ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PII-DETRI-2021-06: MEJORA DE LAS TÉCNICAS DE EVALUACIÓN DE LA INTEGRIDAD CONTEXTUAL DE LA PRIVACIDAD DE LAS APLICACIONES MÓVILES

DESARROLLO DE MICROSERVICIOS PARA LA EXTRACCIÓN DE POLÍTICAS DE PRIVACIDAD DESDE ARTEFACTOS DE UNA APLICACIÓN MÓVIL ANDROID

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE INFORMACIÓN

JULIO CESAR MIRANDA CARRASCO

julio.miranda01@epn.edu.ec

DIRECTOR: DANNY SANTIAGO GUAMÁN LOACHAMÍN

danny.guaman@epn.edu.ec

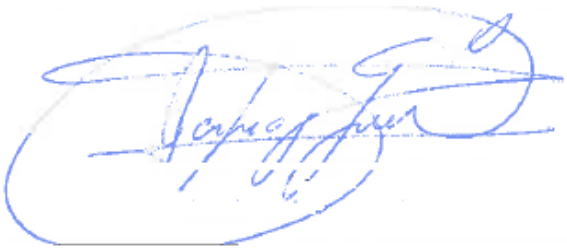
DMQ, septiembre de 2022

CERTIFICACIONES

Yo, Julio Cesar Miranda Carrasco declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

JULIO CESAR MIRANDA CARRASCO

Certifico que el presente trabajo de integración curricular fue desarrollado por Julio Cesar Miranda Carrasco, bajo mi supervisión.



DANNY SANTIAGO GUAMÁN LOACHAMÍN

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Julio Cesar Miranda Carrasco

Danny Santiago Guamán Loachamín

DEDICATORIA

Dedico este trabajo a mi familia, que sin su ayuda y apoyo nada de esto sería posible. Especialmente a mi madre y mi padre que siempre me apoyaron en cada uno de los momentos difíciles de mi vida, los amo mucho.

AGRADECIMIENTO

Agradezco a Dios por brindarme esta oportunidad de pertenecer a la distinguida institución EPN. Agradezco a mi familia, especialmente a mi madre, mi padre y mi abuela, sin su gran apoyo no habría llegado tan lejos. Agradezco a todos y cada uno de los compañeros, amigos quienes me acompañaron durante mi estancia en la universidad.

Agradezco a cada uno de los docentes que ha influido en mi desarrollo personal y académico durante formación profesional.

Agradezco a mi tutor, el PhD. Danny Guamán por el gran apoyo y guía para el desarrollo del presente trabajo.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

PALABRAS CLAVE: privacidad, políticas de privacidad, protección de datos, Android.

Este Trabajo de Integración Curricular presenta el desarrollo de microservicios que contribuyen a la evaluación de cumplimiento de requisitos de privacidad y protección de datos en las aplicaciones móviles. Concretamente, el componente desarrollado consiste en tres microservicios que permiten mejorar ciertos módulos de evaluación, que han sido desarrollados anteriormente como parte del proyecto de investigación interno de la Escuela Politécnica Nacional PII-DETRI-2021-03.

El componente se conforma de 3 microservicios, los cuales fueron desarrollados en un ambiente de desarrollo PyCharm y el lenguaje de programación Python junto a varias bibliotecas que facilitaron el análisis:

- El microservicio 1, basado en los permisos de las aplicaciones, determina automáticamente sí la aplicación debería divulgar una política de privacidad.
- El microservicio 2 extrae las cadenas de caracteres embebidas en un APK y detecta las URLs de documentos que se relacionen con políticas de privacidad.
- El microservicio 3 recupera el contenido de una página web que contiene la política de privacidad de una aplicación.

Los 3 microservicios han sido contenerizados en imágenes Docker para facilitar su despliegue.

Este documento se organiza de la siguiente manera: en el Capítulo 1 se presenta el marco teórico requerido para entender el presenta trabajo, incluyendo una descripción breve de privacidad y protección de datos personales, la Ley Orgánica de Protección de Datos Personales (LOPDP) y las políticas de privacidad. Además, se describen las tecnologías empleadas en el este trabajo, incluyendo los microservicios, el sistema operativo Android y su modelo de permisos Android, y las herramientas y bibliotecas relevantes de Python. En el Capítulo 2 se presenta la metodología usada en el desarrollo de los microservicios, basada en la metodología ágil, Scrum. El proceso de desarrollo se llevó a cabo a través de 3 fases: análisis, diseño e implementación. Además, se presenta los resultados y discusión. Finalmente, en el capítulo 3 se presenta las conclusiones de este trabajo, así como recomendaciones respecto al desarrollo de trabajos futuros.

ABSTRACT

KEYWORDS: privacy, privacy policy, data protection, Android

This work presents the development of microservices that contribute to the compliance assessment of mobile applications with privacy and data protection requirements. Specifically, it developed a component consisting of three microservices that allow improving certain assessment modules, which have been previously developed as part of the research project PII-DETRI-2021-03 in the Escuela Politécnica Nacional .

The component consists of 3 microservices, which were developed in a PyCharm development environment and the Python programming language along with several libraries to support the analysis:

- Microservice 1 automatically determines whether the application should disclose a privacy policy, based on application permissions.
- Microservice 2 extracts strings embedded in an APK and detects document URL's related to privacy policies.
- Microservice 3 retrieves the content of a web page containing the app's privacy policy.

The 3 microservices have been contained in Docker images for ease of deployment.

