Universidad de Valladolid

UNIVERSIDAD DE BURGOS

universidad de león

Universidades de Valladolid, Burgos y León

TRABAJO FIN DE MÁSTER

Máster Universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

Mejorando App-PIMD, un repositorio para el estudio de la privacidad de aplicaciones móviles

Autor:
Alejandro Pérez de la Fuente

Tutores:
Quiliano Isaac Moro Sancho
M. Mercedes Martínez González

Resumen

La importancia de la privacidad de las personas en el ámbito de los datos personales y dispositivos móviles es fundamental en la era digital actual. Los datos personales, como información financiera, médica o de ubicación, son extremadamente sensibles y pueden ser utilizados de manera indebida si caen en manos equivocadas. En el contexto de los dispositivos móviles, que están constantemente conectados a internet y recopilan una gran cantidad de datos sobre nuestras actividades diarias, la protección de la privacidad se vuelve aún más crítica. Surgió así el proyecto *App Privacy Impact* (App-PI), un ecosistema para la evaluación del impacto de las apps para dispositivos móviles sobre la privacidad y seguridad de sus usuarios. Una de las partes fundamentales del ecosistema es su repositorio de metadatos de aplicaciones móviles que permite a investigadores estudiar el impacto sobre la privacidad de los usuarios, el cuál comenzó su desarrollo en el TFG de Alejandro Pérez de la Fuente [1]. En este trabajo Fin de Máster, se trata de mejorarlo mediante el estudio y aplicación de técnicas y arquitecturas *Big Data*, ampliaciones de la funcionalidad del mismo en términos de fuentes de información, y proporcionando paneles de mandos que muestren los contenidos del repositorio mediante KPIs. Este trabajo forma parte de un proyecto más amplio cuya finalidad es empoderar a los usuarios finales en la gestión de su privacidad y de su seguridad en entornos móviles, App-PI.

Abstract

The importance of people's privacy in the realm of personal data and mobile devices is fundamental in today's digital age. Personal data, such as financial, medical, or location information, is extremely sensitive and can be misused if it falls into the wrong hands. In the context of mobile devices, which are constantly connected to the internet and collect a large amount of data about our daily activities, protecting privacy becomes even more critical. Thus, the *App Privacy Impact* (App-PI) project emerged, an ecosystem for evaluating the impact of mobile device apps on the privacy and security of their users. One of the fundamental parts of the ecosystem is its repository of mobile application metadata, which allows researchers to study the impact on user privacy. This repository began its development in the TFG of Alejandro Pérez de la Fuente [1]. In this Master's thesis, the aim is to improve it by studying and applying Big Data techniques and architectures, expanding its functionality in terms of information sources, and providing dashboards that display the repository's contents through KPIs. This work is part of a broader project whose purpose is to empower end users in managing their privacy and security in mobile environments, App-PI.

Agradecimientos

A mis padres y a mi hermana Ana Isabel, Juan José y Lydia por su apoyo y ánimo incondicional.

A mis tutores Quiliano Isaac Moro Sancho y M. Mercedes Martínez González por su tiempo, ayuda y guía durante el desarrollo del trabajo.

Al Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid por prestarme su apoyo y saber.

A todos los profesores del máster por transmitirme sus conocimientos.

A mis amigos por creer en mí.

El proyecto al que aporta este TFM se realiza al amparo de un convenio de colaboración entre la Universidad de Valladolid y Instituto Nacional de Ciberseguridad de España (INCIBE).

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

En la era digital que vivimos, los teléfonos móviles se han convertido en una herramienta esencial de la vida cotidiana, proporcionando acceso instantáneo a información, comunicación, entretenimiento, enseñanza, entre otros. Según un estudio de Kaspersky, una de las empresas multinacionales líderes en seguridad informática, se demostró que existe una gran dependencia tecnológica basándose en una encuesta sobre una muestra representativa de 2.012 españoles [2]. En la figura 1.1 se puede ver como se distribuye esta dependencia tecnológica en las distintas comunidades autónomas.

Figura 1.1: Mapa de dependencia tecnológica en España [2].

Esta constante presencia de dispositivos móviles plantea serios **riesgos sobre la privacidad de los usuarios**. La recopilación masiva de datos personales, el rastreo de ubicaciones y las vulnerabilidades de seguridad son solo algunos de los problemas que amenazan la privacidad de las personas. De hecho, en este mismo estudio de Kaspersky también se demostró la falta de educación ciudadana en materia de ciberseguridad en los dispositivos móviles, ya que tan solo un 40,5 % de los encuestados tenía una solución de ciberseguridad instalada.

Otra evidencia adicional a este respecto se encuentra en la nota técnica publicada por la AEPD[1] en 2019 sobre el avance en un estudio de riesgos para la privacidad de los usuarios en *software* pre-instalado en dispositivos Android [3]. Algunas de las conclusiones preliminares de este estudio apuntan a que el modelo AOSP[2] y los modelos de monetización de contenido actuales permiten a numerosas organizaciones y empresas monitorear y obtener información personal a través del *software* pre-instalado, del cual los usuarios suelen ser inconscientes. Asimismo, la dificultad de eliminar este *software* y la falta de transparencia en los permisos limitan la capacidad de los usuarios para gestionar su información personal.

La tabla 1.1 muestra una lista con una recopilación de la gran cantidad de aplicaciones pre-instaladas en un teléfono móvil Samsung Galaxy S23 [4].

Lista de aplicaciones pre-instaladas en un Samsung Galaxy S23	
Smart Switch	Samsung Pass
SmartThings	Samsung Members
Samsung Wallet	Samsung DeX
Game Launcher	Samsung Kids
Samsung Health	Samsung Global Goals
Galaxy Store	Find My Mobile
Samsung Internet	Samsung Health Monitor
Samsung Cloud	PENUP
Galaxy Themes	Microsoft Office
Bixby	LinkedIn
Samsung Notes	Microsoft Outlook

Tabla 1.1: Lista de aplicaciones pre-instaladas en un Samsung Galaxy S23 [4].

Sistema Operativo	**Cuota de Mercado**
Android	86,2 %
iOS	13,7 %
Windows Phone	0,1 %

Tabla 1.2: Cuota de mercado para cada sistema operativo móvil [5].

Actualmente, los principales sistemas operativos para teléfonos móviles son Android,

[1] AEPD: Agencia Española de Protección de Datos
[2] AOSP: *Android Open Source Project*

iOS y Windows Phone. Android, desarrollado por Google, es un sistema operativo de código abierto ampliamente adoptado por numerosos fabricantes de dispositivos móviles. Por otro lado, iOS, creado por Apple, es un sistema operativo de código cerrado exclusivo para sus propios dispositivos. En último lugar, Windows Phone que ha quedado prácticamente en desuso, desarrollado por Microsoft. Según datos de mercado recientes [5], la cuota de mercado de estos sistemas es del 86,2 % para Android, 13,7 % para iOS y apenas 0,1 % para Windows Phone (ver tabla 1.2). Es por ello que se destaca la importancia de Android en el ámbito considerado de la privacidad debido a su predominio en el mercado.

1.2. Motivación

Como ya se viene mencionando, la privacidad de los usuarios de aplicaciones móviles es de suma importancia, ya que a través de estos dispositivos se procesan y/o gestionan grandes cantidades de datos personales. Los usuarios confían en que su información personal, incluida su ubicación, historial de búsqueda y comunicaciones, se mantiene segura y confidencial. Según la Ley Orgánica 3/2018 de protección de datos, la privacidad de los usuarios, y en especial en aplicaciones móviles, es un derecho fundamental que debe protegerse [6].

Sin embargo, este no es siempre el caso y puede dar lugar a graves violaciones de privacidad. Recientemente ha habido dos casos de filtraciones de datos personales que he vivido de primera mano:

- La **filtración masiva de datos en la cadena de gimnasios Synergym** entre los cuales se incluía nombre, apellido, número de teléfono y número de DNI[3] [7].
- La **filtración masiva de datos en Banco Santander** de la cual no se tiene información sobre que datos han sido filtrados, pero se sabe que son de carácter personal [8].

Por otro lado, los trabajos relacionados con este tema se encuentran desactualizados [9] o solamente estudian un número reducido de aplicaciones [10], [11]. Estas carencias llevaron a plantear una línea de investigación en el Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid cuyo objetivo general es estudiar el impacto de las aplicaciones sobre la privacidad y seguridad de sus usuarios. Bajo esta línea de investigación se haya el **proyecto estratégico de ciberseguridad *App Privacy Impact* (App-PI)**: Un ecosistema para la evaluación del impacto de *apps* para dispositivos móviles sobre la privacidad y seguridad de sus usuarios dirigido por M. Mercedes Martínez González. Realizado al amparo de una colaboración entre la Universidad de Valladolid y el INCIBE [4], y financiado con fondos PRTR[5] y Next Generation por la Unión Europea.

[3]DNI: Documento Nacional de Indentidad
[4]INCIBE: Instituto Nacional de Ciberseguridad
[5]PRTR: Plan de Recuperación, Transformación y Resiliencia

1.2.1. Estado actual de App-PI y App-PIMD

En la figura 1.2 se muestra la arquitectura del proyecto y algunos de los resultados derivados del desarrollado del mismo. Alineado con este proyecto se ha propuesto una métrica de privacidad que ayuda a entender de manera sencilla el impacto que generan las *apps* a sus usuarios [12]. También se ha puesto a disposición de usuarios finales un servicio *online* denominada APKFalcon que les permite obtener evaluaciones orientativas y fácilmente comprensibles sobre el nivel de riesgo que pueden suponer las aplicaciones, y les ayude a tomar decisiones al respecto [13]. Además, se ha aportado a la comunidad investigadora un repositorio de metadatos de aplicaciones Android que sirven para nutrir las distintas métricas y estudios sobre el impacto en la privacidad que puedan surgir denominado App-PIMD[6].

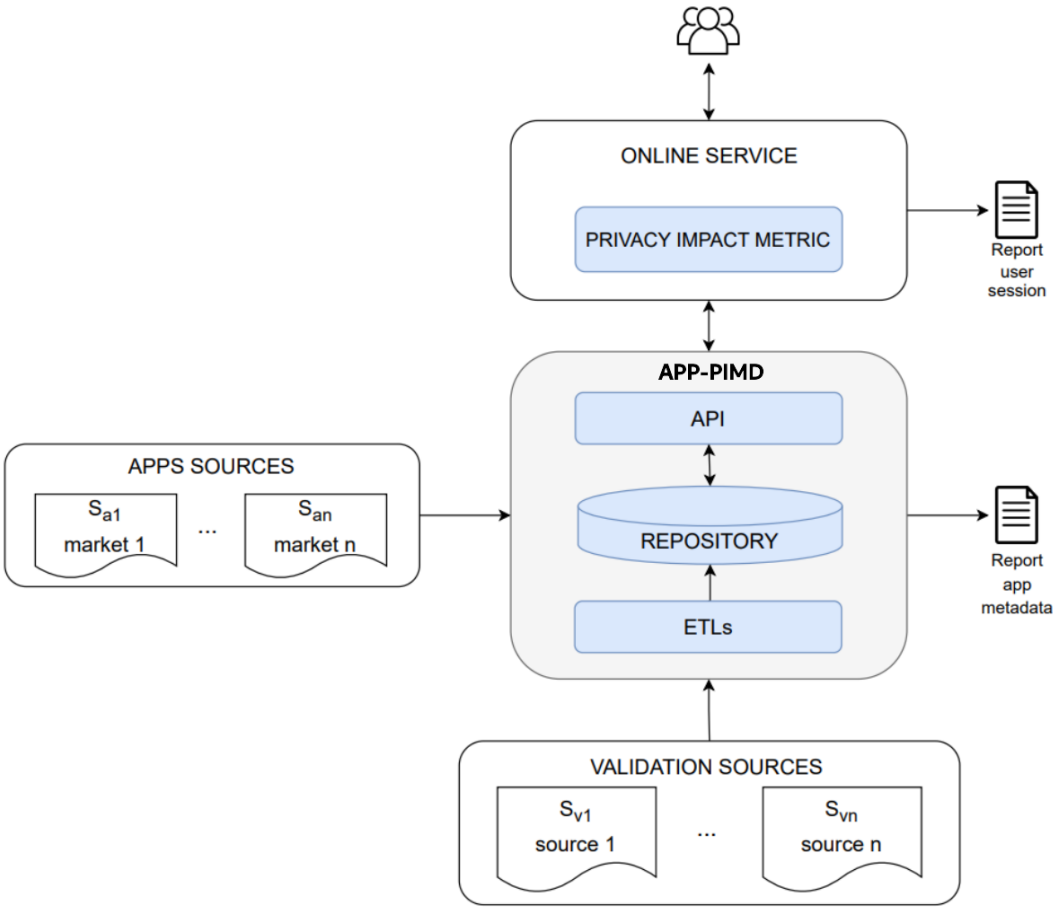

Figura 1.2: Arquitectura de App-PI.

Actualmente todo el ecosistema del proyecto esta siendo explotado por usuarios e investigadores. A este respecto, el servicio *online* APKFalcon, el cual se nutre del repositorio App-PIMD, se ha puesto a prueba en una serie de talleres con usuarios finales

[6]App-PIMD: *App Privacy Impact MetaData*

realizados en los espacios CyL Digital en las fechas 3, 8, 24 de mayo y 6 de junio de 2024.

También se ha puesto a prueba el repositorio por separado con otro tipo de usuario que podría estar interesado en él, los desarrolladores de aplicaciones. De todos estos talleres se ha extraído una gran cantidad de conocimiento sobre el ecosistema y como puede mejorarse.

1.3. Objetivos

El objetivo principal de este Trabajo Fin de Máster es **mejorar el repositorio App-PIMD** por medio de la formalización de su estudio siguiendo una metodología *Big Data*, desde la obtención de los datos, siguiendo por la limpieza y transformación de los mismos, hasta la visualización de la información.

Algunas de las tareas principales a llevar a cabo para conseguir el objetivo principal del trabajo consisten en el análisis, diseño e implementación de los distintos aspectos de mejora del repositorio.

Para la mejora del repositorio App-PIMD se plantean los siguientes subobjetivos o tareas:

- Estudiar y analizar el estado del arte de las distintas arquitecturas *Big Data* para el almacenamiento masivo de información aplicado al caso concreto de App-PIMD.
- Añadir a App-PIMD nuevas fuentes de datos que enriquezcan el volumen de información del repositorio.
- Añadir a App-PIMD nuevos tipos de datos que permitan conocer la popularidad, valoraciones y orígen de las aplicaciones.
- Proporcionar acceso a las nuevas fuentes de datos y tipos de información de forma uniforme con los ya existentes en App-PIMD por medio de técnicas de integración de la información.
- Crear paneles de mandos (*dashboards*) asociados a los contenidos del repositorio que resuman su contenido lo más próximo a tiempo real.
- Ampliación y mejora de la funcionalidad de App-PIMD en base al *feedback* recogido por sus usuarios.

1.4. Organización de la memoria

Este documento se estructura en siete capítulos con los siguientes contenidos:

- **Capítulo 1. Introducción.** Contextualización e introducción al Trabajo Fin de Máster que se va a realizar.
- **Capítulo 2. Metodología.** Selección y determinación de la metodología empleada durante el trabajo.
- **Capítulo 3. Técnicas y herramientas.** Se explican las técnicas, estándares y herramientas utilizadas a lo largo del proyecto.
- **Capítulo 4. Análisis.** Análisis de requisitos de mejora de App-PIMD.
- **Capítulo 5. Diseño.** Diseño detallado de los elementos de mejora de App-PIMD desarrollados en este trabajo.
- **Capítulo 6. Implementación y pruebas.** Se explican los detalles de la implementación y pruebas de la misma.
- **Capítulo 7. Conclusiones y líneas de trabajo futuro.** Reflexión sobre los resultados obtenidos y posibles mejoras o lineas de trabajo futuro.
- **Apéndice A. Guía de instalación de App-PIMD.** Guía de instalación y puesta en marcha del repositorio App-PIMD.
- **Apéndice B. Documentación App-PIMD API v2.** Documentación detallada de la versión 2 de la API de App-PIMD.

Capítulo 2

Metodología

Como se ha visto en la sección 1.3 los objetivos varían desde añadir nuevos tipos de datos hasta la creación de *dashboards*, pasando por la ampliación y mejora de la funcionalidad del repositorio. Es por ello que, debido a la naturaleza heterogénea de los objetivos, se ha decidido optar por un modelo incremental en la realización de este trabajo.

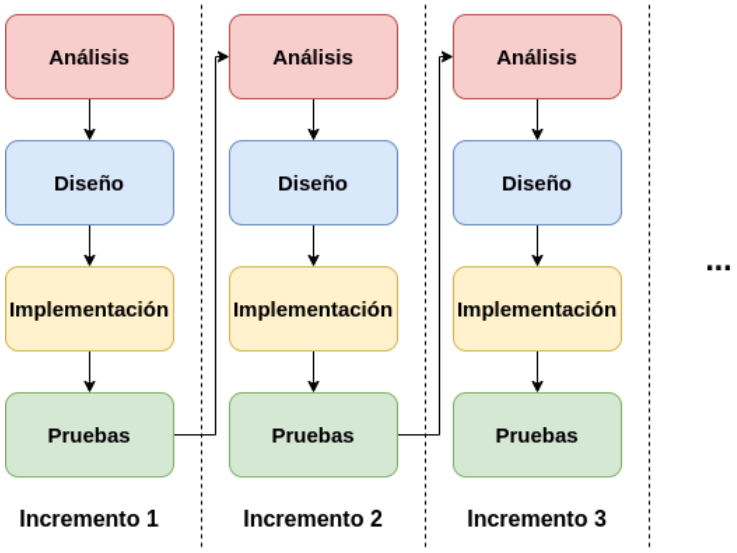

Figura 2.1: Modelo incremental.

Como se puede ver en la figura 2.1, este modelo combina el desarrollo en cascada con enfoques iterativos, donde en cada incremento, haciendo uso del desarrollo en cascada, se añade nueva funcionalidad [14]. Algo muy conveniente para asociar los distintos subobjetivos con cada una de los incrementos del desarrollo del proyecto. Características distintivas que favorecen este modelo:

- **Desarrollo en fases o incrementos:** El análisis, diseño e implementación de los componentes del proyecto, como el *software*, se realiza en distintas etapas. Esto simplifica el desarrollo al abordarse un subproblema pequeño en cada incremento en lugar de la totalidad del proyecto.

- **Rapidez:** Gracias al desarrollo en fases se van alcanzando los subobjetivos de manera progresiva antes del final del proyecto. Esto se traduce en la obtención de resultados rápidamente antes del final del proyecto.

- **Sinergia:** La entrega incremental permite un mejor seguimiento del proyecto en tiempo real e ir ajustando las necesidades del mismo por medio de la retroalimentación. Todo ello se traduce en un proyecto mejor acabado, y una mayor satisfacción final.

- **Gestión de riesgos:** La gestión de los riesgos se facilita al tratar cada incremento de forma independiente, ya que se trata una menor cantidad de riesgos en paralelo. Además, esto permite que los problemas se puedan identificar, y por tanto solucionar, en las primeras fases de cada incremento en lugar de al final del proyecto.

2.1. Metodología específica del proyecto

Puesto que este proyecto de mejora de App-PIMD se trata desde el punto de vista de un proyecto *Big Data*, también es importante especificar la metodología que se llevará a cabo para gestionar cada una de sus distintas etapas.

- **Recopilación de Datos.**

  Las fuentes de datos que se utilizan se escogen en función de su tipo y calidad. Se prefieren tipos de fuentes de datos (APIs[1], páginas web, ...) heterogéneos ya que de este modo se enriquece contenido del repositorio.

  El tipo de dato que se recopila es estructurado por naturaleza, ya que se trata de metadatos bien definidos sobre aplicaciones [15].

  Los procedimientos de recogida de los mismos varían dependiendo del tipo de fuente que se utilice. De este modo, se puede recopilar datos de una fuente de tipo página web por medio de *scraping*, mientras que las de tipo API haciendo uso de peticiones HTTPS[2].

- **Procesamiento de Datos.**

  Para el procesamiento e integración de los datos se utilizan procesos ETL[3]. Estos procesos consisten en extraer los datos de diversas fuentes, transformarlos en un esquema común, y cargarlos en un almacén de datos o *data warehouse*.

  La figura 2.2 muestra un resumen visual de este tipo de procesos[4].

[1] API: *Application Programming Interface*
[2] HTTPS: *Hypertext Transfer Protocol Secure*
[3] ETL: *Extract, Transform, Load*
[4] `https://www.linkedin.com/pulse/tu-proceso-etl-est%C3%A1-optimizado-rossibel-toro/`

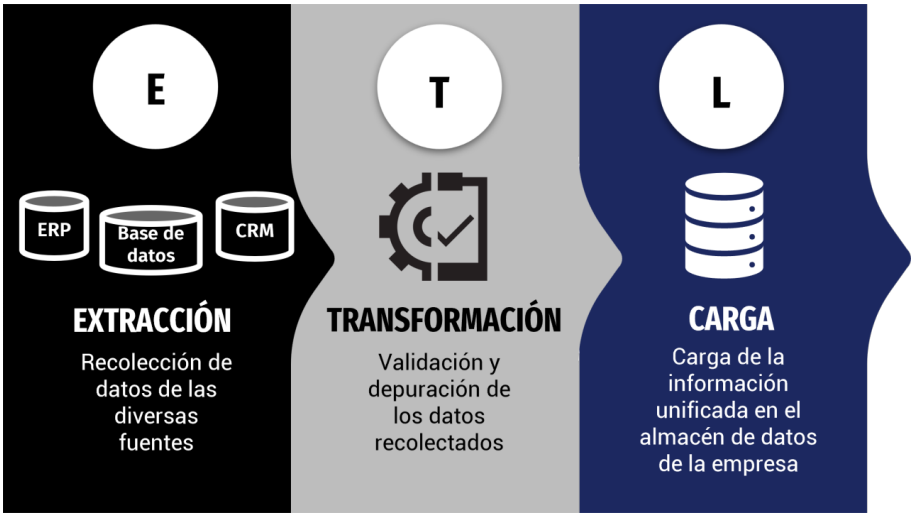

Figura 2.2: Proceso ETL.

- **Almacenamiento y Gestión de Datos.**

  Debido a la naturaleza estructurada de los datos, la infraestructura que se utiliza para su almacenamiento son bases de datos relacionales. Este tipo de almacenamiento permite el uso de SQL[5] para la gestión y manipulación de los datos.

  Un lenguaje que permite realizar operaciones avanzadas de búsqueda y análisis de datos mediante su capacidad de combinar, filtrar, agrupar y ordenar grandes volúmenes de información rápidamente.

- **Visualización de Datos.**

  Para realizar la visualización de los datos se utilizan *dashboards*, ya que se han convertido en el estándar en proyectos *Big Data* debido a su capacidad para consolidar y presentar grandes volúmenes de datos de forma clara y efectiva. Para que resulten así, es necesario abordar las siguientes cuestiones:

  - **Objetivos de la Visualización.** Establecer los objetivos del *dashboard*, como qué preguntas se quieren responder con los datos. Esto permite seleccionar las herramientas y tipos de gráficos más adecuados para cumplir los objetivos.
  - **Identificación de los Usuarios.** Establecer y analizar el tipo de usuario del *dashboard*, sus necesidades, habilidades y objetivos. De este modo, se puede ajustar tanto la interfaz como los tipos de visualizaciones en términos de complejidad y diseño a las necesidades de los usuarios. Esta adaptación asegura que el *dashboard* no solo cumpla su propósito, sino que lo haga de manera óptima y precisa.

