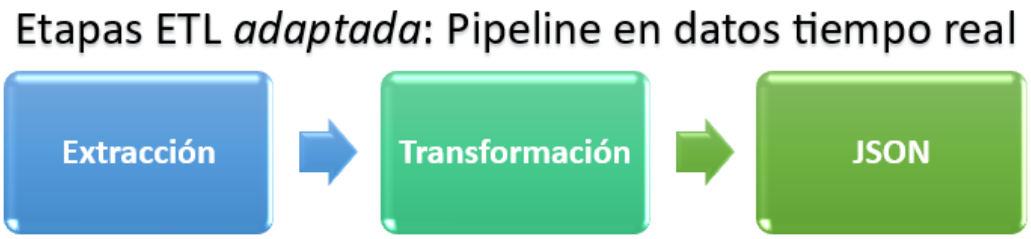

Figura 3.4: Fases ETL adaptada.

Extracción: Adquisición datos mediante APIs RESTful

La primera parte del pipeline consiste en llevar a cabo una extracción de datos del repositorio App-PIMD. Para la obtención de estos datos sobre aplicaciones móviles, se ha empleado una técnica de extracción automatizada basada en una API RESTful. Esta técnica forma parte del proceso de adquisición de información y permite acceder a información externa y de calidad. Para ello, se ha utilizado Python como lenguaje de programación, junto con la biblioteca *requests* para realizar peticiones HTTP a la API del repositorio.

Transformación: Adaptación de datos a consulta

Los datos recibidos en formato JSON son transformados para adaptarse y proporcionar la información solicitada según las diferentes consultas. También, se han utilizado librerías como *statics* o *difflib* que permiten transformar los datos para dar respuesta a las consultas y extraer información de interés del repositorio para llevar a cabo nuestro análisis de privacidad.

Con el uso de esta técnica se ha recolectado información relevante sobre la privacidad en las aplicaciones, cómo el número de permisos solicitados, puntuación de privacidad, entre otras, que serán explicadas más detalladamente en los siguientes capítulos.

JSON: Respuesta proporcionada por el endpoint

Una vez que los datos ya han sido transformados y se ha extraído la información que se quiere proporcionar según el endpoint se pasa a la parte adaptada de la ETL. Con los datos ya transformados, consiste en generar una respuesta JSON que pueda responder a la consulta planteada. El objetivo es devolver un JSON para asegurar la reproducibilidad y que esta información pueda ser utilizada posteriormente en diferentes trabajos. A pesar de que en este Trabajo de Fin de Máster se realiza una visualización de métricas en grafana, al conseguir este tipo de respuesta en formato JSON facilita su utilización para otros proyectos posteriores.

3.2. Herramientas utilizadas

Para materializar el proyecto que se ha desarrollado se requiere utilizar diferentes herramientas para cada una de las partes del que se compone.

Herramientas Generales del Proyecto

El proyecto se desarrolla por medio del lenguaje de **Python** y el entorno de **Visual Studio Code** [16], un editor desarrollado por Microsoft y que proporciona ventajas a la hora de desarrollar nuestro proyecto en Python al contener diferentes extensiones que facilitan su desarrollo.

Se ha utilizado **GitHub** [17] como plataforma de almacenamiento de código basada en **Git**. Esto permite llevar a cabo un control de versiones y contar con un historial que permite conocer de forma detallada los cambios que se han realizado al lo largo del desarrollo del proyecto.

Para desarrollar los diagramas que se van a mostrar durante la fase de diseño, se ha empleado **Draw.Io** [18], es una herramienta fácil con la que poder realizar diagramas de diseño software y poder exportarlo.

Herramientas de Backend e Inteligencia artificial

Para la parte de desarrollo de backend e investigación de inteligencia artificial se han utilizado diferentes herramientas que han permitido la construcción de una API con las consultas que se quieren realizar sobre el repositorio App-PIMD y la gestión de versiones de paquetes.

Se ha utilizada el framework de **fastAPI** [19], este framework permite construir de una forma sencilla toda la API con la que se quieren definir las consultas sobre el repositorio con las que extraer información de privacidad sobre la aplicación solicitada por el usuario.

Para poder gestionar los diferentes paquetes y sus versiones y evitar incompatibilidades entre ellos, se ha utilizado el gestor de paquetes de **Poetry** [20], de esta manera se ha podido definir un entorno virtual sobre el cuál se ha podido indicar los paquetes que se han querido instalar para el desarrollo de la parte de inteligencia artificial. También, este gestor presenta ventajas a la hora de mantener las versiones y que estén actualizadas.

Se utiliza **Docker** [21] con la finalidad de construir un contenedor con todos los servicios de la API que se han creado. Tiene el objetivo de ofrecer un servicio en nuestro equipo Ubuntu y que según la máquina se encienda, este se encuentre levantado. Destacamos esta funcionalidad ya que es de utilidad y simplifica el despliegue de la API.

Herramientas de Visualización

Para la visualización de los datos a través de diferentes dashboards, se utiliza **Grafana** [22]. Una plataforma de código abierto para la visualización de datos. Se realizan diferentes

gráficos sobre las consultas elaboradas, con el objetivo de mostrar la información más relevante para observar los riesgos de privacidad de una aplicación móvil. De esta manera, el usuario cuenta con una herramienta que le pueda ayudar en la toma de decisiones sobre su privacidad y seguridad ante aplicaciones móviles.

4: Análisis

Durante esta fase de Análisis, una vez consultada toda la documentación y literatura mencionada en la parte de Introducción de este Trabajo de Fin de Máster, se da la necesidad de llevar a cabo una selección de indicadores con la finalidad de crear una herramienta que ayude al usuario a poder conocer los riesgos de privacidad de aplicaciones móviles Android. Su objetivo es estar más informado y establecer comparaciones entre aplicaciones Android, sus permisos y categorías. Es por ello que, se han definido una serie de requisitos que van a guiar la investigación que se va a llevar a cabo.

Una vez definidos los requisitos de la investigación, el siguiente paso, es definir las consultas que se realizan al repositorio App-PIMD, con el objetivo de ofrecer información relevante al usuario. De esta manera, se tiene claro que se desea realizar y consultar en el repositorio, lo que facilita el trabajo en las siguientes fases.

A través de las consultas que se realizan al repositorio App-PIMD se obtiene información relacionada con la privacidad y los permisos de aplicaciones Android. A partir de estos datos, se ha evidenciado la gran cantidad de información y datos de privacidad almacenada en la base de datos del repositorio App-PIMD, lo que lleva a plantearse una serie de interrogantes:

- ¿Existen patrones ocultos en esos datos que puedan ofrecernos una comprensión más profunda sobre el comportamiento de las aplicaciones en términos de seguridad y privacidad?
- ¿Podría la inteligencia artificial ayudarnos a descubrir esos patrones, detectar anomalías y anticipar posibles riesgos de intrusividad?

Con estos interrogantes se pretende realizar una investigación teniendo en cuenta las capacidades de la inteligencia artificial para analizar un gran volumen de datos.

4.1. Requisitos

Definir unos requisitos constituyen la base de cualquier investigación y resulta fundamental para el desarrollo de la herramienta y la selección de métricas que se llevan acabo. A través de los requisitos se quiere definir qué es lo que se desea obtener de la investigación, y establecer una serie de condiciones mínimas. Se definen los siguientes requisitos: requisitos funcionales y requisitos no funcionales.

Requisitos funcionales

Los requisitos funcionales indican las características que constituyen la herramienta que se quiere realizar durante la investigación. Los requisitos definen acciones que indican que es lo que la herramienta debe realizar, además de, ofrecer una idea acerca de su funcionamiento y los resultados u objetivos que se desean obtener con ellas.

Estos requisitos muestran las capacidades y el comportamiento que la herramienta debe tener desde el punto de vista del usuario. Se debe tener en cuenta que, estos requisitos deben ser específicos, medibles, verificables y no ambiguos. Además de, no limitarse solo al comportamiento que debe tener la herramienta, sino que, sirvan como base para estructurar las actividades de la investigación. Estos requisitos describen tareas cómo: funciones que la herramienta debe realizar, comportamiento del sistema o restricciones del sistema, entre otros.

Código	**Requisito Funcional**	**Prioridad**
RF1	La herramienta deberá recopilar los datos procedentes del repositorio App-PIMD en función del nombre de la aplicación dada por el usuario.	Alta
RF2	La herramienta deberá recopilar los datos procedentes del repositorio App-PIMD en función del tipo de categoría de la aplicación.	Alta
RF3	La herramienta deberá incluir una serie de consultas sobre el repositorio App-PIMD que permitan obtener indicadores sobre la privacidad y seguridad de una aplicación.	Alta
RF4	La herramienta deberá ofrecer una respuesta a la consulta del usuario en un formato operable y reproducible.	Alta
RF5	La herramienta deberá proporcionar una visualización compuesta por diferentes tipos de gráficas que representen los datos obtenidos a través de las consultas realizadas al repositorio.	Alta

Código	Requisito Funcional	Prioridad
RF6	La herramienta deberá facilitar la interpretación de los riesgos de privacidad asociados a los permisos de la aplicación.	Media
RF7	La herramienta deberá utilizar los datos del repositorio App-PIMD para el entrenamiento de los modelos de inteligencia artificial.	Alta
RF8	La herramienta deberá ofrecer la posibilidad de dada una aplicación poder obtener los datos correspondientes del repositorio e informar por medio de un algoritmo de inteligencia artificial si la aplicación contiene riesgos de privacidad o seguridad.	Media
RF9	La herramienta deberá ofrecer la posibilidad de dada una aplicación poder predecir por medio de un algoritmo de inteligencia artificial su score PIM.	Media
RF10	La herramienta deberá ofrecer la posibilidad de reentrenar el modelo de detección de aplicaciones con riesgos de privacidad a partir de los datos recopilados en el repositorio App-PIMD.	Media
RF11	La herramienta deberá ofrecer la posibilidad de reentrenar el modelo de predicción de score a partir de los datos recopilados en el repositorio App-PIMD.	Media

Tabla 4.6: Requisitos funcionales.

Requisitos no funcionales

Los requisitos no funcionales especifican criterios de calidad, no están relacionados con las funcionalidades específicas del sistema, sino que, se utilizan para definir criterios y como debe ser el comportamiento de la herramienta. A través de ellos, se definen características relativas a especificaciones de calidad, restricciones, rendimiento, seguridad o usabilidad.

De esta forma, se busca garantizar el correcto funcionamiento de la herramienta, no solo cumplir con la funcionalidad requerida, sino también satisfacer estándares de calidad.

Código	Requisito No Funcional
RNF1	La herramienta deberá proporcionar una respuesta a las consultas realizadas sobre el repositorio App-PIMD en formato JSON.
RNF2	La herramienta deberá desarrollar una API por medio de la cual pueda dar respuesta a las consultas que se realizan al repositorio App-PIMD.

Código	Requisito No Funcional
RNF3	La herramienta deberá emplear algoritmos de inteligencia artificial con los que pueda detectar aplicaciones que supongan un riesgo de privacidad en comparación con el resto de aplicaciones de la misma categoría.
RNF4	La herramienta utilizará las bibliotecas de *sklearn* y *keras* para poder crear los diferentes modelos de inteligencia artificial adecuados.
RNF5	La herramienta deberá adaptarse a los tipos de permisos y categorías recogidos en *AndroidManifest.xml*.
RNF6	La herramienta deberá entrenar los modelos de inteligencia artificial a partir de los datos recogidos en el repositorio App-PIMD.

Tabla 4.7: Requisitos no funcionales.

4.2. Elaboración de consultas

Se llega a uno de los puntos más importantes del Trabajo de Fin de Máster, a continuación, se elaboran las consultas que se realizan sobre el repositorio App-PIMD. A partir de estas consultas se realiza un análisis sobre los permisos que solicita una aplicación móvil para determinar su nivel de riesgo de privacidad y su intromisión. También, se pretende realizar este análisis sobre categorías de aplicaciones, con el objetivo de conocer dentro de cada categoría cuales son las mejores y peores aplicaciones respecto a su gestión de privacidad.

Antes de nada, resulta fundamental conocer la arquitectura y los datos con los que se trabaja y que proporciona el repositorio App-PIMD.

Arquitectura y datos App-PIMD

Una vez conocida la arquitectura de App-PIMD [5], se pasa a analizar los datos que este repositorio proporciona. Para ello, a continuación, se muestra un ejemplo donde se hace una solicitud al repositorio de los datos de WhatsApp, una de las aplicaciones más conocidas y utilizadas en los dispositivos móviles. En las siguientes imágenes se puede observar parte de la respuesta que proporciona App-PIMD a la consulta sobre *WhatsApp*.

Response body


```
{
  "App": {
    "hash": "e417fbf251b639e612d1cbc2b72a3c3351fece95edc2d0af000fbafc5ec08cf1",
    "app_name": "WhatsApp",
    "icon": "https://apkfalcon.infor.uva.es:8080/app_icon/hash/e417fbf251b639e612d1cbc2b72a3c3351fece95edc2d0af000fbafc5ec08cf1",
    "package": "com.whatsapp",
    "version_code": 251476003,
    "version_name": "2.25.14.76",
    "min_sdk_version": 21,
    "target_sdk_version": 35,
    "max_sdk_version": null,
    "category": "COMMUNICATION",
    "uses_permission_list": [
      {
        "Permission": {
          "name": "android.permission.ACCESS_COARSE_LOCATION",
          "protection_level": "dangerous|instant",
          "declared_group_list": [
            {
              "PermissionGroup": {
                "name": "android.permission-group.LOCATION"
              }
            }
          ],
          "rank_list": [
            {
              "Rank": {
                "value": 0.0243902,
```


Figura 4.5: Ejemplo respuesta WhatsApp 1.


```
    ],
    "defines_group_list": null,
    "extraction_metadata_list": [
      {
        "ExtractionMetadata": {
          "source": "APKPure",
          "method": "web scraping",
          "timestamp": "2025-05-08T07:53:45"
        }
      }
    ],
    "score_list": [
      {
        "Score": {
          "value": 6.60878,
          "rank_name": "PIM (Privacy Impact Metric)",
          "app_hash": "e417fbf251b639e612d1cbc2b72a3c3351fece95edc2d0af000fbafc5ec08cf1"
        }
      },
      {
        "Score": {
          "value": null,
          "rank_name": "Tosdr",
          "app_hash": "e417fbf251b639e612d1cbc2b72a3c3351fece95edc2d0af000fbafc5ec08cf1"
```


Figura 4.6: Ejemplo respuesta WhatsApp 2.

A continuación, en la siguiente tabla 4.8 se muestran los diferentes campos del repositorio App-PIMD con información relativa a permisos, puntuaciones score, metadatos entre otros.

Campo	Información
hash	Identificador hash de la aplicación en el repositorio App-PIMD.
app_name	Nombre de la aplicación de la que se quiere obtener los datos.
package	Referencia al paquete de la aplicación.
version_code	Muestra la versión de código de la aplicación.
version_name	Informa del nombre de la versión de la aplicación.
min_sdk_version	Versión mínima de SDK en la que se puede ejecutar la aplicación.
target_sdk_version	Versión más común de SDK en la que se puede ejecutar la aplicación.
max_sdk_version	Versión máxima de SDK en la que se puede ejecutar la aplicación.
category	Categoría en la que se agrupa la aplicación.
uses_permission_list	Recoge la lista de permisos solicitados: ▪ name: Nombre del permiso. ▪ protection_level: Tipo de permiso clasificado por nivel. ▪ declared_group_list: Agrupación por nombre. ▪ rank_list: Score de privacidad de métrica PIM.
defines_permissons	Diccionario de permisos personalizados de la app: ▪ name: Nombre del permiso. ▪ protection_level: Tipo de permiso clasificado por nivel. ▪ declared_group_list: Agrupación por nombre. ▪ rank_list: Score de privacidad de métrica PIM.
extraction_metadata	Diccionario donde se informa de la fuente y técnica de extracción: ▪ source: Fuente de origen. ▪ method: Técnica utilizada. ▪ timestamp: Fecha de extracción.
score_list	Diccionario con la puntuación de privacidad de la aplicación: ▪ value: Score numérico. ▪ rank_name: Métrica utilizada. ▪ app_hash: Identificador hash de la aplicación.

Tabla 4.8: Información proporcionada por App-PIMD.

De la tabla anterior, se observa toda la información que proporciona el repositorio App-PIMD sobre cualquier aplicación. A partir de esta información se ha analizado y buscado las consultas que pueden extraer y proporcionar información relevante sobre la privacidad de aplicaciones Android. Esta información permite comparar aplicaciones y categorías entre sí, para poder evaluar los riesgos de privacidad entre ellas.

Se han desarrollado *47 consultas* al repositorio de App-PIMD para realizar un análisis de privacidad y seguridad. Estas consultas se pueden dividir en tres grupos: *Métricas de privacidad por aplicación*, *Métricas de privacidad por categoría* y *Evolución permisos de una aplicación con el paso de versiones.*

Métricas de privacidad por aplicación

Estos endpoints por medio de los cuáles se realizan consultas al repositorio de App-PIMD, se caracterizan porque proporcionan información sobre la privacidad de la aplicación en la que se encuentra interesado el usuario. La información que proporcionan estos endpoints está relacionada con los metadatos que ofrece el repositorio, además de incluir información relativa a los permisos y al nivel de intrusividad de la aplicación, que se representa mediante su score.

Por tanto, el objetivo final de estos endpoints consiste en conocer la información relativa a la privacidad de una aplicación y con ella, poder compararla con otras aplicaciones de manera que, se determine cuál es la menos invasiva de las dos o poder extraer conclusiones según las necesidades del usuario.

Los endpoints que corresponden a este grupo de consultas son:

- *get_package/{app_name}*: Con este endpoint se obtiene el identificador de paquete de la aplicación.
- *get_hash/{app_name}*: Endpoint que muestra el identificador hash con el que se identifica la aplicación en el repositorio.
- *get_version_code/{app_name}*: Este endpoint informa sobre el código de versión de la aplicación.
- *get_target_sdk/{app_name}*: Endpoint que muestra la versión de sdk que se espera que utilice la aplicación.
- *get_permissions/{app_name}*: Este endpoint obtiene todos los permisos que solicita la aplicación con su información como nombre del permiso, tipo y score.
- *get_category/{app_name}*: Endpoint que informa del tipo de categoría de la aplicación.
- *get_extraction_metadata/{app_name}*: Este endpoint muestra la fuente y forma de la que se ha extraido la información de la aplicación.
- *get_defines_permissions/{app_name}*: Endpoint en el que se muestran los permisos personalizados de la aplicación.
- *get_min_sdk/{app_name}*: Endpoint que informa de la versión mínima de sdk necesaria para ejecutar la aplicación.

- *get_name/{app_name}*: Endpoint que muestra el nombre de la aplicación que el usuario ha solicitado.
- *get_scores/{app_name}*: Endpoint que muestra la puntuación de privacidad que tiene la aplicación, utilizando la métrica PIM.
- *count_permissions/{app_name}*: Este endpoint informa sobre el número de permisos que solicita la aplicación.
- *count_permissions_type/{app_name}*: Este endpoint muestra el número de permisos clasificados según su tipo.
- *mean_rank_permissions/{app_name}*: Endpoint que proporciona la puntuación PIM media de todos los permisos de un determinado tipo (dangerous, normal o signature)
- *count_permissions_category/{app_name}*: A través de este endpoint se conoce el número de permisos solicitados por la aplicación según la categoría de permisos.

Métricas de privacidad por categoría

Estas consultas se caracterizan por extraer información relativa al promedio sobre todas las aplicaciones que conforman una categoría, conocer el número de permisos promedio solicitados, el número medio de permisos de cada tipo, también, informa sobre las peores y mejores aplicaciones de una determinada categoría. Para ello, se realizan comparaciones como las cinco aplicaciones de una categoría que solicitan más o menos permisos.