This document is organized as follows: Chapter 1 presents the theoretical framework required to understand this work, including a brief description of privacy and personal data protection, the "Ley Orgánica de Protección de Datos" (LOPDP) and privacy policies. In addition, it describes the technologies employed in this work, including microservices, the Android operating system and its Android permissions model, and the relevant Python tools and libraries. Chapter 2 presents the methodology used in the development of the microservices, based on the agile methodology, Scrum. The development process was carried out through 3 phases: analysis, design and implementation. In addition, the results and discussion are presented. Finally, Chapter 3 presents the conclusions of this work, as well as recommendations as per the development of future work.

1 INTRODUCCIÓN

En Ecuador, a febrero de 2021, el 77,8% de las conexiones a Internet provienen de terminales móviles [1]. En nuestro país, el mercado de estos terminales móviles está dominado principalmente por el sistema operativo Android (alrededor del 88%) [2]. Un terminal móvil, a través de sus aplicaciones, nos proporciona numerosas utilidades y sobre todo permite el acceso a un gran número de servicios (p.ej., educativos, bancarios, sanitarios, redes sociales, ...). No obstante, a través de estas aplicaciones, las organizaciones pueden recolectar una gran cantidad de datos personales [3].

En Ecuador, las organizaciones que prestan estos servicios y que recogen los datos personales de los usuarios desde los terminales móviles (p.ej., datos de contacto, localización e identificadores) tienen que cumplir con un marco regulatorio de protección de datos. En concreto, la recientemente aprobada Ley Orgánica de Protección de Datos Personales (LOPDP) establece un conjunto de principios y requisitos que las organizaciones, y sus sistemas de software, tienen que cumplir cuando procesan datos personales; su incumplimiento puede suponer sanciones de hasta el 1% de la facturación [4]. Por ejemplo, las organizaciones que no mantengan políticas de privacidad afines al tratamiento de datos personales incurrirán en una infracción leve de acuerdo con la LOPDP, por lo que podrían ser sancionadas con 10 Salarios Básicos Unificados o una multa de entre el 0.1% y el 0.7% calculada sobre su volumen de negocio.

Con el fin de contribuir con la provisión de técnicas y herramientas para la evaluación de cumplimiento en el ecosistema de las aplicaciones móviles, investigadores de la Escuela Politécnica Nacional y la Universidad Politécnica de Madrid han desarrollado la “plataforma CLIIP”. Esta plataforma se apoya en una arquitectura basada en microservicios y en un conjunto de técnicas de análisis estático y dinámico que permiten evaluar diversos aspectos de la privacidad y protección de datos. Para mejorar la plataforma CLIIP, actualmente está en marcha el proyecto de investigación PII-DETRI-2021-06, que se centra en mejorar las técnicas para evaluar la integridad contextual de la privacidad. No obstante, durante la ejecución del proyecto de investigación se ha encontrado varios desafíos a la hora de identificar y extraer automáticamente las políticas de privacidad de las aplicaciones, un insumo esencial para este trabajo. Algunos desafíos relacionados con la extracción de políticas de privacidad son:

Desafío 1. Varias aplicaciones disponibles en el Play Store en Ecuador no publican las URLs (Uniform Resource Locator) de sus políticas de privacidad en la consola de Play Store, pese a potencialmente estar procesando datos personales. Por lo tanto, es

necesario un mecanismo rápido que permita determinar automáticamente potenciales incumplimientos, es decir, si una aplicación que podría estar procesando datos personales no divulga su política de privacidad.

Desafío 2. La URL de una política de privacidad puede estar embebida en el interior de una APK (Android PacKage). Por lo tanto, se requiere un mecanismo automático para descubrir y extraer la política de privacidad (su URL) desde un APK usando técnicas de análisis estático de código.

Desafío 3. Varias políticas de privacidad están alojadas en sitios web que usan tecnologías no convencionales como JavaScript y, por tanto, no se puede extraer el texto de la política con una simple petición GET. Además, las políticas no necesariamente están representadas en formato HTML, sino en formato PDF o de Google Drive. Para abordar este desafío se requiere analizar la variada casuística de problemas encontrados y proponer las soluciones pertinentes.

1.1 OBJETIVO GENERAL

Desarrollar un conjunto de microservicios para la extracción de políticas de privacidad desde los artefactos de una aplicación móvil Android.

1.2 OBJETIVOS ESPECÍFICOS

- Analizar los requerimientos de cada microservicio en función de los desafíos encontrados.
- Diseñar un conjunto de microservicios que satisfagan los requerimientos especificados.
- Implementar los microservicios.
- Evaluar los resultados de los microservicios implementados.

1.3 ALCANCE

Se desarrollará un conjunto de tres microservicios que permitan dar solución a los desafíos antes descritos que se han detectado durante la extracción de las políticas de privacidad de aplicaciones Android:

Microservicio 1. Permitirá determinar automáticamente si una aplicación podría caer en el incumplimiento por no divulgar una política de privacidad.

Microservicio 2. Permitirá descubrir y extraer la política de privacidad (su URL) embebida en un APK.

Microservicio 3. Permitirá extraer el texto de una política de privacidad de diversas fuentes, incluyendo páginas HTML que utilizan diversas tecnologías web, documentos PDF y documentos de Google Drive.

1.4 MARCO TEÓRICO

En esta sección se presentan los conceptos relativos a la privacidad, datos personales, protección de datos y modelos de permisos, indispensables para comprender el contexto del presente trabajo. La privacidad se aborda desde el ámbito de la ingeniería según Carmela Troncoso [5]. A continuación, se presenta brevemente el marco legal de la LOPDP y con una descripción de las políticas de privacidad. Finalmente, se exponen de manera breve las herramientas, bibliotecas y programas utilizados en la implementación de los microservicios.

1.4.1 PRIVACIDAD

El mundo moderno condiciona que las interacciones de las personas estén cada vez más relacionadas con el mundo digital (conferencias, música, transporte, comida, …). Por lo que la recopilación y uso de la información conlleva serios problemas de privacidad.