[5]SQL: *Structured Query Language*

- **Diseño de la Visualización.** Diseñar el número de páginas o paneles del *dashboard* y la distribución de cada uno de ellos haciendo uso de métodos de agrupación y patrones de diseño como el "Z" [16]. Escoger qué gráficos se ajustan mejor al tipo de dato que se esta representando, uso correcto de los colores y tipografía adecuada. Incluir interactividad para permitir aplicar filtros y parámetros que permitan personalizar los *dashboard* a los usuarios. Teniendo en cuanta todo este tipo de decisiones, nos aseguramos de que los datos serán fácilmente accesibles, comprensibles y visualmente atractivos una vez se implementen las visualizaciones.
- **Guía de Usuario.** Crear una guía de usuario bien elaborada que proporcione a los usuarios instrucciones claras sobre cómo navegar y utilizar las diversas funciones del *dashboard*, lo que minimiza la curva de aprendizaje y reduce el riesgo de errores por parte de los usuarios. Además, ha de explicar el propósito y la interpretación de cada visualización, lo que garantiza que los usuarios comprendan correctamente los datos presentados.

- **Consideraciones Éticas y Legales.**

  En todos los proyectos en que se recogen y almacenan datos es muy importante seguir unos principios éticos que se basen en minimizar la recolección de datos. Es decir, recopilar solo los datos necesarios, anonimizarlos para proteger la identidad de los usuarios, e implementar medidas de seguridad robustas para protegerlos contra accesos no autorizados y violaciones de seguridad. Además, se ha de llevar a cabo estudios del cumplimiento normativo y legal sobre los datos que se utilizan.

  En el proyecto App-PIMD no se recogen datos de tipo personal en ningún caso. Es por ello que solo tenemos que analizar el cumplimiento legal desde el punto de vista de la recolección y difusión de metadatos de aplicaciones Android que se encuentran públicas en diversos *markets*. Cuando un desarrollador Android publica su app en Google Play se acoge al "Acuerdo de Distribución para Desarrolladores de Google Play (*Google Play Developer Distribution Agreement*)". Según este, se concede a los usuarios una licencia no exclusiva, mundial y perpetua para realizar, instalar y usar la aplicación y cualquier contenido, material y servicios relacionados con tu aplicación a través de Google Play, conforme a los términos del acuerdo de licencia de usuario final.

  En España por norma general no se permite el descompilado de programas, como son las aplicaciones móviles, ni la ingeniería inversa a excepción de lo dispuesto en el artículo 100 de la Ley de Propiedad Intelectual [17]. Por lo que en el presente trabajo se hará uso de análisis estático ya que nos permite obtener todos los metadatos que necesitamos y si esta permitido legalmente.

Capítulo 3

Técnicas y Herramientas

3.1. Técnicas

En esta sección se enumeran y desarrollan las técnicas, procedimientos y estándares utilizados durante el desarrollo del trabajo.

3.1.1. Técnicas de Almacenamiento de Datos en entornos *Big Data*

En la actualidad, el almacenamiento en entornos *Big Data* ha cobrado gran importancia en la gestión de grandes volúmenes de datos provenientes de diversas fuentes, como redes sociales, sensores IoT[1] y sistemas empresariales. Además, estos datos suelen crecer en volumen, velocidad y variedad, lo que requiere la evolución de las técnicas de almacenamiento para garantizar la eficiencia, el acceso y la seguridad de los datos.

A continuación, se describen algunas de las principales técnicas de almacenamiento *Big Data* [18], [19]:

- **Almacenamiento en Bases de Datos NoSQL**

  Las bases de datos NoSQL están diseñadas para manejar grandes volúmenes de datos no estructurados o semi-estructurados. Entre sus características principales se encuentran su modelo de datos flexible, a diferencia del modelo fijo de las bases de datos relacionales, y su capacidad para ser escalables horizontalmente. Esto permite manejar mayores volúmenes de datos añadiendo más instancias en lugar de aumentar la capacidad de una sola.

  Sin embargo, entre sus desventajas se incluye el uso de consistencia eventual, es decir, sacrifican la consistencia inmediata de los datos para lograr una mayor disponibilidad y particionamiento. Además, no existen estándares unificados como en el caso de las bases de datos estructuradas (SQL), lo que puede llevar a una mayor complejidad en el aprendizaje y la integración de este tipo de almacén de datos.

[1] IoT: *Internet Of Things*

- **Sistemas de Archivos Distribuidos**

  Cuando los datos a almacenar carecen de estructura y son de tipos muy variados, como imágenes, vídeos, archivos de texto u otros tipos de archivo, la mejor solución es utilizar sistemas de archivos distribuidos.

  En este tipo de almacenamiento, los archivos se encuentran en múltiples ubicaciones físicas de manera transparente para el usuario. Esto permite una gran escalabilidad horizontal del sistema añadiendo nuevas instancias, alta disponibilidad ante caídas de servidores concretos, recuperación ante desastres si un servidor falla, acceso concurrente y alta eficiencia.

  Sin embargo, estos sistemas de almacenamiento tienen una serie de desafíos, como la gestión de la consistencia entre todos los nodos del sistema y la necesidad de asegurar que el rendimiento no se vea afectado por las tareas de gestión y replicación, que pueden llegar a saturar la red.

- **Almacenamiento en *Data Warehouse***

  Cuando los datos a almacenar tienen una estructura clara y provienen de múltiples fuentes que deben integrarse en un único lugar para realizar análisis y consultas, la mejor solución es utilizar almacenamiento en *data warehouse*. Entre sus características principales se encuentran el uso de un modelo de datos relacional, es decir, los datos se almacenan en tablas organizadas que pueden tener relaciones entre sí. Esto permite utilizar SQL, un lenguaje estándar para consultas, actualización y gestión de bases de datos relacionales. Para resolver el problema de la integración de datos de diversas fuentes, se suelen emplear procesos ETL.

  Entre las ventajas de este tipo de almacén se encuentra la consistencia e integridad de los datos, gracias a las propiedades ACID[2] de los sistemas SQL, así como la capacidad de realizar consultas complejas y análisis sobre grandes volúmenes de datos de manera eficiente. También se destaca la interoperabilidad que proporciona SQL, gracias a ser un estándar, con herramientas de análisis e inteligencia de negocio.

  Sin embargo, los puntos negativos suelen ser en términos de escalabilidad, ya que el sistema permite principalmente escalabilidad vertical para mantener las propiedades ACID. Esto suele conllevar un coste más elevado cuando se trata de grandes volúmenes de datos en comparación con otras soluciones. Además, la complejidad en el diseño de los procesos ETL puede ser un desafío para asegurar que los datos se integran y transforman correctamente.

[2] ACID: Atomicidad, Consistencia, Aislamiento y Durabilidad

Dadas las características de este proyecto, se utiliza un sistema de almacenamiento basado en *data warehouse*, ya que los datos a almacenar son metadatos estructurados provenientes de diversas fuentes. Se requiere que estos datos puedan ser consultados y analizados de manera compleja, además de proporcionar una gran interoperabilidad para facilitar su uso por terceros. Asimismo, dado que los metadatos son textuales y su volumen es relativamente pequeño, las limitaciones de escalabilidad en este tipo de almacén no son un inconveniente significativo. Por otro lado, los metadatos están bien documentados lo que facilita el diseño de los procesos ETL [15], [20], [21].

3.1.2. Proceso ETL

Como se mencionó anteriormente en la sección 2.1, los procesos ETL constituyen una metodología empleada en la gestión de datos para integrar información de diversas fuentes, transformarla según las necesidades de análisis o almacenamiento, y cargarla en un sistema destino, como un *data warehouse* [22]. Esta metodología se compone de tres etapas: extracción de la información, transformación y carga. A continuación, se detallará cada uno de las etapas, así como los distintos estándares y técnicas utilizados en cada fase de los procesos ETL.

Etapa de Extracción

En esta etapa se aborda el problema de la extracción de información de diversas fuentes origen, que probablemente sean heterogéneas. Esto requiere la aplicación de técnicas específicas de extracción de información. Las siguientes técnicas son las que se utilizan en el proyecto:

- **Peticiones a APIs por medio del protocolo HTTPS**

  HTTPS es un protocolo de comunicación web seguro basado en mensajes de texto, cuya estructura se está definida en los RFCs[3] 2616 [23], 7230 [24], 7231 [25] y 5246 [26]. La figura 3.1 muestra la estructura básica de los mensajes que intercambian entre el cliente y el servidor durante una comunicación.

  Los elementos más importantes que se pueden observar en la figura 3.1 son:

  - En la petición del cliente, el método de la petición (en la figura, “GET”, que indica que se quiere obtener información) y la dirección del recurso solicitado (en la figura, “/hello.txt”). Con estos dos parámetros, podemos solicitar información a APIs.
  - En la respuesta del servidor, el cuerpo del mensaje es la parte más importante, ya que contiene la información solicitada.

[3]RFC: *Request For Comments*

```
Client request:

  GET /hello.txt HTTP/1.1
  User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
  Host: www.example.com
  Accept-Language: en, mi


Server response:

  HTTP/1.1 200 OK
  Date: Mon, 27 Jul 2009 12:28:53 GMT
  Server: Apache
  Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
  ETag: "34aa387-d-1568eb00"
  Accept-Ranges: bytes
  Content-Length: 51
  Vary: Accept-Encoding
  Content-Type: text/plain

  Hello World! My payload includes a trailing CRLF.
```

Figura 3.1: Formato de los mensajes HTTP [24].

- ***Web scraping***

  El *web scraping* es una técnica utilizada para extraer información de sitios web de manera automatizada. Consiste en utilizar *scripts* que navegan por las páginas web e identifican y extraen datos específicos, como texto, imágenes o enlaces.

Etapa de Transformación

En esta etapa se aborda el problema de transformar los datos según las necesidades de análisis o almacenamiento del proyecto. Dado que las fuentes de datos son heterogéneas, también es necesario convertir los datos al esquema común utilizado en el almacén del proyecto. La técnica aplicada para la transformación de la información utilizada en el proyecto es principalmente análisis estático de código.

Esta técnica de inspección de *software* examina el archivo de una aplicación sin ejecutarla, con el objetivo de identificar errores, vulnerabilidades y, en el caso de este TFM, metadatos. El proceso se realiza mediante herramientas especializadas que analizan el código para detectar patrones. En este trabajo, se analizan patrones XML[4], ya que son los que definen los metadatos dentro de las aplicaciones Android (ver figura 3.2) [15], [20], [21].

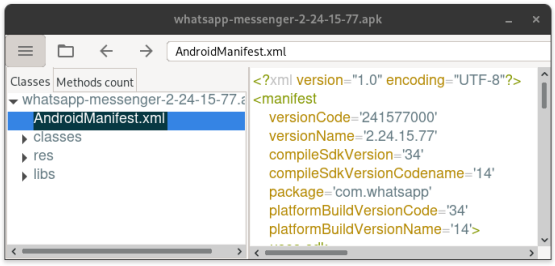

Figura 3.2: Archivo de manifiesto de la app WhatsApp.

[4]XML: *Extensible Markup Language*

Etapa de Carga

En esta etapa se aborda el problema de la carga de los datos en el *data warehouse*. Este proceso es crucial para garantizar que los datos estén disponibles y sean accesibles para su análisis y uso en la toma de decisiones. Para ello, se utiliza el patrón de diseño *software Data Mapper* [27]. Este patrón relaciona los datos entre una base de datos y los objetos de dominio de una aplicación, lo que permite subir de manera fácil y estandarizada todos los datos extraídos al repositorio.

3.1.3. Técnicas de Visualización de Datos

Las técnicas de visualización de datos utilizadas se extraen del libro *Information Dashboard Design: Displaying Data for At-a-glance Monitoring* [16] como se mencionó en la sección 2.1 de metodología específica, y del blog *Dashboards. ¿Cómo diseñar dashboards óptimos?* [28].

La técnica más importante en cuanto a la organización del contenido es aprovechar los espacios de alta relevancia visual aplicando el patrón Z, según el cuál se rigen estos. En la figura 3.3 se muestran los espacios de mayor relevancia. Junto con esto, es muy importante también el uso de espacios en blanco para separar las zonas del *dashboard* y evitar el uso del *scroll*, de manera que toda la información se encuentre disponible de un vistazo.

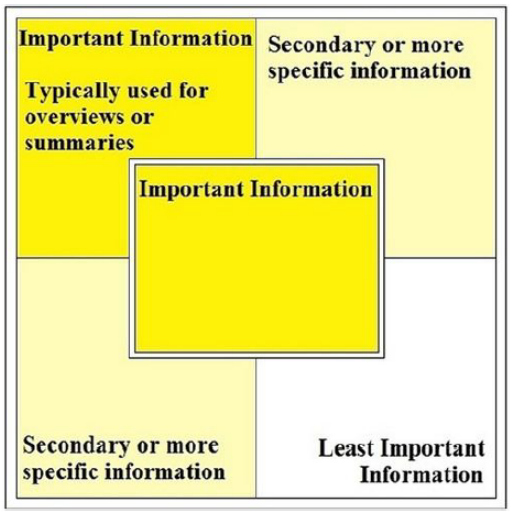

Figura 3.3: Zonas de mayor relevancia visual (Patrón Z) [28].

En cuanto al diseño de los gráficos, es muy importante eliminar cualquier representación innecesaria, manteniendo así una interfaz limpia y clara. Se deben utilizar colores adecuados que permitan separar las distintas secciones y focalizar la atención en los elementos importantes, además de hacer uso de visualizaciones apropiadas para los datos que se están representando.

3.1.4. Técnicas de Integración

Como se ha visto en apartados anteriores, en este trabajo se integra información de diversas fuentes mediante el uso de procesos ETL y un *data warehouse* para almacenar todos los datos. No obstante, dado que la finalidad de un repositorio, además de almacenar información, es ponerla a disposición de sus usuarios, también se emplean técnicas para integrar y unificar las formas de acceso a los distintos servicios del repositorio.

Por un lado, se emplea el estándar OpenAPI para diseñar y documentar la API de acceso a App-PIMD, facilitando así la interoperabilidad e integración con otros servicios. Para la autenticación de usuarios, se utiliza el estándar OAuth 2.0, como se especifica en OpenAPI [29].

Por otro lado, es necesario uniformar el acceso al *dashboard* con el de la API del repositorio. Para ello, se utiliza un *proxy* inverso, que actúa como un servidor intermediario entre los clientes y los servicios *backend*, en este caso, la API de acceso al repositorio y el *dashboard*. Este *proxy* gestiona y redirige las peticiones a los distintos servicios *backend* desde un único punto de entrada [30].

La figura 3.4 muestra la arquitectura que se ha utilizado para uniformar el acceso a los servicios de App-PIMD: el repositorio y el *dashboard*.

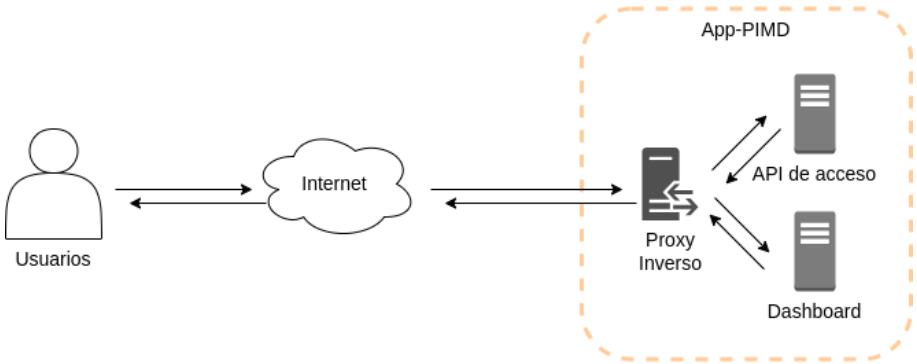

Figura 3.4: Acceso uniforme a los servicios de App-PIMD.

3.2. Herramientas

En esta sección se enumeran las herramientas *software* utilizadas durante el desarrollo del proyecto.

3.2.1. Entorno de Desarrollo

VirtualBox

Para alojar todo el repositorio y sus servicios en el entorno de desarrollo, he decidido utilizar la virtualización. En concreto, se usa VirtualBox, un software de virtualización de

código abierto actualmente desarrollado por Oracle que permite a los usuarios ejecutar múltiples sistemas operativos en una sola máquina física de manera simultánea. Esto se logra creando máquinas virtuales que simulan el *hardware* completo de un ordenador.

Esta herramienta resulta valiosa para el desarrollo y las pruebas de software, ya que nos permite probar las aplicaciones en múltiples versiones y replicar entornos de producción en un entorno seguro y controlado. Entre sus características avanzadas se encuentran las instantáneas, que permiten guardar el estado de una máquina virtual en un momento dado y volver a él posteriormente, lo cual es muy útil a la hora de desarrollar.

Visual Studio Code

Como editor de código fuente utilizo Visual Studio Code (VS Code), desarrollado por Microsoft y diseñado para proporcionar una experiencia de desarrollo ligera pero poderosa. Entre sus ventajas destacan su extensibilidad y personalización mediante una amplia variedad de extensiones disponibles en su *marketplace*, que permiten agregar soporte para nuevos lenguajes de programación, herramientas de depuración y servicios de desarrollo en la nube. Las extensiones que utilizo principalmente son:

- **Python Extension Pack:** Permite el manejo de entornos virtuales de Python y la depuración de código.

- **Autodocstrings y Autopep8:** Facilitan la creación de documentación interna y mantener el estándar de código PEP8, respectivamente.

Además, VS Code es multiplataforma, funcionando en Windows, macOS y Linux, lo que lo hace accesible para una gran variedad de desarrolladores. Otras ventajas incluyen su integración con sistemas de control de versiones como Git, un terminal integrado, y una interfaz de usuario intuitiva y altamente configurable. Todo esto, sumado a su rendimiento eficiente es lo que me hace decantarme por este editor.

GitHub

Para llevar un control del código, utilizo GitHub, una plataforma de desarrollo colaborativo y alojamiento de código fuente basada en Git, un sistema de control de versiones distribuido. GitHub permite almacenar, gestionar y compartir proyectos de manera eficiente. A través de repositorios, los usuarios podemos organizar el código, colaborar con otros desarrolladores y mantener un historial detallado de los cambios realizados en los proyectos.

3.2.2. *Stack* Tecnológico

Python *enviroments*

Como lenguaje de programación base y núcleo de la lógica de los servicios, utilizo Python con entornos virtuales para llevar un control de las versiones de las librerías utilizadas en el proyecto. Esto facilita la gestión de dependencias entre ellas. Además, Python es un lenguaje muy versátil y eficiente. Finalmente, lo escojo también por mi experiencia previa con él y su fácil aprendizaje.

Selenium

Selenium es una librería de Python, disponible también en otros lenguajes de programación, que se utiliza para la automatización de navegadores web. Permite controlar navegadores de manera programática, simulando acciones humanas como hacer clic en botones, rellenar formularios, navegar entre páginas y extraer información de sitios web. Por ello, la utilizo para realizar *scraping* a páginas web y extraer información en la etapa *extract* de los procesos ETL.

Librerías de transformación de los datos

Para transformar los datos a los esquemas finales, utilizo dos librerías:

- **Androguard:** Es una herramienta para la ingeniería inversa y el análisis de aplicaciones Android, desarrollada en Python. Ampliamente utilizada por investigadores en seguridad y analistas, permite descompilar, analizar y manipular los archivos de las aplicaciones. Permite realizar análisis estático de aplicaciones Android, proporcionando una variedad de funcionalidades útiles para examinar el código, los recursos y el comportamiento de las aplicaciones.

- **Lxml:** Es una biblioteca eficiente para el procesamiento de datos XML. Ofrece una interfaz para trabajar con estos formatos de datos, y es conocida por su capacidad para manejar grandes volúmenes de información de manera rápida y precisa. Lxml se basa en las bibliotecas de C libxml2 y libxslt, lo que le permite proporcionar una amplia gama de funcionalidades con una ejecución rápida.

MySQL

Para el almacenamiento de los datos, se ha utilizado el sistema gestor de bases de datos MySQL junto con el conector para Python mysql-connector. Este conector permite ejecutar consultas seguras contra ataques de inyección SQL (SQLi) mediante la validación de los parámetros de las consultas.

Grafana

Para la visualización de datos y la creación de *dashboards*, utilizo Grafana, una plataforma de código abierto para el análisis y la visualización de datos en tiempo real. Es especialmente útil para crear *dashboards* interactivos y personalizados. Su principal ventaja radica en su capacidad para integrarse con una variedad de fuentes de datos, como bases de datos, servicios de monitoreo y sistemas de *logs*, lo que permite centralizar información de diferentes sistemas. Grafana facilita la creación de gráficos, tablas y alertas dinámicas a través de una interfaz intuitiva, lo que ayuda a identificar tendencias y tomar decisiones basadas en datos. Además, su flexibilidad y extensibilidad mediante *plugins* permiten adaptar los *dashboards* a necesidades específicas, mejorando así la comprensión de los datos.

Traefik

Traefik es un *proxy* inverso y balanceador de carga moderno, diseñado para facilitar la gestión del tráfico en aplicaciones distribuidas y contenedorizadas. Ofrece administración centralizada del tráfico, con características como enrutamiento basado en reglas, balanceo de carga eficiente y soporte para HTTPS con certificados automatizados. Su interfaz de usuario intuitiva y su capacidad para actualizarse dinámicamente sin necesidad de reiniciar los servicios lo convierten en una solución ideal para entornos de alta disponibilidad y escalabilidad.

Capítulo 4

Análisis

En este capítulo se analizan los requisitos necesarios para alcanzar el objetivo y los subobjetivos planteados en la sección 1.3. De este modo, se establece una especie de "contrato" a seguir durante el desarrollo del trabajo, en el que se refleja su alcance y que sirve como punto de partida para su ejecución. Para una mejor comprensión de los capítulos de Análisis y Diseño de este trabajo, se recomienda leer los capítulos correspondientes del TFG en el que se desarrolló inicialmente App-PIMD [1].

Dado que los objetivos son diversos, se ha decidido separar el análisis de requisitos de este trabajo en tres grandes bloques: análisis de requisitos para la ampliación de fuentes y tipos de datos en App-PIMD, análisis de requisitos para la mejora de la funcionalidad de la API de App-PIMD, y análisis de requisitos para los *dashboards* de App-PIMD.

4.1. Análisis de requisitos para la ampliación de fuentes y tipos de datos en App-PIMD

Como es bien sabido, las fuentes de datos de un repositorio son fundamentales para garantizar la calidad de la información almacenada, lo que, a su vez, impacta directamente en la validez y calidad de cualquier análisis o decisión basados en estos datos. App-PIMD dispone de un total de cinco fuentes de datos: APKPure, Evozi, Apkmonk, Apkfollow y Androzoo. Todas estas fuentes ofrecen aplicaciones que, en su mayoría, se encuentran en la tienda oficial de Android, Play Store. Sin embargo, su contenido es limitado en términos de heterogeneidad, ya que solo disponen de los archivos de las aplicaciones en su mayoría de software propietario, sin proporcionar información adicional sobre ellas más que la contenida en su propio archivo.