Con ello, se persiguen dos objetivos. Por un lado buscar las mejores y peores aplicaciones de una determinada categoría comparándolas también con sus valores promedios. En segundo lugar, comparar entre categorías, para conocer cuál de ellas suponen un mayor o menor riesgos de privacidad.

- *category_score_permissions/{category_name}*: Con este endpoint se obtiene la puntuación PIM de todas las aplicaciones que conforman la categoría.
- *category_count_permissions/{category_name}*: A través de este endpoint se muestra la información relativa al número de permisos solicitados por todas las aplicaciones de la categoría
- *category_count_normal_permissions_type/{category_name}*: Este endpoint informa sobre el número de permisos de tipo normal solicitados por cada aplicación de la categoría.
- *category_count_signature_permissions_type/{category_name}*: Por medio del endpoint se informa sobre el número de permisos de tipo signature solicitados por todas las aplicaciones que conforman la categoría.

- *category_count_dangerous_permissions_type/{category_name}*: Muestra el número de permisos solicitados de tipo dangerous por todas las aplicaciones de la categoría.
- *category_count_permissions_type/{category_name}*: Este endpoint muestra el número medio de permisos solicitados por cada tipo de categoría.
- *category_avg_dangerous_permissions_type/{category_name}*: Endpoint que muestra el número medio de permisos de tipo dangerous solicitados por las aplicaciones de la categoría.
- *category_avg_normal_permissions_type/{category_name}*: Este endpoint muesta el número medio de permisos del tipo normal que solicitan las aplicaciones de una categoría.
- *category_avg_signature_permissions_type/{category_name}*: Endpoint que informa del número medio de permisos del tipo signature solicitados por las aplicaciones de la categoría solicitada.
- *category_mean_permissions/{category_name}*: Endpoint que informa del número de permisos medio que solicita una aplicación de la categoría indicada.
- *category_mean_score/{category_name}*: Este endpoint muestra el score medio de las aplicaciones de esa categoría y el número de aplicaciones que la conforman.
- *get_category_name/{category_name}*: Con este endpoint se obtiene el nombre de la categoría de la que se esta extrayendo información.
- *category_count_best_permission/{category_name}/{val}*: Este endpoint informa sobre las aplicaciones de la categoría que menos permisos solicitan.
- *category_count_worst_permission/{category_name}/{val}*: Por medio de este endpoint se puede conocer las aplicaciones de la categoría que más permisos solicitan.
- *category_score_permission_best/{category_name}/{val}*: Este endpoint se utiliza para conocer las aplicaciones de la categoría con mejor puntuación.
- *category_score_permission_worst/{category_name}/{val}*: A través del endpoint se obtiene las apliacaciones de la categoría con peor puntuación PIM.
- *category_count_normal_permission_best/{category_name}/{val}*: Con este endpoint se obtiene la información de las aplicaciones que menos permisos del tipo normal solicitan en la categoría.
- *category_count_normal_permission_worst/{category_name}/{val}*: Con este endpoint se obtiene la información de las aplicaciones que más permisos del tipo normal solicitan en la categoría.

- *category_count_dangerous_permission_best/{category_name}/{val}*: Con este endpoint se obtiene la información de las aplicaciones que menos permisos del tipo dangerous solicitan en la categoría.
- *category_count_dangerous_permission_worst/{category_name}/{val}*: Con este endpoint se obtiene la información de las aplicaciones que más permisos del tipo dangerous solicitan en la categoría.
- *category_count_signature_permission_best/{category_name}/{val}*: Este endpoint muestra las aplicaciones que menos permisos del tipo signature solicitan en la categoría.
- *category_count_signature_permission_worst/{category_name}/{val}*: Este endpoint muestra las aplicaciones que más permisos del tipo signature solicitan en la categoría.

Evolución permisos de una aplicación con el paso de versiones

A través de este grupo de consultas se analiza la evolución de las aplicaciones según sus versiones más recientes. Para ello, se ha desarrollado una serie de consultas que permiten al usuario ver esta evolución. Estas consultas incluyen, por ejemplo, el número de permisos solicitados en cada versión de la aplicación, así como el número de permisos solicitados en función de su tipo (*dangerous*, *signature*, *normal*). Además se ha incluido consultas con los que poder conocer el número medio de permisos solicitados por versión.

Es importante tener en cuenta algunas cuestiones a la hora de detallar las consultas realizadas. En este Trabajo de Fin de Máster se ha pensado solo en una serie de aplicaciones y versiones debido a que para cada aplicación que se quería probar, se debe previamente subir el apk para que el repositorio cuente con esa versión. Para ello, hay que subir mediante el método post */post/app/file"* el apk de la versión de la aplicación en la que se está interesado previamente, por lo que en líneas futuras sería interesante poder incluir esta información en el repositorio con un volumen de datos más grande.

Las aplicaciones y versiones que se utilizan en este grupo de consultas son:

Aplicación	**Versiones**
BBVA	2021, 2022, 2023, 2024, 2025
WhatsApp	2021, 2022, 2023
Telegram	2021, 2022, 2023, 2024
Netflix	2021, 2025
Glovo	2021, 2022, 2023, 2024

Tabla 4.9: Aplicaciones y sus versiones.

Las consultas relativas a esta parte son las siguientes:

- *avg_evolution_permission_app/{app_name}*: Media de permisos solicitados por la aplicación en las últimas versiones.
- *avg_evolution_dangerous_permission_app/{app_name}*: Endpoint en el que se informa del número medio de permisos dangerous solicitados en las últimas versiones.
- *avg_evolution_normal_permission_app/{app_name}*: Número medio de permisos del tipo normal solicitados en las últimas versiones de la aplicación.
- *avg_evolution_signature_permission_app/{app_name}*: Este endpoint informa sobre el número medio de permisos signature durante las últimas versiones de la aplicación.
- *avg_evolution_score_app/{app_name}*: Muestra la media de la puntuación PIM durante las últimas versiones de la aplicación.
- *evolution_count_permission_app/{app_name}*: Endpoint que muestra la evolución del número de permisos solicitados según la aplicación elegida.
- *evolution_count_dangerous_permission_app/{app_name}*: Consulta con la que se informa sobre la evolución de permisos del tipo dangerous solicitados por la aplicación.
- *evolution_count_normal_permission_app/{app_name}*: Se muestra información relativa a la evolución del número de permisos solicitados por la aplicación.
- *evolution_count_signature_permission_app/{app_name}*: Endpoint que informa sobre la evolución del número de permios signature.
- *evolution_score_app/{app_name}*: Muestra la evolución del score que recibe la aplicación según el paso de versiones.

Endpoints personalizados para este TFM

Una vez identificados y definidos todos los endpoints necesarios para realizar consultas sobre el repositorio App-PIMD, con el objetivo de construir una visualización que permita un análisis comparativo de los riesgos de privacidad entre distintas aplicaciones Android, es importante destacar cómo se han construido dichos endpoints.

Algunos de los endpoints proporcionan información que se extrae directamente de estos datos como la categoría de la aplicación o los metadatos de la misma, pero hay un gran número de endpoints que para extraer esa información y crear esa métrica ha sido necesaria llevar a cabo una serie de operaciones a partir de la información que proporciona el repositorio. A continuación, se detallan los endpoints que se han creado con información personalizada en función de los datos proporcionados por el repositorio App-PIMD.

count_permissions

Este endpoint proporciona el número de permisos que solicita una aplicación para poder ser instalada. Se obtiene a partir de la cuenta de todos los permisos que se solicita. La respuesta proporcionada por este endpoint se refleja en la tabla 4.10.

Nombre aplicación	**Número de permisos**
WhatsApp	81

Tabla 4.10: Respuesta endpoint: count_permissions.

count_permissions_type

A partir de este endpoint, se cuenta el número de permisos que solicita en función de su tipo. Estos pueden ser:

- *signature*: Permisos que solo pueden ser otorgados si la aplicación esta firmada con la misma clave que la aplicación que define el permiso.
- *normal*: Permisos que son de bajo riesgo y que Android concede automáticamente.
- *dangerous*: Permisos que son peligrosos porque pueden acceder a datos sensibles.

Para obtener esta información se agrupan todos los permisos por su tipo y se realiza un conteo de cuantos hay en cada grupo. La respuesta proporcionada por este endpoint se refleja en la tabla 4.11.

Nombre aplicación	**Nº permisos *normal***	**Nº permisos *signature***	**Nº permisos *dangerous***
WhatsApp	33	5	22

Tabla 4.11: Respuesta endpoint: count_permissions_type.

mean_rank_permissions

Este endpoint se utiliza para conocer el valor promedio del score en función de cada tipo de permiso que solicita la aplicación. Para ello, se agrupan todos los permisos por su tipo y se calcula la media de cada conjunto. La respuesta proporcionada por este endpoint se refleja en la tabla 4.12.

Aplicación	**Score *dangerous instant***	**Score *dangerous***	**Score *normal instant***	**Score *normal***	**Score *signature***
WhatsApp	0.0242	0.1203	null	null	null

Tabla 4.12: Respuesta endpoint: mean_rank_permissions.

count_permissions_category

Este endpoint cuenta el número de permisos que solicita la aplicación por su categoría por ejemplo, Location, Nearby Devices, Storage, SMS, Conctacts, Phone, Microphone, System, entre otros. Esta información se obtiene a partir de todos los permisos que solicita la aplicación agrupándolos por su categoría y realizando un conteo sobre cada una de ellas. La respuesta proporcionada por este endpoint se refleja en la tabla 4.13.

Aplicación	**Categoría permisos**	**Nº permisos solicitados**
WhatsaApp	Accounts	4
WhatsaApp	Camera	2
WhatsaApp	Contacts	2
WhatsaApp	Foreground	2
WhatsaApp	Location	4
WhatsaApp	Microphone	2
WhatsaApp	Nearby Devices	12
WhatsaApp	Notifications	1
WhatsaApp	Personalizados	8
WhatsaApp	Phone	5
WhatsaApp	Read Media (Aural)	1
WhatsaApp	Read Media (Visual)	3
WhatsaApp	SMS	9
WhatsaApp	Sensors	2
WhatsaApp	Storage	2
WhatsaApp	System	17

Tabla 4.13: Respuesta endpoint: Permisos por categoría.

category_score_permissions

A través de este endpoint se podrá conocer las puntuaciones score de todas las aplicaciones de una determinada categoría, por ejemplo si se busca la categoría de COMMUNICATION, la respuesta proporcionada por este endpoint se refleja en la tabla 4.14.

Categoría	Aplicación	Score
COMMUNICATION	Stay Updated	0.1556
COMMUNICATION	WF Connection	0.2334
COMMUNICATION	Jitsi Meet	0.3112
COMMUNICATION	Opera browser	0.3890
COMMUNICATION	Azar	0.3890
COMMUNICATION	RPM ACADEMY	0.2334

Tabla 4.14: Respuesta endpoint: Score por categoría.

Esta información se obtiene a través de la búsqueda de todas las aplicaciones de una categoría y almacenado su score en una lista.

category_count_permissions

Con este endpoint se conoce el número de permisos solicitados de todas las aplicaciones de una determinada categoría, por ejemplo, si se busca la categoría de COMMUNICATION, la respuesta proporcionada por este endpoint se refleja en la tabla 4.15.

Categoría	Aplicación	Nº permisos
COMMUNICATION	Voissy	5
COMMUNICATION	Mi Lowi	7
COMMUNICATION	RPM ACADEMY	8
COMMUNICATION	Mi Pepephone	12
COMMUNICATION	MasMovil	13
COMMUNICATION	AlertCops	15

Tabla 4.15: Respuesta endpoint: Permisos por categoría.

El número de permisos de cada aplicación de la categoría, se obtiene a través de la búsqueda de todas las aplicaciones de una categoría y almacenando el conteo de los permisos que solicita cada categoría en una lista.

category_count_normal_permissions

Por medio de este endpoint se conoce el número de permisos del tipo normal solicitados por cada una de las aplicaciones de una determinada categoría, por ejemplo, si se busca la categoría de COMMUNICATION, la respuesta proporcionada por este endpoint se refleja en la tabla 4.16.

Categoría	**Aplicación**	**Nº permisos *normal***
COMMUNICATION	feedhom	1
COMMUNICATION	idSide	4
COMMUNICATION	Voissy	5
COMMUNICATION	JouwLoon	6
COMMUNICATION	Mi Lowi	7
COMMUNICATION	Tor Browser	8

Tabla 4.16: Respuesta endpoint: Permisos *normal*.

La información se obtiene a través de la búsqueda de todas las aplicaciones de una categoría y filtrando por los permisos de tipo normal, tras ello, se hace un conteo que almacena su valor en una lista.

category_count_signature_permissions

Por medio de este endpoint se conoce el número de permisos del tipo *signature* solicitados por cada una de las aplicaciones de una determinada categoría, por ejemplo si se busca la categoría de COMMUNICATION, la respuesta proporcionada por este endpoint se refleja en la tabla 4.17.

Categoría	**Aplicación**	**Nº permisos *signature***
COMMUNICATION	Azar	0
COMMUNICATION	RPM ACADEMY	0
COMMUNICATION	Google Meet	1
COMMUNICATION	Mi Movistar	1
COMMUNICATION	Mi Pepephone	1
COMMUNICATION	Mi Vodafone	2

Tabla 4.17: Respuesta endpoint: Permisos *signature*.

La información se obtiene a través de la búsqueda de todas las aplicaciones de una categoría y filtrando por los permisos de tipo signature, tras ello, se hace un conteo que almacena su valor en una lista.

category_count_dangerous_permissions

Por medio de este endpoint se conoce el número de permisos del tipo *dangerous* solicitados por cada una de las aplicaciones de una determinada categoría, por ejemplo si se busca la categoría de COMMUNICATION, la respuesta proporcionada por este endpoint se refleja en la tabla 4.18.

Categoría	**Aplicación**	**Nº permisos *dangerous***
COMMUNICATION	WaveCast	0
COMMUNICATION	PPP Widget	1
COMMUNICATION	Mi Lowi	2
COMMUNICATION	RPM ACADEMY	3
COMMUNICATION	Ecumeni	4
COMMUNICATION	ASTRNT	5

Tabla 4.18: Respuesta endpoint: Permisos *dangerous*.

La información se obtiene a través de la búsqueda de todas las aplicaciones de una categoría y filtrando por los permisos de tipo dangerous, tras ello se hace un conteo que almacena su valor en una lista.

category_count_permissions_type

Este endpoint proporciona información sobre el número medio de categoría de permisos solicitados por las aplicaciones de una determinada categoría, clasificados según la categoría de permiso.

Para obtener esta información para cada una de las aplicaciones que conforman la categoría solicitada, se agrupan los permisos por su categoría y se realiza un conteo calculando el promedio de cada una de ellas, el valor obtenido se almacena en una lista. La respuesta proporcionada por este endpoint se refleja en la tabla 4.19.

Categoría	**Permiso**	**Nº permisos**
COMMUNICATION	Personalizados	2.14
COMMUNICATION	Location	1.1
COMMUNICATION	Nearby Devices	2.78
COMMUNICATION	Accounts	1.06
COMMUNICATION	System	5.61
COMMUNICATION	Storage	1.54
COMMUNICATION	Phone	1.12

Categoría	Permiso	Nº permisos
COMMUNICATION	SMS	0.63
COMMUNICATION	Camera	0.58
COMMUNICATION	Contacts	0.6
COMMUNICATION	Microphone	0.51
COMMUNICATION	Sensors	0.2
COMMUNICATION	Bindings	0.47
COMMUNICATION	Notification	0.18
COMMUNICATION	Calendar	0.21
COMMUNICATION	Read Media (Aural)	0.04
COMMUNICATION	Read Media (Visual)	0.1
COMMUNICATION	Call log	0.13
COMMUNICATION	Foreground	0.04
COMMUNICATION	Activity Recognition	0.03

Tabla 4.19: Respuesta endpoint: Permisos según categoría.

category_avg_dangerous_permissions_type

Por medio de este endpoint se obtiene el número medio de permisos del tipo *dangerous* que son solicitados por las aplicaciones en función de su categoría. Para ello, se filtran los permisos de cada aplicación quedándonos solo con los de tipo *dangerous* y se calcula la media del conjunto. La respuesta proporcionada por este endpoint se refleja en la tabla 4.20.

Categoría	Media permisos *dangerous*
COMMUNICATION	6.3

Tabla 4.20: Respuesta endpoint: Avg permisos *dangerous*.

category_avg_normal_permissions_type

Por medio de este endpoint se obtiene el número medio de permisos del tipo normal que son solicitados por las aplicaciones en función de su categoría. Para ello, se filtran los permisos de cada aplicación quedándonos solo con los de tipo normal y se calcula la media del conjunto. La respuesta proporcionada por este endpoint se refleja en la tabla 4.21.

Categoría	**Media permisos *normal***
COMMUNICATION	6.3

Tabla 4.21: Respuesta endpoint: Avg permisos *normal*.

category_avg_signature_permissions_type

Por medio de este endpoint se obtiene el número medio de permisos del tipo signature que son solicitados por las aplicaciones en función de su categoría. Para ello, se filtran los permisos de cada aplicación quedándonos solo con los de tipo signature y se calcula la media del conjunto. La respuesta proporcionada por este endpoint se refleja en la tabla 4.22.

Categoría	**Media permisos *signature***
COMMUNICATION	6.3

Tabla 4.22: Respuesta endpoint: Avg permisos *signature*.

category_mean_permissions

Este endpoint proporciona el número medio de permisos solicitados por una aplicación de una determinada categoría. Se obtiene a través del conteo del número de permisos solicitados por todas las aplicaciones de la categoría y calculando su promedio. La respuesta proporcionada por este endpoint se refleja en la tabla 4.23.

Categoría	**Media permisos**
COMMUNICATION	19.1

Tabla 4.23: Respuesta endpoint: Avg permisos.

category_mean_score

A través de este endpoint se proporciona información sobre la puntuación PIM media de todas las aplicaciones que conforman una determinada categoría. La respuesta proporcionada por este endpoint se refleja en la tabla 4.24.

Categoría	**Media score**
COMMUNICATION	1.77

Tabla 4.24: Respuesta endpoint: Avg score.

Se obtiene recogiendo todos los valores score de las aplicaciones y calculando su promedio, se representa por medio de un velocímetro que muestra su valor.

category_count_best_permission

Con este endpoint, se conocen las aplicaciones que menos permisos solicitan. Se obtienen a través del conteo de los permisos que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con menor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.25.

Categoría	Aplicación	Permisos solicitados
COMMUNICATION	Uighur Basic Phrases	5
COMMUNICATION	Voissy	5
COMMUNICATION	estherrani	2
COMMUNICATION	feedhome.feed	1
COMMUNICATION	idSide - Echo	4

Tabla 4.25: Respuesta endpoint: Top 5 permisos.

category_count_worst_permission

A través de este endpoint se conocen las aplicaciones que más permisos solicitan. Se obtiene a través del conteo de los permisos que solicita cada aplicación de una categoría y almacenando en una lista las cinco aplicaciones que mayor número de permisos solicitan. La respuesta proporcionada por este endpoint se refleja en la tabla 4.26.

Categoría	Aplicación	Permisos solicitados
COMMUNICATION	Rakuten	47
COMMUNICATION	Telegram	71
COMMUNICATION	Truecaller	56
COMMUNICATION	WhatsApp	81
COMMUNICATION	WhatsApp Business	48

Tabla 4.26: Respuesta endpoint: Worst 5 permisos.

category_score_permission_best

A partir de este endpoint, se conocen las aplicaciones que mejor puntuación PIM tienen. Se obtienen a través de conocer el score de cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con mejor puntuación. La respuesta proporcionada por este endpoint se refleja en la tabla 4.27.