La privacidad es reconocida como un derecho humano fundamental [6] y es abordada desde múltiples perspectivas, incluyendo la ética, legal, tecnológica, entre otras. Este hecho ha dado lugar a múltiples concepciones de la privacidad y también a múltiples tecnologías que buscan contribuir a preservar la privacidad en sus diferentes concepciones. Carmela Troncoso organiza estas tecnologías considerando tres paradigmas [4]:

- **Privacidad como confidencialidad**

Proporcionar protección para evitar que la información personal sea accesible por cualquier entidad o persona, minimizando la cantidad de información expuesta. La información se refiere a los datos intercambiados explícitamente con los proveedores de servicio.

- **Privacidad como control**

Proporcionar a los usuarios los medios para decidir qué información se expondrá a los servicios, manteniendo la confidencialidad de la información recopilada y minimizando la cantidad de información que se puede inferir.

- **Privacidad como transparencia**

Informar al usuario qué información ha sido expuesta y qué o quién ha accedido o tratado esta información. Analizar la actividad de los usuarios con el propósito de: a) retroalimentar al usuario sobre sus acciones y b) realizar auditorías para para reivindicar que la privacidad del usuario no fue infringida.

Este trabajo se enfoca en la privacidad como control.

1.4.2 PROTECCIÓN DE DATOS PERSONALES

El dato personal se define como "*dato que identifica o hace identificable a una persona natural, directa o indirectamente*" [3]. Los diferentes elementos de información, que una vez recopiladas, pueden llevar a la identificación de una determinada persona, constituyen datos de carácter personal (p.ej., nombre, domicilio, cédula, etc.) [7].

El artículo 4 de la LOPDP[1] proporciona definiciones adicionales como: dato biométrico, dato genético, dato crediticio, datos sensibles y datos relativos.

Por otro lado, la protección de datos se refiere a los principios, prácticas, y salvaguardas fundamentales establecidas para proteger los datos personales del usuario [8].

Formalmente, ciertos principios y prácticas de protección de datos son definidos en marcos regulatorios de protección de datos. En el caso ecuatoriano, la Ley Orgánica de Protección de Datos Personales define un conjunto de principios, derechos y obligaciones que buscan la protección de los titulares durante el tratamiento de sus datos personales. En la siguiente sección se desarrolla brevemente este marco normativo.

1.4.3 LEY ORGÁNICA DE PROTECCIÓN DE DATOS PERSONALES

El Proyecto de Ley Orgánica de Protección de Datos Personales (PLOPDP) fue aprobado el 21 de mayo de 2021 y publicado en el Registro Oficial Suplementario No. 459 el 26 de mayo del 2021. A continuación, se resumen algunos artículos relevantes para este trabajo.

La LOPDP en su Artículo 1.- Objetivo y finalidad define la necesidad de "*Garantizar el derecho a la protección de datos personales, incluyendo el acceso y decisión sobre la información y datos de este carácter, así como su correspondiente protección. Para dicho efecto regula, prevé y desarrolla principios, derechos y obligaciones de tutela*." [4].

[1] LOPDP: Ley Orgánica de Protección de Datos Personales

En el artículo 10 menciona los principios para la protección de datos, entre los que se mencionan:

- Transparencia.
- Finalidad.
- Confidencialidad.
- Consentimiento.
- Seguridad de datos personales.

Para el presente trabajo el principio de Transparencia es relevante dado que la información sobre el tratamiento de los datos personales debe ser fácilmente accesible para los usuarios y de lenguaje sencillo para que pueda ser comprendido

El Capítulo III menciona los Derechos para la Protección de Datos, entre los artículos relevantes se destacan:

- Artículo 12.- Derecho a la información.
- Artículo 13.- Derecho de acceso.
- Artículo 22.- Derecho de consulta.
- Artículo 24.- Ejercicio de derechos.

En el Capítulo IV menciona las Categorías Especiales de Datos, entre los artículos relevantes se destacan:

- Artículo 25.- Categorías especiales de datos personales.
- Artículo 26.- Tratamiento de datos sensibles.

1.4.4 POLÍTICAS DE PRIVACIDAD/PROTECCIÓN DE DATOS

Una política de privacidad es un documento a través de la cual una organización describe las operaciones detalladas que realiza con los datos del usuario y cómo aplica los principios de protección de datos. Los datos serán recopilados siempre que se cuente con el consentimiento del usuario [9].

Las políticas de privacidad son el principal medio para informar al titular la manera en que un servicio hace uso de sus datos personales [4]. El objetivo principal de las políticas de privacidad es informar a los usuarios sobre el tratamiento de sus datos personales (p.ej., recopilación, almacenamiento, transferencia a terceros, etc.) y cómo aplicar la protección de datos para que puedan comprender los riesgos de privacidad que enfrentan. Así, las políticas son empleadas para evaluar ciertos requisitos de transparencia [10]

En la Constitución Ecuatoriana el artículo 66, numeral 19, define "*El derecho a la protección de datos de carácter personal, que incluye el acceso y la decisión sobre información y datos de este carácter, así como su correspondiente protección. La recolección, archivo, procesamiento, distribución o difusión de estos datos o información requerirán la autorización del titular o el mandato de la ley.*" [11].

1.4.5 MICROSERVICIOS

En los últimos años los microservicios han ganado gran popularidad entre los desarrolladores de software. Los microservicios, en el desarrollo de software, permiten desarrollar aplicaciones que se componen de módulos independientes. Los módulos pueden ser implementados en diferentes lenguajes y modificados sin afectar a la funcionalidad de la aplicación. Así, estos benefician el desarrollo de software en velocidad de implementación, escalabilidad, maleabilidad y equipos de trabajo mínimo [12].