Dadas las limitaciones de las fuentes de información actuales de App-PIMD expuestas en el párrafo anterior, se propone ampliarlas de la siguiente manera: en primer lugar, añadir una fuente de información adicional que disponga de aplicaciones de código abierto, mejorando así la heterogeneidad de las aplicaciones almacenadas en el repositorio;

en segundo lugar, incorporar una fuente de información que proporcione información adicional, como la popularidad y valoraciones de las aplicaciones, información sobre el desarrollador, y la política de privacidad correspondiente.

4.1.1. Requisitos

- Añadir una fuente de información que sea de libre acceso.
- La fuente de información ha de disponer de un gran catálogo de aplicaciones. Al menos 1000 aplicaciones.
- La fuente de información ha de permitir la descarga de aplicaciones en formato fichero.
- Añadir una segunda fuente de información que disponga de información sobre al menos la valoración, desarrollador y política de privacidad de las *apps*.
- La fuente de información ha de disponer de información sobre versiones antiguas de las aplicaciones.

4.2. Análisis de requisitos para la mejora de la funcionalidad de la API de App-PIMD

Como se mencionó en la sección 1.2.1, el repositorio ha sido probado en múltiples ocasiones por usuarios generales, cuyo comportamiento básicamente se limita a realizar consultas sobre metadatos de aplicaciones y solicitar la carga de nuevos al repositorio. Gracias a todas estas pruebas, se ha obtenido conocimiento sobre la corrección del funcionamiento del repositorio y sus limitaciones actuales, lo que ha permitido identificar las áreas de mejora en su funcionalidad que se abordarán en este trabajo.

4.2.1. Requisitos

En esta sección se especifican los requisitos que el sistema ha de cumplir para considerarse exitoso. Es importante tener en cuenta que estos requisitos se añaden de forma incremental a los elicitados durante el desarrollo inicial del repositorio en el TFG [1].

Requisitos funcionales

- El sistema ha de permitir a los usuarios generales la descarga de una lista de versiones almacenadas para una aplicación concreta a través de su nombre de paquete.

- El sistema ha de permitir a los usuarios generales hagan peticiones de carga de aplicaciones concretas al repositorio a través de su archivo.
- El sistema ha de permitir a los usuarios generales la descarga de los metadatos de una aplicación a través de su nombre.
- El sistema ha de permitir a los usuarios generales la descarga de los metadatos detallados de una aplicación a través de su hash.

Requisitos no funcionales

- El sistema deberá mantener retrocompatibilidad.
- El sistema deberá aportar la nueva funcionalidad de modo uniforme a la ya presente.

4.2.2. Casos de uso

En esta sección se identifican los casos de uso que el sistema debe implementar para satisfacer los requisitos. En la figura 4.1 se observa que, respecto al análisis del desarrollo inicial [1], solo se ha añadido el caso de uso "Ver las versiones almacenadas de una *app*", ya que el resto de los requisitos pueden cumplirse mediante la ampliación y modificación de los casos de uso existentes.

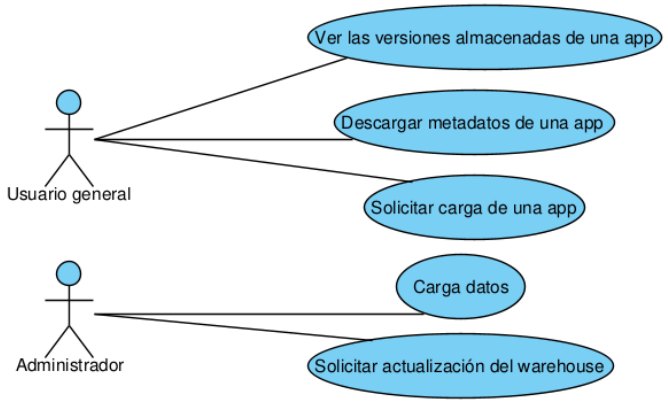

Figura 4.1: Diagrama de casos de uso.

A continuación, se detallan únicamente el nuevo caso de uso y aquellos que han sufrido cambios. Para facilitar la visualización de las modificaciones en los diagramas respecto a los del desarrollo inicial, se utiliza el color verde para destacar los cambios.

UC-1: Ver las versiones almacenadas de una *app*

En la tabla 4.1 se encuentra la especificación del caso de uso de ver las versiones almacenadas de una *app* y en la figura 4.2 se encuentra el diagrama de actividades asociado a ese caso de uso.

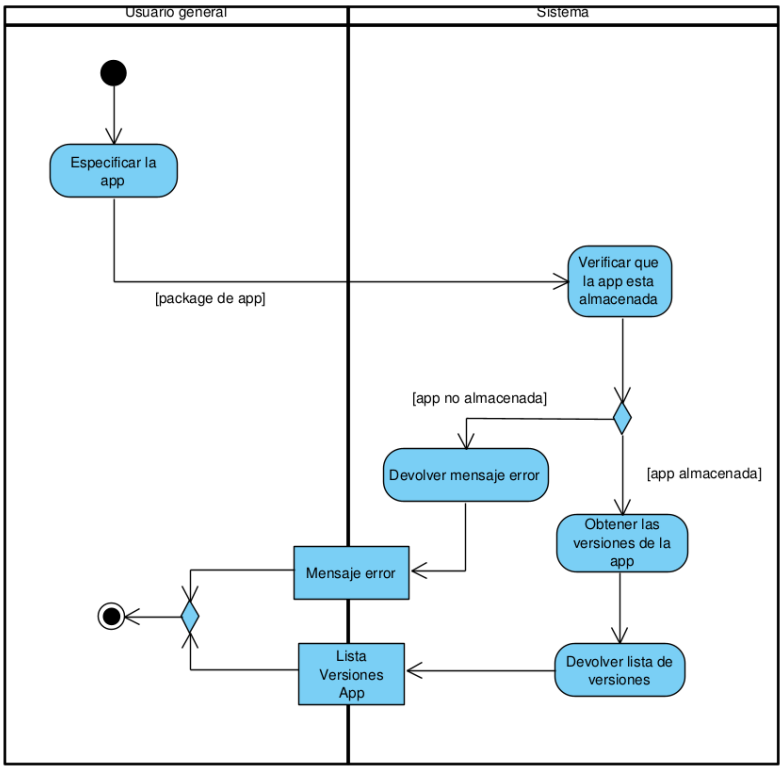

Figura 4.2: Diagrama de actividades del UC-1.

UC-1	**Ver las versiones almacenadas de una *app***
Descripción	Ver las versiones almacenadas de una *app* en el repositorio.
Secuencia normal	1. El usuario general introduce el *package* de la *app* que desea. 2. El sistema comprueba que la *app* se encuentra almacenada. 3. El sistema obtiene la lista de versiones almacenadas. 4. El sistema devuelve la lista de versiones.
Flujos alternativos	2a-1. La *app* no se encuentra en el sistema. 2a-2. El sistema devuelve un error indicando que la *app* solicitada no se encuentra en el sistema. 2a-3. El caso de uso queda sin efecto.
Postcondición	El usuario general ha obtenido la lista de versiones almacenadas de la *app*.

Tabla 4.1: Caso de uso: Ver las versiones almacenadas de una *app*.

UC-2: Descargar metadatos de una *app*

En la tabla 4.2 se encuentra la especificación del caso de uso de descargar los metadatos de una *app* y en la figura 4.3 se encuentra el diagrama de actividades asociado a ese caso de uso.

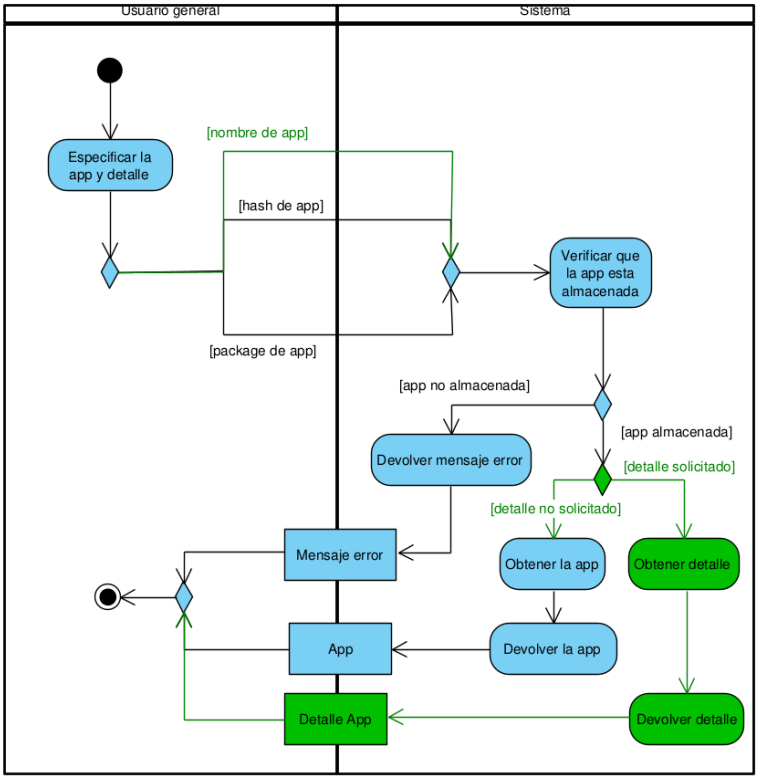

Figura 4.3: Diagrama de actividades del UC-2.

UC-2	**Descargar metadatos de una *app***
Descripción	Descarga los metadatos de una *app* del repositorio.
Secuencia normal	1. El usuario general introduce el *package*, nombre o hash de la *app* que desea. 2. El sistema comprueba que la *app* se encuentra almacenada. 3. El sistema comprueba si se desea el detalle. 4. El sistema obtiene el detalle de la *app* almacenada. 5. El sistema devuelve el detalle de la *app*.
Flujos alternativos	2a-1. La *app* no se encuentra en el sistema. 2a-2. El sistema devuelve un error indicando que la *app* solicitada no se encuentra en el sistema. 2a-3. El caso de uso queda sin efecto. 3a-1. No se desea el detalle. 3a-2. El sistema obtiene los metadatos de la *app*. 3a-3. El sistema devuelve los metadatos de la *app*.
Postcondición	El usuario general ha obtenido los metadatos almacenados de la *app*.

Tabla 4.2: Caso de uso: Descargar metadatos de una *app*.

UC-3: Solicitar carga de una *app*

En la tabla 4.3 se encuentra la especificación del caso de uso de solicitar carga de una *app* y en la figura 4.4 se encuentra el diagrama de actividades asociado a ese caso de uso.

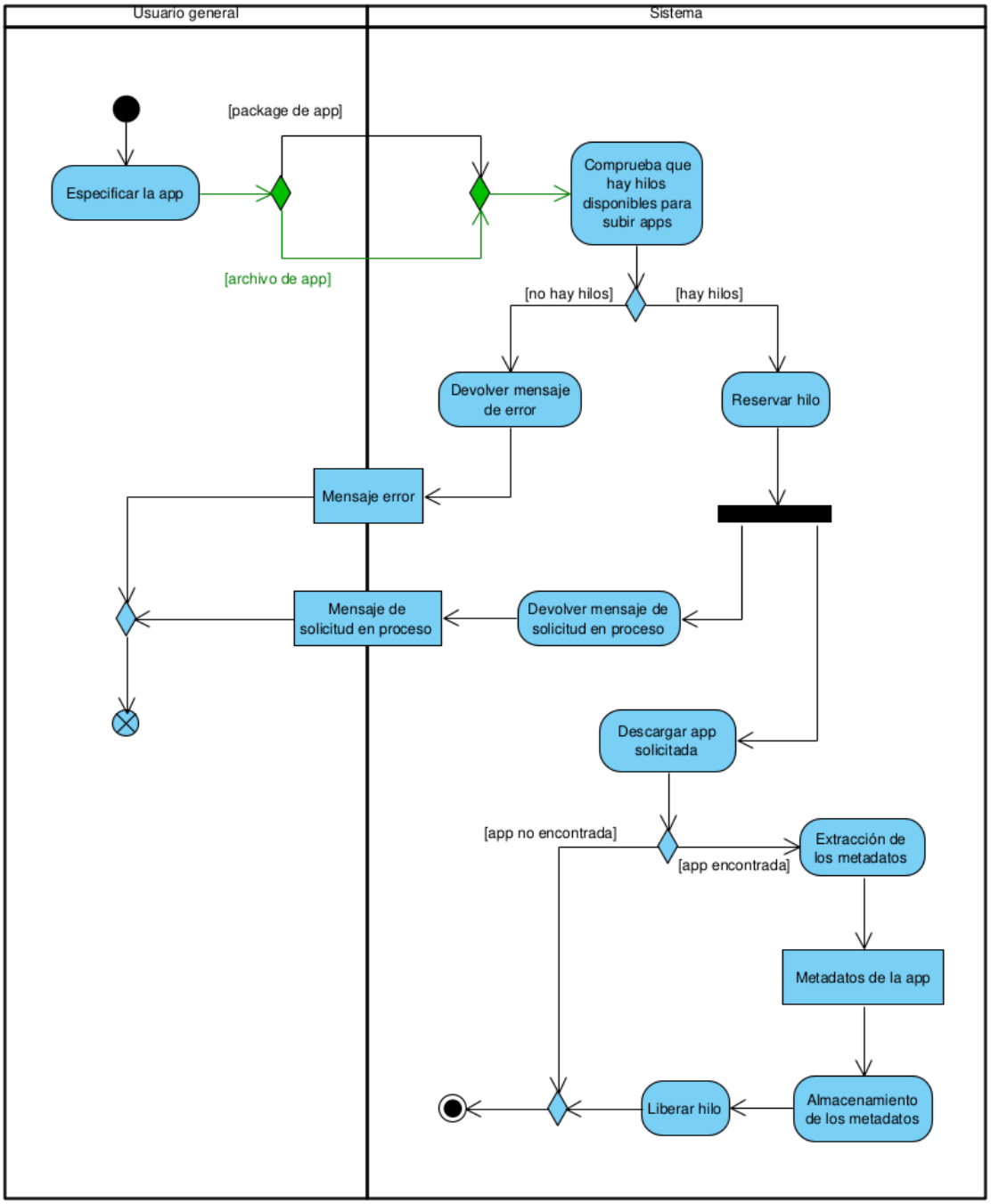

Figura 4.4: Diagrama de actividades del UC-3.

UC-3	**Solicitar carga de una *app***
Descripción	Solicita la carga de una *app* al *warehouse*.
Secuencia normal	1. El usuario general introduce el *package* o archivo de la *app* que desea cargar. 2. El sistema comprueba que hay hilos disponibles para subir *apps*. 3. El sistema reserva un hilo para subir la *app*. 4. El sistema indica al usuario general que la petición se esta procesando. 5. El sistema descarga la *app* solicitada. 6. El sistema extrae la información de la *app* descargada. 7. El sistema almacena la información extraída. 9. El sistema libera el hilo reservado.
Flujos alternativos	2a-1. Todos los hilos de subida se encuentran ocupados. 2a-2. El sistema informa al usuario general de que el sistema se encuentra ocupado. 2a-3. El caso de uso queda sin efecto. 5a-1. No se encuentra la *app* solicitada en las fuentes de datos. 5a-2. El sistema libera el hilo reservado. 5a-3. El caso de uso queda sin efecto.
Postcondición	La *app* solicitada queda almacenada en el *warehouse*.

Tabla 4.3: Caso de uso: Solicitar carga de *app*.

4.3. Análisis de requisitos para los *dashboards* de App-PIMD

En esta sección se realiza el análisis de requisitos de los *dashboards* de App-PIMD. Es fundamental definir claramente los objetivos de estos, así como los datos que deben visualizarse, y los indicadores y métricas clave necesarios para crear *dashboards* útiles que apoyen la toma de decisiones.

El objetivo principal de los *dashboards* es la trasmisión de conocimiento. Basándonos en esto, se requiere que esta transmisión se divida en tres áreas distintas: por un lado, la trasmisión de conocimiento sobre AOSP; por otro, la trasmisión de conocimiento de aplicaciones; y, finalmente, la trasmisión de conocimiento sobre el impacto de las aplicaciones en la privacidad de sus usuarios. En base a esta distinción en áreas se elicitarán los requisitos de cada uno de los *dashboards* o paneles. Se ha de poder navegar entre todos los paneles.

4.3.1. Requisitos sobre el panel de AOSP

Con este panel de mandos se ha de ser capaz de responder las preguntas que abarcan los siguientes requisitos:

- El panel de mandos ha de mostrar el nombre de todos los grupos de permisos definidos en el sistema operativo Android.
- El panel de mandos ha de mostrar el número total de grupos de permisos definidos en el sistema operativo Android.
- El panel de mandos ha de mostrar el número total de permisos de cada grupo definido en el sistema operativo Android.
- El panel de mandos ha de mostrar el nivel de SDK[1] en que se añadió cada permiso al sistema operativo Android.
- El panel de mandos ha de mostrar la distribución del número de permisos añadido en función del nivel de SDK.

4.3.2. Requisitos sobre el panel de aplicaciones

Con este panel de mandos se ha de ser capaz de responder las preguntas que abarcan los siguientes requisitos:

- El panel de mandos ha de permitir interactividad para seleccionar una *app* concreta según el nombre de paquete que la identifica y su versión.
- Requisitos sobre información general:
    - El panel de mandos ha de mostrar el nombre de la aplicación seleccionada.
    - El panel de mandos ha de mostrar la categoría de la aplicación seleccionada.
    - El panel de mandos ha de mostrar el nombre de paquete de la aplicación seleccionada.
    - El panel de mandos ha de mostrar la fecha de publicación de la aplicación seleccionada.
    - El panel de mandos ha de mostrar la versión de la aplicación seleccionada.
    - El panel de mandos ha de mostrar el tamaño de la aplicación seleccionada.
    - El panel de mandos ha de mostrar la versión mínima de SDK Android necesaria para la ejecución de la *app* seleccionada.

[1] SDK: *Software Development Kit*

- Requisitos sobre la popularidad y valoraciones de una aplicación:
  - El panel de mandos ha de mostrar la valoración media de la aplicación seleccionada.
  - El panel de mandos ha de mostrar la distribución de las valoraciones de la *app* seleccionada.
  - El panel de mandos ha de mostrar el número total de valoraciones de la *app* selecionada.
  - El panel de mandos ha de mostrar el número de descargas de la aplicación selecionada.
- Requisitos sobre el desarrollador de una aplicación:
  - El panel de mandos ha de mostrar el nombre del desarrollador de la aplicación seleccionada.
  - El panel de mandos ha de mostrar el e-mail del desarrollador de la aplicación seleccionada.
  - El panel de mandos ha de mostrar la dirección del desarrollador de la aplicación seleccionada.
  - El panel de mandos ha de mostrar la dirección de la página web de la *app* seleccionada.

4.3.3. Requisitos sobre los paneles de privacidad

Deben existir al menos dos paneles de mandos dedicados a la privacidad. El primero se centrará en el análisis general de la privacidad de las aplicaciones Android, permitiendo responder preguntas como qué categorías de aplicaciones son más intrusivas en la privacidad de sus usuarios o qué versión de SDK de Android presenta aplicaciones más intrusivas. El segundo panel se enfocará en el análisis de la privacidad de aplicaciones Android específicas, y permitirá realizar comparaciones directas entre ellas.

Requisitos del primer panel de mandos de privacidad:

- El panel de mandos ha de permitir interactividad para seleccionar la métrica de privacidad, categoría de aplicaciones, y versión de SDK a utilizar en el análisis.
- Análisis de la privacidad por categoría:
  - El panel de mandos ha de mostrar la categoría seleccionada.
  - El panel de mandos ha de mostrar el número de aplicaciones involucradas en el análisis.

    - El panel de mandos ha de mostrar la puntuación media obtenida con la métrica de privacidad escogida para la selección de aplicaciones realizada según categoría.
    - El panel de mandos ha de mostrar la distribución de las puntuaciones obtenidas con la métrica de privacidad seleccionada para la selección de aplicaciones realizada según categoría.
    - El panel de mandos ha de mostrar el valor medio de todas las métricas de privacidad del repositorio en una misma visualización que permita compararlas para la selección de aplicaciones realizada según categoría.

- Análisis de la privacidad por SDK:

    - El panel de mandos ha de mostrar el SDK seleccionado.
    - El panel de mandos ha de mostrar el número de aplicaciones involucradas en el análisis.
    - El panel de mandos ha de mostrar la puntuación media obtenida con la métrica de privacidad escogida para la selección de aplicaciones realizada según SDK.
    - El panel de mandos ha de mostrar la distribución de las puntuaciones obtenidas con la métrica de privacidad seleccionada para la selección de aplicaciones realizada según SDK.
    - El panel de mandos ha de mostrar el valor medio de todas las métricas de privacidad del repositorio en una misma visualización que permita compararlas para la selección de aplicaciones realizada según SDK.

- Análisis de la privacidad misceláneos:

    - El panel de mandos ha de mostrar el valor medio de la métrica de privacidad escogida según dependencias de la aplicación.
    - El panel de mandos ha de mostrar la distribución del valor medio de la métrica de privacidad escogida según la valoración de la aplicación.
    - El panel de mandos ha de mostrar el valor de la métrica de privacidad escogida según el tamaño de la aplicación.

Requisitos del segundo panel de mandos de privacidad:

- El panel de mandos ha de permitir interactividad para seleccionar las dos aplicaciones a comparar y la métrica de privacidad a utilizar.

- Para cada aplicación seleccionada:

- El panel de mandos ha de mostrar el nombre de la aplicación.
- El panel de mandos ha de mostrar el número total de permisos que la aplicación solicita para su funcionamiento.
- El panel de mandos ha de mostrar el número total de permisos "*custom*" que la aplicación solicita para su funcionamiento. Siendo permisos "*custom*" permisos que no se encuentran definidos por el sistema operativo Android.
- El panel de mandos ha de mostrar la puntuación obtenida según la métrica de privacidad seleccionada.
- El panel de mandos ha de mostrar las aportaciones de cada grupo de permisos a la puntuación obtenida con la métrica de privacidad.
- El panel de mandos ha de mostrar la evolución del valor de la métrica en el paso de versiones de la aplicación.
- El panel de mandos ha de sugerir aplicaciones de la misma categoría con mejor puntuación en base a la métrica de privacidad seleccionada.
- El panel de mandos ha de mostrar la dirección a la página web de la política de privacidad de la *app*.

Capítulo 5

Diseño

En este capítulo se lleva a cabo el diseño detallado de todos los componentes necesarios para cumplir con los requisitos establecidos en el capítulo anterior. Al igual que en dicho capítulo, se recomienda consultar el TFG en que se desarrolló inicialmente este repositorio [1], ya que solo se presentarán los diagramas y gráficos que sean nuevos o hayan sido modificados respecto al desarrollo inicial, omitiendo todo el diseño general del mismo, dado que se ha mantenido intacto. Para resaltar las modificaciones, se utiliza el color verde en los diagramas.