Categoría	Aplicación	Permisos solicitados
COMMUNICATION	WaveCast	0
COMMUNICATION	Uighur	0
COMMUNICATION	Estherrani	0
COMMUNICATION	Bukharimc	0
COMMUNICATION	Halley VPN	0

Tabla 4.27: Respuesta endpoint: Top 5 score.

category_score_permission_worst

A partir de este endpoint, se conocen las aplicaciones que peor puntuación PIM tienen. Se obtienen a través de conocer el score de cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con peor puntuación. La respuesta proporcionada por este endpoint se refleja en la tabla 4.28.

Categoría	Aplicación	Score
COMMUNICATION	Contacts+	7.8992
COMMUNICATION	Messenger	7.9770
COMMUNICATION	Signal	8.1326
COMMUNICATION	CallApp	8.2104
COMMUNICATION	TrueCaller	8.5995

Tabla 4.28: Respuesta endpoint: Worst 5 score.

category_count_normal_permissions_best

Con este endpoint, se conocen las aplicaciones que menos permisos solicitan del tipo normal. Se obtienen a través del conteo de los permisos de tipo normal que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con menor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.29.

Categoría	Aplicación	Permisos ***normal***
COMMUNICATION	feedhom	1
COMMUNICATION	Wasa	2

Categoría	**Aplicación**	**Permisos *normal***
COMMUNICATION	Estherrani	2
COMMUNICATION	IdSide - Echo	3
COMMUNICATION	YolkaApp	3

Tabla 4.29: Respuesta endpoint: Top 5 normal.

category_count_normal_permissions_worst

Con este endpoint, se conocen las aplicaciones que más permisos solicitan del tipo normal. Se obtienen a través del conteo de los permisos de tipo normal que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con mayor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.30.

Categoría	**Aplicación**	**Permisos *normal***
COMMUNICATION	Signal	26
COMMUNICATION	Mini-Download	27
COMMUNICATION	Rakuten	27
COMMUNICATION	WhatsApp Business	28
COMMUNICATION	WhatsApp	33

Tabla 4.30: Respuesta endpoint: Worst 5 normal.

category_count_dangerous_permissions_best

Con este endpoint, se conocen las aplicaciones que menos permisos solicitan del tipo *dangerous*. Se obtienen a través del conteo de los permisos de tipo *dangerous* que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con menor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.31.

Categoría	**Aplicación**	**Permisos *dangerous***
COMMUNICATION	WaveCAST	0
COMMUNICATION	Voissy	0
COMMUNICATION	Deleted Messages Recovery	0
COMMUNICATION	Uighur	0
COMMUNICATION	Feedhom	0

Tabla 4.31: Respuesta endpoint: Top 5 *dangerours*.

category_count_dangerous_permissions_worst

Con este endpoint, se conocen las aplicaciones que más permisos solicitan del tipo *dangerous*. Se obtienen a través del conteo de los permisos de tipo *dangerous* que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con mayor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.32.

Categoría	**Aplicación**	**Permisos *dangerous***
COMMUNICATION	Signal Private Messenger	17
COMMUNICATION	CallAPP	18
COMMUNICATION	Telegram	20
COMMUNICATION	WhatsApp	22
COMMUNICATION	TrueCaller	23

Tabla 4.32: Respuesta endpoint: Worst 5 *dangerous*.

category_count_signature_permissions_best

Con este endpoint, se conocen las aplicaciones que menos permisos solicitan del tipo *signature*. Se obtienen a través del conteo de los permisos de tipo *signature* que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con menor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.33.

Categoría	**Aplicación**	**Permisos *signature***
COMMUNICATION	WaveCAST	0
COMMUNICATION	Voissy	0
COMMUNICATION	Azar	0
COMMUNICATION	Tor Browser	0
COMMUNICATION	Talkchat	0

Tabla 4.33: Respuesta endpoint: Top 5 *signature*.

category_count_signature_permissions_worst

Con este endpoint, se conocen las aplicaciones que más permisos solicitan del tipo *signature*. Se obtienen a través del conteo de los permisos de tipo *signature* que solicita cada aplicación de una determinada categoría y almacenando en una lista las cinco aplicaciones con mayor número de permisos solicitados. La respuesta proporcionada por este endpoint se refleja en la tabla 4.34.

Categoría	**Aplicación**	**Permisos *signature***
COMMUNICATION	CallAPP	6
COMMUNICATION	Emojikey	6
COMMUNICATION	WeChat	7
COMMUNICATION	Messenger	7
COMMUNICATION	Gmail	7

Tabla 4.34: Respuesta endpoint: Worst 5 *signature*.

avg_evolution_permission_app

A través de este endpoint se obtiene el número medio de permisos solicitados por una aplicación, permitiendo así comparar su evolución a lo largo de las distintas versiones. Para obtener esta información se realiza un conteo de los permisos solicitados por la aplicación en cada versión y se calcula su valor promedio. La respuesta proporcionada por este endpoint se refleja en la tabla 4.35.

Aplicación	**Nº medio de permisos**
WhatsApp	64.3

Tabla 4.35: Respuesta endpoint: Avg evolución permisos.

avg_evolution_dangerous_permission_app

A través de este endpoint se obtiene el número medio de permisos solicitados de tipo *dangerous* por una aplicación, permitiendo así comparar su evolución a lo largo de las distintas versiones. Para obtener esta información se realiza un conteo de los permisos de tipo *dangerous* solicitados por la aplicación en cada versión y se calcula su valor promedio. La respuesta proporcionada por este endpoint se refleja en la tabla 4.36.

Aplicación	**Nº medio de permisos *dangerous***
WhatsApp	15

Tabla 4.36: Respuesta endpoint: Evolución Avg *dangerous*.

avg_evolution_normal_permission_app

A través de este endpoint se obtiene el número medio de permisos solicitados de tipo *normal* por una aplicación, permitiendo así comparar su evolución a lo largo de las distintas versiones. Para obtener esta información se realiza un conteo de los permisos de tipo *normal* solicitados por la aplicación en cada versión y se calcula su valor promedio. La respuesta proporcionada por este endpoint se refleja en la tabla 4.37.

Aplicación	**Nº medio de permisos *normal***
WhatsApp	28.6666

Tabla 4.37: Respuesta endpoint: Evolución Avg *normal*.

avg_evolution_signature_permission_app

A través de este endpoint se obtiene el número medio de permisos solicitados de tipo *signature* por una aplicación, permitiendo así comparar su evolución a lo largo de las distintas versiones. Para obtener esta información se realiza un conteo de los permisos de tipo *signature* solicitados por la aplicación en cada versión y se calcula su valor promedio. La respuesta proporcionada por este endpoint se refleja en la tabla 4.38.

Aplicación	**Nº medio de permisos *signature***
WhatsApp	4.3333

Tabla 4.38: Respuesta endpoint: Evolución Avg *signature*.

avg_evolution_score_app

A través de este endpoint se obtiene la puntuación PIM media de las aplicaciones que conforman una categoría, permitiendo así comparar su evolución a lo largo de las distintas versiones. Para obtener esta información se obtiene la puntuación PIM de todas las apicaciones de una categoría y se calcula su media. La respuesta proporcionada por este endpoint se refleja en la tabla 4.39.

Aplicación	**Score**
WhatsApp	6.0641

Tabla 4.39: Respuesta endpoint: Evolución Avg score.

evolution_count_permission_app

Por medio de este endpoint se conoce la evolución del número de permisos solicitados entre las versiones de una aplicación representados con el año de lanzamiento de esa versión. Se obtiene a partir del número de permisos solicitados en cada una de las versiones y creando su gráfica de líneas correspondiente. La respuesta proporcionada por este endpoint se refleja en la tabla 4.40.

Aplicación	**Versión**	**Nº permisos**
WhatsApp	2021	57
WhatsApp	2022	61

Aplicación	Versión	Nº permisos
WhatsApp	2023	75

Tabla 4.40: Respuesta endpoint: Permisos por versión.

evolution_count_dangerous_permission_app

Por medio de este endpoint se conoce la evolución del número de permisos del tipo *dangerous* solicitados entre las versiones de una aplicación representados con el año de lanzamiento de esa versión. Se obtiene a partir del número de permisos de tipo *dangerous* solicitados en cada una de las versiones y creando su gráfica de líneas correspondiente. La respuesta proporcionada por este endpoint se refleja en la tabla 4.41.

Aplicación	Versión	Nº permisos *dangerous*
WhatsApp	2021	13
WhatsApp	2022	13
WhatsApp	2023	19

Tabla 4.41: Respuesta endpoint: Permisos *dangerous* versión.

evolution_count_normal_permission_app

Por medio de este endpoint se conoce la evolución del número de permisos del tipo *normal* solicitados entre las versiones de una aplicación representados con el año de lanzamiento de esa versión. Se obtiene a partir del número de permisos de tipo *normal* solicitados en cada una de las versiones y creando su gráfica de líneas correspondiente. La respuesta proporcionada por este endpoint se refleja en la tabla 4.42.

Aplicación	Versión	Nº permisos *normal*
WhatsApp	2021	27
WhatsApp	2022	28
WhatsApp	2023	31

Tabla 4.42: Respuesta endpoint: Permisos *normal* versión.

evolution_count_signature_permission_app

Por medio de este endpoint se conoce la evolución del número de permisos del tipo *signature* solicitados entre las versiones de una aplicación representados con el año de lanzamiento de esa versión. Se obtiene a partir del número de permisos de tipo *signature* solicitados en cada una de las versiones y creando su gráfica de líneas correspondiente. La respuesta proporcionada por este endpoint se refleja en la tabla 4.43.

Aplicación	**Versión**	**Nº permisos *signature***
WhatsApp	2021	4
WhatsApp	2022	4
WhatsApp	2023	5

Tabla 4.43: Respuesta endpoint: Permisos *signature* versión.

evolution_score_app

Por medio de este endpoint se conoce la evolución de la puntuación PIM entre las versiones de una aplicación representados con el año de lanzamiento de esa versión. Se obtiene a partir del score de cada versión de la aplicación y creando su gráfica de líneas correspondiente. La respuesta proporcionada por este endpoint se refleja en la tabla 4.44.

Aplicación	**Versión**	**Score**
WhatsApp	2021	5.9085
WhatsApp	2022	5.9085
WhatsApp	2023	6.3754

Tabla 4.44: Respuesta endpoint: Score versión.

4.3. Inteligencia artificial y Privacidad

Una vez analizada la gran cantidad de información que proporciona el repositorio App-PIMD, se realiza una investigación teniendo en cuenta las capacidades de la inteligencia artificial para analizar un gran volumen de datos. Este trabajo representa un primer acercamiento, cuyo objetivo ha sido valorar la viabilidad de esta propuesta. Se plantea aplicar técnicas de detección de anomalías sobre los permisos solicitados por las aplicaciones con las que generar un modelo capaz de identificar comportamientos inconsistentes, sospechosos o anómalos.

Asimismo, a partir de la información relativa a la puntuación PIM que recibe cada aplicación, se genera un modelo capaz de realizar una predicción del score de privacidad de una aplicación, que permite estimar su grado de intrusividad en relación con otras aplicaciones de la misma categoría.

Este trabajo en el que se incorpora la inteligencia artificial, enriquece el análisis que se ha realizado hasta este punto, proporciona a la herramienta una funcionalidad predictiva.

Para ello, se elaboran dos modelos de inteligencia artificial:

- Un modelo de *aprendizaje supervisado* capaz de predecir la puntuación de intrusividad de la aplicación.
- Un modelo de *aprendizaje no supervisado* basado en técnicas de *clustering*, con el objetivo de que, para cada categoría de aplicación, identificar agrupamientos de permisos entre las aplicaciones que forman la categoría. A partir de estos clusters el modelo detecta aquellas aplicaciones que solicitan permisos atípicos respecto al patrón general de su categoría, permitiendo así señalar posibles casos de comportamiento invasivo o sospechoso en términos de privacidad.

4.3.1. Modelo de aprendizaje supervisado

Para predecir la puntuación de intrusividad que tiene una cierta aplicación, se requiere de un modelo de aprendizaje supervisado [25].

El *aprendizaje supervisado* es una técnica del campo de la inteligencia artificial más concretamente del aprendizaje automático. Se basa en la idea de crear un algoritmo que pueda aprender a partir de ejemplos previamente etiquetados, es decir, datos que están previamente asociados a una salida deseada. Esto implica que el aprendizaje supervisado proporcione al algoritmo un conjunto de datos de entrenamiento los cuales incluyen tanto las características de entrada como las salidas deseadas. El objetivo es que el modelo aprenda la relación entre las características de entrada y la salida deseada, de modo que pueda predecir la etiqueta correspondiente cuando se le presenten nuevos datos no etiquetados.

Algunas de las *características* del aprendizaje supervisado y que son importantes tener en cuenta para aplicarlas correctamente a las características de nuestro trabajo son:

- Se cuenta con datos etiquetados en el entrenamiento del modelo, por ello, se debe disponer tanto de las variables de entrada como del valor de la salida deseada de manera que, el modelo pueda aprender a base de ejemplos.
- Los modelos de aprendizaje supervisado pueden realizar tanto tareas de *regresión* en las que la variable objetivo a predecir es un valor numérico como de *clasificación* dónde se buscan predecir categorías discretas.

- Un factor clave a la hora de realizar un modelo de aprendizaje supervisado consiste en dividir el conjunto de datos por lo menos en dos conjuntos; un *conjunto de entrenamiento* en el que se entrena el modelo y un *conjunto de prueba* donde se evalúa la capacidad de generalización del modelo para evitar el sobreajuste (aprender el conjunto de datos de manera memorística).

  Por ello, se evalúa la calidad del modelo a través de diferentes métricas. Destacar la independencia entre el conjunto de entrenamiento y conjunto de prueba para garantizar la calidad del modelo y su capacidad de generalización. Además de, tener en cuenta que los datos se encuentren balanceados y una distribución representativa de la clase.

Se ha pensado en dos modelos para iniciar esta investigación, estos modelos han sido un *árbol gradient boosting* [30] y una *regresión lineal* [31].

Árbol gradient boosting

Gradient boosting [30] es una técnica de aprendizaje supervisado que se emplea tanto para tareas de regresión como de clasificación. Para nuestro trabajo, esta pensado para desarrollar una tarea de regresión.

Este algoritmo consiste en construir un modelo complejo tomando como punto de partida varios modelos simples que normalmente se emplea a partir de árboles de decisión. Se empieza con árboles de poca profundidad y se inicia un proceso iterativo en el que en cada iteración se entrena un nuevo modelo en el que se corrigen los errores que comete el modelo anterior.

Este proceso se repite hasta que se alcanza el nivel de precisión indicado como hiperparámetro o se cumple con la condición de parada a partir de la cual finaliza el algoritmo. Su gran *ventaja* se encuentra en que se puede ajustar a diferentes funciones de perdida y a incorporar técnicas de regularización con las que evitar el sobreajuste.

Regresión lineal

Respecto al segundo modelo que se ha utilizado para poder predecir el nivel de intrusividad de una aplicación, es la *regresión lineal* [31]. La regresión lineal es un algoritmo de aprendizaje supervisado que permite modelar las relaciones entre una o mas variables independientes y la variable objetivo.

Este algoritmo trata de representar la relación por medio de una línea recta o un hiperplano. Para realizar esta técnica, la relación entre las variables independientes y la variable dependiente debe ser lineal. Esto significa que la variable dependiente puede expresarse como una combinación lineal de las variables independientes. El modelo intenta encontrar los coeficientes óptimos que definen esa recta o hiperplano, minimizando el error cuadrático medio entre las predicciones y los valores reales.

4.3.2. Modelo de aprendizaje no supervisado: Clustering

El segundo modelo que se ha desarrollado no se basa en un aprendizaje supervisado, sino que se realiza un algoritmo de *aprendizaje no supervisado* aplicando técnicas de *clustering*. Como se ha mencionado previamente, a partir de todos los datos que se tiene de privacidad y de solicitud de permisos de aplicaciones recogidas en el repositorio, es interesante aplicar una de estas técnicas con el objetivo de descubrir patrones ocultos en los datos y que todavía no se han descubierto o no sean apreciables a simple vista. Para ello, se utilizan estas técnicas de clustering, propias del aprendizaje no supervisado.

El aprendizaje no supervisado [26] se caracteriza por ser una rama de la inteligencia artificial en la que no se emplean etiquetas que clasifiquen la salida deseada de los datos, ya que su principal objetivo consiste en descubrir estructura, patrones o relaciones ocultas en los datos. Esta rama consta de numerosas técnicas como son clustering, detección de anomalías o reducción de la dimensionalidad.

Nuestro trabajo se basa en emplear técnicas de clustering [27], a través de esta técnica se agrupan elementos similares entre sí formando grupos conocidos como cluster. Los elementos que forman parte del mismo grupo comparten características. En nuestro trabajo se emplea esta técnica debido a que se busca para cada categoría conformar los cluster que existen en función de los permisos que solicita y a partir de ello, detectar aplicaciones que solicitan permisos anómalos.

Las *características* más significativas de este tipo de aprendizaje y las técnicas de clustering son:

- Estos algoritmos no utilizan etiquetas ya que no disponen de una variable objetivo, se trabaja con los datos de entrada.
- El objetivo principal es identificar patrones o estructuras ocultas en los datos sin conocimiento previo.
- Las técnicas de clustering se caracterizan por la formación de grupos (clusters). En ellos, se agrupa los datos en subconjuntos de forma que los elementos dentro de un grupo sean lo más similares posible y lo más distintos respecto a otros grupos.
- Utiliza métricas como Distancia Euclidiana, Manhattan o Cosine para definir la similitud entre elementos que conforman el cluster.

Para ello, se ha probado con algoritmos como DBSCAN [28] y K-Means [41], con el objetivo de ver si es factible aplicar técnicas de clustering para la identificación de permisos anómalos y detectar aplicaciones que supongan riesgos de privacidad.

DBSCAN

DBSCAN se caracteriza por ser un algoritmo de agrupamiento que se basa en la densidad de los datos. A partir de esta densidad, se puede identificar regiones de alta

densidad de puntos como grupos mientras que a los puntos aislados se les identifica como ruido. Una característica importante de DBSCAN es que no requiere de especificar el número de cluster que deseas formar (esto no ocurre con K-means) sino que, utiliza dos parámetros: *eps* (radio de vecindad) y *minPts* (mínimo de puntos para formar una región densa). El funcionamiento del algoritmo parte de un punto arbitrario y expande una región mientras los vecinos dentro del radio cumplan con el umbral de densidad.

K-means

Es otro tipo de algoritmo de agrupamiento en este caso, basado en particiones. Se busca dividir un conjunto de datos en grupos distintos, de tal forma que, los elementos dentro de cada grupo sean lo más similares entre sí, y lo más diferentes respecto a los de otros grupos. El algoritmo parte de una inicialización aleatoria de centroides y, de forma iterativa, asigna cada punto al centro más cercano utilizando la distancia euclídea como criterio principal.

5: Diseño

Una vez definidos los requisitos sobre los que se desarrolla el proyecto y las consultas sobre el repositorio App-PIMD para extraer información sobre aspectos de privacidad, así como, los modelos de inteligencia artificial empleados para la detección de anomalías sobre los permisos solicitados por las aplicaciones, se pasa a la fase de diseño.

En esta fase se crean los modelos de inteligencia artificial con los que dar respuesta a las preguntas planteadas en la etapa de Análisis, se establece la arquitectura que conforma la herramienta que se desarrolla para este Trabajo de Fin de Máster y por último, se presentan los bocetos que conforman la visualización.

5.1. Inteligencia artificial y Privacidad

En esta etapa del Trabajo de Fin de Máster, se aplican técnicas para dar respuestas a los interrogantes sobre si existen patrones ocultos en los datos que ofrecen las aplicaciones en cuanto a términos de privacidad y seguridad. Y si la inteligencia artificial ayuda a descubrir patrones, detectar anomalías o anticiparse a riesgos de intrusividad. Para ello, se han creado distintos modelos de inteligencia artificial pensados en la búsqueda de modelos tanto para aprendizaje supervisado como aprendizaje no supervisado.