1.4.6 SISTEMA OPERATIVO ANDROID

Android es un sistema operativo open source[2], desarrollado por Google para dispositivos móviles. En Ecuador, el S.O. Android lidera el mercado de dispositivos móviles, con una cuota de mercado que se aproxima al 88% [2].

1.4.6.1 Estructura de una aplicación Android

Los archivos que componen una aplicación Android se empaquetan dentro de un APK (Application Package)[3] [13]. En la Figura 1.1 se ilustran los componentes de un APK.

[2] **open source**: programas informáticos que permiten el acceso a su código de programación,
[3] **APK**: es el formato de archivo del paquete utilizado para distribuir e instalar software de aplicación en Android.

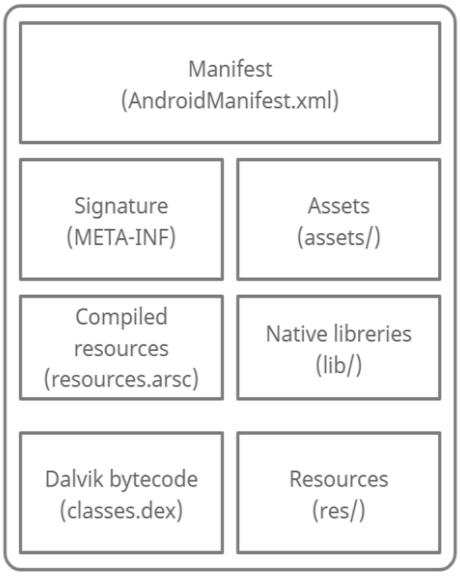


Figura 1.1 Estructura de un APK

- **AndroidManifest.xml**: Fichero principal de la aplicación Android, que contiene información cómo: Actividades, Servicios, Proveedores de Contenido, permisos, entre otros.
- **assets**: Directorio que contiene los recursos de la aplicación como: imágenes, ficheros, entre otros.
- **classes.dex**: Fichero que contiene el código Java compilado en Dalvik[4] bytecode.
- **Resources:** Directorio que contiene los recursos no compilados.
- **META-INF**: Directorio que contiene información para mantener la integridad de la aplicación y la seguridad del sistema.
- **resource.arsc**: Fichero que contiene los recursos precompilados como los ficheros XML binarios.

1.4.6.2 Modelo de Permisos Android

Las aplicaciones Android emplean un sistema de permisos para restringir el acceso de la aplicación a recursos de software y hardware del dispositivo. Todos los permisos requeridos para el correcto funcionamiento de la aplicación son descritos en el fichero manifest.xml (AndroidManifest.xml) de la aplicación [14].

[4] **Dalvik**: es una máquina virtual (VM) basada en registros. Cada aplicación de Android se ejecuta en su propio proceso con su propia instancia de Dalvik VM.

Los permisos de Android son una cadena de texto única declarada y solicitada en el fichero manifiest.xml de la aplicación Android. En la versión actual, hay 206 permisos de Android de diferentes grupos para acceder a recursos y datos [15].

Los permisos solicitados pueden afectar a los recursos del S.O. Android, así como recursos de aplicaciones de terceros instaladas en un dispositivo [16]. En el Anexo I se presenta una lista de permisos sensibles[5].

1.4.6.3 Tipos de Permisos Básicos

El modelo de permisos gestionado por Android define 4 tipos de permisos [17]:

- **"normal"**: Permisos de bajo riesgo, concedidos automáticamente durante la instalación de la aplicación sin la aprobación explícita del usuario.
- **"dangerous"**: Permisos de alto riesgo, solicitados explícitamente por la aplicación para tener acceso a datos del usuario o control del dispositivo.
- **"signature"**: Permisos concedidos automáticamente solo si la aplicación solicitante está firmada con el mismo certificado del paquete de software que declara el permiso.
- **"signatureOrSystem"**: Permisos concedidos solo a las aplicaciones que tienen una carpeta dedicada en la imagen del sistema Android o que están firmadas con el mismo certificado que la aplicación que declara el permiso.

1.4.7 PÁGINA WEB

Una página web es un documento electrónico publicado por una persona o empresa. Una página web contiene información variada (p.ej., textos, sonidos, imágenes, ...). Las páginas web son programadas principalmente en formato HTML[6] o XML[7], normalizadas por los estándares de la World Wide Web (WWW) y accesibles a través de un navegador web [18]. No obstante, en los últimos años han aparecidos bibliotecas y frameworks para enriquecer las aplicaciones de forma similar a una aplicación de escritorio. Por ejemplo, React es una biblioteca de JavaScript que permite implementar aplicaciones de una sola página (One Single Applications).

[5] **permisos sensibles**: permisos que infringen la privacidad de los usuarios al acceder a su información personal.
[6] **HTML**: HyperText Markup Language.
[7] **XML**: eXtensible Markup Language.

1.4.7.1 Tipos de Páginas web

Según las necesidades para presentar la información se tiene diferentes tipos de páginas web [19]:

- **Páginas web Estáticas**

  Son páginas web de contenido fijo. El usuario no puede interactuar con estas páginas, por lo que el contenido se modifica manualmente en el lado del servidor.

- **Páginas web Dinámicas**

  Son páginas web de contenido dinámico. El usuario puede interactuar con estas páginas, por lo que el contenido se modifica fácilmente.

- **Páginas web Protegidas**

  Son páginas web que requieren de una autenticación y autorización para acceder a su contenido.

- **Landing Pages**

  También conocidas como páginas de aterrizaje, son webs sencillas de una sola página con suficiente información para permitir acceder a múltiples recursos web.