Al igual que en el capítulo anterior, el diseño se separará en tres bloques: diseño de la ampliación de fuentes y tipos de datos en App-PIMD, diseño de la mejora de la funcionalidad de la API de App-PIMD, y diseño de los *dashboards* de App-PIMD.

5.1. Diseño de la ampliación de fuentes y tipos de datos en App-PIMD

Como primera fuente de información a incorporar, se ha decidido utilizar F-Droid[1], ya que cumple con los siguientes requisitos: acceso libre, un catálogo extenso de aplicaciones, la posibilidad de descargar aplicaciones en formato archivo y que estas sean de código abierto. Esta fuente de datos cuenta con un catálogo de más de 2000 aplicaciones de código abierto, además de una lista de APIs que facilitan su acceso. Por tanto, ya que cumple con todos los requisitos elicitados en la sección 4.1.1, F-Droid ha sido seleccionada como la primera fuente de datos a incluir en el repositorio.

Como segunda fuente de datos a incorporar, la única fuente de datos que consta de información de calidad sobre valoración, desarrollador y política de privacidad de las aplicaciones es la tienda de aplicaciones oficial Play Store. Sin embargo, esta no permite el acceso a estos metadatos en versiones anteriores. Una opción sería recolectar manualmente todos los metadatos desde de la tienda oficial y almacenarlos a medida que se

[1] `https://f-droid.org/es/`

lanzan nuevas versiones. No obstante, AndrozooGP, un proyecto de la Universidad de Luxemburgo que recolecta metadatos directamente de Play Store a través de los servicios de Google, ha estado recolectando esta información desde mayo de 2020, por lo que es más interesante utilizar este proyecto como fuente de datos para nuestro repositorio [31].

5.1.1. Inclusión de F-Droid en App-PIMD

Para incluir F-Droid, es necesario modificar algunos componentes de software del repositorio. En este aspecto, se presentarán únicamente los diagramas de diseño que han sido modificados, haciendo especial hincapié en las zonas que han sufrido cambios.

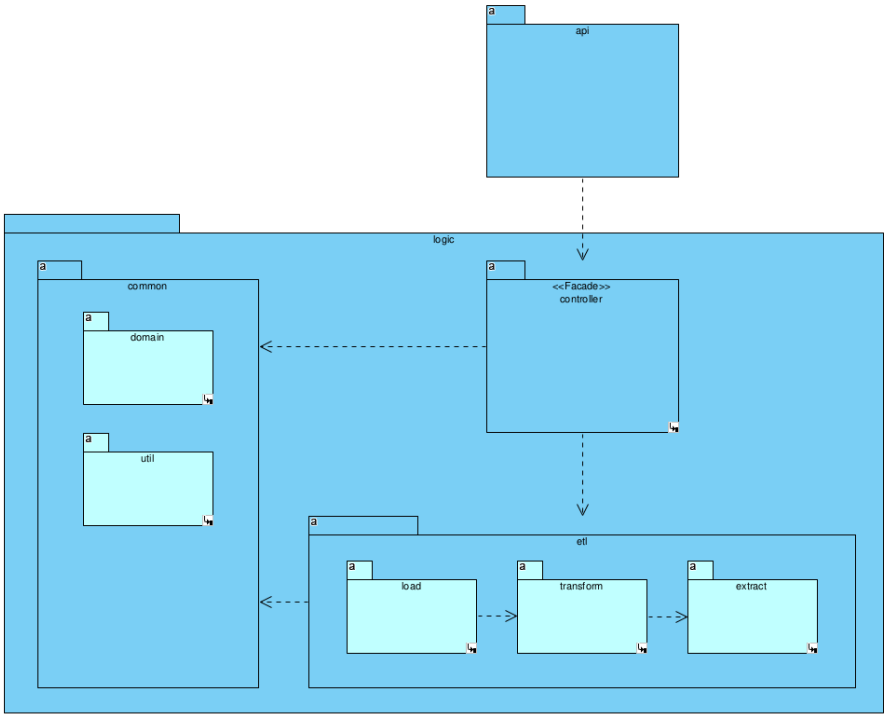

Figura 5.1: Diagrama de paquetes general de App-PIMD.

La figura 5.1 recuerda la estructura general de paquetes del repositorio para situar claramente que zonas sufren cambios o modificaciones. A la hora de añadir una nueva fuente de datos, es necesario incluir un nuevo paquete que la represente dentro de “extract”, el paquete que agrupa todos los procesos de extracción de datos. La figura 5.2 muestra la nueva clase que se ha de añadir, correspondiente a una fuente de datos de tipo web. Esta tendrá que implementar las funciones de búsqueda y descarga de una aplicación dado su nombre de paquete.

Para esta implementación, se hace uso de las APIs que aporta F-Droid[2]. Para la búsqueda, F-Droid aporta una web-api básica que permite realizar búsquedas de texto en el repositorio, incluyéndose búsquedas por nombre de paquete, a través de la dirección:

[2]`https://f-droid.org/es/docs/All_our_APIs/`

`https://search.f-droid.org/api/search_apps?q=TEXTO_BUSQUEDA`. Para la descarga de las aplicaciones, se hace uso de web *scraping*, ya que los enlaces de descarga de su repositorio tienen el mismo formato: `https://f-droid.org/repo/NOMBRE_DE_PAQUETE_DE_APP_VERSION_APP.apk`.

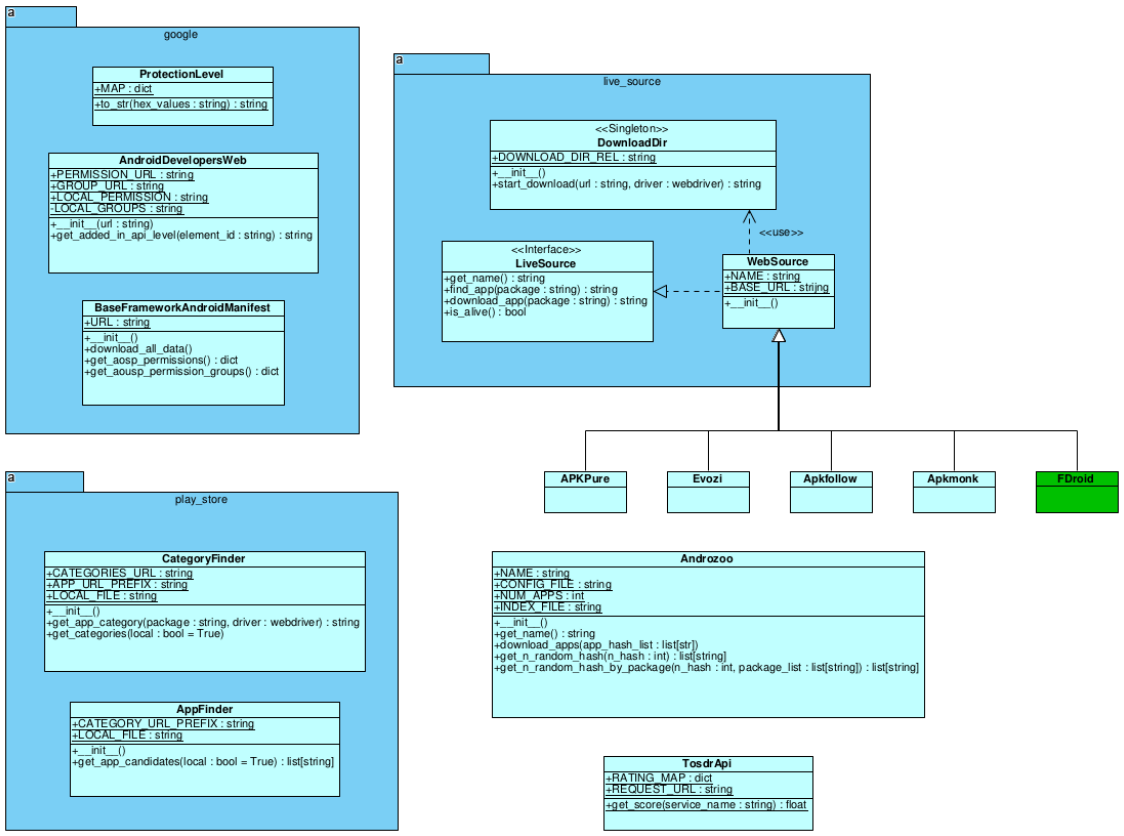

Figura 5.2: Diagrama del paquete *extract* con F-Droid.

5.1.2. Inclusión de AndrozooGP a App-PIMD

Para incorporar AndrozooGP, es necesario modificar más componentes que para incluir F-Droid. Esto se debe a que, además de añadir una nueva fuente de datos, estamos incorporando nuevos tipos de datos, como valoraciones de aplicaciones, información sobre los desarrolladores, políticas de privacidad, entre otros. Por tanto, en primer lugar se ha de modificar el modelo de datos del repositorio para incorporar toda esta nueva información.

Modificaciones al modelo de datos de App-PIMD

Primero, se modelan las nuevas entidades necesarias para añadir la nueva información. En este caso, se han añadido dos entidades nuevas: “az_metadata” y “az_dependency”. La primera representa los metadatos detallados de una aplicación concreta, incluyendo

valoraciones, información sobre el desarrollados y la política de privacidad. La segunda representa las dependencias de una aplicación, es decir, otras aplicaciones de las que depende para su correcto funcionamiento. Como se ve en el diagrama gracias al doble rectángulo y diamante, la nueva entidad "az_metaadata" es débile y sus relaciones identificativas, esto es, que no pueden existir sin que la entidad aplicación, con la que se relacionan, exista. Sin embargo, las dependencias no son una entidad débil ya que pueden existir independientemente.

Como se observa en las relaciones, una aplicación puede tener varias instancias de "az_metadata" asociadas. Esto se debe a que los metadatos se recopilan en momentos específicos y pueden variar con el tiempo, incluso si la versión de la aplicación no ha cambiado. Por otro lado, no todas las aplicaciones necesariamente tienen dependencias para su funcionamiento.

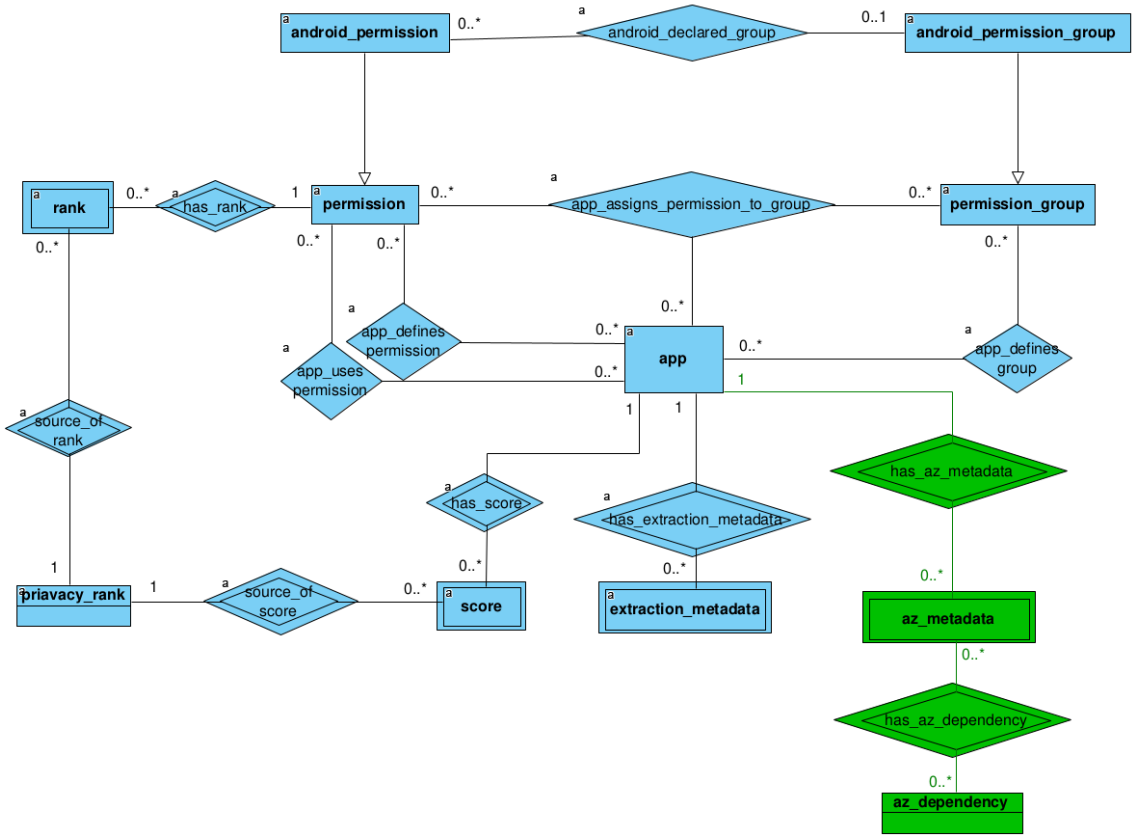

Figura 5.3: Modelo de datos conceptual.

A continuación, se refina el diagrama de la figura 5.3 añadiendo los atributos de cada entidad, su tipo, y las tablas intermedias necesarias para implementar las relaciones. Todo esto se muestra en la figura 5.4, donde se puede observar que ha sido necesario incluir una tabla intermedia para implementar la relación muchos a muchos entre las entidades asociadas a los metadatos y las dependencias.

En esta figura también se muestran las claves primarias de cada tabla. En el caso de los metadatos, van asociados a una aplicación concreta, la cual se identificaba por su *hash*, y a la fecha concreta en la que se recopilaron los metadatos. Para las dependencias, la clave primaria está compuesta por el nombre del paquete y la versión, ya que ambos atributos identifican de manera única cada dependencia. En las tablas 5.1 y 5.2 se muestra una breve descripción de cada nuevo atributo.

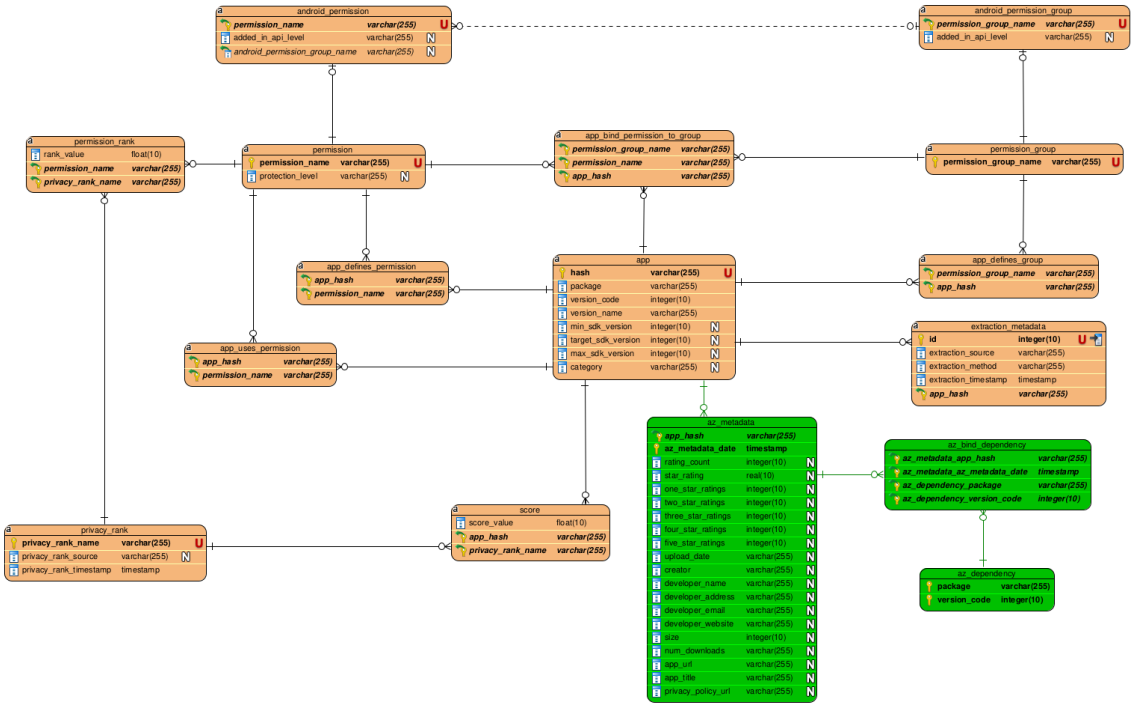

Figura 5.4: Modelo de datos físico.

az_dependency	
package	Nombre de paquete de la dependencia
version_code	Código de versión de la dependencia

Tabla 5.1: Atributos de *az_dependency*.

az_metadata	
app_hash	Hash de la aplicación asociada a los metadatos
az_metadata_date	Marca temporal de la recolección de los metadatos
rating_count	Número de valoraciones de la aplicación
star_rating	Valoración media de la aplicación [0, 5]
one_star_ratings	Número de valoraciones de una estrella
two_star_ratings	Número de valoraciones de dos estrellas
three_star_ratings	Número de valoraciones de tres estrellas
four_star_ratings	Número de valoraciones de cuatro estrellas
five_star_ratings	Número de valoraciones de cinco estrellas
upload_date	Fecha en que se subió la *app* a Play Store
creator	Nombre del creador de la *app*
developer_name	Nombre del desarrollador de la *app*
developer_address	Dirección del desarrollador de la *app*
developer_email	E-mail del desarrollador de la *app*
developer_website	Página web del desarrollador de la *app*
size	Tamaño de la *app* en Bytes
num_downloads	Número aproximado de descargas de la *app*
app_url	Página web de la *app* en Play Store
app_title	Nombre de la *app*
privacy_policy_url	Política de privacidad de la *app*

Tabla 5.2: Atributos de *az_metadata*.

Modificaciones a los componentes software de App-PIMD

Al igual que al añadir F-Droid, también se han de modificar los componentes software de App-PIMD para incorporar AndrozooGP como una nueva fuente de datos del repositorio. En este caso, dado que esta nueva fuente aporta tipos de datos adicionales, se requiere ajustar un mayor número de componentes software.

La figura 5.6 muestra un nuevo paquete que engloba las dos fuentes de Androzoo: la estándar y AndrozooGP. En esta figura, se ve que la clase "AndrozooGP" ha de implementar un método que devuelva los metadatos recogidos en el modelo de datos bajo la entidad "az_metadata", a partir de un nombre de paquete, versión y *hash* de una aplicación.

También es necesario añadir en el paquete *load* tres nuevas clases para gestionar la subida de las entradas correspondientes a las tres nuevas tablas del modelo de datos al repositorio. Estas clases, las cuales se muestran en la figura 5.5, seguirán el patrón de diseño ya utilizado a este respecto, el patrón *data mapper*.

Finalmente, es necesario modificar la función *load_app* del controlador para que, al cargar una aplicación en el repositorio, se verifique si existen metadatos asociados a esta en AndrozooGP haciendo uso de la nueva clase del paquete "extract". En caso afirmativo, estos metadatos deberán ser almacenados en el repositorio haciendo uso de las modificaciones del paquete "load".

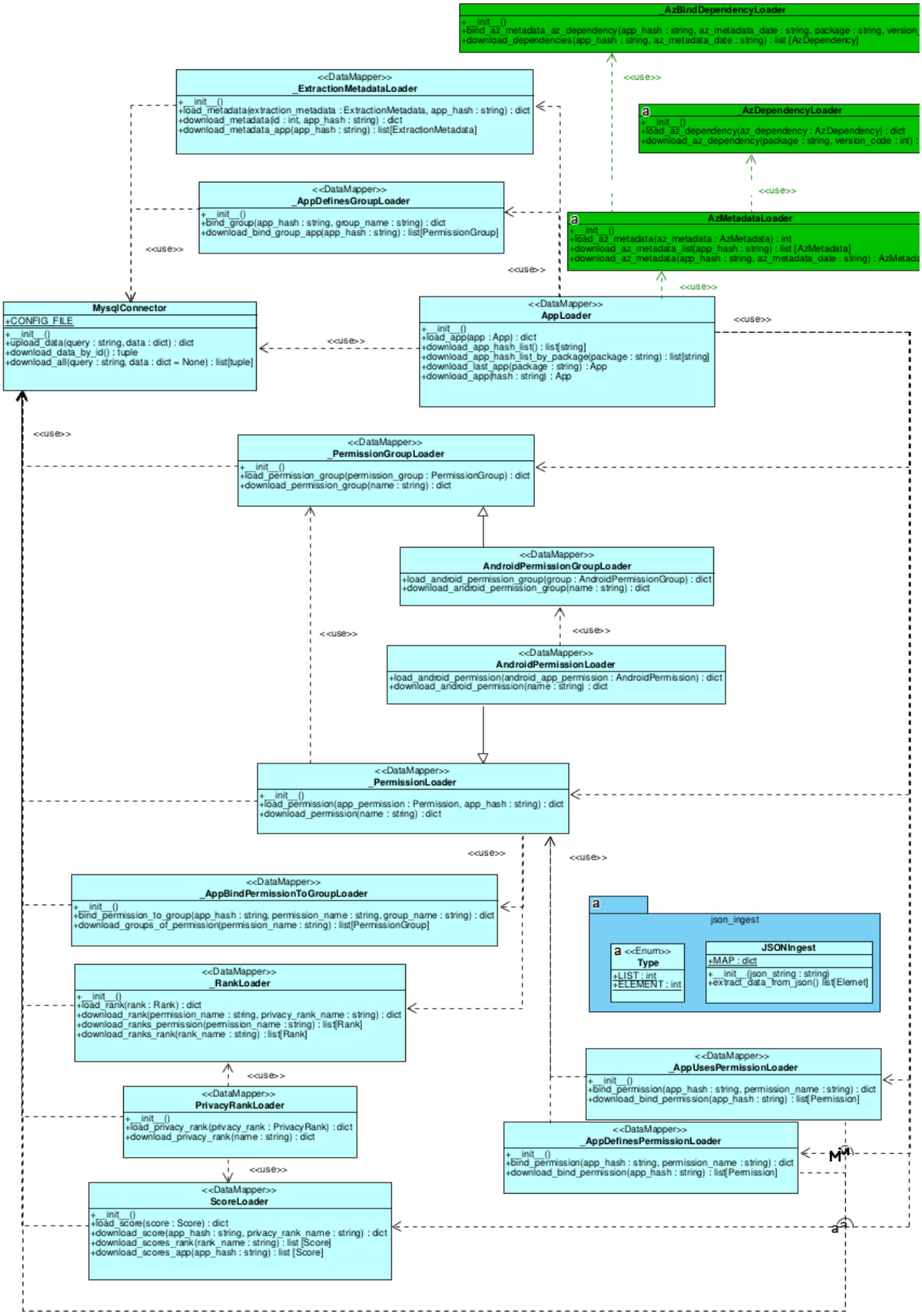

Figura 5.5: Diagrama del paquete *load* con AndrozooGP.