5.1.1. Aprendizaje supervisado

Esta técnica de inteligencia artificial por la cuál se predice la puntuación de intrusividad que tiene una cierta aplicación, crea modelos con los que predecir el nivel de intrusividad de una aplicación según la métrica PIM. Para ello, se tiene en cuenta el *conjunto de datos* con el que se trabaja, su *preprocesamiento* y la *creación* y *evaluación* de modelos con los que determinar si existe posibilidad o interés en el uso de inteligencia artificial para predecir el nivel de privacidad de una aplicación.

Conjunto de datos

Para desarrollar un modelo de inteligencia artificial capaz de predecir el score de intrusividad de una aplicación, se han utilizado los datos recopilados del repositorio de App-PIMD. Estos datos se encuentran presentes en diferentes tablas que se han exportado en formato .CSV para el entrenamiento del modelo.

Una vez se ha analizado el contenido de cada una de las tablas se ha creado un dataset para llevar a cabo la creación del modelo. Este dataset está compuesto por información relativa de los metadatos de las aplicaciones, los permisos solicitados por cada aplicación y la puntuación PIM de intrusividad de la aplicación que constituye nuestra variable objetivo.

De esta manera, el dataset que se ha formado para la predicción del score de una aplicación esta compuesto por las siguientes columnas:

- *package*: Tipo de dato categórico que representa la aplicación a la que hace referencia.
- *min_sdk_version*: Tipo de dato numérico continuo que hace referencia a la versión mínima de sdk sobre la que se ejecuta la aplicación.
- *target_sdk_version*: Tipo de dato numérico continuo que informa sobre la versión principal de sdk sobre la que se ejecuta la aplicación.
- *max_sdk_version*: Tipo de dato numérico que representa la versión máxima de sdk sobre la que la aplicación se puede ejecutar.
- *category*: Tipo de dato categórico que representa la categoría de la aplicación.
- *Permission*: Tipo de dato categórico que refleja los permisos solicitado por la aplicación.
- score: tipo de dato numérico que muestra la puntuación PIM que recibe la aplicación.

Con todo ello, se ha obtenido un conjunto de datos que cuenta con 50 categorías diferentes, 4947 permisos, de los que se puede extraer información de las 11471 aplicaciones de las que se tiene datos.

Limpieza y transformación de datos

Una vez que se ha construido el dataset para el entrenamiento del modelo, se pasa a la parte de *limpieza y transformación de los datos*. En esta parte se empieza por comprobar los valores nulos del dataset y se observa como la única columna que tiene datos nulos es *category*. Debido a la gran cantidad de valores nulos, ya que se cuenta con 6163 aplicaciones que no tiene asignado categoría, se ha decidido crear un *árbol de regresión* con el objetivo de predecir los valores nulos y no perder una columna que puede ser interesante para el modelo que se quiere realizar.

Inferencia categoría de la aplicación

El objetivo de crear este pequeño modelo, en el que se predice el tipo de categoría de una aplicación, reside de la parte de limpieza de datos y poder imputar valores nulos a la categoría de la aplicación.

Para llevarlo a cabo, se parte del conjunto de datos inicial que se había descrito previamente. De él, se elimina la columna de *score* debido a que se considera que ese valor no es relevante a la hora de determinar la categoría. Además, es probable que si no se conoce el valor de la categoría de la aplicación, tampoco, se pueda conocer su puntuación de nivel de intromisión, es por ello que, se decide eliminar esta columna.

Otra columna que se desea eliminar es la de *package*. Esta columna representa el identificador de la aplicación. Debido a que esto puede restringir el aprendizaje del modelo y dificultar su generalización se decide eliminar esta columna.

La última transformación que se lleva a cabo, consiste en eliminar todos los *valores nan de la columna categoría* ya que el objetivo de este modelo es precisamente predecirlos. Por lo que, *categoría* pasa a ser la variable objetivo del modelo, mientras que, las variables de *min_sdk*, *target_sdk*, *max_sdk* y *Permissions* son las features del modelo.

A la hora de entrenar y validar el modelo, se cuenta con datos de 5308 aplicaciones. Por lo que, se ha decidido realizar una *validación cruzada* con el objetivo de poder aprovechar al máximo los datos de los que se dispone tanto para la parte de entrenamiento como para le evaluación del modelo y poder comprobar su grado de generalización.

La *validación cruzada* [33] es una técnica que se basa en dividir los datos en carpetas y en cada iteracción se prueba con una de esas carpetas mientras que el resto se utilizan para el entrenamiento. De esa manera, todos los datos se utilizan tanto para entrenar como para validar el modelo. El resultado final está marcado por la *tasa de acierto* y una *desviación estándar* que muestra la variación que tiene el resultado. Se calcula a través de la media de cada uno de los *accuracy* calculados para cada carpeta.

En nuestro caso, se han dividido los datos en 5 folds y tras aplicar un modelo de un *árbol de regresión* se han conseguido unos resultados de una *media de 0.9018* y una *desviación de 0.0034*, lo cuál, son unos resultados positivos y muestran que a partir de este pequeño modelo se pueden rellenar los valores nulos. En la siguiente tabla 5.45 se pueden observar los resultados obtenidos respecto a la media y desviación del modelo de árbol de regresión.

Modelo	**Media**	**Desviación**
Árbol de regresión	0.9018	0.0034

Tabla 5.45: Evaluación predicción categoría.

Una vez que ya se dispone de un modelo con el que poder hacer una imputación de valores ausentes sobre la columna *categoría*, se pasa a tener en cuenta el resto de operaciones de limpieza. En nuestro caso debido a que no tenemos más valores ausentes y que los datos se toman de una fuente confiable se supone que estos datos no sufren ni problemas de ruidos, ni tampoco de la presencia de outliers. Por todo ello, se pasa a la etapa de *transformación de datos*.

Transformación de los datos

Durante la transformación de los datos, se observa como existen algunas variables sobre las que se debe tener en cuenta una serie de operaciones. Lo primero de todo es transformar las variables de tipo categórico a un tipo numérico para ser tratadas por el modelo, es por ello que, se debe prestar atención a dos columnas que son: *permissions* y *category*.

La columna que más problemas ha dado es la de *permissions*. Esta columna representa todos los permisos que solicita una aplicación. Debido a la gran variedad de estos permisos, no resultaba adecuado aplicar un *label encoder*, debido a que el número de valores que tendrían serían muy grande y permisos que podrían considerarse similares al transformarse en valores numéricos se podrían encontrarse muy separados por lo que, llevaría a interpretaciones erróneas al modelo. Mientras que, aplicar un *one-hot encoder* tampoco es apropiado, debido a que, representar las columnas de forma binaria aumenta mucho el coste computacional del modelo y su complejidad.

La solución más óptima ha consistido en basarse en que los permisos formasen parte de las columnas y atributos del modelo. De esta manera, se representa información booleana de si esa aplicación solicita (1) o no solicita (0) ese permiso.

Respecto al resto de atributos numéricos, no se realiza ninguna transformación. Se pensó en realizar un *one-hot encoder* pero, no se vio necesario ya que la distancia entre los permisos puede tener un significado oculto.

Para las categorías se decidió aplicar un *label encoder* debido a que no era muy alto el número de variables y no se quería aumentar más la complejidad del modelo.

Una vez realizada la transformación de los datos, se pasa a la división de los mismos. Se aplica un *train-test split* en el que el 80 % de los datos se emplean para el entrenamiento mientras que, el 20 % restante se usan para la validación. De esta manera, se comprueba la capacidad de generalización del modelo.

Creación de modelos

A la hora de pensar en los modelos sobre los que se va a construir un algoritmo para predecir la puntuación de intrusividad que recibe una aplicación en función a metadatos

como: su categoría, versión mínima, máxima, media de sdk y los permisos que solicita, se ha pensado en el desarrollo de dos modelos de aprendizaje automático. El objetivo es poder compararlos y seleccionar el mejor de los dos modelos. Como se planteó en un primer momentos en el análisis, se pensó en dos modelos para iniciar esta investigación. Estos modeles han sido un *árbol gradient boosting* [30] y una *regresión lineal* [31]. A continuación, se detallan cada uno de los modelos que se han elaborado.

Árbol gradient boosting

Gradient boosting [30] es una técnica de aprendizaje supervisado que consiste en construir un modelo complejo tomando como punto de partida varios modelos simples que normalmente se emplean a partir de árboles de decisión. Se empieza con árboles de poca profundidad y se inicia un proceso iterativo en el que en cada iteración se entrena un nuevo modelo en el que se corrigen los errores que comete el modelo anterior. Este proceso se repite hasta que se alcanza el nivel de precisión indicado como hiperparámetro o se cumple con la condición de parada a partir de la cuál finaliza el algoritmo. Los *hiperparámetros* más interesantes que se han tenido en cuenta al generar el modelo han sido:

- *n_estimators*: Número de árboles que se construye, con este hiperparámetro se construyen 100 árboles para desarrollar el modelo.
- *loss*: Establece la función de perdida que sigue en el algoritmo, al tratarse de un problema de una regresión, se utiliza el error cuadrático.

Regresión lineal

En cuanto al segundo modelo que se ha probado para predecir el nivel de intrusividad de una aplicación, se ha utilizado una *regresión lineal* [31]. Uno de los motivos por los que se ha seleccionado este algoritmo reside en su simplicidad. Gracias a él, se observa la influencia que toma cada permiso en la decisión y por lo tanto, determina cuales son los permisos que tienen una mayor influencia.

Evaluación de modelos

Una vez que ya se han construido y entrenado ambos modelos, se pasa a realizar la evaluación para conocer la calidad de cada uno de los modelos generardos [32]. Para ello, se calcula las métricas de *MSE* y R^2, a través del conjunto de datos de prueba independiente al de entrenamiento. A partir de estas métricas se conoce cuál es el modelo de mayor calidad y mejor comportamiento.

Coeficiente de determinación

El *coeficiente de determinación* también conocido como R^2 [34], es una métrica que calcula la proporción de variabilidad de la variable dependiente que puede ser explicada por medio del modelo de regresión que se ha elaborado.

Su valor se encuentra dentro del rango 0 - 1, dónde:

- 1: El modelo presenta un ajuste perfecto.
- 0: El modelo no explica ninguna variabilidad.

De esta manera, esta métrica mide como de bueno es el modelo en comparación a predecir el valor medio de la variable objetivo. Cuanto más alto sea el valor de R^2 significa que el modelo captura bien la relación entre las variables independientes y la dependiente. El coeficiente de correlación puede ser engañoso en modelos con muchas variables, ya que tiende a aumentar su valor. Por lo que, lo más adecuado es comparar este valor junto a otras métricas como MSE.

Modelo	**Valor R^2**
Gradient boosting	0.9665
Regresión lineal	0.9646

Tabla 5.46: Evaluación modelos R^2

MSE

La métrica MSE (Error cuadrático medio), es una de las métricas más conocidas y utilizada para evaluar el rendimiento de los modelos de regresión. A través de ella, se puede conocer la diferencia entre los valores predichos y los valores reales. Para ello, se calcula la diferencia entre el valor predicho y el valor real y se eleva la diferencia al cuadrado. A continuación, se promedian los errores al cuadrado. De esta manera, cuanto más bajo sea el valor significa que el modelo tendrá un mejor ajuste, es por ello que, un valor de MSE de 0 es considerado perfecto. Los modelos que se han generado presentan los siguientes MSE:

Modelo	**Valor MSE**
Gradient boosting	0.048
Regresión lineal	0.0507

Tabla 5.47: Evaluación modelos MSE

Conclusión sobre modelo predicción score

Una vez que se ha pasado por todas las etapas de construcción del modelo, se puede observar como para el caso de querer *predecir el nivel de intromisión de una aplicación*, el mejor modelo a partir de la evaluación efectuada es el relativo a *Gradient boosting*. Con este modelo se obtiene unos valores de MSE de 0.048 los cuales son muy bajos y el error que se tiene es mínimo. Respecto al valor del coeficiente de determinación, también es bastante bueno de 0.96 es por ello que al proporcionar buenos resultados, unidos a que son ligeramente superiores a los de la regresión lineal se ha decidido utilizar este modelo para poder predecir el nivel de intromisión de una aplicación.

5.1.2. Aprendizaje no supervisado

A partir de esta técnica de inteligencia artificial se pretende descubrir la solicitud de permisos anómalos con el objetivo de encontrar patrones ocultos con los que poder determinar aplicaciones que supongan un riesgo de privacidad. Para ello, nos centramos en aplicar técnicas de clustering con las que poder agrupar cada categoría en función de los permisos que solicitan las aplicaciones. De esta manera, las aplicaciones que se clasifiquen dentro de un cluster y se encuentren alejadas de su centroide, se puede inferir que existe riesgo de privacidad con ella.

A continuación, se detallan las diferentes fases por las que se pasa para la creación de los modelos: Conjunto de datos, limpieza y transformación de los datos y creación y evaluación de los modelos generados. A través de ello, se busca conocer sí las técnicas empleadas proporcionan o no resultados interesantes sobre la detección de riesgos de privacidad.

Conjunto de datos

Para crear un modelo de *aprendizaje no supervisado* basado en técnicas de clustering, primero se debe tener en cuenta los datos con los que se va a trabajar. Al igual que en el modelo de aprendizaje supervisado anterior, tras una búsqueda de los atributos más interesantes que pueden resultar en este modelo, lo primero que se hace es una unión entre varias tablas que contienen información relativa a los metadatos de la aplicación y las tablas con información de los permisos solicitados.

Destacar que, en este conjunto de datos no se incluye el *score* porque el principal objetivo es que a partir de los metadatos y permisos de los que se dispone, poder detectar patrones que sean capaces de mostrar riesgos de privacidad y si se mantiene el atributo score está directamente relacionado con ellos. Es por ello que, se decide no tener en cuenta este atributo.

En cambio, se debe tener en cuenta que, se está ante un problema de aprendizaje no supervisado, es por ello que, no se va a contar con etiquetas que informen al modelo de la salida deseada, sino que, se busca encontrar patrones en los datos que logren explicar un cierto comportamiento, en nuestro caso, la detección de posibles riesgos de privacidad.

Los *atributos* que conforman el conjunto de datos sobre las que se busca un modelo de aprendizaje no supervisado van a ser:

- *package*: Tipo de dato categórico que representa la aplicación a la que hace referencía.
- *min_sdk_version*: Tipo de dato numérico continuo que hace referencía a la versión mínima de sdk sobre la que se ejecuta la aplicación.
- *target_sdk_version*: Tipo de dato numérico continuo que informa sobre la versión principal de sdk sobre la que se ejecuta la aplicación.
- *max_sdk_version*: Tipo de dato numérico que representa la versión máxima de sdk sobre la que la aplicación se puede ejecutar.
- *category*: Tipo de dato categórico que representa la categoria de la aplicación.
- *Permission*: Tipo de dato categórico que refleja los permisos solicitado por la aplicación.

De esta manera, se ha obtenido un conjunto de datos compuesto por 11471 aplicaciones a las que se puede aplicar técnicas de clustering.

Limpieza y transformación de datos: Embedding

Una vez que ya se conocen los datos sobre los que se van a aplicar la técnicas de clustering, se pasa a la parte de *limpieza y transformación*. Primero, se realizan las operaciones relativas a la limpieza de los datos, al igual que ocurría en el modelo anterior, lo primero que se comprueba es la existencia de valores nulos en el dataset.

El único atributo que tenia valores ausentes era *category*, pero gracias al modelo que se creó anteriormente que calcula la categoría de una aplicación en función de sus metadatos y de los permisos que esta solicita, se infirieron estos valores nulos sin necesidad de perder información. Respecto al *ruido de los datos* y presencia de *outliers*, no se tuvieron en cuenta ya que se partía de una fuente oficial y de confiabilidad de los datos por lo cuál se consideraron válidos.

Transformación de datos: Embedding

Una vez finalizada la parte de limpieza de los datos, se pasa a realizar la *transformación de datos*. En este caso se tienen que tener en cuenta varias cuestiones:

- La primera cuestión hace referencia a que los permisos solicitados como se vió en el conjunto de datos, eran datos de tipo categórico, por lo que, se debía transformar su tipo. Es por ello que, como se estudió en el modelo anterior ante las dificultades de realizar un *one-hot encoder* y la perdida o generación de información errónea por parte de *label encoder*. Hicieron que fuese necesario transformar los permisos en

columnas del dataset, y que se conviertan en atributos booleanas que indican si la aplicación solicita o no ese permiso.

- Otro aspecto importante en la transformación de los datos vino en que parecía interesante poder añadir y crear un nuevo atributo que proporcionase más información sobre los permisos. De ahí que, se decidiese añadir la columna *número de permisos* que se encarga de realizar un conteo del número de permisos solicitados por cada aplicación.

  Una vez que se ha generado este nuevo atributo se elimina la columna *paquetes* ya que es un indicador de la aplicación y puede dificultar el encontrar patrones al ser todos los datos diferentes en ese campo, puede generarse un cluster por cada aplicación.

- Respecto a la última decisión de transformación, esta es relativamente importante, ya que el objetivo de este modelo consiste en detectar patrones entre datos y poder detectar anomalías de las solicitudes de permisos de una aplicación. Si se genera un cluster sobre todo el conjunto de datos debido a la gran variabilidad da lugar a unas muestras que no son del todo representativas.

  Es por ello que, se ha decidido aplicar estas técnicas de clustering sobre *subconjuntos*, pensando en la opción de separar por categoría. De esta manera, se crea grupos más precisos entorno a su categoría y son capaces de detectar mejor los patrones que existen entre los datos que pueden llevar a que se detecte que una aplicación hace una solicitud inadecuada de sus permisos.

Teniendo en cuanta de que se dispone inicialmente de las siguientes categorías:

education, business, books_and_reference, finance
productivity, music_and_audio, personalization, lifestyle
travel_and_local, communication , entertainment, health_and_fitness
game_puzzle, game_simulation, food_and_drink, maps_and_navigation
shopping, game_action, social, game_casual, photography
sports, news_and_magazine, game_card, game_role_playing
art_and_design, game_educational, game_board, medical
auto_and_vehicle, beauty, events, game_arcade
game_adventure, comics, video_players, game_racing
game_sports, house_and_home, game_casino, dating
game_strategy, weather, libraries_and_demo, game_word
parenting, game_trivial, game_music

Se ha observado que existe un gran número de categorías lo que dificulta el desarrollo de un modelo de clustering por cada categoría. Hay algunas de ellas que no tiene los suficientes valores como para aplicar esta técnica. Es por ello que, se ha decidido agrupar las categorías entorno a su funcionalidad dando lugar a los siguientes conjuntos.

Supercategoría	**Categoría**
Game	Game_Action, Game_Adventure, Game_Arcade, Game_Board, Game_Card, Game_Casino, Game_Casual, Game_Educational, Game_Music, Game_Puzzle, Game_Racing, Game_Role_Playing, Game_Simulation, Game_Sports, Game_Strategy, Game_Trivia, Game_Word
Utility	Tools, Productivity, Personalization, Libraries_And_Demo
Media	Music_And_Audio, Video_Players, Photography
Social	Social, Communication, Dating
Health	Health_And_Fitness, Medical, Parenting
Education	Education, Books_And_Reference
Lifestyle	Lifestyle, Beauty, House_And_Home, Food_And_Drink, Auto_And_Vehicles, Art_And_Design
News	News_And_Magazines, Comics
Travel	Travel_And_Local, Maps_And_Navigation, Weather
Business	Business, Finance, Shopping, Events
Entertainment	Entertainment
Sport	Sports

Tabla 5.48: Agrupación categorías por funcionalidad

Estas supercategorias sobre las que se realiza el clustering se han obtenido según las funcionalidades que desarrolla cada categoría, es decir, se han agrupado entorno al desarrollo de funcionalidades similares.