1.4.7.2 Estructura de una Página HTML

Los 3 elementos principales para la creación de una página web se ilustran en la Figura 1.2 [20]:

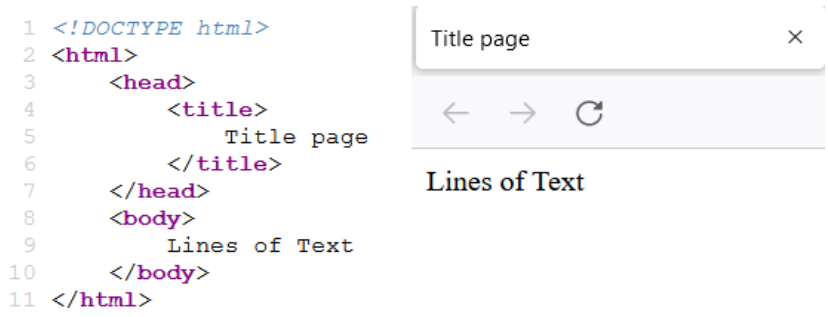


Figura 1.2 Ejemplo de una página web de estructura HTML

- **<!DOCTYPE>**: Etiqueta que especifica la versión de HTML utilizada.
- **<html> </html>**: Etiqueta de inicio y cierre del documento HTML.
- **<head> </head>:** Etiqueta de apertura y cierre de la cabecera, que contiene la información sobre el título de la página, el autor, palabras clave, etc.

- **<body> </body>**: Etiqueta de apertura y cierre del cuerpo, donde se encuentra el contenido de la página web (p.ej., el texto de una política de privacidad o protección de datos).

1.4.7.3 Navegador Web

Un navegador Web es una aplicación que permite visualizar páginas web, ya sean páginas almacenadas localmente (intranet) o provenientes de Internet (extranet). Estos permiten "entrar" en Internet y ver distintos contenidos a través de direcciones URLs o del uso de servicios de búsqueda datos en línea (conocidos como buscadores web).

1.4.8 HERRAMIENTAS UTILIZADAS EN EL PROYECTO

En esta sección se presentan los programas, bibliotecas y herramientas utilizadas para el desarrollo e implementación de los microservicios.

1.4.8.1 Android Asset Packaging Tool (AAPT)

AAPT es una herramienta de empaquetado de archivos Android. Esta herramienta analiza, indexa y compila archivos de formatos: ZIP, JAR[8] y APK. AAPT manipula ficheros de aplicaciones, tales como: el fichero AndroidManifest.xml y los ficheros XML [21].

1.4.8.2 Bibliotecas de Python relevantes para este trabajo

Las ventajas de Python[9] es utilizar su extenso repositorio de bibliotecas para integrar funcionalidades que se requieran. Las bibliotecas de Python se encuentran en el repositorio de software Python Package Index[10](PyPI). Entre las bibliotecas utilizadas se mencionan:

- python_json_logger[11]: es una biblioteca permite el registro estándar de Python para generar datos de registro como objetos JSON[12].
- requests[13]: es una biblioteca que permite crear y manejar peticiones HTTP [22].
- tld[14]: es una biblioteca que permite extraer el dominio y subdominio de una URL.
- json: es una biblioteca que permite crear y exportar archivos JSON.
- ast: es una biblioteca que permite evaluar caracteres de origen no confiable.

---

[8] **JAR**: Java ARchive
[9] **Python**: es un lenguaje interpretado, orientado a objetos, de alto nivel y con semántica dinámica.
[10] https://pypi.org
[11] https://pypi.org/project/python-json-logger/
[12] **JSON**: Es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript
[13] https://pypi.org/project/requests/
[14] https://pypi.org/project/tld/

- docx2text: es una biblioteca que permite manejar archivos XML, pertenecientes a archivos DOCX.
- tika[15]: es una biblioteca que permite extraer el contenido de un documento de formato PDF.

1.4.8.3 Herramientas de web scraping

Entre las herramientas utilizadas para web scraping con Python están:

- Selenium[16] es un API basada en HTML y JavaScript que permite realizar pruebas automatizadas en navegadores web [23].
- Selenium WebDriver es una herramienta que permite controlar al navegador web de forma nativa, como lo haría un usuario real [24].
- Beautifulsoup[17] es una biblioteca que permite realizar búsquedas de información en formato HTML o XML [25].

1.4.8.4 Androguard

Androguard[18] provee un marco de trabajo para aplicaciones Android. Provee un API de Python que permite analizar diferentes artefactos contenidos en un APK. A partir de esto se pueden realizar pruebas de penetración de malware y vulnerabilidades. Androguard es una herramienta multiplataforma, es decir, compatible con Linux, Windows y OSX.

Esta herramienta proporciona información específica sobre la aplicación como: permisos, servicios, actividades, entre otros. Esta herramienta, al estar escrita en Python permite incorporar sus funcionalidades en scripts personalizados [26].

1.4.8.5 Docker

Docker[19] es un proyecto de código abierto que permite crear, implementar y administrar de forma sencilla aplicaciones a través del de contenedores. Los contenedores son unidades estandarizadas de software que nos proporciona un mecanismo de empaquetamiento lógico en el que las aplicaciones pueden abstraerse de su entorno de ejecución [27].

Las herramientas del contenedor ofrecen un modelo de implementación mediante imágenes. Una imagen Docker incluye todos las dependencias, herramientas y bibliotecas

[15] https://pypi.org/project/tika/
[16] https://pypi.org/project/selenium/
[17] https://pypi.org/project/beautifulsoup4/
[18] https://github.com/androguard/androguard
[19] https://www.docker.com/why-docker

necesarios para la ejecución de una aplicación independientemente del anfitrión. En la Figura 1.3 se ilustra cinco contenedores Docker que son instancias de una imagen.