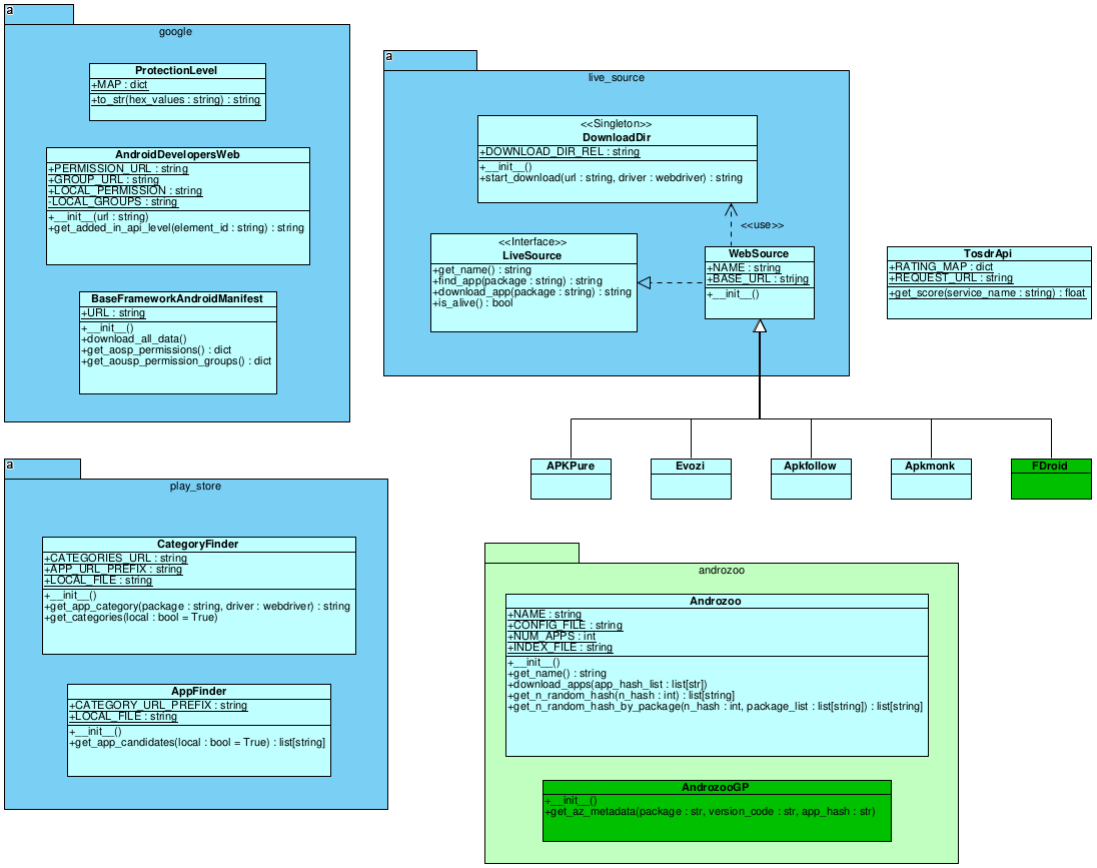

Figura 5.6: Diagrama del paquete *extract* con AndrozooGP.

5.2. Diseño de la mejora de la funcionalidad de la API de App-PIMD

Como se indicó en la sección 4.2.1, la API ha de incorporar cuatro nuevas funcionalidades, manteniendo la uniformidad con las ya existentes y garantizando la retrocompatibilidad. Por ello, se decide englobar esta nueva funcionalidad bajo una nueva versión de la API, la "v2". De este modo, toda la funcionalidad existente permanecerá intacta y se podrá hacer uso de la nueva bajo el prefijo "v2" en las peticiones.

En el apéndice B se encuentra el manual de esta nueva versión de la API, donde se encuentra el diseño formal de las nuevas funcionalidades, los parámetros de cada petición, los posibles estados de respuesta y respuestas asociadas, así como algunos ejemplos.

5.3. Diseño de los *dashboards* de App-PIMD

5.3.1. Identificación de los hechos y dimensiones

En un proyecto BI[3] es esencial identificar los hechos y las dimensiones. Los hechos representan lo que se desea medir, mientras que las dimensiones definen cómo lo queremos medir. Para identificar los hechos y dimensiones hay que tener en cuenta los requisitos elicitados en la sección 4.3.

Los hechos asociados a una aplicación identificados son:

- Nombre.
- Versión.
- Fecha de lanzamiento.
- Tamaño.
- Nombre del desarrollador.
- E-mail del desarrollador.
- Dirección del desarrollador.
- Página web del desarrollador.
- Política de privacidad.
- Lista de permisos.

Las dimensiones o segmentaciones identificadas son:

- *App*. Que se subdivide en dos dimensiones que forman una jerarquía: nombre de paquete y versión.
- Categoría de la *app*.
- Versión mínima de SDK requerido para ejecutar la *app*.
- Fecha. Que se subdivide en día, mes y año, que forman una jerarquía.

Para cumplir con el objetivo de realizar paneles de mandos lo más cercanos posibles a tiempo real (ver sección 1.3), se ha decidido simplificar el modelado de hechos y dimensiones adaptándonos al modelo de datos que utiliza el repositorio (ver sección 5.1.2). Por tanto, a continuación solo se mostrarán las métricas y KPIs[4] que requieran algún tipo de cálculo, ya que no se encuentren en el modelo de datos del repositorio directamente.

[3] BI: *Business Intelligence*
[4] KPI: *Key Performance Indicator*

5.3.2. Métricas y KPIs representados

Teniendo en cuenta los hechos y dimensiones encontrados, la tabla 5.3 muestra la definición de todas las métricas, que no son directamente atributos del modelo de datos (ver sección 5.1.2), y se representarán en las distintas visualizaciones.

Nombre métrica	**Definición**
Nº de *apps* almacenadas	Representa el número total de metadatos de aplicaciones almacenados en el repositorio.
Nº de fuentes de datos	Representa el número total de fuentes de datos de aplicaciones que utiliza el repositorio.
Nº de permisos	Representa el número total de permisos, ya sean oficiales o *custom*, almacenados en el repositorio.
Nº de grupos	Representa el número total de grupos de permisos oficiales almacenados en el repositorio.
Nº de descargas	Representa el número total de descargas de una *app* concreta en Play Store.
Nº de valoraciones	Representa el número total de valoraciones recibido por una *app* concreta en Play Store.
Nº de *apps* del análisis	Representa el número de aplicaciones seleccionadas según los filtros aplicados, que se utilizarán en las visualizaciones.

Tabla 5.3: Definición de las métricas representadas.

La tabla 5.4 recoge los KPI asociados a la consecución del objetivo de minimizar el impacto de las aplicaciones en la privacidad de sus usuarios. Estos indicadores permiten evaluar y monitorizar de manera objetiva el nivel de intrusividad de las aplicaciones y la efectividad de las medidas que se implementen para proteger la privacidad de los usuarios. En todos los casos, los KPI se representan mediante un valor numérico entre 0 y 10, donde 10 indica una aplicación con una intrusividad elevada en la privacidad, y 0 una aplicación con nula intrusividad en la privacidad. Estos valores facilitan la comparación y el seguimiento del impacto de cada aplicación en la privacidad de los usuarios.

Nombre KPI	**Definición**
Métrica RPNDroid	Definida en base a los grupos de permisos Android que solicita la *app* para su funcionamiento [12].
Métrica ToS;DR	En base a los términos de servicio de las *apps* [32].
Métrica TUDelft *intrusiveness*	En base a cuestionarios realizados [11].
Métrica PTaP *privacy impact*	En base a los permisos Android que solicita la *app* [10].
Métrica PTaP *threats by review*	En base a las reseñas de las personas [10].

Tabla 5.4: Definición de los KPI representados.

5.3.3. Diseño de los paneles de mandos

El diseño de los *dashboards* se ha realizado a base de *mockups*, que son prototipos de cómo se espera que sean finalmente cada uno de los paneles de mandos. Estos diseños se han desarrollado siguiendo los siguientes principios [16]:

- **Modularidad**: Separar las visualizaciones de datos en áreas distintas para facilitar al usuario la asociación entre los datos y gráficos relacionados.
- **Guiar la atención**: Uso de colores vivos y visualizaciones atractivas para guiar la atención del usuario hacia los elementos más relevantes del *dashboard*.
- **Facilidad de aprendizaje**: Uso de paneles informativos de ayuda que faciliten el aprendizaje del uso del *dashboard* a medida que se utiliza.
- **Personalizable**: Permitir la personalización de los paneles mediante filtros de información, para que el usuario pueda extraer una mayor cantidad de información relevante.

En total se han realizado cinco *mockups*, uno para cada uno de los paneles de mandos que se han de implementar en base al análisis de la sección 4.3, y otro para un menú principal que servirá como único punto de entrada a los distintos paneles.

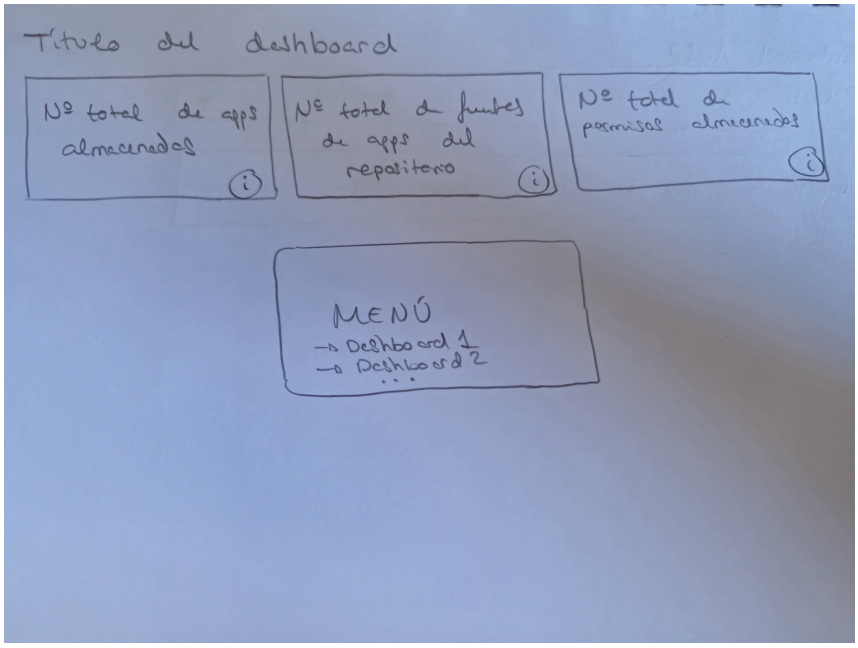

Figura 5.7: *Mockup* del menú de los *dashboards*.

En la figura 5.7 se encuentra el *mockup* realizado para el menú general de los *dashboards*. Este panel incluye un título, tres paneles con información general sobre el repositorio, y una zona central con los botones que dirigen al resto de paneles de mandos. Se ha utilizado patrón "Z" para alinear el contenido, y usado la zona central del panel para los

elementos más importantes, los botones hacia el resto de paneles. Además, cada visualización de datos y gráfico incluye un botón de ayuda que proporciona información sobre el contenido y su interpretación.

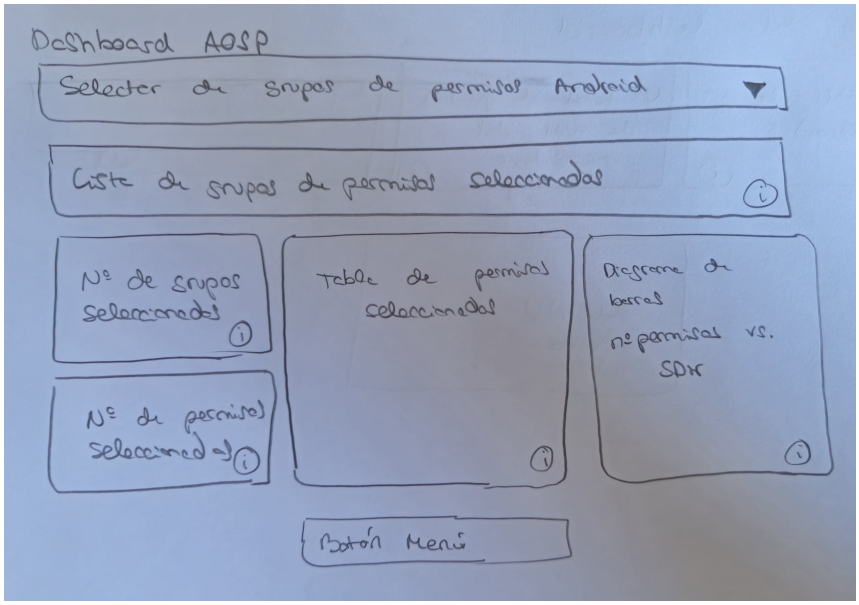

Figura 5.8: *Mockup* del *dashboards* de AOSP.

La figura 5.8 muestra el prototipo del panel de mandos dedicado a la información sobre AOSP. Este panel permite interactuar para seleccionar los grupos de permisos Android que se desean visualizar. En la zona superior, se muestra el nombre de los grupos de permisos seleccionados. A continuación, se presentan el número de grupos de permisos seleccionado, el número total de permisos que engloban, y una tabla que detalla los nombres de estos permisos junto con la versión de SDK de Android en la que fueron añadidos. Finalmente, un diagrama de barras ilustra la distribución del número de permisos añadidos en cada versión del SDK de Android. Se decide utilizar un diagrama de barras en lugar de un histograma u otra visualización dada la naturaleza de número entero de la versión de SDK Android.

La figura 5.9 muestra el *mockup* del panel de mandos centrado en la información sobre *apps*. Este panel permite la selección interactiva del nombre de paquete y la versión de la aplicación deseada. Siguiendo el principio de modularidad, el panel se divide en tres secciones horizontales: la primera proporciona información general sobre la aplicación, la segunda corresponde con información sobre su popularidad y las valoraciones, y la tercera ofrece información sobre el desarrollador.

En la primera sección, se muestra en texto el nombre de la *app* seleccionada, su categoría, fecha de publicación en la tienda oficial, nombre de paquete, versión, versión mínima de SDK requerido para su ejecución y el tamaño en MB. El tamaño de la aplicación se representa mediante la metáfora del semáforo, donde el texto cambia de color en función del tamaño: verde si la aplicación es poco pesada (menos de 50MB), amarillo si

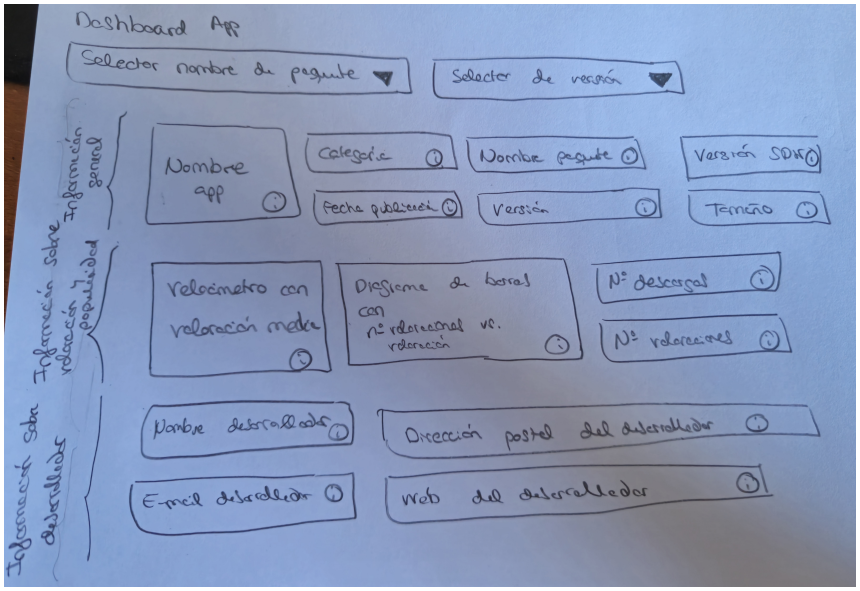

Figura 5.9: *Mockup* del *dashboards* de aplicaciones.

es medianamente pesada (de 50 a 100MB), o rojo si es muy pesada (más de 100MB).

En la segunda sección, se utiliza un velocímetro para mostrar la valoración media de la aplicación, siguiendo también la metáfora del semáforo: rojo para malas valoraciones (0-2), amarillo para valoraciones medias (2-4), y verde para buenas valoraciones (4-5). A continuación, se muestra la distribución de valoraciones en un diagrama de barras, permitiendo identificar si la media esta influenciada por valores atípicos o si existe bimodalidad, entre otros. Además, se muestra el número aproximado de descargas y el número de valoraciones de la aplicación para ofrecer una visión general de su popularidad.

Finalmente, en la tercera sección se proporciona información sobre el desarrollador de la aplicación, incluyendo su nombre, dirección de correo electrónico, dirección postal y página web.

En la figura 5.10 se encuentra el prototipo del primer *dashboard* dedicado al estudio y análisis del impacto de las aplicaciones en la privacidad de los usuarios. El panel ofrece interactividad mediante menús desplegables que permiten seleccionar la métrica de privacidad a utilizar en las visualizaciones, la categoría de las aplicaciones y la versión mínima de SDK de Android requerida para ejecutarlas.

Este panel se divide en tres áreas principales, cada una dedicada a un tipo de análisis específico:

- **Análisis de la Privacidad por Categoría:**

  En la primera zona del primer área, se muestra la categoría seleccionada y el número de aplicaciones que participan en el análisis. Seguidamente, se encuentran tres representaciones gráficas que ayudan a entender la intrusividad en la privacidad de las aplicaciones de dicha categoría.

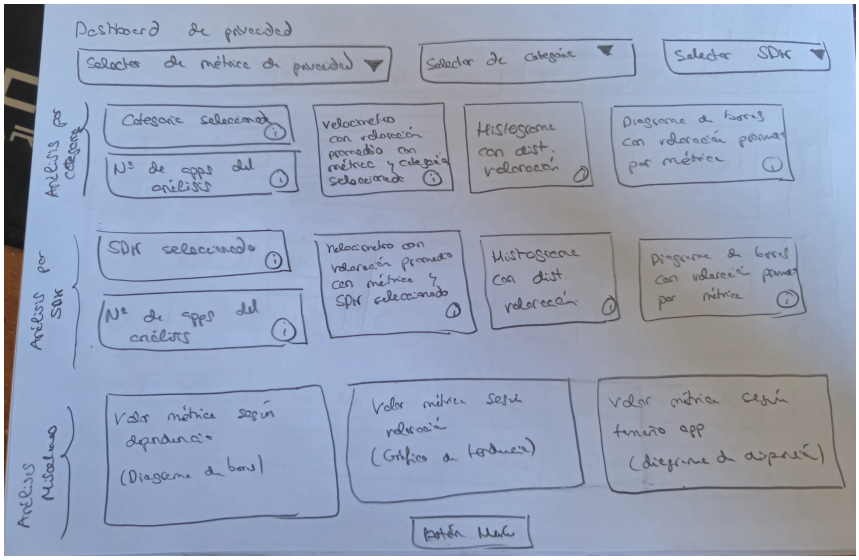

Figura 5.10: *Mockup* del *dashboards* de privacidad general.

- **Gráfico de Velocímetro:** Muestra la puntuación media obtenida con la métrica de privacidad seleccionada por las aplicaciones de la categoría seleccionada. Además, utiliza la metáfora del semáforo para indicar en una escala del verde al rojo cuán buena es la puntuación obtenida. Verde indica una puntuación baja (buena privacidad), amarillo una puntuación media, y rojo una puntuación alta (mala privacidad).
- **Histograma:** Muestra la distribución de las puntuaciones de privacidad para hacerse una idea de la intrusividad general de las aplicaciones de la categoría seleccionada. De este modo se revela si hay un predominio de aplicaciones con buenas o malas puntuaciones, y si hay grupos de puntuaciones atípicas.
- **Gráfico de Barras:** Muestra las diferentes valoraciones medias obtenidas con cada métrica de privacidad. Permite comparar las distintas métricas de privacidad para la categoría seleccionada, ayudando a evaluar si los resultados son de buena o mala calidad si las métricas son parecidas o distan mucho. De modo similar al velocímetro, se ha utilizado la metáfora del semáforo.

- **Análisis de la Privacidad por Versión de SDK:**

  En el segundo área se muestra una réplica de la primera, pero aplicada a la versión mínima de SDK de Android necesaria para ejecutar la aplicación. Esto permite ver si las aplicaciones más intrusivas se encuentran aglomeradas en alguna versión específica del SDK de Android.

- **Análisis de la Privacidad - Misceláneos:**

  Finalmente, en el tercer área se encuentran tres visualizaciones con análisis misceláneos sobre la privacidad.

- **Gráfico de Barras:** Muestra el valor medio de la métrica de privacidad seleccionada por dependencia de las aplicaciones. De este modo se puede identificar si la presencia de alguna dependencia concreta es un buen indicador de la intrusividad en la privacidad de una aplicación. De modo similar a los velocímetros anteriores, se ha utilizado la metáfora del semáforo.
- **Gráfico de Líneas:** Muestra la tendencia del valor medio de la métrica de privacidad seleccionada según el número de estrellas obtenidas en la valoración de una *app*. Esta visualización permite estudiar si las aplicaciones mejor valoradas tienden a ser más o menos intrusivas en la privacidad.
- **Gráfico de Dispersión:** Muestra el valor de la métrica obtenido por tamaño en MB de una aplicación. Permite ver si existe una relación entre la intrusividad en la privacidad y el tamaño de una aplicación.

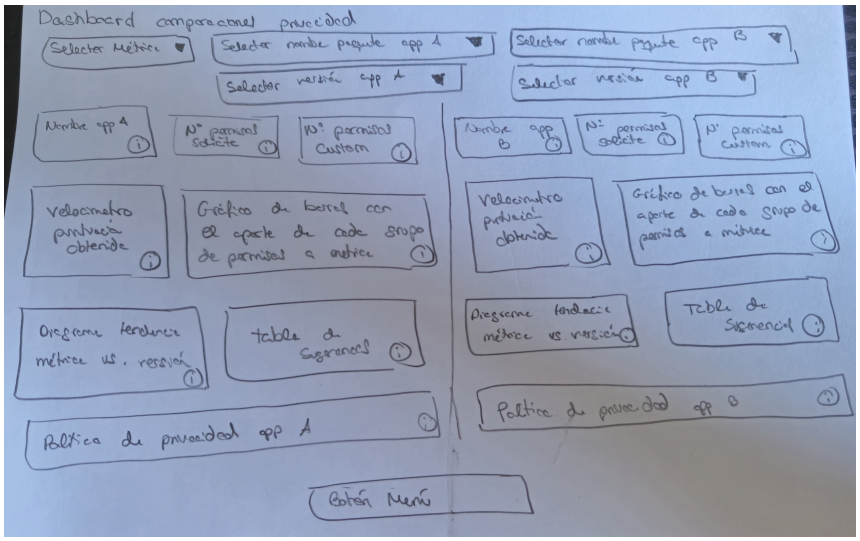

Figura 5.11: *Mockup* del *dashboards* de comparaciones de privacidad.

En la figura 5.11 se presenta el *mockup* del segundo y último *dashboard* dedicado al estudio y análisis del impacto de las aplicaciones en la privacidad de los usuarios. El panel ofrece interactividad mediante menús desplegables que permiten seleccionar la métrica de privacidad a utilizar en el análisis y elegir dos aplicaciones según su nombre de paquete y versión.