Embedding

Una parte importante de la transformación de los datos reside en poder reducir la complejidad de los mismos. El hecho de tener tantas columnas hace que se incremente la complejidad del modelo, es por ello que, se ha decidido en la etapa de transformación aplicar la técnica de embedding [35],

La *técnica de embedding* consiste en representar variables en forma de vectores de manera que se reduzca la dimensión del conjunto de datos. A pesar de reducir la dimensión se busca mantener la mayor cantidad posible de información. Esta transformación permite reemplazar representaciones de alta dimensionalidad por vectores más compactos. Esta

técnica facilita el procesamiento y mejora el rendimiento de los modelos de aprendizaje automático.

Aunque su uso más común ha sido con variables categóricas, en este proyecto, se han aplicado esta técnica sobre un conjunto de columnas booleanas que indican la presencia o ausencia de distintos permisos [36]. Esto se debe a que, al tratar las columnas como un conjunto, estas columnas conforman un conjunto binario de alta dimensionalidad que puede dificultar la eficiencia del modelo. Gracias al uso de embeddings, ha sido posible reducir de forma significativa el número de columnas sin una pérdida relevante de información, lo que contribuye a disminuir la complejidad del modelo. Es por todo ello que, para reducir el número de columnas booleanas relacionadas con permisos, se ha implementado un *autoencoder*, una red neuronal no supervisada diseñada para aprender representaciones comprimidas de los datos.

Arquitectura embedding

La arquitectura utilizada con la que se ha construido el autoencoder consta de una capa de entrada cuyo tamaño es el número de columnas originales, seguida por dos capas densas con 32 y 16 neuronas respectivamente, ambas con función de activación ReLU [37].

A continuación, se incorpora una *capa densa* intermedia denominada *embedding*, compuesta por 10 neuronas. Esta capa es la clave del modelo, su salida es utilizada como representación comprimida del conjunto original. La parte decodificadora simula la estructura inversa, con capas densas de 16 y 32 neuronas, finalizando en una capa de salida con el mismo número de neuronas que la entrada, utiliza como función de activación la función *sigmoide* para reconstruir la entrada.

El modelo se ha entrenado con el optimizador *Adam* [38] y la función de pérdida *binary_crossentropy* [39] se ha empleado esta función pensando en que nos encontramos ante datos binarios, se ha entrenado durante 50 épocas y con un tamaño de lote de 32 batch. Como resultado, se ha obtenido una representación vectorial de 10 dimensiones que resume de forma eficiente la información contenida en todas las columnas booleanas originales permitiendo reducir significativamente la complejidad computacional.

Evaluación modelo embedding

Para evaluar la calidad del modelo embedding generado, se ha realizado un proceso de reconstrucción a partir de la representación comprimida obtenida. Esta reconstrucción se ha comparado con los datos originales utilizando como métrica el error cuadrático medio (MSE), tal y como se explicó en apartados anteriores.

El resultado obtenido ha sido un MSE de*0.13*, un valor relativamente cercano a cero, lo que indica que el modelo captura de forma efectiva la información presente en las variables originales, generando así una representación de buena calidad.

Creación de modelos

El objetivo que se lleva persiguiendo en esta investigación de modelos de IA consiste en desarrollar un algoritmo con el que realizar una búsqueda de patrones. Con ello, se pretende en cada categoría generar cluster de los permisos más comunes solicitados. Al incluir una aplicación dentro de un cluster se evalúa la distancia a la que se encuentra del centroide, si su distancia es muy lejana se podrá decir que nos encontramos ante una anomalía y que la aplicación supone un riesgo de privacidad.

Para ello, una vez que se ha desarrollado toda la parte de transformación de datos, se prueba sobre los algoritmos de DBSCAN [40] y K-means [41] con el objetivo de agrupar los cluster que conforman cada uno de los grupos.

DBSCAN

Este algoritmo se caracteriza por realizar el agrupamiento teniendo en cuenta la densidad de los datos. A partir de esta densidad, se puede identificar regiones de alta densidad de puntos como grupos, mientras que, a los puntos aislados se les identifica como ruido. El funcionamiento del algoritmo parte de un punto arbitrario y expande una región mientras los vecinos dentro del radio cumplan con el umbral de densidad.

Se ha probado con unos valores de *eps* de 0.5 y *min_samples* de 5. Se decidió no seguir investigando este modelo debido a que se generaban muchos cluster y hacían que el conjunto de datos estuviese muy separado. Además, se pudo observar que los resultados obtenidos por K-means aconsejaban seguir este modelo como se verá a continuación.

K-means

K-means es otro tipo de algoritmo de agrupamiento basado en particiones. En él, se divide un conjunto de datos en grupos distintos, de manera que los elementos dentro de cada grupo sean lo más similares entre sí, y lo más diferentes respecto a los de otros grupos. El algoritmo parte de una inicialización aleatoria de centroides y, de forma iterativa, asigna cada punto al centro más cercano utilizando la distancia euclídea como criterio principal. El hiperparámetro más interesante que se ha tenido en cuenta durante la investigación y sobre el que se define K-means es el *número de cluster*. En este caso, se realiza una búsqueda del número óptimo de cluster para cada categoría entre 2 y 23 cluster.

Evaluación de modelos

Como se ha mencionado en la etapa de transformación de datos y creación de modelos, se ha realizado un entrenamiento sobre todas las categorías que se han mencionado anteriormente. Para evaluar la calidad de cada uno de los cluster de los modelos y poder seleccionar el mejor de ellos, se ha decidido utilizar la métrica de *Silhouette Score* [42]

Esta métrica es utilizada para evaluar la calidad de los clusters generados sin necesidad de etiquetas externas. Compara la cohesión interna de un clúster con la separación respecto al clúster más cercano. Su rango de valores oscila entre –1 y +1. De manera que:

- *Valores próximos a 1*: Indican que los puntos están bien agrupados y separados de otros clústeres.
- *Valores cercanos a 0*: Reflejan solapamiento entre clústeres.
- *Valores cercanos a -1*: Indican una agrupación incorrecta.

Gracias a esta métrica de *Silhouette Score* se ha interpretado la estructura del clustering, considerándose valores por encima de 0.7 como indicadores de agrupamientos bien definidos.

En la siguiente tabla 5.49 se muestran los resultados obtenidos por la métrica de Silhouette score en cada una de las categorias.

Categoría	**Silhouette Score**
ENTERTAINMENT	0.7813
UTILITY	0.7493
NEWS	0.7496
GAME	0.8381
SOCIAL	0.7494
HEALTH	0.7713
BUSINESS	0.7257
SPORT	0.8778
TRAVEL	0.7711
EDUCATION	0.7714
MEDIA	0.7684
LIFESTYLE	0.7720

Tabla 5.49: Silhouette Score por categoría

Como se puede observar en la tabla 5.49 los resultados obtenidos son superiores a 0.70 lo que nos quiere decir que se han agrupado los datos correctamente y se han generado un modelo de calidad para cada categoría.

Si se quiere más información adicional que se ha generado durante la investigación como: el valor de la métrica para cada k de cada una de las categorías o cuales son los permisos más comunes de cada cluster, en el repositorio se ha generado un *archivo log* con dicha información, no se ha incluido en esta memoria debido a su gran longitud.

Tras haber realizado esta exploración se ha creado 4 endpoints adicionales a nuestra herramienta con la que poder realizar la parte de IA. Para cada modelo se ha generado 2 endpoints uno de reentrenamiento y otro de predicción y con el que utilizar los modelo que se acaban de generar. Los enpoints que se han generado para cada uno de los modelos son:

Predicción score	
train_score	Reentrenamos el modelo de predicción de intrusividad de una aplicación.
inference_score/{app_name}	Pasando como parámetro la aplicación en la que se esta interesado se realiza una inferencia devolviendo la predicción de intrusividad de una aplicación.

Tabla 5.50: Endpoints: modelo score

Predicción cluster	
train_cluster	Reentrenamos el modelo basado en técnicas cluster de detección de asolicitud anómala de permisos por parte de una aplicación.
inference_score/{app_name}	Pasando como parámetro la aplicación en la que se esta interesado se realiza una inferencia en la que se informa si la aplicación solicita permisos que suponen un riesgo de privacidad.

Tabla 5.51: Endpoints: modelo cluster

Como **conclusión** a los interrogantes que se planteaban al principio de la investigación, se puede afirmar que sí existen patrones ocultos en los datos sobre el comportamiento de las aplicaciones en términos de seguridad y privacidad y que el uso de técnicas de inteligencia artificial, más concretamente de clustering, podrían ayudar a detectar estos patrones, anomalías o incluso anticiparse a posibles riesgos de intrusividad. De ahí que, se haya iniciado una investigación en este campo como una posible linea futura a seguir investigando.

5.2. Arquitectura de la herramienta

Para el desarrollo de la herramienta, es importante establecer las bases sobre las que se sostentan su arquitectura, para ello hay que tener en cuenta los principios que se van a seguir para construir una herramienta organizada de una forma modular y escalable centrada en la creación de una API en la que poder realizar consultas sobre el repositorio App-PIMD de la que extraer información de privacidad.

En el desarrollo de la arquitectura de la herramienta, hay que tener en cuenta la parte de *inteligencia artificial*, se ha realizado una pequeña investigación de cual sería el mejor modelo que se puede desarrollar. También, se debe implementar en nuestra API endpoints con los que poder realizar la inferencia y el reentrenamiento del modelo.

Por todo ello, se ha decidido realizar y llevar a cabo una *arquitectura por capas* [23] y basada en el principio de *Single Responsibility Principle* [24], con el objetivo de dividir las funcionalidades del sistema en componentes independientes y bien definidos. Se ha diseñado una arquitectura basada en los siguientes principios:

- *Separación por capas funcionales*, permite una clara división entre obtención de datos, procesamiento, análisis mediante modelos, y exposición de resultados a través de la API.
- *Aplicación del Principio de Responsabilidad Única*, asegurando que cada módulo o componente del sistema se centre exclusivamente en una tarea concreta.

Primero vamos a conocer los conceptos teóricos de cada uno de ellos y el motivo por el que se han seleccionado.

Arquitectura por capas

Esta arquitectura permite dividir la aplicación en capas en las que cada una de ellas se encarga de desarrollar una funcionalidad o tarea especifica y que se comunica con el resto de capas, esto facilita la modularidad y el mantenimiento mejorando su escalabilidad [23].

La adopción de esta arquitectura en nuestra herramienta, ha permitido estructurar el flujo de trabajo: desde la capa de extracción de datos, pasando por el procesamiento y análisis mediante inteligencia artificial, hasta la exposición de resultados a través de la API. Gracias a esta división por capas, se ha conseguido una arquitectura modular y escalable, en la que es posible modificar una capa sin afectar al resto del sistema.

Principio de responsabilidad única

El principio de responsabilidad única, *Single Responsibility Principle*, establece que cada componente o módulo del sistema debe ser encargado de una sola responsabilidad dentro de la herramienta, es decir, solo debe tener una única funcionalidad. A través de este principio, se reduce el acoplamiento y aumenta la cohesión entre los componentes del sistema, lo que favorece a su mantenimiento y escalabilidad [24].

Este principio se ha aplicado a la hora de desarrollar la herramienta mediante una separación de responsabilidades y de funciones que deben desarrollar entre los distintos modulos y componentes del sistema. Cada uno de ellos se encargan de realizar una tarea especifica como la extracción de datos, prerpocesamiento, entrenamiento de inteligencia artificial o implementación endpoints de la API. Esto ha permitido crear una arquitectura modular y facil de mantener lo que faverece a la escalabilidad y la independecia entre las partes que consituyen la herramienta.

Una vez que se conocen la manera en la que se va a estructurar la arquitectura de la herramienta, se pasa a partir de los diagramas de diseño software que se han elaborado

poder mostrar como es el flujo de datos y trabajo de la herramienta y cual es su estructura de conexiones interna tanto a nivel de componentes como a un nivel más detallado.

Diagrama de flujo externo

A continuación se presenta el *Diagrama de flujo externo*, se quiere visualizar de la manera más esquemática posible como va a ser el flujo de información y cómo va a interaccionar el usuario con la herramienta que estamos desarrollando. Además de cómo esta obtiene los datos por medio del repositorio App-PIMD.

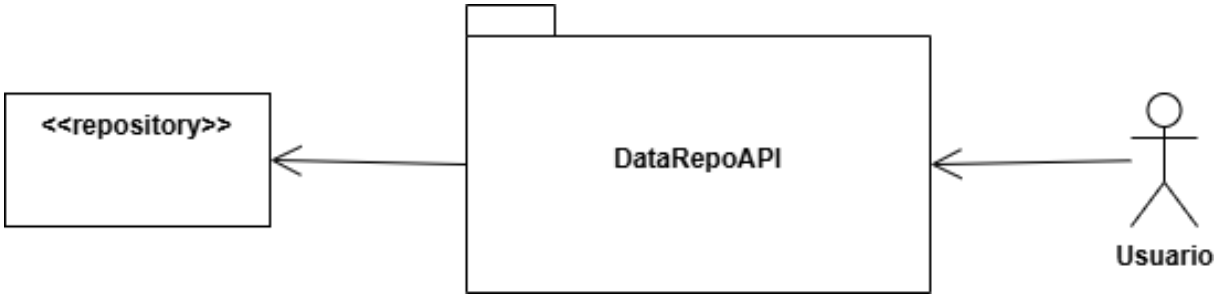

Figura 5.7: Conexiones alto nivel.

Como se observa en el diagrama de la figura 5.7, la herramienta que se esta desarrollando en este Trabajo de Fin de Máster, obtiene los datos de privacidad por medio de una fuente externa, el repositorio App-PIMD. Esta información es procesada por parte de DataRepoAPI, el cual, en función de la consulta realizada por el usuario, ejecuta operaciones con las que poder extraer la información en la que esta interesada el usuario.

De esta manera, la herramienta es capaz de transformar la información bruta de privacidad que proporciona el repositorio en una respuesta estructurada en función de la petición del usuario y que responda a la información que este solicita.

A partir del diagrama se observa como el usuario hace una petición a DataRepoAPI sobre la consulta al repositorio que quiera realizar. La herramienta se encarga de a partir de los datos propocionados en el repositorio poder transformarlos en la información de privacidad en la que esta interesado el usuario.

Diagrama arquitectura simplificada

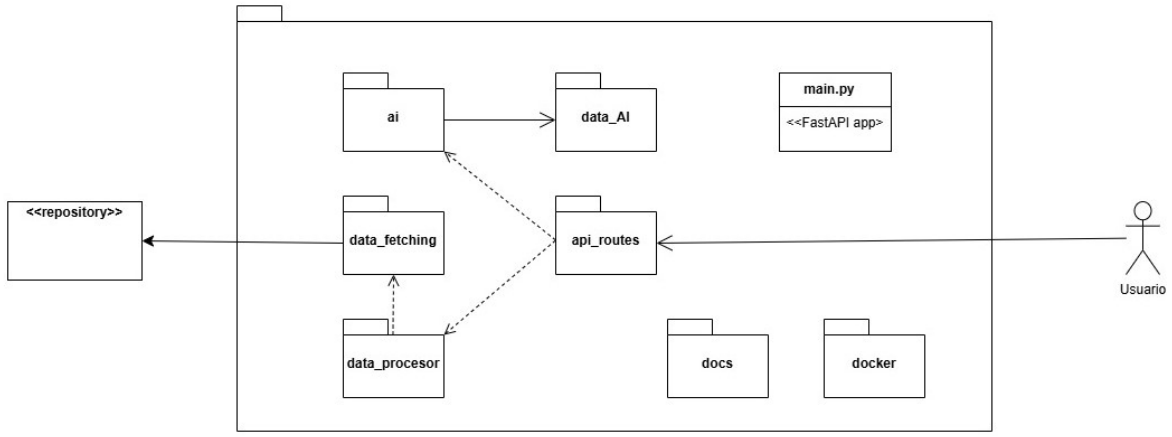

Figura 5.8: Diagrama simplificado paquetes.

El *Diagrama de arquitectura simplificado* figura 5.8 proporciona una visión clara y general del funcionamiento de la herramienta desarrollada, destacando únicamente los componentes clave y el flujo de datos sin entrar en detalles de cada módulo. Su objetivo es ofrecer una comprensión rápida del sistema, mostrando de forma sencilla la relación entre el usuario, la API y la fuente de datos principal, el repositorio App-PIMD.

En este diseño, se observa que el usuario interactúa directamente con los endpoints definidos en la carpeta api_routes, siendo cada uno de ellos la solicitud que se desea realizar al repositorio y que constituye el núcleo de la herramienta. Esta API actúa como punto de entrada de todas las solicitudes, recibe la petición del usuario, interpreta el tipo de consulta que desea realizar sobre el repositorio, y gestiona el flujo de trabajo para proporcionar una respuesta adecuada.

Esta respuesta no se construye a partir de datos internos o estáticos, sino que se genera dinámicamente consultando una fuente de datos externa, el repositorio App-PIMD, que contiene información sobre permisos y metadatos de aplicaciones móviles. A partir de la solicitud del usuario, la API se encarga de conectar con el repositorio, extraer los datos necesarios, procesarlos en función de los criterios especificados en la consulta y devolverlos en un formato estructurado de JSON.

Además, el diseño refleja el *Principio de responsabilidad única*, mantiene separadas las responsabilidades de cada bloque, api_routes se ocupa de la gestión de peticiones y respuestas. La parte relativa a la extracción de datos del repositorio recae sobre data_fetching y el procesamiento está delegada al componente de data procesor. Esta separación permite mantener el sistema organizado, facilitar su mantenimiento y permitir futuras modificaciones o poder conectarse a otros repositorios.

Otro de los puntos a destacar de este diagrama es el relativo al componente de Inteligencía artificial *ai*. En él, se encuentra la lógica de entrenamiento de modelos y los scripts con los que realizar la inferencia. Este componete se encuentra relacionado con data_AI donde se encuentran los archivos *.csv* con los que se ha realizado el entrenamiento de los modelos. Respecto a los componenetes de docs solo cuenta con documentación de los modelos de inteligencia artificial desarrollados, mientras que, el componenete docker contiene el dockerfile con el que se va a construir el contenedor.

Diagrama de componentes detallado

En este último diagrama, se presenta el *Diagrama de componentes detallado* figura 5.9, ofrece una representación más profunda de la arquitectura de la herramienta, descomponiendo los módulos que componen DataRepoAPI, así como reflejando los flujos de datos entre ellos. Se representa explícitamente el pipeline completo que sigue la información desde su origen hasta su entrega al usuario, incluyendo el uso de modelos de inteligencia artificial.