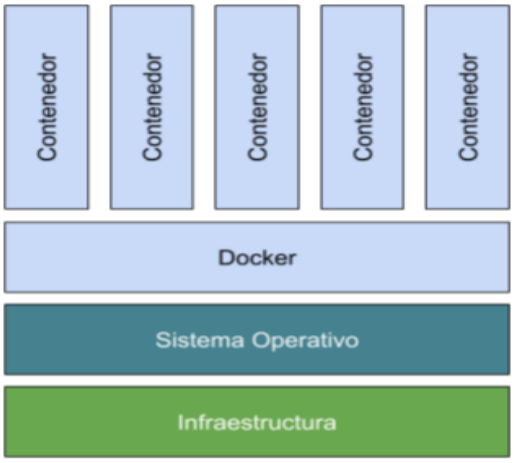


Figura 1.3 Estructura de funcionamiento Docker [28]

La creación de imágenes Docker personalizadas se realiza mediante un Dockerfile. Dockerfile es un documento de instrucciones para especificar el sistema operativo, las dependencias, las bibliotecas, los ejecutables, entre otros para el funcionamiento de microservicio.

2 METODOLOGÍA

En el siguiente capítulo se presenta la metodología de desarrollo utilizada en las fases de desarrollo de los microservicios. En la primera sección, se presenta la metodología ágil, Scrum [29], En la segunda sección, se presenta el análisis de los requisitos para cada uno de los microservicios, el *product backlog* y la planificación de *sprints*. En la tercera sección, se presenta el diseño de los microservicios. En la cuarta sección, se presenta el desarrollo de *sprints* en la implementación de los microservicios. Finalmente, en la quinta sección, se presentan los resultados de cada microservicio y discusión de los resultados.

2.1 METODOLOGÍA DE DESARROLLO

En esta sección se describe como Scrum fue adaptada para el desarrollo del componente. Scrum es ideal para proyectos expuestos a modificaciones y revisiones constantes, por lo que se consideró adecuado para el desarrollo del presente trabajo.

Scrum define los siguientes recursos, roles y responsabilidades [30]:

- *Product Owner:* representante del cliente dentro del equipo de trabajo. Es el responsable de la definición del *product backlog*, la planificación y la calendarización de las iteraciones.

- *Scrum Master*: responsable del equipo, quien supervisa los avances del desarrollo y garantiza que el equipo cuenta con las herramientas necesarias para el desempeño de su actividad.
- *Development Team*: equipo de trabajo.

Scrum define una serie de fases que se describen a continuación [30]:

- *Product Backlog*: Es una lista con todas las especificaciones del producto entregable por parte del *product owner*.
- *Sprint Planning Meeting:* reunión en la que se plantean las soluciones para las funcionalidades requeridas.
- *Sprint*: proceso de desarrollo de las funcionalidades definidas en la fase anterior.
- *Daily Scrum*: reuniones en la que se hace seguimiento a *sprint*; en cada fase colaboran el *development team* y el *scrum master*.
- *Sprint Review*: reunión en la que se verifica el cumplimiento de las metas y objetivos del sprint.
- *Sprint Retrospective*: reunión en que se analizan los resultados del sprint anterior, pudiendo encontrar falencias o mejoras aplicables en el siguiente sprint.

En la Tabla 2.1 se representan los roles principales definidos en este proyecto.

Tabla 2.1 Roles de Scrum

Rol	Nombre
Scrum master	Ing. Danny Guamán
Product owner	Ing. Danny Guamán
Development team	Cesar Miranda

2.2 ANÁLISIS

En esta sección se describen los requisitos definidos en reuniones con el *product owner*.

2.2.1 DEFINICIÓN DE REQUERIMIENTOS

En esta subsección se presentan los requisitos de cada uno de los microservicios, como se ilustra en la Tabla 2.2.

Tabla 2.2 Requisitos de cada uno de los microservicios

Artefacto	Requisito
Microservicio 1	● Proveer un insumo para determinar potenciales incumplimientos, basado en los permisos utilizados por la aplicación. ● Determinar una lista de permisos sensibles a la privacidad del usuario, que han sido solicitados por la aplicación.
Microservicio 2	● Descubrir y extraer la política de privacidad (su URL) desde un APK usando análisis estático. ● Determinar la URL de la política de privacidad a partir de cadenas de caracteres.
Microservicio 3	● Extraer y almacenar el texto y código HTML de una política de privacidad de diversas fuentes, incluyendo páginas HTML que utilizan diversas tecnologías web, documentos PDF y documentos web.

2.2.2 PRODUCT BACKLOG

En la Tabla 2.3 se presenta el *product backlog*, junto con la estimación de tiempo requerida para cada requerimiento.

Tabla 2.3 Product Backlog

N	Descripción	Estimación
1	Definición de interfaces del microservicio 1.	1
2	Extracción de un paquete APK desde un terminal móvil.	3
3	Extracción de permisos solicitados por la aplicación.	5
4	Elaboración de una lista de permisos sensibles.	5
5	Comparación de permisos de la aplicación con permisos sensibles.	5
6	Generación de resultados del microservicio 1	1
7	Definición de interfaces del microservicio 2.	1
8	Selección del fichero APK para la recuperación de cadenas de caracteres.	3
9	Análisis estático de un APK.	5
10	Recuperación de las cadenas de caracteres embebidas en un APK	3
11	Filtrar de URLs de políticas de privacidad.	4
12	Generación de resultados del microservicio 2	3
13	Definición de interfaces del microservicio 3.	1
14	Verificar la disponibilidad de una URL.	1
15	Análisis de problemas durante la extracción de texto de las políticas de privacidad desde una página web	2
16	Proponer soluciones a los problemas durante la descarga del texto de una política de privacidad.	8
17	Desarrollo de métodos utilizando web scraping, Selenium, bibliotecas Python, etc.	6
18	Generación de resultados del microservicio 3	6
19	Listar los componentes, bibliotecas y dependencias necesarias para los ambientes de ejecución de cada microservicio	1
20	Generar las imágenes Docker de cada microservicio	3
21	Verificar el funcionamiento de programa implementado	5