El panel de mandos se divide en dos zonas en paralelo, una por *app* seleccionada. Por lo tanto, se especificará el diseño de una de estas, ya que el de la otra es idéntico. En primer lugar, se muestra el nombre de la aplicación seleccionada, el número total de permisos que solicita para su funcionamiento y el número de permisos "*custom*" que requiere. Los permisos "custom" son aquellos que no se encuentran definidos por el sistema Android, es decir, son definidos por aplicaciones de terceros.

Seguidamente, haciendo uso de la metáfora del semáforo, se muestra un gráfico de velocímetro que permite comparar la puntuación obtenida con la métrica seleccionada para ambas aplicaciones. Además, se incluye un gráfico de barras que muestra el aporte de cada grupo de permisos a la métrica. También se presenta una linea de tendencia del valor de la métrica en función de la versión de la *app*, que permite evaluar si su desarrollo va en buen camino o no. Finalmente, se ofrece una tabla con sugerencias de aplicaciones de la misma categoría seleccionada pero con mejor valor de la métrica de privacidad, junto con un enlace a la política de privacidad de la aplicación seleccionada.

Capítulo 6

Implementación y pruebas

En este capítulo se detallan los aspectos clave y las modificaciones realizadas en la implementación, así como las nuevas librerías empleadas además de las utilizadas en el desarrollo inicial. Para una lista completa de todos los detalles de la implementación y librerías utilizadas, se recomienda ver el capítulo correspondiente del TFG [1]. Todo el código desarrollado se encuentra en el repositorio público Github: `https://github.com/peres317/app-pimd-tfm.git`. Su puesta en marcha se encuentra documentada en la guía de instalación contenida en el apéndice A.

6.1. Detalles sobre la implementación

6.1.1. Configuración del *proxy* inverso

Como se mencionó en la sección 3.1.4 sobre técnicas de integración, se utiliza un *proxy* inverso para el acceso uniforme a la API y los *dashboards* del proyecto App-PIMD. Traefik, el *proxy* escogido, tiene una configuración muy sencilla dentro del directorio */etc/traefik*. Las configuraciones realizadas son las siguientes:

- *traefik.yml:* Este es el archivo de configuración general de Traefik. En él se especifican los puntos de entrada que queremos abrir, así como el directorio para el resto de archivos de configuración. Se configura un único punto de entrada en el puerto 9999 y el subdirectorio *config/* para el resto de configuraciones (ver figura 6.1).

```
entryPoints:
  App-PIMD:
    address: ":9999"

providers:
  file:
    directory: /etc/traefik/config
traefik.yml (END)
```

Figura 6.1: Configuración en *traefik.yml*.

- *config/api.yml:* En este archivo de configuración se especifica que el prefijo */api* identificará las entradas a la API y deberán enrutarse al servicio local en el puerto 8080, la API. Además, también se especifican los archivos correspondientes al certificado SSL a utilizar (ver figura 6.2).

```
tls:
  certificates:
    - certFile: /etc/traefik/certificates/cert.pem
      keyFile: /etc/traefik/certificates/key.pem

http:
  routers:
    router0:
      entryPoints:
        - "App-PIMD"
      middlewares:
        - "App-PIMD-API-mid"
      service: App-PIMD-API
      rule: "PathPrefix(`/api`)"
      tls: {}

  middlewares:
    App-PIMD-API-mid:
      stripPrefix:
        prefixes:
          - "/api"

  services:
    App-PIMD-API:
      loadBalancer:
        servers:
          - url: "http://localhost:8080/"
config/api.yml (END)
```

Figura 6.2: Configuración de la entrada a la API.

- *config/grafana.yml:* En este archivo se especifica que bajo el prefijo */dashboard* se encuentran los *dashboards*. Por tanto, el *proxy* deberá enrutar estas conexiones al servicio local en el puerto 3000 que corresponde a Grafana (ver figura 6.3).

```
tls:
  certificates:
    - certFile: /etc/traefik/certificates/cert.pem
      keyFile: /etc/traefik/certificates/key.pem

http:
  routers:
    router1:
      entryPoints:
        - "App-PIMD"
      middlewares:
        - "App-PIMD-Grafana-mid"
      service: App-PIMD-Grafana
      rule: "PathPrefix(`/dashboard`)"
      tls: {}

  middlewares:
    App-PIMD-Grafana-mid:
      stripPrefix:
        prefixes:
          - "/dashboard"

  services:
    App-PIMD-Grafana:
      loadBalancer:
        servers:
          - url: "http://localhost:3000/"
config/grafana.yml (END)
```

Figura 6.3: Configuración de la entrada a los *dashboards*.

- *certificates/:* En este directorio se introduce el certificado SSL que se utilizará para encriptar el tráfico en tránsito que salga del *proxy* hacia Internet.

6.1.2. Implementación de las nuevas fuentes de datos

Implementación de F-Droid

Para implementar el proceso ETL de F-Droid, ya se mencionó en el capítulo de diseño qué APIs y procesos de extracción se utilizaban. A continuación, se detallan.

1. Para la búsqueda de aplicaciones se hace uso de su web-api. Se formatea la URL: `https://search.f-droid.org/?q=NOMBRE_DE_PAQUETE` con el nombre de paquete que se desea buscar y se carga la página web asociada. En la figura 6.4 se muestra la búsqueda del paquete *org.telegram*.

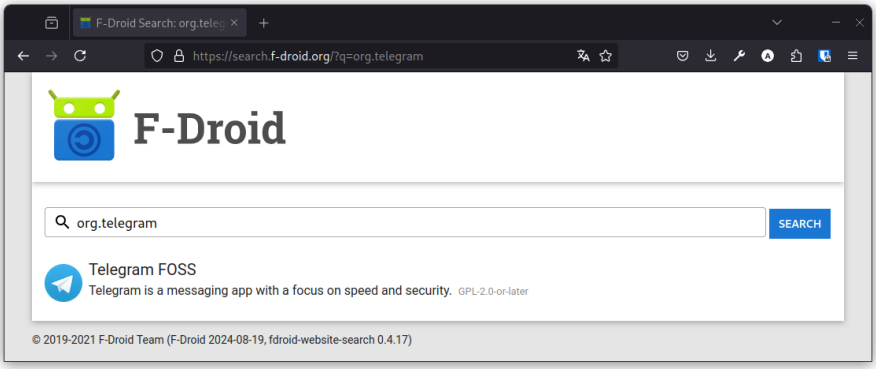

Figura 6.4: Búsqueda del paquete *org.telegram* en F-Droid.

   Todos los paquetes del resultado son enlaces (etiqueta "a" en HTML) que contienen la palabra "package". De este modo, podemos filtrarlos. Si hay más de un resultado en la búsqueda, se extrae el primero de ellos.

2. Para la descarga de aplicaciones, se carga el enlace obtenido al finalizar el paso anterior y se extraen los enlaces que finalizan en ".apk", ya que son enlaces de descarga. Estos enlaces están ordenados de forma decreciente por versión, por lo que el primer enlace de descarga corresponderá a la versión más moderna y será el que se utilice por defecto.

Implementación de AndrozooGP

Para implementar el proceso ETL de AndrozooGP se realiza una petición a su API formateando la siguiente URL: `https://androzoo.uni.lu/api/get_gp_metadata/NOMBRE_DE_PAQUETE/VERSION_CODE` con el nombre de paquete de la aplicación a buscar y su código de versión.

La respuesta que se recibe (ver figura 6.5), de donde habrá que extraer los atributos definidos en la tablas 5.2 y 5.1 de la sección 5.1.2, se encuentra en formato JSON. La tabla 6.1 muestra las rutas a cada atributo dentro del JSON.

```
alejandro@torre<~>$ curl -G 'https://androzoo.uni.lu/api/get_gp_metadata/occam.hammer.drone/65?apikey=

[{"aggregateRating":{"commentCount":"0","fiveStarRatings":"0","fourStarRatings":"0","oneStarRatings":"0","ratingsCo
unt":"0","starRating":0.0,"threeStarRatings":"0","twoStarRatings":"0","type":2},"availability":{"restriction":1},"a
z_metadata_date":"2023-09-17 14:03:39.220798","backendDocid":"occam.hammer.drone","backendId":3,"creator":"Hammer M
issions","descriptionHtml":"Hammer provides *adaptive* mission planning for drones, enabling high-quality data coll
ection for a number of use-cases.<br><br>Whatever the mission, Hammer&#39;s got you covered!<br><br>Hammer is the o
ne-stop solution for planning and flying different types of drone flights automatically with high levels of precisi
on and control. Say goodbye to constantly switching between apps or learning new systems all the time.<br><br>With
```

Figura 6.5: Ejemplo de respuesta de AndrozooGP.

az_dependency	
package	*details/appDetails/dependencies/dependency/packageName*
version_code	*details/appDetails/dependencies/dependency/version*
az_metadata	
app_hash	Es un parámetro ya conocido.
az_metadata_date	*az_metadata_date*
rating_count	*aggregateRating/ratingsCount*
star_rating	*aggregateRating/starRating*
one_star_ratings	*aggregateRating/oneStarRatings*
two_star_ratings	*aggregateRating/twoStarRatings*
three_star_ratings	*aggregateRating/threeStarRatings*
four_star_ratings	*aggregateRating/fourStarRatings*
five_star_ratings	*aggregateRating/fiveStarRatings*
upload_date	*details/appDetails/uploadDate*
creator	*creator*
developer_name	*details/appDetails/developerName*
developer_address	*details/appDetails/developerAddress*
developer_email	*details/appDetails/developerEmail*
developer_website	*details/appDetails/developerWebsite*
size	*details/appDetails/installationSize*
num_downloads	*details/appDetails/numDownloads*
app_url	*shareUrl*
app_title	*title*
privacy_policy_url	*relatedLinks/privacyPolicyUrl*

Tabla 6.1: Mapeo de atributos y estructura JSON de AndrozooGP.

6.1.3. Implementación de las mejoras de la API

- **Implementación de la lista de versiones almacenadas:**

  La figura 6.6 muestra mediante un ejemplo la consulta SQL a la tabla *app* del modelo de datos que permite devolver la lista de versiones almacenadas de una aplicación.

```
mysql> SELECT hash, version_name FROM app WHERE package = "org.telegram.messenger";
+------------------------------------------------------------------+--------------+
| hash                                                             | version_name |
+------------------------------------------------------------------+--------------+
| 007667996e3b3d8fb808d9a4930198fd58dda4c84d8571ef2b7fa03662958e18 | 1.3.23       |
| 0236fc7bb949554f054bb4f22c33fdc4ff8d2c97efb152ad5b42308572b038f5 | 1.3.17       |
| 0727652e8662e20c2ceaa9ef2dfc2ac68131890755705a84dbbe86191a2c8738 | 5.4.0        |
| 9d542895a7f6877e843c6860a07b2e3b432a2fe587d087ba8ea50423fbd58fa2 | 10.14.3      |
+------------------------------------------------------------------+--------------+
4 rows in set (0.01 sec)
```

Figura 6.6: Ejemplo de obtención de la lista de versiones almacenadas de Telegram.

- **Implementación de la subida de una *app* por archivo:**

  El único detalle que se debe tener en cuenta a la hora de subir archivos mediante una petición a la API es que sean el tipo de archivo esperado, en este caso, aplicaciones. Para ello, la API debe comprobar que la petición contiene la cadena "application/vnd.android.package-archive", la cual identifica un archivo de aplicación Android, en la cabecera *content_type*.

- **Implementación del motor de búsqueda de *apps* por nombre:**

  Para implementar la búsqueda de una aplicación por su nombre, la forma más sencilla es utilizar un motor de búsqueda existente. Y, ¿qué mejor motor de búsqueda que el de la tienda de aplicaciones oficial de Android, Play Store? Este se basa en una web-api, que al cargar la siguiente URL: `"https://play.google.com/store/search?q=NOMBRE_DE_APP"`, nos devuelve una lista ordenada de aplicaciones.

6.1.4. Implementación de los *dashboards*

Como se mencionó en la sección 3.2, se utiliza Grafana para implementar los *dashboards*. Grafana permite establecer un intervalo de tiempo de actualización de los paneles, haciéndolos *near real time*. Para implementar los paneles diseñados en la sección 5.3, se han empleado los botones y menús interactivos disponibles en la interfaz de Grafana. Las figuras 6.7, 6.8, 6.9, 6.10 y 6.11 muestran la implementación final de los paneles de mandos de App-PIMD.

Figura 6.7: Implementación del panel de mandos Menú.

Figura 6.8: Implementación del panel de mandos AOSP.

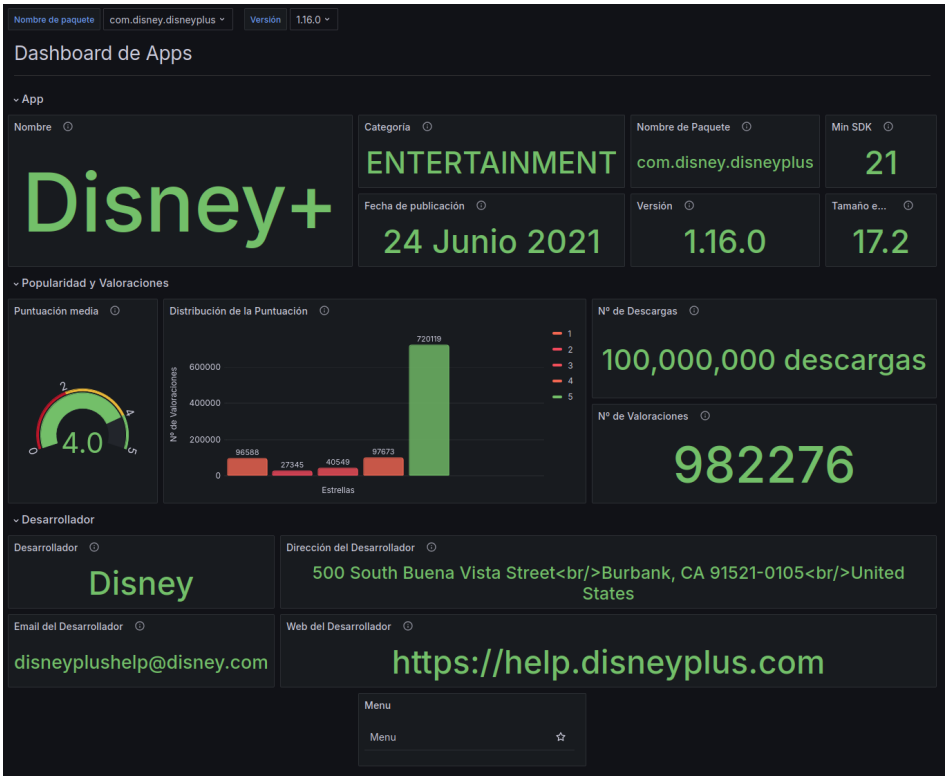

Figura 6.9: Implementación del panel de mandos Aplicaciones.

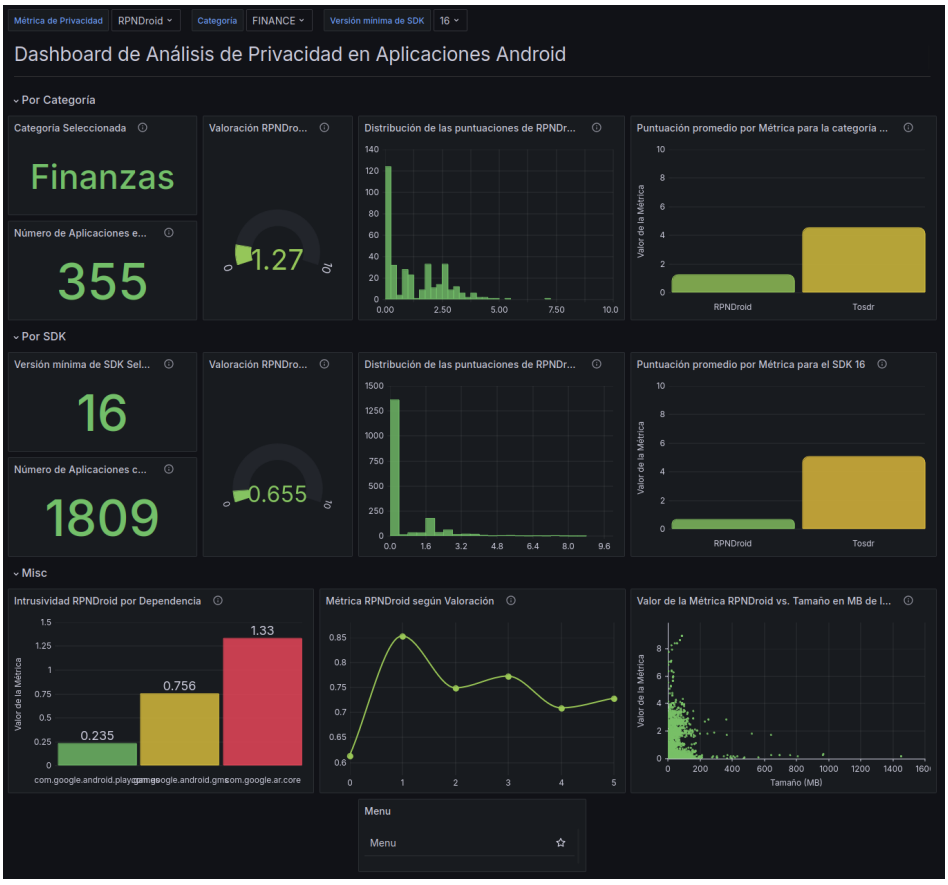

Figura 6.10: Implementación del panel de mandos Privacidad.

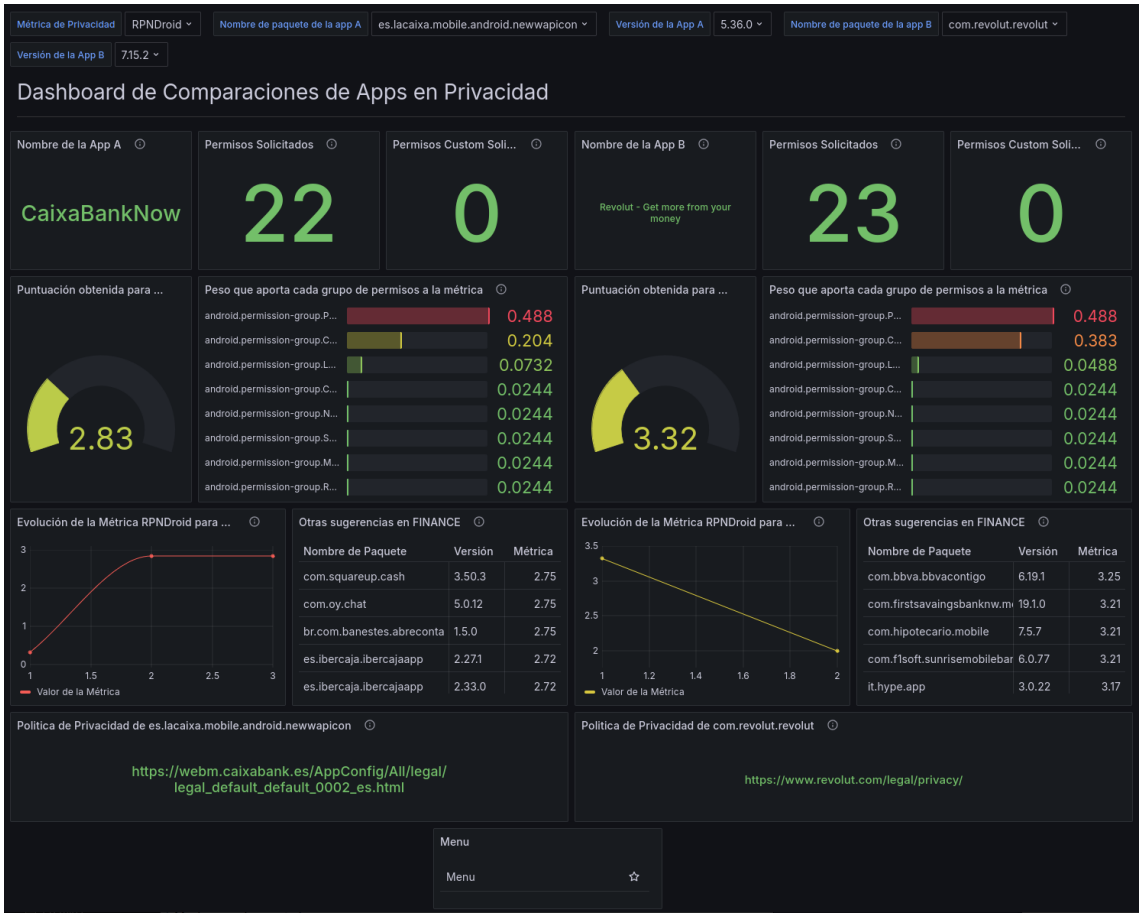

Figura 6.11: Implementación del panel de mandos Comparaciones de Privacidad.

6.2. Pruebas

La funcionalidad desarrollada de la API, así como los paneles de mandos, se han probado dinámicamente en términos de rendimiento. No se ha detectado ninguna carencia en el rendimiento de estos dos componentes. A continuación, se cuantificará el rendimiento de la versión 2 (v2) de la API del repositorio en términos de tiempo que tarda en completar las distintas consultas que se pueden realizar.

6.2.1. Rendimiento de la API v2 de App-PIMD

La figura 6.12 muestra una consulta de prueba para retornar las versiones de Telegram almacenadas a través del cliente Postman. En esta figura se muestra como la consulta se completa en tan solo 33 ms.

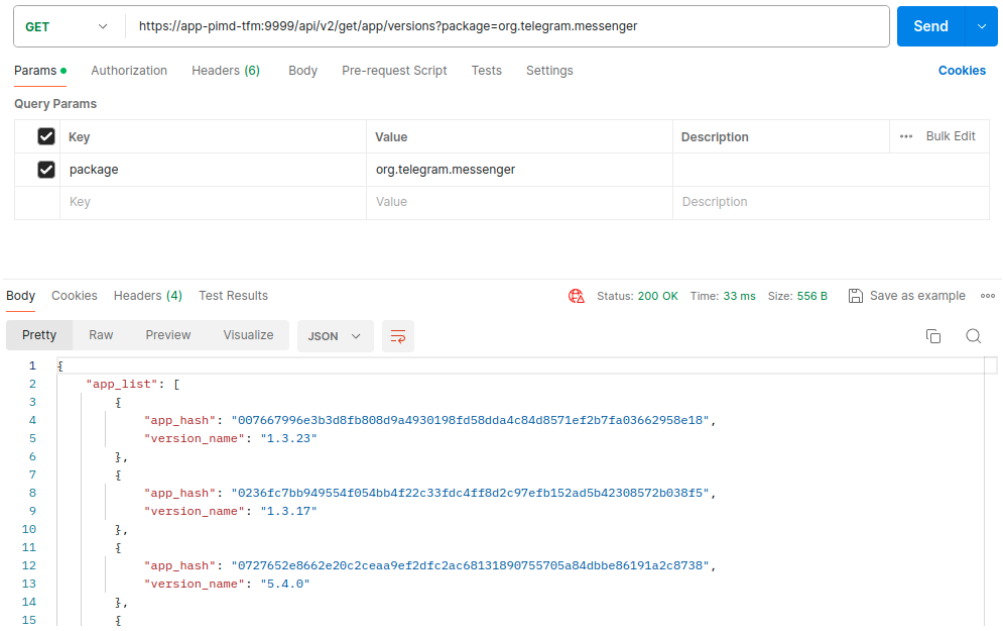

Figura 6.12: Prueba de retorno de lista de versiones almacenadas de una *app*.