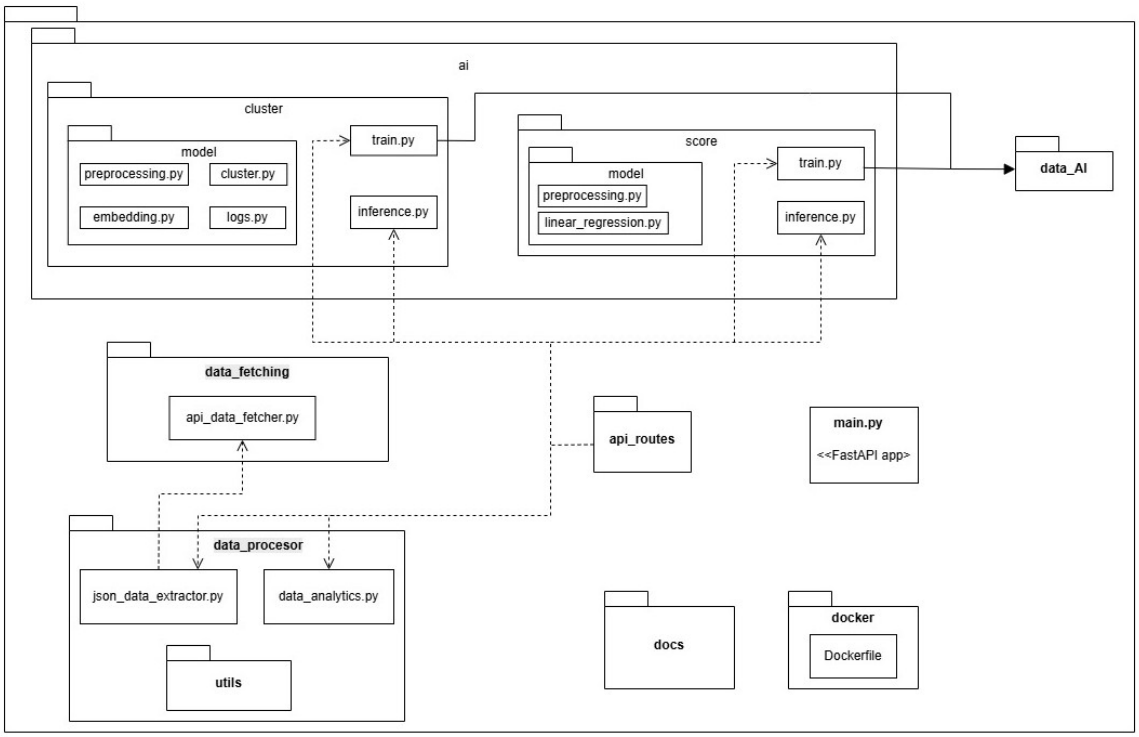

Figura 5.9: Diagrama detallado sistema.

En primer lugar, se identifican los componenetes encargados de la extracción y transformación de datos desde el repositorio. Respecto a los componentes de extracción, se cuenta con el script api_data_fetcher.py con el que se extrae los datos de privacidad de

una aplicación Android del repositorio App-PIMD en formato JSON por medio de una petición HTTP a su API RESTfull.

En cuanto a los componente de la parte de transformación de datos se detacan los script json_data_extractor.py con el que se obtiene la información del JSON obtenido de api_data_fetcher.py. El script date_analysis.py, se ocupa de la creación de métricas de interés que no se obtiene directamente del JSON proporcionado por App-PIMD pero que se pueden conseguir con la información que dá como: el número de permisos solicitados, número de permisos de cada categoria a partir de todas las aplicaciones de una categoría conocer el top de peores y mejores aplicaciones de la categoría, como se ha explicado previamente en la parte de Anális al detallar los endpoints que conforman las consultas.

Posteriormente, los datos procesados pueden pasar por la parte de inferencia de modelos de inteligencia artificial, con el objetivo de generar información más precisa y útil para el análisis de privacidad. También, se cuenta con la posibilidad de poder ser reentrenados los modelos con la información almacenada en formato .csv de la carpeta de data_AI.

Por último, los resultados son devueltos a través de la API, utilizando endpoints organizados por cada una de las solicitudes en las que puede estar interesado el usuario y que ya se explicó previamente en la parte de análisis. El diagrama también muestra la existencia de un archivo Dockerfile a partir del cual se construye una imágen y se podrá lanzar un contenedor con toda la funcionalidad de la herramienta que se ha desarrollado.

5.3. Bocetos visualización

A continuación se presentan los bocetos que se han realizado para diseñar la visualización, por medio de dashboards que se tilizan para realizar una comparativa de privacidad entre aplicaciones y categorías. A través de estos bocetos se plasma las diferentes gráficas que lo conforman. La visualización se compone de tres secciones en las que se representan diferentes informaciones sobre privacidad en función de si se desea comparar aplicaciones, categorías u observar la evolución de una aplicación con el paso de versiones.

Métricas de privacidad por aplicación

En esta sección se busca realizar una comparativa de riesgos de privacidad entre dos aplicaciones. Por medio de un selector de entrada, el usuario puede indicar las dos aplicaciones en las que está interesado en realizar la comparativa. Tras ello, se realiza las consultas sobre el repositorio App-PIMD para extraer la información necesaria.

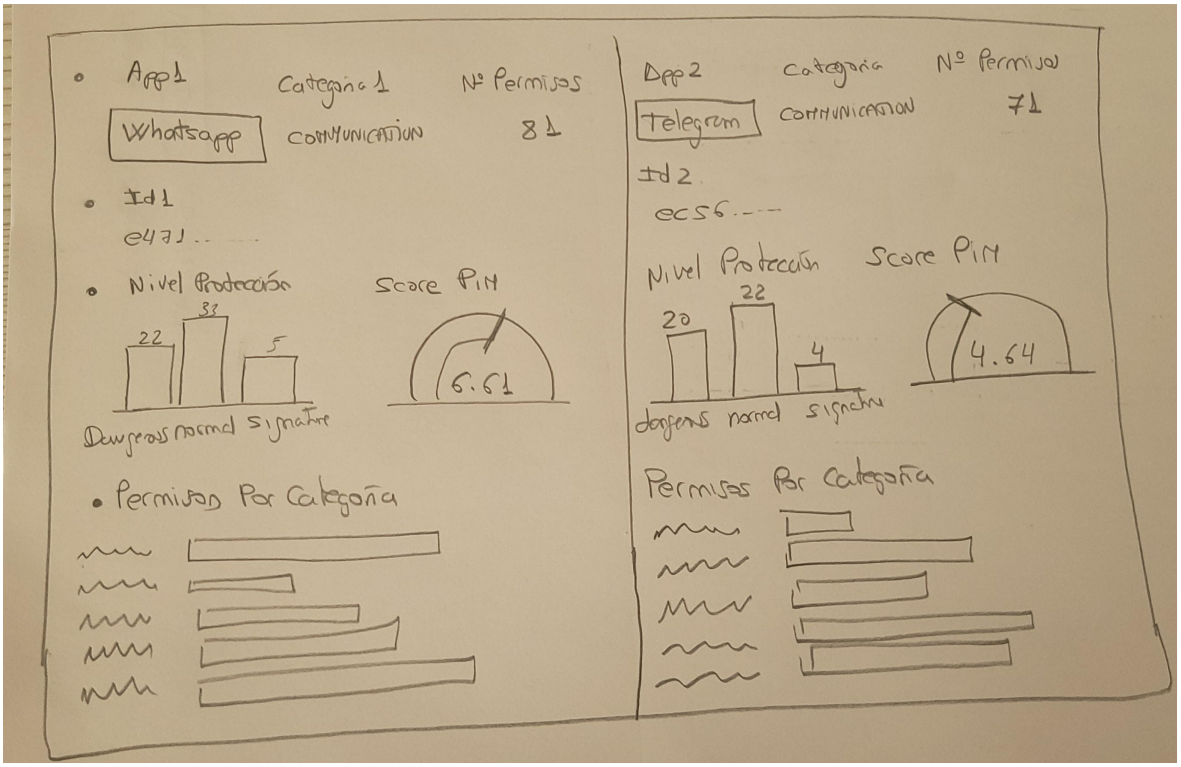

Figura 5.10: Boceto: Comparativa aplicaciones.

Esta sección se encuentra dividida en dos partes con el objetivo de facilitar la comparación entre aplicaciones. Cada parte muestra la información correspondiente a una de las aplicaciones seleccionadas. Esta disposición en paralelo permite realizar un análisis más intuitivo, ya que los datos de ambas aplicaciones pueden ser comparados de forma directa y visual, sin necesidad de cambiar de vista. La información que se va a mostrar en este dashboard y que se corresponde con la figura 5.10 es la siguiente:

- *Aplicación*: Nombre de la aplicación que el usuario solicita para realizar la comparativa.
- *Categoría*: Categoría a la que pertenece la aplicación solicitada por el usuario.
- *Número de permisos*: Muestra el número de permisos totales que la aplicación solicita al usuario. Se representa por medio de una gráfica stat para visualizar su valor
- *Hash*: Identificador hash único de la aplicación en el repositorio, nos permite asegurarnos de que se esta analizando la aplicación en la que realmente estamos interesados. Se representa también por medio de una gráfica stat que visualiza su valor.
- *Nivel de protección:* Por medio del endpoint *count_permissions_type* se pretende conocer el número de permisos de cada tipo que solicita la aplicación. Se representa

por medio de un gráfico de barras para poder realizar una comparación entre los tipos de permisos solicitados. Este gráfico pretende mostrar como se distribuyen los permisos solicitados por la aplicación y si hay una mayor cantidad de algun tipo, prestando especial interés en el tipo dangerous.

- *Ranking score*: Con esta visualización se observa la puntuación PIM de privacidad que recibe la aplicación. Se emplea un velocímetro ya que se busca comparar el valor PIM con la calificación que este recibe. El valor PIM oscila entre 0-10 siendo 0 una aplicación que no supone riesgos de privacidad mientras que el 10 es el valor de riesgo máximo. El gráfico en forma de velocímetro ayuda a realizar una comparación de cómo de alto es el nivel.

  Para realizar una comparación de como de *mejor* o *peor* es la métrica, se ha pensado en utilizar un rango intuitivo como es la valoración de 0-10 pero a la inversa. En el que el rango de 0-5 supone una aplicación segura y se representa por medio del color verde, de 5-8 supone una aplicación con riesgos de privacidad y su color sería amarillo y a apartir del 8 hasta el valor máximo 10 supone una aplicación con alto riesgo de privacidad, por lo que se representa con rojo de peligro.

- *Permisos por categoría*: Por medio de esta gráfica se representa el número de permisos solicitados por categoría. Se representa por medio de un gráfico de barras horizontal que facilita la usuario realizar una comparación a simple vista de que tipo de categorias de permisos es el más solicitado en función de la longitud de la barra. Aunque también, se incluye su valor para ser más claros. Esta gráfica se obtiene a partir de la información propocionada en el endpoint de *count_permissions_category*.

Métricas de privacidad por categoría

Esta sección del dashboard se caracteriza por mostrar una comparación por categorías. En ella, se ofrece una lista de categorías y el usuario selecciona las dos categorias en las que está interesado en conocer sus riesgos de privacidad. Esta sección se encuentra dividida en dos partes debido a que se pertende al igual que en la anterior poder establecer una comparativa a simple vista de los riesgos de privacidad por categoria.

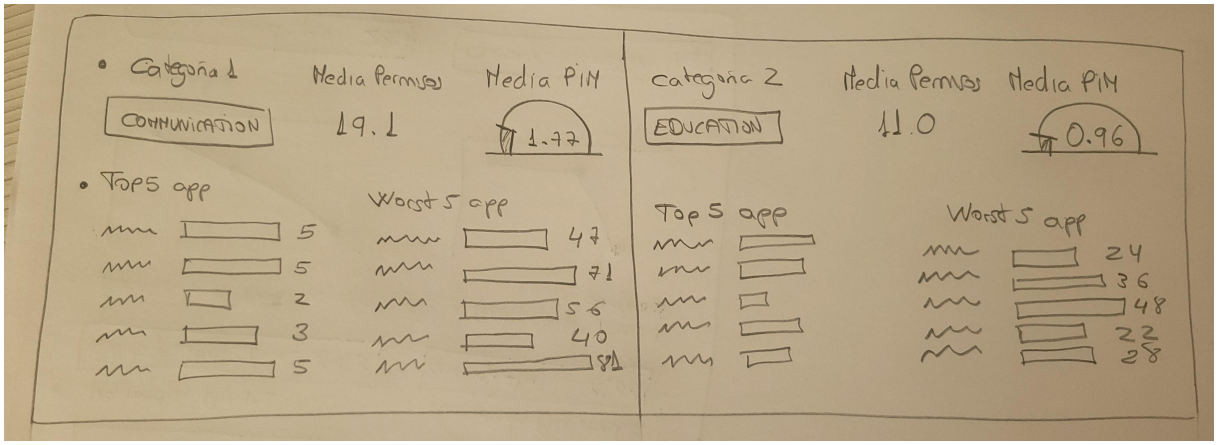

Figura 5.11: Boceto: Comparativa categorias.

Como se puede observar a través de la figura 5.11, en esta segunda sección de la visualización se proporciona la siguiente información:

- *Categoria*: Nombre de la categoría que queremos conocer sus riesgos de privacidad.
- *Media permisos*: Número medio de permisos solicitados por las aplicaciones que conforman esta categoría. Se visualiza por medio de un gráfico de tipo stat donde se muestra su valor.
- *Media PIM*: Representa el valor PIM promedio de todas las aplicaciones de la categoría. Se representa por medio de un gráfico de tipo velocímetro, se ha elegido este tipo de visualización debido a su fácilidad de interpretación por parte del usuario ya que se puede relacionar con escalas del 0 al 10, lo que facilita al usuario interpretar el valor obtenido como *mejor* o *peor*.

  El velocímetro está dividido en tres rangos diferenciados por colores para reforzar la interpretación de los datos: De 0 a 5 en color verde, representa un riesgo bajo o aceptable, de 5 a 8 en color amarillo, es un riesgo moderado pero indicando que entramos en una zona de precaución y de 8 a 10 en color rojo, representa un riesgo alto, indicativo de una mala puntuación en términos de privacidad.
- *Top 5 aplicaciones*: Informa de las 5 mejores aplicaciones de esa categoria comparandolas respecto al número de permisos totales que estas solicitan, se representa por medio de un gráfico de barras horizontal con su valor correspondiente. Se emplea este gráfico ya que a través de la longitud de la barra nos ayuda a observar visualmente cual es la aplicación que menos permisos solicita, facilitando una visualización más rápida e intuitiva. Esta gráfica se representa en color verde para distinguirla y facilitar la comparación de la siguiente gráfica *worst 5 app*, la cual, es de color rojo reflejando que se trata de las peores aplicaciones de la categoría respecto al número de permisos.

- *Worst 5 aplicaciones*: Esta gráfica muestra las 5 peores aplicaciones de la categoría. Al igual que la gráfica anterior se utiliza un diagrama de barras horizontal para facilitar la comparación entre ellas, a simple vista y de forma intuitiva, se puede conocer las aplicaciones que más permisos solicitan. Al igual que la gráfica anterior se utiliza el color para facilitar la interpretación del usuario.

Evolución permisos de una aplicación con el paso de versiones

En esta última sección de la visualización, se representa la evolución en el tiempo de los permisos y puntuaciones PIM con el paso de versiones de ciertas aplicaciones.

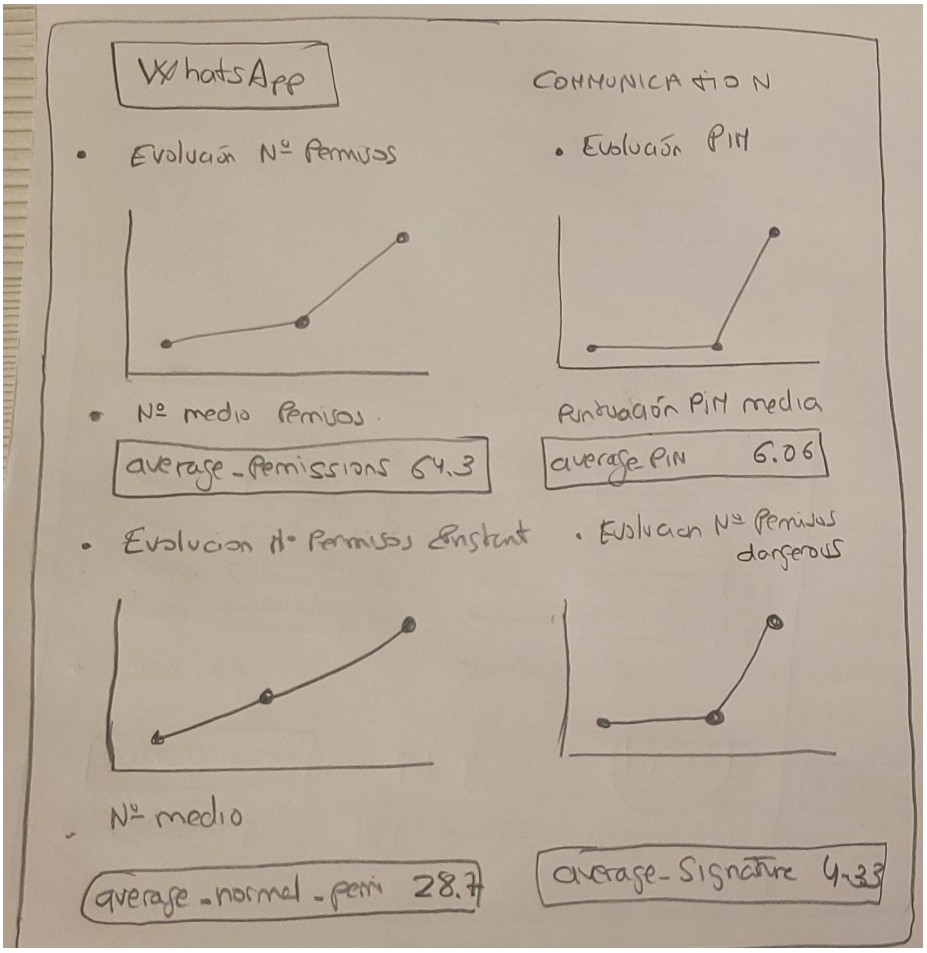

Figura 5.12: Boceto: Evolución permisos de una aplicación.

El objetivo de esta visualización es observar como ha variado en función del tiempo la solicitud de permisos y su puntuación de privacidad, de forma que, se pueda observar si ha variado el comportamiento de la aplicación convertiendose en una app más instrusiva o por el contrario, esto se ha visto reducido o mantenido.

Como se puede observar en la imágen figura 5.12 a diferencia de las dos secciones anteriores, esta no se encuentra partida en dos partes comparativas, sino que es una sola parte en la que se trata de comparar unicamente una aplicación consigo mismo, es decir, su evolución según el paso de versiones.

Esta visualización se compone de gráficas de líneas que muestran la evolución temporal del número de permisos totales, número de permisos en función de su tipo y la evolución de la puntuación PIM que recibe la aplicación. Debajo de cada uno de estas gráficas de líneas, encontramos una gráfica de tipo stat que informa sobre el valor promedio de dicha variable con el objetivo de observar si a través del paso de versiones estas variables han ido aumentado o disminuyendo.

Las graficas que conforman esta sección son:

- *Nombre de la aplicación*: Nombre de la aplicación de la que se quiere conocer su evolución con el paso de versiones.
- *Categoría*: Categoría a la que se corresponde la aplicación.
- *Evolución número de permisos*: Gráfica de líneas que muestra la evolución en el tiempo del número de permisos totales solicitados de una aplicación.
- *Promedio de permisos solicitados entre las versiones*: Gráfica de tipo stat que muestra el valor promedio de permisos solicitados por esa aplicación en ese periodo de tiempo.
- *Evolución PIM*: Gráfica de líneas que muestra la evolución de la puntuación PIM recibida por la aplicación.
- *Promedio PIM entre las versiones*: Gráfica de tipo stat que muestra el valor promedio puntuación PIM recibida por la aplicación.
- *Evolución Número de permisos instant*: Gráfica de líneas que muestra la evolución en el tiempo del número de permisos del tipo instant solicitados de una aplicación.
- *Promedio de permisos instant solicitados entre versiones*: Gráfica de tipo stat que muestra el valor promedio de permisos solicitados del tipo instant por esa aplicación en ese periodo de tiempo.
- *Evolución Número de permisos dangerous*: Gráfica de líneas que muestra la evolución en el tiempo del número de permisos del tipo dangerous solicitados de una aplicación.
- *Promedio de permisos dangerous solicitados entre versiones*: Gráfica de tipo stat que muestra el valor promedio de permisos solicitados del tipo dangerous por esa aplicación en ese periodo de tiempo.
- *Evolución Número de permisos normal*: Gráfica de líneas que muestra la evolución en el tiempo del número de permisos del tipo normal solicitados de una aplicación.