2.2.3 PLANIFICACIÓN DE SPRINTS

A continuación, se presenta un breve resumen de las funcionalidades a desarrollar en cada *sprint*:

1. **Primer Sprint - microservicio 1**
   El microservicio 1 provee un insumo para determinar posibles incumplimientos en el marco legal de la LOPDP. Este sprint contiene el diseño e implementación de los requisitos del microservicio 1. El tiempo destinado para este *sprint* es de 4 semanas.
2. **Segundo Sprint - microservicio 2**
   El microservicio 2 permite analizar y extraer la política de privacidad (su URL) embebida en un APK de una aplicación Android. Este sprint contiene el diseño e implementación de los requisitos del microservicio 4. El tiempo destinado para este *sprint* es de 3 semanas.
3. **Tercer Sprint - microservicio 3**
   El microservicio 3 permite descargar y almacenar el texto de una política de privacidad. Este sprint contiene el diseño e implementación de los requisitos del microservicio 3. El tiempo destinado para este *sprint* es de 6 semanas.
4. **Cuarto Sprint – contenedores Docker**

   Este sprint contiene la implementación de los contenedores Docker pertenecientes a cada uno de los microservicios. El tiempo destinado para este *sprint* es de 2 semana.

Se organizaron diferentes *sprints* de manera que cubran cada uno de los microservicios y la generación de contenedores Docker.

En la Tabla 2.4 se muestra la planificación de *sprints* de cada microservicio.

Tabla 2.4 Planificación de sprints

Actividad	N	Descripción	Estimación
microservicio 1	1	Definición de interfaces del microservicio 1.	1
	2	Extracción de un paquete APK desde un terminal móvil.	3
	3	Extracción de permisos solicitados por la aplicación.	5
	4	Elaboración de una lista de permisos sensibles.	5
	5	Comparación de permisos de la aplicación con permisos sensibles.	5
	6	Generación de resultados del microservicio 1	1
microservicio 2	7	Definición de interfaces del microservicio 2.	1
	8	Selección del fichero APK para la recuperación de cadenas de caracteres.	3
	9	Análisis estático de un APK.	5
	10	Selección del objeto AndroGuard para la extracción de cadenas de caracteres.	3
	11	Recuperación de las cadenas de caracteres embebidas en un APK	4
	12	Filtrar de URLs de políticas de privacidad.	3
	13	Generación de resultados del microservicio 2	1
microservicio 3	14	Definición de interfaces del microservicio 3.	1
	15	Verificar la disponibilidad de una URL.	2
	16	Análisis de problemas durante la extracción de texto de las políticas de privacidad desde una página web	8
	17	Proponer soluciones a los problemas durante la descarga del texto de una política de privacidad.	6
	18	Desarrollo de métodos utilizando web scraping, Selenium, bibliotecas Python, etc.	6
	19	Generación de resultados del microservicio 3	1
Contenedor Docker	20	Listar los componentes, bibliotecas y dependencias necesarios para los ambientes de ejecución de cada microservicio	3
	21	Generar las imágenes Docker de cada microservicio	5
	22	Verificar el funcionamiento de programa implementado	2

2.3 DISEÑO

En esta sección se presentan los artefactos generados durante el diseño de los microservicios.

2.3.1 DISEÑO DEL MICROSERVICIO 1

Como se ha explicado en la sección 2.2.1, este microservicio proveerá los insumos para determinar si un proveedor de aplicaciones necesita divulgar su política de privacidad. A continuación, se presenta la arquitectura del microservicio y el diagrama de actividades que modela su comportamiento.

2.3.1.1 Arquitectura del microservicio 1

En la Figura 2.1 se ilustra la arquitectura del microservicio 1, cuyos componentes esenciales son:

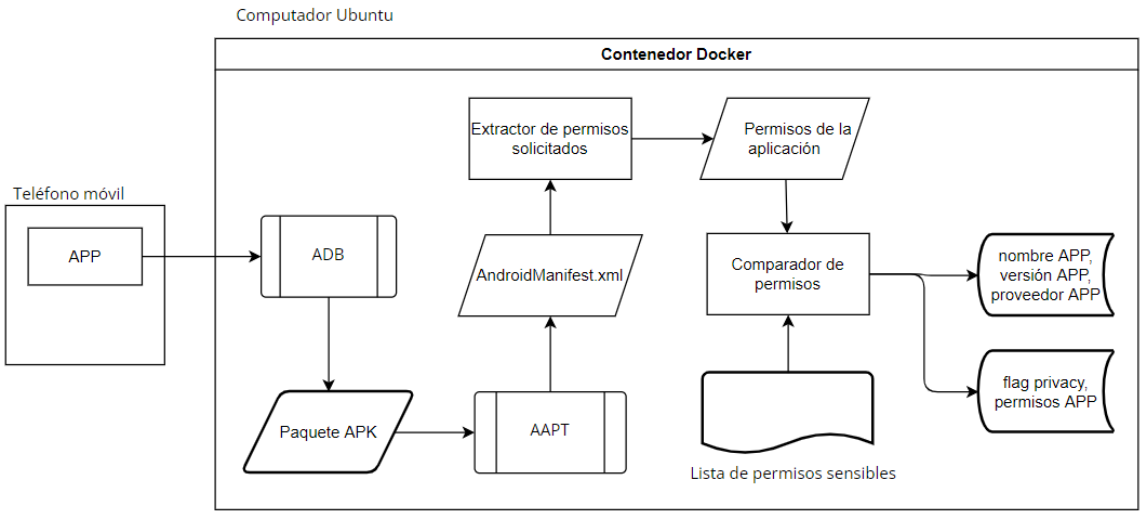


Figura 2.1 Arquitectura del microservicio 1

a) **Teléfono móvil:** Dispositivo móvil con capacidad para descargar aplicaciones Android desde Play Store.