La figura 6.13 muestra la prueba de subida de una aplicación a través de la documentación interactiva de la API. La aplicación subida es el conector de datos de salud de Android, que ocupa aproximadamente 7,1 MB. La consulta tarda aproximadamente 10 segundos en resolverse, de los cuales la mayoría se gastan en esperar a que se transmita el archivo de la aplicación al repositorio a través de la red.

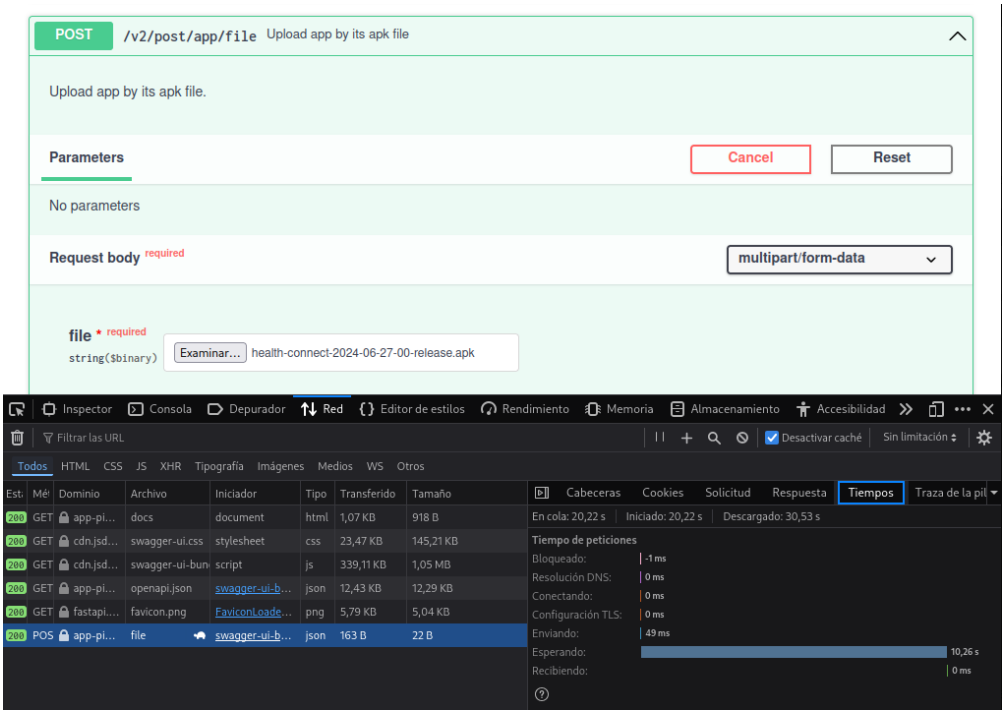

Figura 6.13: Prueba de subida de una *app* por archivo.

En la figura 6.14 se muestra la descarga de los metadatos de Telegram por su nombre en lugar de por el nombre de paquete o el *hash*, como se venía haciendo hasta ahora. Como se observa en la figura, la consulta tarda unos 7 segundos en completarse, ya que debe realizar la petición al motor de búsqueda de Play Store a través de su web-api.

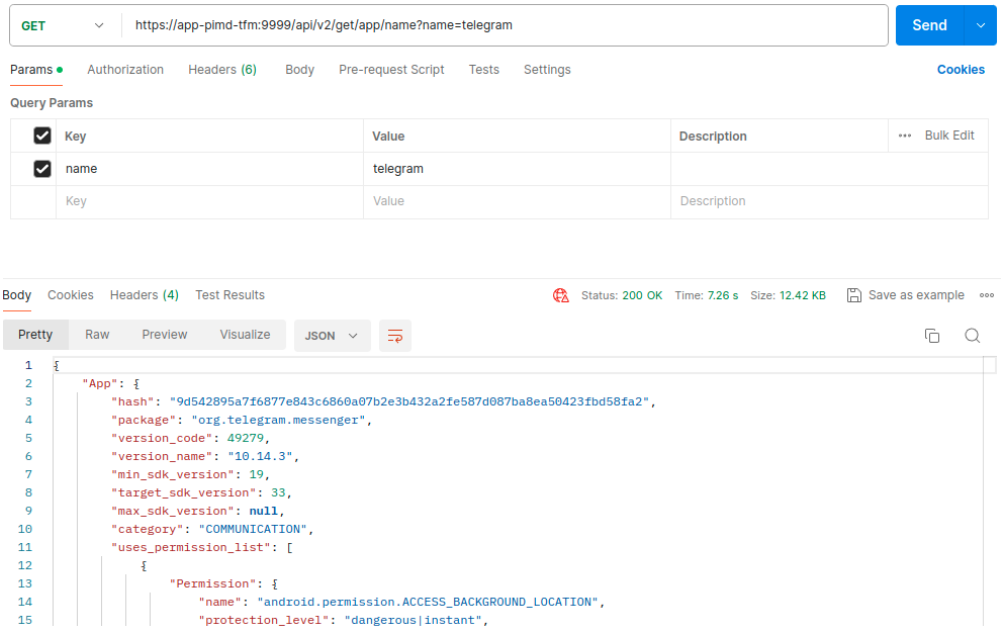

Figura 6.14: Prueba de descarga de los metadatos de una *app* por nombre.

Finalmente, la figura 6.15 muestra la consulta de los metadatos detallados de la aplicación Discord. Esta consulta tarda aproximadamente 270 ms en completarse.

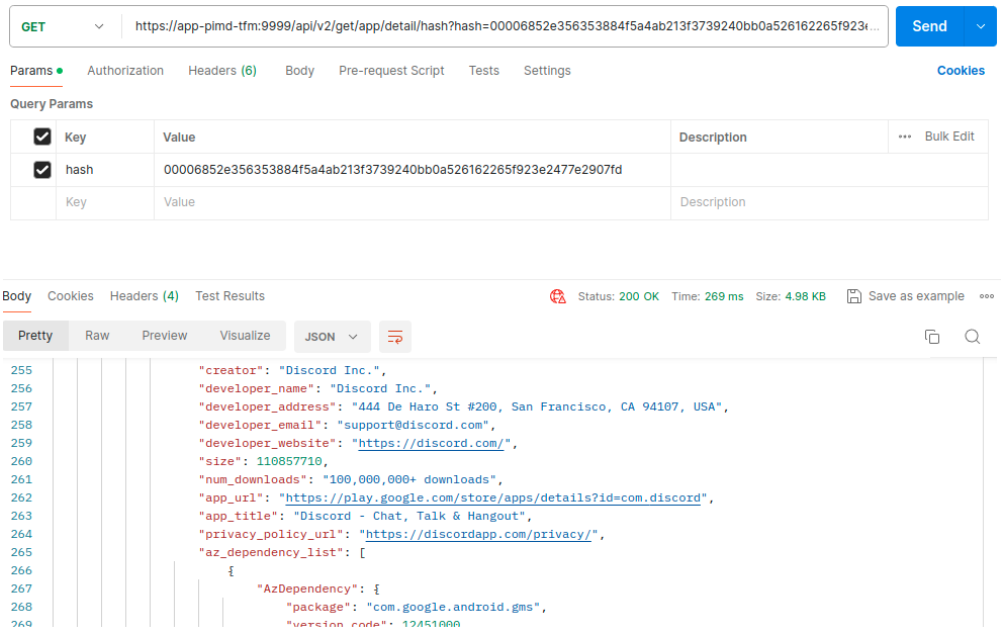

Figura 6.15: Prueba de descarga de metadatos detallados de una *app*.

Capítulo 7

Conclusiones y líneas de trabajo futuro

En este capítulo se desarrollan las conclusiones del trabajo realizado, evaluando el nivel de consecución de los objetivos y los resultados obtenidos. Finalmente, se enumeran las posibles líneas de trabajo futuro que surgen a partir de este proyecto.

7.1. Conclusiones

En este trabajo se ha conseguido poner en valor la importancia de la metodología *Big Data* al alcanzar todos los objetivos planteados en la sección1.3. Gracias la aplicación de esta metodología, se han creado herramientas efectivas para la extracción de conocimiento, como los cinco paneles de mandos desarrollados, y se ha mejorado el repositorio en términos de tipos de datos almacenados y volumen de información. Además, también se ha puesto en valor el *feedback* recibido por todos los usuarios de App-PIMD. Esta retroalimentación ha permitido dar servicio a nueva funcionalidad, como la búsqueda de aplicaciones por nombre, que facilita enormemente la consulta de aplicaciones, y la capacidad de subir aplicaciones mediante archivo, una característica crucial para los desarrolladores que aún no han publicado sus aplicaciones en los principales *markets*.

Partiendo del estado inicial del repositorio mencionado en la sección 1.2.1, se ha conseguido aportar a todos los usuarios de App-PIMD y toda la comunidad investigadora un repositorio que ahora cuenta con 4.420 nuevas aplicaciones provenientes de F-Droid y un total de 7.012 nuevos metadatos sobre valoraciones, dependencias, políticas de privacidad, etc., provenientes de AndrozooGP. Además, se ha proporcionado una nueva versión de la funcionalidad de la API de acceso al repositorio, basándose en el *feedback* recogido por los usuarios del mismo, así como una herramienta de extracción de conocimiento que permite a investigadores y profesionales realizar análisis sobre la privacidad de las aplicaciones con los contenidos del repositorio en *near real time*.

No obstante, pese a las contribuciones que realiza este trabajo, también presenta limitaciones. Debido a las restricciones de Play Store, no es posible acceder a todas sus aplicaciones y metadatos de forma directa, lo que limita las extracciones de contenido a otras fuentes de información y dificulta la obtención de datos. Además, el uso de técnicas de extracción de información mediante *scraping* en los ETL obliga a que el repositorio sea mantenido por una persona con experiencia en este tipo de técnicas para que siga funcionando a lo largo del tiempo. Finalmente, las limitaciones en *hardware* del mismo lo limitan a tener que almacenar solamente metadatos textuales sobre las aplicaciones, debido a que almacenar archivos, iconos u otros tipos de información que pudieran ser de interés requeriría una capacidad de almacenamiento significativamente mayor que la actual necesaria para manejar las 13.398 aplicaciones almacenadas a día de hoy[1].

7.2. Líneas de trabajo futuro

Este trabajo de fin de máster presenta distintas líneas de trabajo futuro posibles, principalmente de tres tipos distintos: las de ampliación de contenido del repositorio, las de mejora en la funcionalidad del repositorio, y las de mejora de los paneles de mandos:

- Ampliar el contenido del repositorio con nuevas fuentes y tipos de datos, lo que permitirá realizar análisis más complejos y avanzados sobre el impacto de las aplicaciones en la privacidad de sus usuarios.
- Implementar nuevas métricas de privacidad con toda la información de la que dispone el repositorio, que funcionen como buenos indicadores del nivel de intrusividad en la privacidad de las aplicaciones.
- Mejorar la funcionalidad del repositorio a medida que se detecten posibles limitaciones o necesidades adicionales, ajustándolo continuamente para garantizar su eficiencia y adaptabilidad. Esto incluye la implementación de nuevas características o la optimización de las ya existentes, con el fin de mantener la relevancia del repositorio y asegurar que siga cumpliendo con las expectativas de los usuarios e investigadores.
- Mejorar la mantenibilidad del repositorio por medio de automatizaciones o con herramientas que simplifiquen su gestión.
- Ampliar y/o mejorar las visualizaciones gráficas de los paneles de mandos para facilitar los análisis más intuitivos y accesibles. Esto podría incluir la incorporación de nuevas visualizaciones, el refinamiento de las gráficas existentes, y la mejora de la interactividad de los paneles.

[1] martes, 20 de agosto de 2024

Bibliografía

[1] A. Pérez-Fuente, «Data Warehouse para el estudio de la privacidad de aplicaciones móviles,» Trabajo Fin de Grado, Universidad de Valladolid, 2023.

[2] Alberto SD. «¿Estás enganchado a la tecnología? El 70 % de los españoles admite estarlo.» (2023), dirección: https://www.kaspersky.es/blog/influencia-tecnologia-spain/28625/. (último acceso: 26/06/2024).

[3] J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador y N. V. Rodriguez, *NOTA TÉCNICA Avance del estudio de IMDEA NETWORKS y UC3M:"Análisis del Software Pre-instalado en Dispositivos Android y sus Riesgos para la Privacidad de los Usuarios"*, 2019.

[4] M. Ramirez. «Tu móvil Samsung tiene todas estas apps preinstaladas que puedes borrar desde ya.» (2023), dirección: https://www.elespanol.com/elandroidelibre/noticias-y-novedades/20230214/samsung-movil-todas-apps-preinstaladas-puedes-borrar-desde-ya/741426139_0.html. (último acceso: 26/06/2024).

[5] Rina, «A Comparative Analysis of mobile Operating Systems,» dirección: https://www.ijcseonline.org/pub_paper/11-IJCSE-05378.pdf.

[6] España. «Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.» (2018), dirección: https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf. (último acceso: 16/05/2023).

[7] Q. Sesé. «La cadena de gimnasios Synergym sufre una filtración de los datos personales de sus usuarios.» (2024), dirección: https://www.elmundo.es/espana/2024/05/10/663e335ae9cf4a340a8b45a1.html. (último acceso: 16/05/2023).

[8] M. García. «El robo de datos de Banco Santander afecta, sobre todo, a los trabajadores.» (2024), dirección: https://www.elindependiente.com/economia/2024/05/15/robo-datos-banco-santander-afecta-trabajadores/. (último acceso: 16/05/2023).

[9] H. Jin, M. Liu, K. Dodhia et al., «Why Are They Collecting My Data?: Inferring the Purposes of Network Traffic in Mobile Apps,» *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, n.º 4, 173:1-173:27, 2018. DOI: 10.1145/3287051. dirección: https://doi.org/10.1145/3287051.

[10] M. Hatamian, N. Momen, L. Fritsch y K. Rannenberg, «A Multilateral Privacy Impact Analysis Method for Android Apps,» en *Privacy Technologies and Policy*, M. Naldi, G. F. Italiano, K. Rannenberg, M. Medina y A. Bourka, eds., Cham: Springer International Publishing, 2019, págs. 87-106, ISBN: 978-3-030-21752-5.

[11] S. Barth, M. de Jong, M. Junger, P. H. Hartel y J. C. Roppelt, «Putting the privacy paradox to the test: Online privacy and security behaviors among users with technical knowledge, privacy awareness, and financial resources,» *Telematics Informatics*, vol. 41, págs. 55-69, 2019.

[12] A. Aparicio, M. Martínez González y V. Cardeñoso, «Métrica basada en grupos de permisos para entender el impacto de las aplicaciones Android sobre la privacidad,» en *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, 2022, págs. 1-5. DOI: 10.23919/CISTI54924.2022.9820147.

[13] J. Crespo Guerrero, «APKFalcon: Servicio de usuarios para la evaluación y comprensión del impacto sobre la privacidad de aplicaciones móviles,» Universidad de Valladolid, 2023.

[14] G. Mancuzo. «Qué es el modelo incremental.» (2021), dirección: https://blog.comparasoftware.com/que-es-el-modelo-incremental/. (último acceso: 01/07/2024).

[15] Android Developers. «Descripción general del manifiesto de la app.» (2024), dirección: https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419. (último acceso: 02/07/2024).

[16] S. Few, *Information Dashboard Design: Displaying Data for At-a-glance Monitoring*. Analytics Press, 2013, ISBN: 9781938377006. dirección: https://books.google.es/books?id=7k0EnAEACAAJ.

[17] España, *Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia*, Boletín Oficial del Estado, núm. 97, 1996.

[18] W. Weichen, G. Jing y C. Rui, «Survey of Big Data Storage Technology,» *Internet of Things and Cloud Computing*, vol. 4, n.º 3, págs. 28-33, 2016. DOI: 10.11648/j.iotcc.20160403.13. dirección: https://doi.org/10.11648/j.iotcc.20160403.13.

[19] A. Siddiqa, A. Karim y A. Gani, «Big Data Storage Technologies: A Survey,» *Frontiers in Information Technology & Electronic Engineering*, vol. 18, n.º 10, págs. 1040-1070, 2017. DOI: 10.1631/FITEE.1500441. dirección: https://doi.org/10.1631/FITEE.1500441.

[20] Android Developers. «Permission tag.» (2022), dirección: https://developer.android.com/guide/topics/manifest/permission-element. (último acceso: 20/07/2024).

[21] Android Developers. «Permission-group tag.» (2023), dirección: https://developer.android.com/guide/topics/manifest/permission-group-element. (último acceso: 20/07/2024).

[22] R. Kimball y J. Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Indianapolis, IN: Wiley, 2004.

[23] R. Fielding, et al., *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, Internet Engineering Task Force, 1999. dirección: https://tools.ietf.org/html/rfc2616.

[24] R. Fielding, et al., *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, RFC 7230, Internet Engineering Task Force, 2014. dirección: https://tools.ietf.org/html/rfc7230.

[25] R. Fielding, et al., *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, RFC 7231, Internet Engineering Task Force, 2014. dirección: https://tools.ietf.org/html/rfc7231.

[26] T. Dierks, E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, Internet Engineering Task Force, 2008. dirección: https://tools.ietf.org/html/rfc5246.

[27] DesignPatternsPHP. «Patrón Data Mapper.» (2024), dirección: https://designpatternsphp.readthedocs.io/es/latest/Structural/DataMapper/README.html. (último acceso: 21/07/2024).

[28] S. Lopez. «Dashboards. ¿Cómo diseñar dashboards óptimos?» (2020), dirección: https://www.hiberus.com/crecemos-contigo/como-disenar-dashboards/.

[29] O. I. (OAI). «OpenAPI.» (2022), dirección: https://www.openapis.org/. (último acceso: 21/07/2024).

[30] Cloudflare. «¿Qué es un proxy inverso? — Servidores proxy explicadosr.» (2024), dirección: https://www.cloudflare.com/es-es/learning/cdn/glossary/reverse-proxy/. (último acceso: 21/07/2024).

[31] M. Alecci, P. J. Ruiz Jiménez, K. Allix, T. F. Bissyandé y J. Klein, «AndroZoo: A Retrospective with a Glimpse into the Future,» en *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, 2024, págs. 389-393.

[32] H. Roy. «Terms of Service Didn't Read.» (2012), dirección: `https://tosdr.org/`. (último acceso: 21/08/2024).

[33] J. Pomeyrol. «Guía de instalación de Ubuntu 20.04 LTS.» (2020), dirección: `https://www.muylinux.com/2020/05/21/guia-instalacion-ubuntu-20-04-lts/`. (último acceso: 28/06/2024).

[34] K. Allix, T. F. Bissyandé, J. Klein e Y. Le Traon, «AndroZoo: Collecting Millions of Android Apps for the Research Community,» en *Proceedings of the 13th International Conference on Mining Software Repositories*, ép. MSR '16, Austin, Texas: ACM, 2016, págs. 468-471, ISBN: 978-1-4503-4186-8. DOI: 10.1145/2901739.2903508. dirección: http://doi.acm.org/10.1145/2901739.2903508.

Apéndice A

Guía de instalación de App-PIMD

En este apéndice se redacta una guía de instalación de todos los componentes del repositorio App-PIMD así como su puesta en marcha. Por motivos de seguridad, las contraseñas utilizadas se encuentran censuradas con la palabra "*password*" en las figuras y/o códigos que se muestren.

Como precondición al proceso de instalación del repositorio, se ha de tener un sistema Linux instalado y en funcionamiento. En el caso concreto de esta instalación, se utiliza un sistema Linux (Ubuntu 24.04 LTS) ya que tiene soporte de actualizaciones de seguridad y mantenimiento hasta Junio de 2029. El sistema se ejecuta sobre una máquina virtual con 4 núcleos de CPU, 4 GB de memoria RAM y 25 GB de disco duro de almacenamiento. Existen múltiples guías de instalación de este sistema operativo [33].

A.1. Descarga del código fuente de App-PIMD

En primer lugar, es necesario tener instalado el sistema de control de versiones Git en nuestro sistema. Para ello, en el caso de los sistemas Linux basados en Ubuntu, basta con ejecutar en una terminal la siguiente línea de código:

```
sudo apt install git
```

Una vez tenemos Git en nuestro sistema, nos dirigimos con el comando *cd* hasta el directorio en que queremos instalar el repositorio. Una vez en el, ejecutamos el comando:

```
> git clone https://github.com/peres317/app-pimd-tfm.git
```

Este comando descargara todo el código necesario para la instalación de App-PIMD. Por último, una vez descargado el código, nos movemos al directorio del proyecto desde donde se partirá para la instalación y configuración del resto de componentes.

```
> cd ./app_warehouse/
```

A.2. Instalación y configuración de MySQL

Para instalar el sistema gestor de bases de datos MySQL, en el caso de los sistemas Linux basados en Ubuntu, ejecutamos la siguiente línea de código:

```
> sudo apt install mysql-server
```

Una vez instalado el sistema gestor de bases de datos, pasamos a configurarlo siguiendo los siguientes pasos:

1. Modificamos *"password"* en el archivo de configuración *utils/mysql-config.sql* por una contraseña segura que utilizará la API para acceder a su base de datos en MySQL.

   Es muy recomendable establecer contraseñas diferentes para el usuario de acceso al repositorio de datos (*load*) y el de acceso a la base de datos de credenciales (*credentials*). De este modo, en caso de filtración de una de ellas, el acceso a la otra base de datos queda protegido.

2. Entramos en MySQL con el usuario administrador con el comando:

   ```
   > sudo mysql
   ```

   Y modificamos la contraseña de acceso *root* del MySQL:

   ```
   mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH
          mysql_native_password BY 'password';
   mysql> EXIT
   ```

3. Una vez configurado el usuario administrador de la base de datos, entramos y ejecutamos el archivo de creación de los usuarios que usará la API para acceder a la base de datos con el siguiente comando:

   ```
   > mysql -u root -p
   mysql> source utils/mysql-config.sql
   ```

4. Por último, ejecutamos los *scripts* de creación de las tablas y entradas del último estado guardado del repositorio:

   ```
   > mysql -u root -p app_warehouse < utils/warehouse-backup.sql
   > mysql -u root -p app_warehouse_credentials <
                              utils/warehouse_credentials-backup.sql
   ```

A.3. Instalación y configuración del entorno Python

Primero se instala Python 3.10 y una dependencia para crear entornos virtuales con los siguientes comandos:

```
> sudo add-apt-repository ppa:deadsnakes/ppa -y
> sudo apt update
> sudo apt install python3.10 python3.10-venv python3.10-dev
```

Una vez instalado Python, se ejecuta el siguiente *script* que creará un entorno virtual Python con las dependencias necesarias para ejecutar App-PIMD. Es posible que de error de permiso denegado, en cuyo caso hay que revisar que el archivo que contiene el *script* tiene permiso de ejecución.

```
> ./utils/python-config.sh
```

Por último, hay que modificar el código de la librería *pyandrozoo* para que descargue las aplicaciones en el directorio correcto, y con un *timeout* de seguridad sensato para que no se quede bloqueado en descargas muy largas.

```
nano venv/lib/python3.10/site-packages/pyandrozoo/api.py
```

```
  GNU nano 6.2   venv/lib/python3.10/site-packages/pyandrozoo/api.py *
20      def __write_file(self, r, **kwargs):
21          sha256 = parse_qs(urlparse(r.url).query)['sha256'][0]
22          if r.status_code == 200:
23               with open(sha256+'.apk', "wb") as apk_file:
24                   apk_file.write(r.content)
25          else:
26              print('Request failed for: {}'.format(sha256))
27          return r
28
29      def get(self, sha256_list):
30          reqs = []
31          parts = urlparse(self.root_url)
32          for sha256 in sha256_list:
33              payload = {'sha256': sha256}
34              payload.update(self.payload)
35              url = urlunparse(parts[:4] + (urlencode(payload),) + parts[5:])
36              reqs.append(grequests.get(url,
37                                        hooks={'response': [self.__write_file]}))
38          grequests.map(reqs, size=30, exception_handler=exception_handler)
```

Figura A.1: Paquete original *pyandrozoo*.