- *Promedio de permisos normal solicitados entre versiones*: Gráfica de tipo stat que muestra el valor promedio de permisos solicitados del tipo normal por esa aplicación en ese periodo de tiempo.

6: Resultados obtenidos

Este capítulo presenta los resultados finales que se han conseguido en el proyecto de investigación que se plantea al inicio del informe. Estos resultados ayudan a comprender el trabajo que se ha realizado y el porqué. Es importante destacar que los datos que se han utilizado para conseguir estos resultados provienen del repositorio App-PIMD y junto a una API interna que se ha creado se proporciona 47 consultas que realizar sobre el repositorio con los que extraer información de privacidad facilitando así el análisis de la información.

Por ello, se muestran estos resultados junto a un ejemplo en el que un usuario puede analizar esta información. Este usuario tiene ciertos interrogantes sobre privacidad y va a representar a cualquier persona que tenga alguna inquietud a la hora de descargarse cualquier aplicación en un dispositivo Andoid. Especialmente, cuando le solicite aceptar permisos y revisar su política de privacidad. ¿Que permisos son realmente necesarios y cuáles no?.

Resultados: Comparativa entre aplicaciones

El primer resultado que se puede observar a través de la visualización que se ha creado es una comparativa entre aplicaciones que pretenden evaluar su privacidad. El usuario introduce el nombre de las dos aplicaciones sobre las que quiere conocer su información de privacidad.

Para ello, una vez que el usuario introduce el nombre de las aplicaciones puede compararlas a simple vista puesto que el campo de visión se parte en dos. La parte izquierda, ofrece toda la información relativa a la *aplicación 1* de color azul, mientras que, la parte de la derecha ofrece la información relativa a la *aplicación 2* de color verde. De esta manera, en la mismas visualización y sin tener que cambiar de vista, puede realizarse una comparativa de ambas aplicaciones.

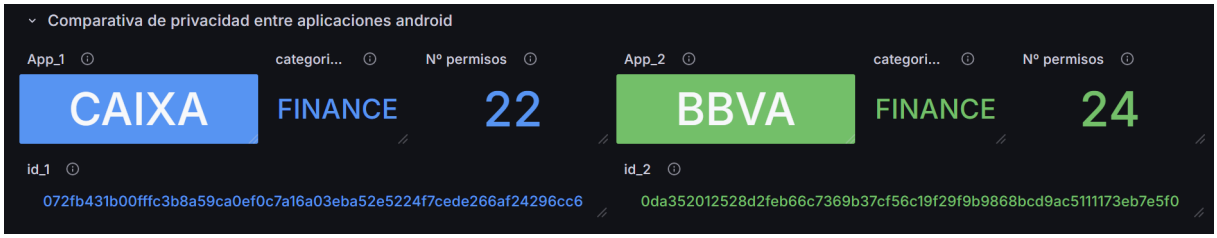

Figura 6.13: Resultados comparativa aplicaciones 1.

Lo primero que se encuentra el usuario en esta sección de la visualización como se observa en la figura 6.13, es el nombre de la aplicación con su categoría y con el número de permisos totales que solicita la aplicación. Debajo se encuentra el número de identificador de la aplicación representado mediante el valor hash que tiene la aplicación en el repositorio. Las aplicaciones tienen un color diferente para facilitar la visualización y su comparación.

De esta manera, el usuario al introducir la aplicación bancaria la CAIXA y BBVA observa que se corresponde con aplicaciones de la categoría de finanzas. El BBVA solicita más permisos que la CAIXA haciéndose idea en un primer momento de cuál de las dos aplicaciones es la que solicita un mayor número de permisos. Este indicador es insuficiente para tener una idea completa de la privacidad de una aplicación, ya que no especifica otra información relevante cómo el tipo de permisos, nivel de intrusividad, entre otros. Por ello, se pasa a un segundo nivel, debajo de esta información el usuario observa unas nuevas métricas que se corresponden con la siguiente imagen.

Figura 6.14: Resultados comparativa aplicaciones 2.

En esta imagen, figura 6.14, el usuario conoce por medio de distintos gráficos el tipo de permiso y su número. Pudiendo comparar a través de un gráfico de barras cuál de las dos aplicaciones contiene más permisos de tipo *dangerous*, *normal* o *signature* de ambas aplicaciones. Esta gráfica, además de contener el valor del número de permisos de cada tipo, también utiliza la longitud de la barra para facilitar la comprensión y comprobar a simple vista cuál es el tipo de permisos que mayor número tiene y cuál el menor.

El usuario dispone de un velocímetro que se corresponde con la puntuación PIM de cada aplicación, pudiendo de nuevo comparar entre ambas aplicaciones, cuál tiene mayor

o menor valor PIM. Esta métrica proporciona un valor que cuanto más alto sea indica un mayor impacto sobre la privacidad, es decir, mayor riesgo y menor privacidad.

Observando la imagen, el PIM de la aplicación de la CAIXA tiene mayor privacidad que el PIM de BBVA que tiene un mayor riesgo. De todas las maneras, ambos valores representan un valor bajo puesto que el velocímetro se encarga también de informar al usuario a través del color cuando se está en riesgo y cuando no. Un marcador en verde significa que no hay peligro, mientras que, uno amarillo alerta de que el riesgo de privacidad es mayor y por último, un marcador rojo detalla que la aplicación supone un riesgo de privacidad.

Figura 6.15: Resultados comparativa aplicaciones 3.

Para completar la comparativa entre aplicaciones, se crea una nueva gráfica que se observa en la figura 6.15 en la que el usuario a través de un diagrama de barras horizontales, conoce cuál es el número de permisos por categoría que solicita una aplicación. Esta gráfica es útil al usuario porque le permite analizar y comparar los permisos solicitados por cada aplicación, en nuestro caso CAIXA y BBVA.

Permite al usuario ver que aplicación es la que solicita más permisos sensibles como por ejemplo, cámara, micrófono y localización, lo que puede indicar posibles riesgos de privacidad. También, permite identificar si cierta aplicación está solicitando permisos innecesarios para su funcionamiento y así, poder decidir sobre que aplicación de las dos instalar basándose en la cantidad y el tipo de permisos.

A través de nuestro ejemplo y viendo la imagen, se observa que la aplicación del BBVA solicita más permisos Accounts que CAIXA, BBVA solicita tres permisos frente a CAIXA que solo solicita uno. O como ambas, tienen un permiso para micrófono, pero en la categoría de location CAIXA solicita tres permisos frente a los dos de BBVA.

Una vez analizada toda la información que proporciona esta primera sección de la visualización, el usuario dispone de datos necesarios para tomar una decisión informada sobre cuál de las dos aplicaciones descargar, en función de sus intereses y prioridades. Si el objetivo principal es minimizar el impacto sobre la privacidad, podrá seleccionar la aplicación con el valor más bajo en la métrica PIM en nuestro caso la CAIXA. Por otro lado, si el usuario prefiere centrarse en el tipo o la cantidad de permisos solicitados, por

ejemplo, eligiendo la aplicación que solicita menos permisos de la categoría *accounts* o *location*, también podrá identificar fácilmente cuál representa la opción más adecuada, así como, si presta atención al número de permisos dangerous solicitados o el número total de permisos. De este modo, el usuario podrá tomar decisiones basadas en diferentes criterios de privacidad.

A continuación, se muestra una imagen general figura 6.16 de esta primera sección, en la que el usuario podrá comparar a simple vista todos los datos sobre los permisos de dos aplicaciones en las que el usuario está interesado conocer para su privacidad.

Figura 6.16: Resultados Comparativa app total.

Resultados: Comparativa entre categorías

En la herramienta que se ha creado, el usuario además de comparar aplicaciones, puede comparar aplicaciones por sus categorías y conocer más datos e información sobre permisos y privacidad en función de su categoría.

Para ello, en esta segunda sección se proporciona en la cabecera dos listas con todas las categorías de aplicaciones de las que se tiene información en el repositorio App-PIMD. De esta manera, el usuario selecciona dos de ellas sobre las que quiere realizar la comparación. Nuevamente, el dashboard se ha dividido en dos secciones para facilitar la comparación visual. También, se ha mantenido el uso del color para diferenciar ambas categorías. Al igual que en la sección anterior, se utiliza el color azul para hacer referencia a la *categoría 1*, mientras que, el color verde hace referencia a la *categoría 2*.

En este punto, el usuario siente inquietud por conocer si las aplicaciones de *finanzas* son más seguras en cuanto a términos de privacidad frente a aplicaciones de *entretenimiento* que pueden ser más permisivas e intrusivas en cuanto a los permisos solicitados. De ahí, que surjan diferentes interrogantes en el usuario cómo: ¿Cuál de las dos categorías pide

un mayor número de permisos? ¿Cuáles son las mejores y peores aplicaciones de cada categoría?

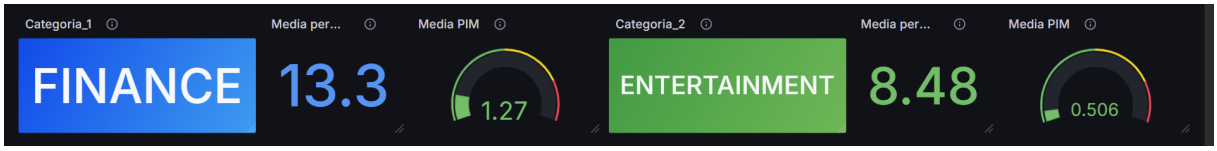

Figura 6.17: Resultados Comparativa categorías 1.

En la primera parte de esta sección, figura 6.17, se muestra información sobre el tipo de categoría que ha seleccionado el usuario e información relativa al número medio de permisos solicitados, así como, la puntuación PIM medio de las aplicaciones de cada categoría. Respecto al tipo de gráficas se puede observar como para la parte de mostrar el número medio de permisos se utiliza un stat con el que se muestra su valor. Mientras que, para conocer la puntuación media de intrusividad de las aplicaciones de la categoría, se utiliza un velocímetro en el que además de mostrar su valor, se hace uso del color para mostrar como es la puntuación promedio.

El usuario introduce en la primera categoría el termino de *finanzas* observando que las aplicaciones de esta categoría tiene un número medio 13.3 permisos y presentan una puntuación PIM de 1.27. Se compara con aplicaciones de la categoría de entretenimiento con las que se ha observafo que el número de permisos promedio solicitados por sus aplicaciones es de 8.48 y la puntuación PIM media de las aplicaciones de la categoría es de 0.506.

Resulta especialmente relevante observar cómo las aplicaciones de la categoría de entretenimiento tienden a ser menos intrusivas en términos de privacidad que las aplicaciones financieras. Estos valores reflejan una tendencia más favorable en cuanto a privacidad para las aplicaciones de entretenimiento en comparación con las de finanzas.

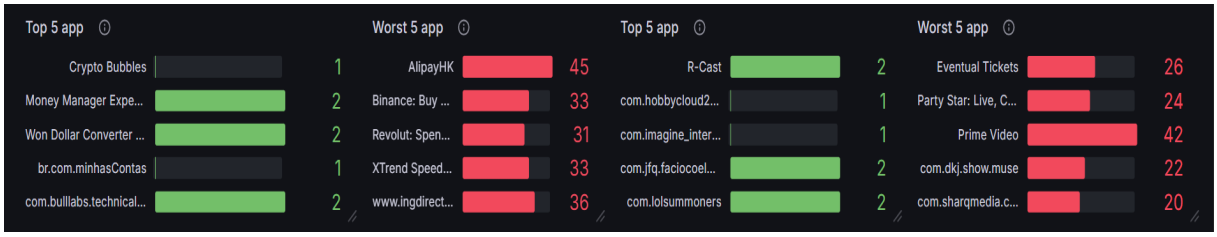

Figura 6.18: Resultados Comparativa categorías 2.

Después de la cabecera, el usuario puede observar dos rankings diferentes en los que se muestran el *top 5 de mejores* aplicaciones de la categoría respecto al número de permisos solicitados y el *top 5 de peores* aplicaciones de la categoría respecto a la misma métrica. Esta información se representa a través de un gráfico de barras horizontal, en el que además

de diferenciarse por el color y por la longitud de la barra, también, se cuenta con el valor del número de permisos que solicita cada una de las aplicaciones.

De esta manera, el usuario puede comparar las mejores aplicaciones entre diferentes categoría. Pero además, puede hacer una segunda comparación al poder comparar con las 5 peores aplicaciones por categoría que también se representan en esta sección. Por ello, se ve necesario el uso del color para facilitar al usuario su diferencia siendo verde para las aplicaciones *top 5 mejores* y rojo para las *top 5 peores*.

El usuario, a través de la figura 6.18 observa que entre las mejores aplicaciones de *finanzas* destaca *crypto bubles* que solicita solo un permiso. Mientras que, la peor aplicación de finanzas es *AlipayHK* que solicita cuarenta y cinco permisos. Además, esta información se puede comparar con los datos de la categoría de entretenimiento, en la que una de las mejores 5 aplicaciones es *Imagine* con solo un permiso solicitado, mientras que, una de las peores es *Prime vídeo* con cuarenta y dos permisos solicitados.

Con estas gráficas, el usuario puede comparar entre las mejores y peores aplicaciones dentro de una misma categoría y entre diferentes categorías.

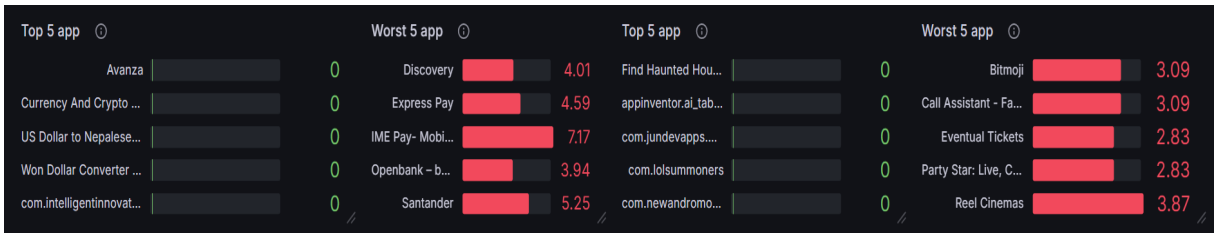

Figura 6.19: Resultados Comparativa categorías 3.

En esta parte de la sección, el usuario observa y conoce los rankings de aplicaciones de la categoría que más y menos puntuación score tienen. De esta manera, el usuario identifica cuáles son las aplicaciones más seguras en términos de privacidad y aquellas de la categoría que más riesgos de seguridad pueden dar.

Se representan nuevamente a través de un gráfico de barras horizontal, haciendo uso del color, su longitud y el valor que corresponde. Al mantener una representación visual similar a la anterior, se busca que el usuario interprete la información de forma rápida e intuitiva.

En la siguiente imagen figura 6.20, se puede observar esta segunda sección al completo sobre comparativa entre las diferentes categorías. El usuario compara a simple vista datos sobre las cinco mejores y peores aplicaciones de cada categoría tanto por el número de permisos solicitados como por su score.

Figura 6.20: Resultados Comparativa categorías total.

Resultados: Evolución permisos solicitados por aplicación

En esta tercera y última sección de la visualización, el usuario a través de distintas gráficas de líneas, conoce la evolución en cuanto al número de permisos solicitados de una aplicación con el paso de las versiones. De esta manera, el usuario puede observar el comportamiento de la aplicación con el paso del tiempo y conocer si una aplicación ha aumentado, disminuido o mantenido su número de permisos solicitados. Se pretende observar como ha evolucionado una aplicación en cuanto el nivel de intromisión.

Figura 6.21: Resultados Evolución aplicación 1.

En la primera parte de la visualización figura 6.21 el usuario introduce el nombre de la aplicación de la que desea conocer la evolución de permisos y la herramienta le responde informándole de la categoría a la que pertenece, mostrando una información meramente informativa. En nuestro ejemplo, el usuario introduce la aplicación de *Glovo* y le informa que pertenece a la categoría de *Business*.

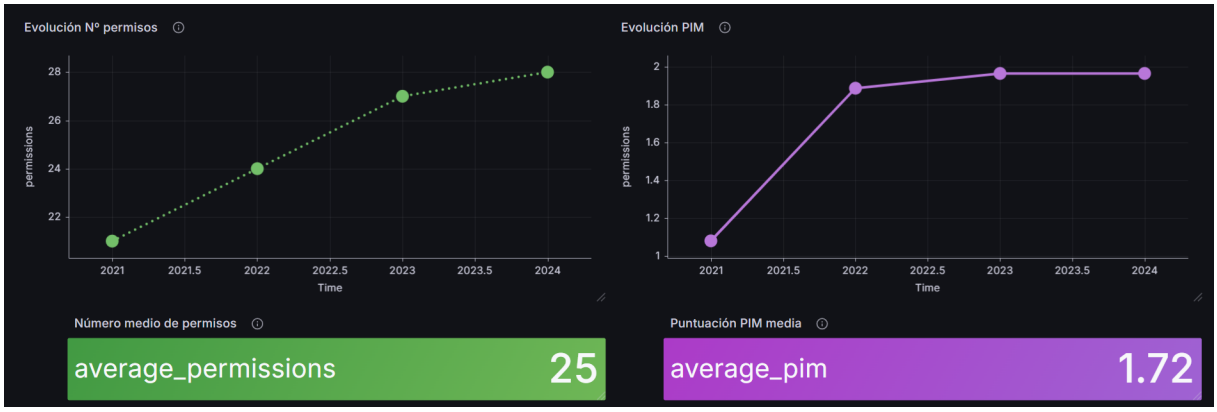

Figura 6.22: Resultados Evolución aplicación 2.

Una vez que el usuario ha introducido el nombre de la aplicación en la que está interesado. Esta sección genera diferentes gráficos de línea con los que se observa la evolución de diferentes métricas. En la figura 6.22 el primer gráfico de líneas se encuentra relacionado con la evolución de los permisos. Se observa cómo esta aplicación ha ido con el paso del tiempo aumentando el número de permisos solicitados, siendo en 2021 menos de 22 permisos para pasar en 2024 a 28 permisos solicitados. Esto refleja un incremento en el número de permisos solicitados por la aplicación de Glovo, incluso superando la media de permisos solicitados en este periodo, lo cuál, se puede observar a través de average_permissiones con 25 permisos de media solicitados en este periodo de tiempo.

En esta sección, también se puede observar la evolución de la métrica PIM, es muy interesante observar como en 2021 esta métrica se encontraba alrededor del 1 mientras que en 2024 llega hasta una puntuación de 2. Su media 1.72 durante este periodo la refleja como una aplicación no muy intrusiva pero que no se ha mantenido en un nivel bajo sino que ha aumentado.

Figura 6.23: Resultados Evolución aplicación 3

En los dos siguientes gráficos, figura 6.23 se observa la evolución del número de permisos tanto de tipo *normal* como *signature* solicitados por la aplicación Glovo en este periodo de tiempo. En la gráfica de la izquierda, se pueda apreciar la evolución que han seguido los permisos de tipo *normal* destacando que desde el 2021 se han mantenido estables hasta 2024, en el que se han incrementando. Consiguiendo una media en este periodo de tiempo de 9.25 permisos de tipo *normal*. Respecto a la gráfica de la derecha, es significativo ver como los permisos de tipo *signature* se han mantenido estables en 1 durante todo este periodo.