Las figuras A.1 y A.2 muestra como se modifica el paquete en las líneas 23 y 37. Para almacenar el fichero si se ha utilizado el editor propuesto (*nano*), se usa la combinación de teclas Ctrl + X.

```
  GNU nano 6.2   venv/lib/python3.10/site-packages/pyandrozoo/api.py *
20      def __write_file(self, r, **kwargs):
21          sha256 = parse_qs(urlparse(r.url).query)['sha256'][0]
22          if r.status_code == 200:
23               with open('data/androzoo_downloads/'+sha256+'.apk', "wb") as ap>
24                   apk_file.write(r.content)
25          else:
26              print('Request failed for: {}'.format(sha256))
27          return r
28
29      def get(self, sha256_list):
30          reqs = []
31          parts = urlparse(self.root_url)
32          for sha256 in sha256_list:
33              payload = {'sha256': sha256}
34              payload.update(self.payload)
35              url = urlunparse(parts[:4] + (urlencode(payload),) + parts[5:])
36              reqs.append(grequests.get(url,
37                                        hooks={'response': [self.__write_file]},
38                                        timeout=300))
39          grequests.map(reqs, size=30, exception_handler=exception_handler)
```

Figura A.2: Paquete modificado *pyandrozoo*.

A.4. Configuración de App-PIMD

En primer lugar, se crea el fichero de configuración de App-PIMD *data/config.json*. Este ha de contener las contraseñas que hemos puesto a la base de datos, la clave de API facilitada por Androzoo, el directorio absoluto del repositorio y el directorio del fichero índice comprimido de aplicaciones de Androzoo [34]. A continuación se muestra la estructura que ha de tener este fichero:

```
{
    "ABSOLUTE_PATH_OF_WAREHOUSE": "/PATH/TO/app_warehouse/",
    "MYSQL": {
        "host": "127.0.0.1",
        "database": "app_warehouse",
        "user": "load",
        "password": "YOUR_MYSQL_LOAD_USER_PASSWORD"
    },
    "MYSQL_CREDENTIALS": {
        "host": "127.0.0.1",
        "database": "app_warehouse_credentials",
        "user": "credentials",
        "password": "YOUR_MYSQL_CREDENTIALS_USER_PASSWORD"
    },
    "ANDROZOO": {
        "API_KEY": "YOUR_ANDROZOO_API_KEY",
```

```
        "INDEX_FILE": "/PATH/TO/app_warehouse/data/latest.csv.gz"
    }
}
```

Puesto que este archivo de configuración contiene contraseñas es muy importante modificar sus permisos y eliminar el permiso de lectura a todos los usuarios a excepción del que ejecutará el repositorio. A continuación, creamos la estructura de directorios de trabajo del repositorio con los siguientes comandos:

```
> mkdir data/androzoo_downloads
> mkdir data/app_downloads
> mkdir data/google_downloads
> mkdir data/play_store_downloads
```

Descargamos el índice de aplicaciones de Androzoo:

```
> cd data && wget https://androzoo.uni.lu/static/lists/latest.csv.gz
> cd ..
```

Y finalmente, se modifica la línea 4 del archivo *api/main.py* para que contenga el directorio base del proyecto.

A.5. Instalación y configuración de Grafana

En primer lugar, se instala Grafana siguiendo su guía de instalación[1] para sistemas basados en Ubuntu, y se inicia con los siguientes comandos:

```
> sudo systemctl enable grafana-server
> sudo systemctl start grafana-server
```

En segundo lugar, se configura. Para ello, se accede a su página alojada por defecto en el puerto 3000 con el *login* por defecto admin/admin y se modifica la contrseña (ver figura A.3).

También es necesario incluir la fuente de datos y los *dashboards* desarrollados. Para ello, se añade una nueva conexión de datos tipo MySQL a *localhost:3306* como se muestra en la figura A.4, y se importan los JSON de los *dashboards* que se encuentran en la carpeta "*utils*" del proyecto con el sufijo "*.dashboard.json*".

Finalmente, se configura para que sepa que se encuentra detrás de un *proxy* inverso. Para ello, en el archivo de configuración */etc/grafana/grafana.ini* se descomenta la línea que especifica el *root_url* y se añade el prefijo *dashboard* al final:

[1] `https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/`

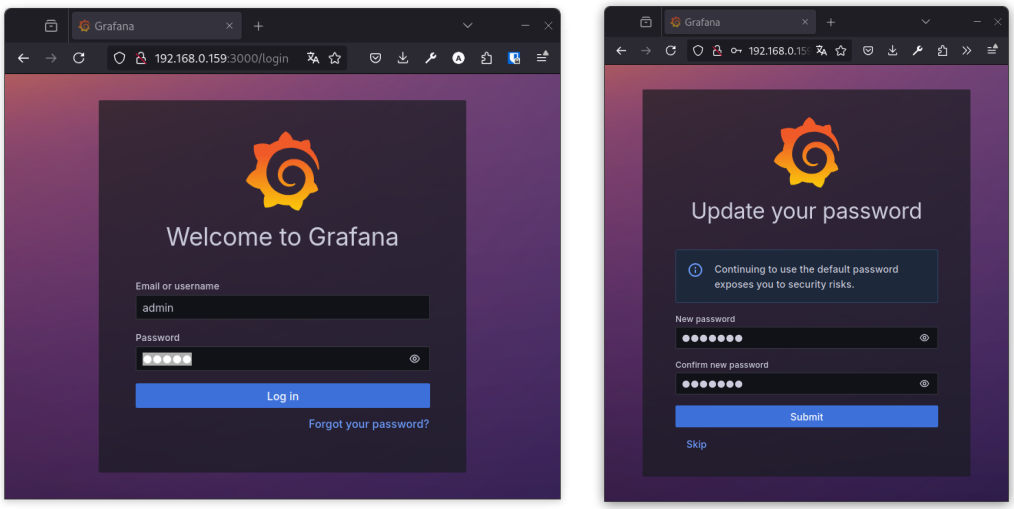

Figura A.3: Primer inicio en Grafana.

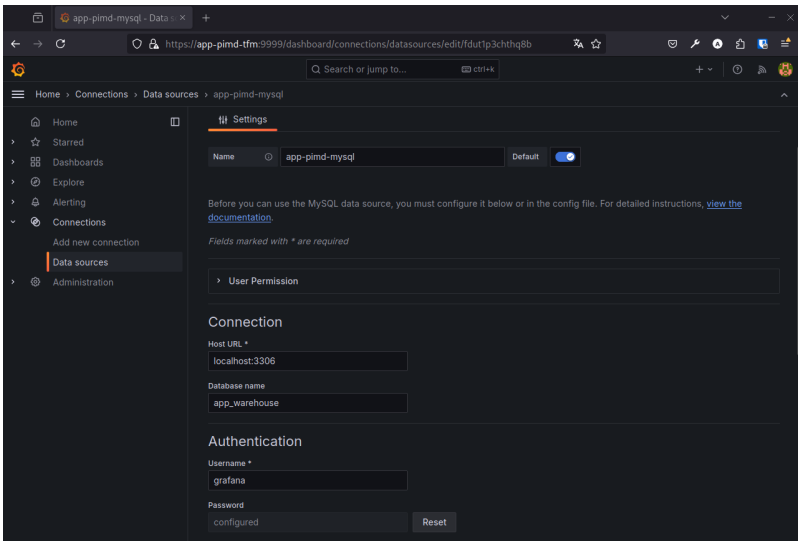

Figura A.4: Configuración de la fuente de datos de Grafana.

```
root_url = %(protocol)s://%(domain)s:%(http_port)s/dashboard
```

Para que los cambios se vean aplicados es necesario reiniciar el servicio.

```
> sudo systemctl restart grafana-server
```

A.6. Instalación y configuración del *proxy*

Para poder servir tanto la API del repositorio como el *dashboard* en Grafana bajo una mismo puerto, y de modo uniforme, se utiliza un *proxy* inverso. Además, de este modo se pueden securizar ambos recursos tras el protocolo SSL[2]. El *proxy* utilizado es Traefik

[2]SSL: *Secure Sockets Layer*

por su gran lista de ventajas entre las que se encuentran su facilidad de uso, seguridad integrada, ligereza, y eficiencia.

Para su instalación, descargamos la última *release* de su Git[3] correspondiente a nuestro sistema operativo, en nuestro caso *traefik_v3.1.0-rc2_linux_amd64.tar.gz*. Una vez descargado, extraemos el contenido, movemos el binario al directorio de binarios del sistema operativo, y creamos y copiamos los directorios de trabajo con las configuraciones ya hechas en *utils/traefik/*.

```
> wget https://github.com/traefik/traefik/releases/download/
                v3.1.0-rc2/traefik_v3.1.0-rc2_linux_amd64.tar.gz
> tar -zxvf traefik_v3.1.0-rc2_linux_amd64.tar.gz
> sudo mv traefik /bin/
> sudo cp -R utils/traefik /etc/
> sudo mkdir /var/log/traefik
```

Para cifrar el tráfico HTTPS tenemos que generar un certificado en la ruta */etc/traefik/-certificates/* con el siguiente comando:

```
> openssl req -x509 -newkey rsa:4096
                            -keyout /etc/traefik/certificates/key.pem
                            -out /etc/traefik/certificates/cert.pem
                            -sha256 -days 365
```

A.7. Ejecución de App-PIMD

Para iniciar el repositorio partiendo de dos terminales en el directorio base del código. En primer lugar, se ha de lanzar la API de acceso al mismo en una de ellas:

```
> source venv/bin/activate
> python3.10 api/main.py
```

En segundo lugar, se ha de lanzar en paralelo el *proxy* en la otra terminal:

```
> sudo traefik
```

El resto de servicios como MySQL o Grafana deberían estar iniciados desde el arranque del sistema si ha seguido la guía de instalación paso por paso.

[3] `https://github.com/traefik/traefik/releases`

Apéndice B

Documentación App-PIMD API v2

API Reference

App-PIMD

2: App-PIMD

La API de App-PIMD te permite descargar metadatos sobre apps Android.

v2

La funcionalidad disponible es:

- Obtener todas las versiones almacenadas de una app.
- Subir una app al repositorio por archivo.
- Descargar los metadatos de una app por su nombre.
- Descargar los metadatos detallados de una app por su hash.

NAME: app-warehouse

EMAIL: app.warehouse.uva@gmail.com

INDEX

API

1. V2

Funcionalidad disponible en API v2.

1.1 GET /v2/get/app/versions

Obtener todas las versiones almacenadas de una app.

Descarga los pares (hash, version) almacenados para la app solicitada (ej. *net.universia.uva*).

PATH PARAMS

NAME	TYPE	REQUIRED	EXAMPLE	DESCRIPTION
package	string	+		nombre de paquete de la app a descargar

RESPONSES

STATUS CODE - 200: Pares (hash, version) almacenados para la app solicitada.

RESPONSE MODEL - application/json

```
{
    app_list [
        {
            app_hash        string   required
            version_name    string   required
        }
    ]
}
```

EXAMPLE

```
{
  "app_list": [
    {
      "app_hash": "string",
      "version_name": "string"
    }
  ]
}
```

STATUS CODE - 404: La app solicitada no se encuentra almacenada.

RESPONSE MODEL - application/json

```
{
    detail   string   required
}
```

EXAMPLE

```
{
```

```
  "detail": "string"
}
```


STATUS CODE - 422: Error de validación.

RESPONSE MODEL - application/json

```
{
    detail [
        {
            loc [
                ANY OF
                OPTION 1
                string
                OPTION 2
                integer
            ]
            msg   string   required
            type   string   required
        }
    ]
}
```

EXAMPLE


```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```


1.2 POST /v2/post/app/file

Subir una app al repositorio por archivo.

Permite cargar una app subiendo su archivo apk.

REQUEST

REQUEST BODY - multipart/form-data

```
{
    file   string   required
}
```

EXAMPLE


```
{
  "file": "string"
}
```


RESPONSES

STATUS CODE - 200: Estado de la petición (busy, requested)

RESPONSE MODEL - application/json

```
{
    detail   string   required
}
```

EXAMPLE

```
{
  "detail": "string"
}
```

STATUS CODE - 400: El archivo subido no es válido.

RESPONSE MODEL - application/json

```
{
    detail   string   required
}
```

EXAMPLE

```
{
  "detail": "string"
}
```

STATUS CODE - 422: Error de validación.

RESPONSE MODEL - application/json

```
{
    detail [
        {
            loc [
                    ANY OF
                    OPTION 1
                    string
                    OPTION 2
                    integer
            ]
            msg   string   required
            type   string   required
        }
    ]
}
```

EXAMPLE

```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

1.3 GET /v2/get/app/name

Descargar los metadatos de una app por su nombre.

Permite descargar los metadatos de una app por su nombre en Play Store (ej. *Whatsapp* o *Telegram Messenger*).

PATH PARAMS

NAME	TYPE	REQUIRED	EXAMPLE	DESCRIPTION
name	string	+		Nombre de la app a descargar.

RESPONSES

STATUS CODE - 200: Metadatos de la app.

RESPONSE MODEL - application/json

```
{
    App {
        hash                        string          required
        package                     string          required
        version_code                integer         required
        version_name                string          required
        min_sdk_version             integer
        target_sdk_version          integer
        max_sdk_version             integer
        category                    string
        uses_permission_list [
            {
                Permission {
                    name                    string      required
                    protection_level        string
                    declared_group_list [
                        {
                            PermissionGroup {
                                name  string  required
                            }
                        }
                    ]
                    rank_list [
                        {
                            Rank {
                                value           number  required
                                rank_name       string  required
                                permission_name string  required
                            }
                        }
                    ]
                }
            }
        ]
        defines_permission_list [
            {
                Permission {
                    name                    string      required
                    protection_level        string
                    declared_group_list [
                        {
                            PermissionGroup {
                                                        required
```

```
                        name   string
                    }
                }
            ]
            rank_list [
                {
                    Rank {
                        value                number   required
                        rank_name            string   required
                        permission_name      string   required
                    }
                }
            ]
        }
    }
]
defines_group_list [
    {
        PermissionGroup {
            name   string   required
        }
    }
]
extraction_metadata_list [
    {
        ExtractionMetadata {
            source      string   required
            method      string   required
            timestamp   string   required
        }
    }
]
score_list [
    {
        Score {
            value       number
            rank_name   string   required
            app_hash    string   required
        }
    }
]
}
}
```

EXAMPLE


```
{
  "App": {
    "hash": "string",
    "package": "string",
    "version_code": 0,
    "version_name": "string",
    "min_sdk_version": 0,
    "target_sdk_version": 0,
    "max_sdk_version": 0,
    "category": "string",
    "uses_permission_list": [
      {
        "Permission": {
          "name": "string",
          "protection_level": "string",
```

```
      "declared_group_list": [
       {
        "PermissionGroup": {
         "name": "string"
        }
       }
      ],
      "rank_list": [
       {
        "Rank": {
         "value": 0,
         "rank_name": "string",
         "permission_name": "string"
        }
       }
      ]
     }
    }
   ],
   "defines_permission_list": [
    {
     "Permission": {
      "name": "string",
      "protection_level": "string",
      "declared_group_list": [
       {
        "PermissionGroup": {
         "name": "string"
        }
       }
      ],
      "rank_list": [
       {
        "Rank": {
         "value": 0,
         "rank_name": "string",
         "permission_name": "string"
        }
       }
      ]
     }
    }
   ],
   "defines_group_list": [
    {
     "PermissionGroup": {
      "name": "string"
     }
    }
   ],
   "extraction_metadata_list": [
    {
     "ExtractionMetadata": {
```

```
        "source": "string",
        "method": "string",
        "timestamp": "string"
       }
     }
    ],
    "score_list": [
     {
      "Score": {
       "value": 0,
       "rank_name": "string",
       "app_hash": "string"
      }
     }
    ]
  }
}
```


STATUS CODE - 404: La app solicitada no se encuentra almacenada.

RESPONSE MODEL - application/json

```
{
    detail   string   required
}
```

EXAMPLE


```
{
  "detail": "string"
}
```


STATUS CODE - 422: Error de validación.

RESPONSE MODEL - application/json

```
{
    detail [
        {
            loc [
                ANY OF
                OPTION 1
                string
                OPTION 2
                integer
            ]
            msg   string   required
            type   string   required
        }
    ]
}
```

EXAMPLE


```
{
  "detail": [
   {
    "loc": [
     "string"
    ],
    "msg": "string",
```

```
    "type": "string"
  }
 ]
}
```

1.4 GET /v2/get/app/detail/hash

Descargar los metadatos detallados de una app por su hash.

Permite descargar los metadatos detallados de una app por su hash. (ej. *00006852e356353884f5a4ab213f3739240bb0a526162265f923e2477e2907fd*).

PATH PARAMS

NAME	TYPE	REQUIRED	EXAMPLE	DESCRIPTION
hash	string	+		Hash de la app a descargar.

RESPONSES

STATUS CODE - 200: Metadatos detallados de la app.

RESPONSE MODEL - application/json

```
{
  AppDetail {
    hash                    string          required
    package                 string          required
    version_code            integer         required
    version_name            string          required
    min_sdk_version         integer
    target_sdk_version      integer
    max_sdk_version         integer
    category                string
    uses_permission_list [
      {
        Permission {
          name                  string        required
          protection_level      string
          declared_group_list [
            {
              PermissionGroup {
                name  string  required
              }
            }
          ]
          rank_list [
            {
              Rank {
                value           number  required
                rank_name       string  required
                permission_name string  required
              }
            }
          ]
        }
      }
```

```
]
defines_permission_list [
    {
        Permission {
            name                    string          required
            protection_level        string
            declared_group_list [
                {
                    PermissionGroup {
                        name  string  required
                    }
                }
            ]
            rank_list [
                {
                    Rank {
                        value                number  required
                        rank_name            string  required
                        permission_name      string  required
                    }
                }
            ]
        }
    }
]
defines_group_list [
    {
        PermissionGroup {
            name  string  required
        }
    }
]
extraction_metadata_list [
    {
        ExtractionMetadata {
            source      string  required
            method      string  required
            timestamp   string  required
        }
    }
]
score_list [
    {
        Score {
            value       number
            rank_name   string  required
            app_hash    string  required
        }
    }
]
az_metadata_list [
    {
        AzMetadata {
            app_hash                string  required
            az_metadata_date        string  required
            ratings_count           integer
            star_rating             number
            comment_count           integer
            one_star_ratings        integer
            two_star_ratings        integer
            three_star_ratings      integer
            four_star_ratings       integer
            five_star_ratings       integer
            upload_date             string
```

```
                creator                 string
                developer_name          string
                developer_address       string
                developer_email         string
                developer_website       string
                size                    integer
                num_downloads           string
                app_url                 string
                app_title               string
                privacy_policy_url      string
                az_dependency_list [
                    {
                        AzDependency {
                            package         string    required
                            version_code    integer   required
                        }
                    }
                ]
            }
        }
    ]
  }
}
```

EXAMPLE


```
{
  "AppDetail": {
    "hash": "string",
    "package": "string",
    "version_code": 0,
    "version_name": "string",
    "min_sdk_version": 0,
    "target_sdk_version": 0,
    "max_sdk_version": 0,
    "category": "string",
    "uses_permission_list": [
      {
        "Permission": {
          "name": "string",
          "protection_level": "string",
          "declared_group_list": [
            {
              "PermissionGroup": {
                "name": "string"
              }
            }
          ],
          "rank_list": [
            {
              "Rank": {
                "value": 0,
                "rank_name": "string",
                "permission_name": "string"
              }
            }
          ]
        }
```

```
    }
   ],
   "defines_permission_list": [
    {
     "Permission": {
      "name": "string",
      "protection_level": "string",
      "declared_group_list": [
       {
        "PermissionGroup": {
         "name": "string"
        }
       }
      ],
      "rank_list": [
       {
        "Rank": {
         "value": 0,
         "rank_name": "string",
         "permission_name": "string"
        }
       }
      ]
     }
    }
   ],
   "defines_group_list": [
    {
     "PermissionGroup": {
      "name": "string"
     }
    }
   ],
   "extraction_metadata_list": [
    {
     "ExtractionMetadata": {
      "source": "string",
      "method": "string",
      "timestamp": "string"
     }
    }
   ],
   "score_list": [
    {
     "Score": {
      "value": 0,
      "rank_name": "string",
      "app_hash": "string"
     }
    }
   ],
   "az_metadata_list": [
    {
```

```
      "AzMetadata": {
        "app_hash": "string",
        "az_metadata_date": "string",
        "ratings_count": 0,
        "star_rating": 0,
        "comment_count": 0,
        "one_star_ratings": 0,
        "two_star_ratings": 0,
        "three_star_ratings": 0,
        "four_star_ratings": 0,
        "five_star_ratings": 0,
        "upload_date": "string",
        "creator": "string",
        "developer_name": "string",
        "developer_address": "string",
        "developer_email": "string",
        "developer_website": "string",
        "size": 0,
        "num_downloads": "string",
        "app_url": "string",
        "app_title": "string",
        "privacy_policy_url": "string",
        "az_dependency_list": [
          {
            "AzDependency": {
              "package": "string",
              "version_code": 0
            }
          }
        ]
      }
    }
  ]
}
```


STATUS CODE - 404: La app solicitada no se encuentra almacenada.

RESPONSE MODEL - application/json

```
{
    detail   string   required
}
```

EXAMPLE


```
{
  "detail": "string"
}
```


STATUS CODE - 422: Error de validación.

RESPONSE MODEL - application/json

```
{
    detail [
        {
            loc [
                ANY OF
```

OPTION 1
string

OPTION 2
integer

]
msg string required
type string required
}
]
}

EXAMPLE

```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

API List

v2

METHOD	API
get	/v2/get/app/versions Obtener todas las versiones almacenadas de una app.
post	/v2/post/app/file Subir una app al repositorio por archivo.
get	/v2/get/app/name Descargar los metadatos de una app por su nombre.
get	/v2/get/app/detail/hash Descargar los metadatos detallados de una app por su hash.