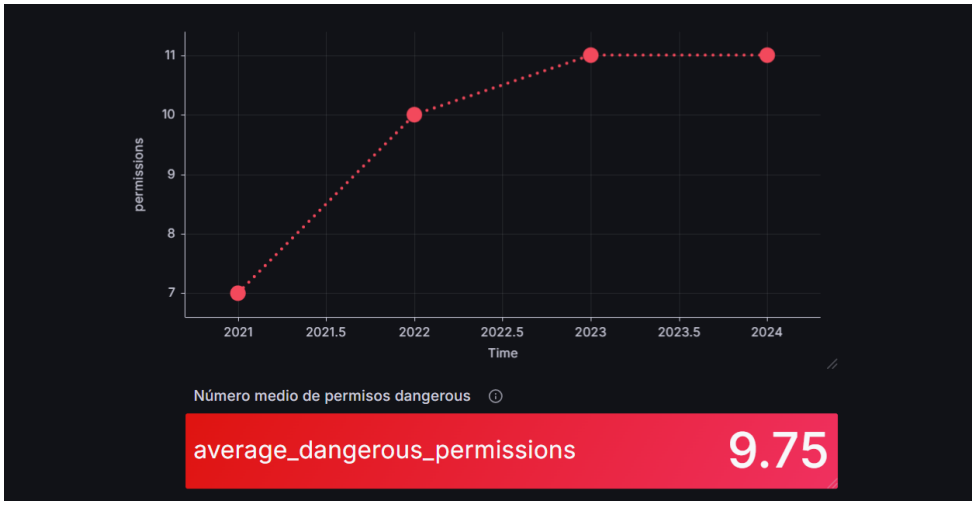

Figura 6.24: Resultados Evolución aplicación 4

En esta última gráfica, figura 6.24, representa la evolución del número de permisos de tipo *dangerous* solicitados por Glovo. Se observa un incremento en este periodo siendo en 2021 de 7 permisos *dangerous*, mientras que en 2024 ha ascendido a 11 permisos *dangerous*, lo que lleva al usuario a cuestionarse porqué este aumento de permisos. Este tipo de permisos *dangerous* implican un mayor riesgo de privacidad, por lo tanto el usuario presta más atención a ello a la hora de tomar decisiones.

Tras analizar todas las gráficas de la evolución de los permisos solicitados por una aplicación, se observa su importancia de cara a la privacidad y la protección de datos. A medida que una aplicación se actualiza con el paso de las versiones, es posible que introduzca nuevas funcionalidades que requieren permisos adicionales a recursos sensibles de nuestros dispositivos. Esto puede representar un riesgo de privacidad y seguridad. A través de esta información, el usuario puede decidir y tomar una decisión adecuada de si dichos permisos se justifican o no con la funcionalidad ofrecida.

Resultados: Inteligencia artificial

Ante el gran volumen de datos con los que se ha trabajado y observado que contienen el repositorio App-PIMD, se plantea la idea en este Trabajo de Fin de Máster de iniciar

una investigación sobre la detección de permisos y riesgos de privacidad mediante el uso de técnicas de inteligencia artificial. Se debe tener en cuenta que se está abordando una investigación en un campo amplio y complejo que requiere de un desarrollo más profundo. Este trabajo pretende ser un primer acercamiento que ha servido para evaluar su viabilidad. Como resultado de esta fase inicial, se han propuesto dos métodos que permiten obtener los siguientes resultados.

Primero, se ha predicho la *puntuación score de intrusividad* de una aplicación. Siguiendo con los ejemplos anteriores, nuestro usuario quiere conocer la puntuación de intrusividad de una nueva aplicación que ha salido al mercado, pero del que aún no se ha calculado su puntuación PIM, ni el usuario conoce los permisos que solicita. El usuario utiliza este endpoint para conocer la puntuación PIM, la herramienta obtiene los permisos y metadatos de la aplicación a partir del repositorio App-PIMD y a través del modelo generado obtiene su puntuación de intromisión. En las siguientes imágenes se puede observar ejemplos de distintas aplicaciones y su uso con este endpoint: *inference_score/{app_name}*.

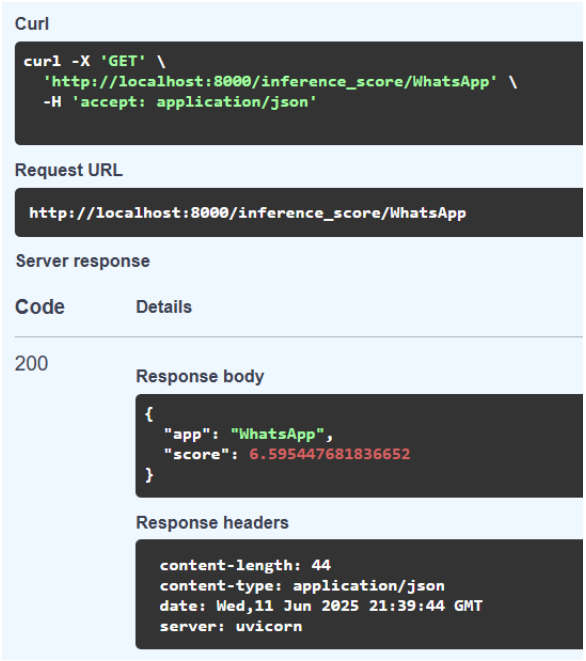

Figura 6.25: Predicción WhatsApp score

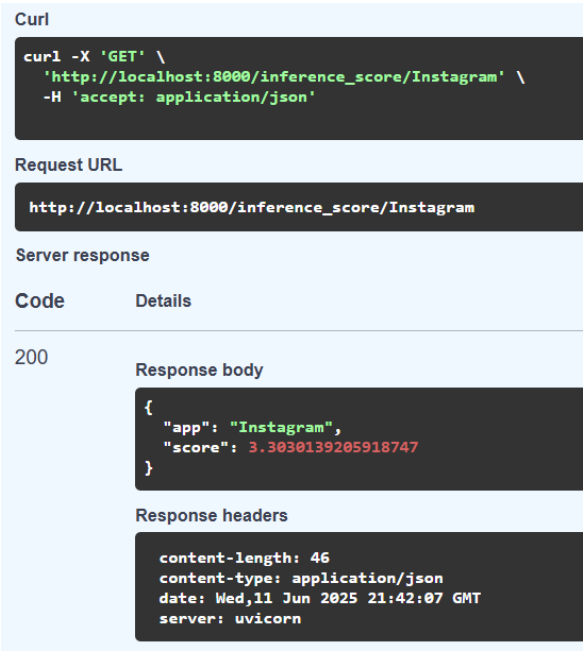

Figura 6.26: Predicción Instagram score

Otro de los objetivos que se pretende investigar se basa en conocer aplicaciones que supongan riesgos de privacidad por medio de técnicas de clustering. De esta manera, se pretende detectar permisos atípicos solicitados por una aplicación.

Por ejemplo, el usuario quiere conocer si la aplicación de WhatsApp se puede considerar una aplicación invasiva o no. A través de este endpoint *inference_cluster/{app_name}*, el sistema obtendrá del repositorio App-PIMD todos los datos necesarios para el modelo generado y tendrá como resultado *True* o *False*. Si el resultado es *True*, el usuario comprende

que sí existen permisos intrusivos para su privacidad en esa aplicación, mientras que, si el resultado es *False*, el usuario puede confiar en la aplicación ya que no se ha detectado riesgos de privacidad. El siguiente paso en líneas futura sería dentro de las aplicaciones de las que se detecta riesgos de privacidad, detectar y predecir cuál de todos los permisos solicitados por la aplicación son los que provocan que sea invasiva.

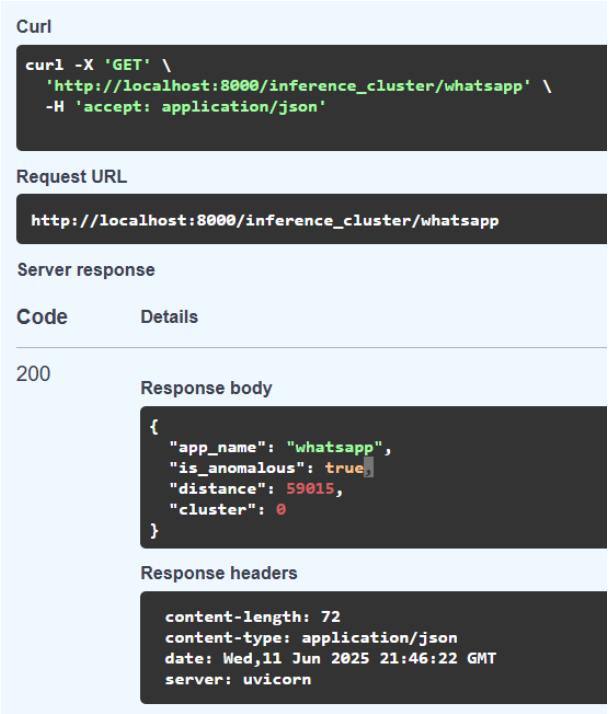

Figura 6.27: Predicción WhatsApp cluster

7: Conclusiones y Líneas de trabajo futuras

En este punto del proyecto se hace hincapié en las *conclusiones* que se han obtenido tras el desarrollo del Trabajo de Fin de Máster: *Selección de indicadores basada en análisis de datos: Aplicación a la gobernanza de la privacidad y la seguridad en aplicaciones móviles*. Se refleja tanto sus aspectos positivos, como los negativos y se comprueba si se han alcanzado los objetivos propuestos al inicio del proyecto. Además de, proponer una serie de posibles mejoras dentro de la sección de *líneas futuras*.

7.1. Conclusiones

A lo largo de este trabajo se ha alcanzado satisfactoriamente el objetivo principal, se ha seleccionado y elaborado una serie de métricas que permiten el análisis y la comparación de aplicaciones Android en términos de privacidad. Para ello, se ha creado una herramienta basada en consultas personalizadas sobre la información proporcionada por el repositorio App-PIMD. Se ha conseguido extraer información relevante sobre permisos y riesgos de privacidad mediante el desarrollo de una API con 47 endpoints, accesible y reutilizables para otros proyectos al exportar la información en formato JSON.

Con ello, se ha elaborado una visualización a partir de los datos proporcionados por esta API que permiten al usuario comparar aplicaciones, categorías y analizar la evolución histórica de los permisos. Mostrando de una manera sencilla y comprensible indicadores de intromisión de aplicaciones móviles. Esto contribuye a facilitar la toma de decisiones por parte del usuario y buscar crear una mayor conciencia crítica sobre la solicitud de permisos y datos personales en aplicaciones móviles.

Asimismo, se ha iniciado una exploración sobre el uso de modelos de inteligencia artificial aplicados a la detección de riesgos de privacidad. En este sentido, se han implementado dos modelos: un modelo supervisado, capaz de predecir la puntuación de intromisión de una aplicación, y otro no supervisado, basado en técnicas de clustering, que permite detectar comportamientos anómalos o sospechosos en función de los patrones asociados

a las categorías de aplicaciones. Ambos modelos se integran en la API, permitiendo su reentrenamiento y consulta mediante endpoints específicos. Es cierto que, de los dos modelos desarrollados el relativo al clustering tiene un mayor interés ya que a partir de patrones anómalos de permisos es capaz de detectar aplicaciones que supongan riesgos de privacidad. Se propone que sería interesante seguir investigando en esta linea, mientras que para el desarrollo de predecir la puntuación de intromisión, es cierto que sería útil para automatizar el proceso de obtención de puntuación PIM de una aplicación, aunque esta misma métrica se puede generar aplicando la fórmula de la métrica PIM.

Desde el punto de vista personal, este trabajo me ha servido para conocer aspectos relativos a la privacidad de las aplicaciones. Sin olvidarse que el desarrollo de este Trabajo de Fin de Máster, me ha sido de utilidad para conocer y adentrarme en el ámbito de una investigación. Además de, conocer nuevos conceptos del área de privacidad y poder afianzar algunos conceptos y técnica vistas durante el máster. Todo ello, ha permitido enriquecerme tanto a nivel profesional como personal.

7.2. Lineas futuras

El trabajo que se ha desarrollado no constituye más que un punto de partida de las tareas que aún se pueden seguir desarrollando y continuar en un futuro. A continuación, se van a detallar algunas *lineas futuras* que se han identificando durante el desarrollo de este trabajo.

- Proporcionar en la visualización un mayor número de gráficas que permita aumentar las opciones y posibilidades de análisis por parte del usuario además de, una mayor interactividad. Hay algunos endpoints personalizados de la herramienta que no han sido utilizados en la parte de visualización y se pueden utilizar en ella.
- Incluir más información en el repositorio App-PIMD, sobretodo entre las versiones de una determinada aplicación con el objetivo de poder visualizar más versiones en la parte de evolución de una aplicación. O añadir nuevos endpoints que proporcionen mayor información sobre las categorías de las aplicaciones.
- Probar nuevos modelos de aprendizaje supervisado con el objetivo de encontrar mejores algoritmos con los que realizar la predicción de puntuación del grado de intromisión de una aplicación, e investigar nuevas técnicas de clustering con los que poder agrupar y descubrir patrones ocultos de los que extraer los datos, siendo capaces de detectar los permisos que provocan que una aplicación sea más o menos intrusiva.

Bibliografía

[1] Saleem, M. S., Mišić, J., & Mišić, V. B. (2022). Android malware detection using feature ranking of permissions. arXiv preprint arXiv:2201.08468.

[2] Wang, Y., Zheng, J., Sun, C., & Mukkamala, S. (2013, July). Quantitative security risk assessment of android permissions and applications. In IFIP Annual Conference on Data and Applications Security and Privacy (pp. 226-241). Berlin, Heidelberg: Springer Berlin Heidelberg.

[3] Sung, Y. T., Chang, K. E., & Liu, T. C. (2016). The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. Computers & Education, 94, 252-275.

[4] Pérez-Fuente, A., Martínez-González, M. M., Aparicio, A., & Moro, Q. I. (2024). Un data warehouse para el estudio de la privacidad y seguridad de aplicaciones móviles. En A. M. Reina Quintero, R. Ceballos Guerrero, & Á. J. Varela Vaca (Eds.), Actas de las IX Jornadas Nacionales de Investigación en Ciberseguridad, JNIC 2024 (pp. 624–631). Sevilla. https://idus.us.es/items/508e612b-3df5-458a-98be-2d5f7c3fb7d2

[5] Pérez-Fuente, A. (2024). Mejorando App-PIMD, un repositorio para el estudio de la privacidad de aplicaciones móviles.

[6] Android Developers. (s.f.). Introducción al archivo del manifiesto. Android Developers. https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

[7] Instituto Nacional de Ciberseguridad (INCIBE). (2022, 28 de noviembre). ¿Por qué piden tantos permisos las apps?. `https://www.incibe.es/ciudadania/blog/por-que-piden-tantos-permisos-las-apps`

[8] Instituto Nacional de Ciberseguridad (INCIBE). Permisos de apps y riesgos para tu privacidad. INCIBE. `https://www.incibe.es/ciudadania/formacion/infografias/permisos-de-apps-y-riesgos-para-tu-privacidad`

[9] Statista. (2024, 6 de noviembre). Dispositivos de Internet móvil y consumo de apps en España. Statista. https://es.statista.com/temas/4179/dispositivos-de-internet-movil-y-consumo-de-apps-en-espana/

[10] Pickaso. (2023, 28 de noviembre). State of mobile apps. https://pickaso.com/blog/state-of-mobile-apps

[11] Briskman, J. (2019, julio). Top Apps Worldwide for Q2 2019 by Downloads. Sensor Tower. https://sensortower.com/blog/top-apps-worldwide-q2-2019-downloads-data-digest

[12] NordVPN. (2023, 8 de diciembre). Privacidad móvil: ¿qué quieren saber tus apps? NordVPN. https://nordvpn.com/es/research-lab/mobile-app-research/

[13] Crehana. (2023, 2 de noviembre). Modelo en cascada: ¿Qué es y cómo aplicarlo en tus proyectos? https://www.crehana.com/blog/transformacion-digital/modelo-en-cascada/

[14] Equipo editorial de IONOS. (2019, 3 de noviembre). El modelo en cascada: desarrollo secuencial de software. IONOS. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/

[15] Oracle España. (2021, 18 de junio). ¿Qué es ETL? Tu guía para los aspectos esenciales de la integración de datos. Oracle. https://www.oracle.com/es/integration/what-is-etl/

[16] Microsoft. (s.f.). Visual Studio Code: el editor de código fuente optimizado para la web. https://code.visualstudio.com/

[17] GitHub. (s.f.). GitHub: plataforma de desarrollo colaborativo. https://github.com/

[18] JGraph Ltd. (s.f.). Draw.io - Diagrams for everyone, everywhere. https://www.drawio.com/

[19] Ramírez, S. (2018). FastAPI: Framework web de alto rendimiento para construir APIs con Python 3.8+. https://fastapi.tiangolo.com/

[20] Poetry Team. (s.f.). Poetry: Gestión de dependencias y empaquetado para Python. https://python-poetry.org/

[21] Docker, Inc. (s.f.). Docker: Accelerated container application development. https://www.docker.com/

[22] Grafana Labs. (s.f.). Grafana: Plataforma abierta y componible de observabilidad. https://grafana.com/

[23] De la Torre Llorente, C., Castro, U. Z., Barros, M. A. R., & Nelson, J. C. (2010). Guía de Arquitectura N-Capas orientada al Dominio con .NET. España: Microsoft Iberica.

[24] Martin, R. C. (2023). Functional Design: Principles, Patterns, and Practices. Addison-Wesley Professional.

[25] Valenzuela González, G. (2022). Aprendizaje Supervisado: Métodos, Propiedades y Aplicaciones.

[26] Portela González, J. (2024). Machine learning: aprendizaje no supervisado.

[27] Rokach, L., & Maimon, O. (2005). Clustering methods. Data mining and knowledge discovery handbook, 321-352.

[28] Deng, D. (2020, September). DBSCAN clustering algorithm based on density. In 2020 7th international forum on electrical engineering and automation (IFEEA) (pp. 949-953). IEEE.

[29] Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. Electronics, 9(8), 1295.

[30] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

[31] Chiok, C. H. M. (2014). Modelos de regresión lineal con redes neuronales. In Anales científicos (Vol. 75, No. 2, pp. 253-260). Universidad Nacional Agraria La Molina.

[32] Sitiobigdata. (2018, 27 de agosto). Machine learning: métricas de regresión – MSE. Sitiobigdata. `https://sitiobigdata.com/2018/08/27/machine-learning-metricas-regresion-mse/`

[33] Krstajic, D., Buturovic, L. J., Leahy, D. E., & Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of cheminformatics, 6, 1-15.

[34] Saunders, L. J., Russell, R. A., & Crabb, D. P. (2012). The coefficient of determination: what determines a useful R2 statistic?. Investigative ophthalmology & visual science, 53(11), 6830-6832.

[35] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

[36] Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737.

[37] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[38] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[39] Mao, A., Mohri, M., & Zhong, Y. (2023, July). Cross-entropy loss functions: Theoretical analysis and applications. In International conference on Machine learning (pp. 23803-23828). PMLR.

[40] Gholizadeh, N., Saadatfar, H., & Hanafi, N. (2021). K-DBSCAN: An improved DBSCAN algorithm for big data. The Journal of supercomputing, 77(6), 6214-6235.

[41] Jin, X., & Han, J. (2016). K-medoids clustering. In Encyclopedia of machine learning and data mining (pp. 1-3). Springer, Boston, MA.

[42] Januzaj, Y., Beqiri, E., & Luma, A. (2023). Determining the Optimal Number of Clusters using Silhouette Score as a Data Mining Technique. International Journal of Online & Biomedical Engineering, 19(4).

[43] Bhaskaran, S. V. (2019). A comparative analysis of batch, real-time, stream processing, and lambda architecture for modern analytics workloads. Applied Research in Artificial Intelligence and Cloud Computing, 2(1), 57-70.

[44] Anjan, K., Andreopoulos, W., & Potika, K. (2021, December). Prediction of higher-order links using global vectors and Hasse diagrams. In 2021 IEEE International Conference on Big Data (Big Data) (pp. 4802-4811). IEEE.