b) **Computador Ubuntu**: Máquina anfitrión que albergará todos los componentes del microservicio.

c) **Contenedor Docker:** Ambiente instalado en el Computador Ubuntu que permitirá el despliegue y ejecución del microservicio, encapsulado en un contenedor.

d) **APP**: Aplicación móvil alojada en el teléfono móvil.

e) **ADB (Android Debug Bridge):** Herramienta de línea de comandos que permitirá interactuar con el dispositivo móvil para, por ejemplo, extraer el APK de una APP.

f) **Paquete APK:** Paquete de Android que contiene todos los artefactos necesarios para ejecutar una aplicación.

g) **AAPT (Android Asset Packaging Tool):** Herramienta de línea de comando que permitirá realizar múltiples operaciones sobre un APK (p.ej., extraer el fichero AndroidManifest.xml y realizar búsquedas dentro de éste).

h) **AndroidManifest.xml:** Fichero principal de una aplicación móvil que contiene información sobre los componentes que lo conforman (es decir, actividades, servicios, receptores de difusión o proveedores de contenido), permisos solicitados, entre otros.

i) **Permisos de la aplicación:** Representa los permisos solicitados por la aplicación para acceder a diversos recursos o datos en el teléfono móvil.

j) **Lista de permisos sensibles:** Es una lista de permisos que tienen acceso a datos personales y que por lo tanto son considerados sensibles para la privacidad de un usuario.

k) **Comparador de permisos:** Realizará una comparación entre los permisos solicitados por una aplicación y los permisos considerados sensibles para la privacidad de los usuarios.

2.3.1.2 Diseño de actividades del microservicio 1

En la Figura 2.2 se ilustra el diagrama de actividades del microservicio 1**.**

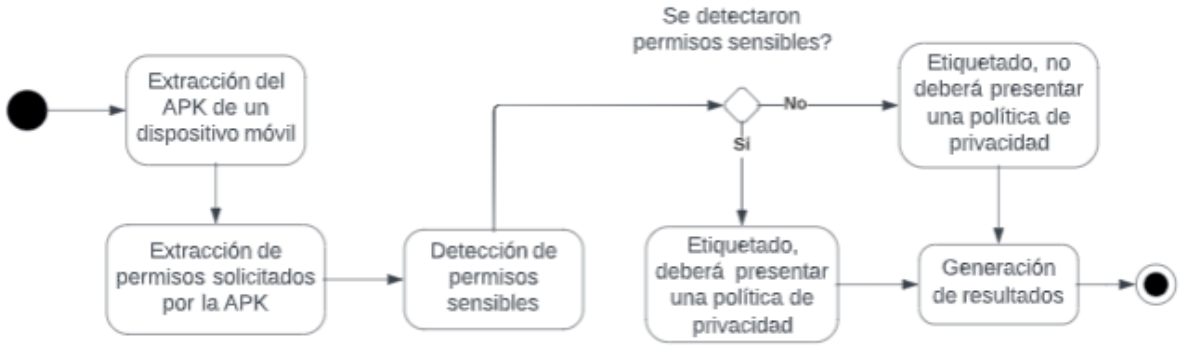


Figura 2.2 Diagrama de actividades del microservicio 1

1. **Extracción del Paquete APK de un dispositivo móvil**

Este módulo tiene por objetivo extraer el APK desde el dispositivo móvil. Para ello, se utiliza ADB, que proporciona una interfaz de línea de comandos para comunicarse con el dispositivo móvil. Una vez establecida la comunicación, se recupera el APK del móvil, que será usada para extraer los recursos necesarios para el análisis.

2. Extracción de permisos solicitados por una aplicación

Este módulo tiene por objetivo extraer los permisos solicitados por un APK. El APK está constituido de varios componentes[20], por lo que se necesitará un proceso desempaquetamiento para aislar el fichero AndroidManifest.XML. Del fichero AndroidManifest.xml, se extraerán los permisos solicitados utilizando la herramienta APPT. Los permisos extraídos serán comparados con una lista de permisos sensibles.

3. Detección de permisos sensibles

Este módulo tiene como objetivo comparar los permisos solicitados por una aplicación con una lista de permisos sensibles. Para ello, se compararán los permisos suministrados en el módulo anterior con una lista de permisos sensibles, en caso de encontrarse elementos comunes, se considerará que el proveedor de la aplicación estaría obligado a presentar una política de privacidad, de lo contrario, el proveedor de la aplicación no estaría obligado a presentar una política de privacidad.

4. Generación de resultados

Los resultados de la ejecución del microservicio 1 son almacenados en formato JSON, que contiene el nombre del paquete, versión de la aplicación, nombre del proveedor de la aplicación, etiqueta de divulgación de política de privacidad, una lista de los permisos de la aplicación y una lista de permisos de permisos sensibles de la aplicación.

2.3.2 DISEÑO DEL MICROSERVICIO 2

Como se ha explicado en la sección 2.2.1, este microservicio proveerá la política de privacidad (su URL) embebida en una aplicación móvil. A continuación, se presenta la arquitectura del microservicio y el diagrama de actividades que modela su comportamiento.

2.3.2.1 Arquitectura del microservicio 2

En la Figura 2.3 se ilustra la arquitectura del microservicio 2. Los componentes principales de la arquitectura son:

[20] Estos componentes se describieron en la subsección 1.4.6.1.

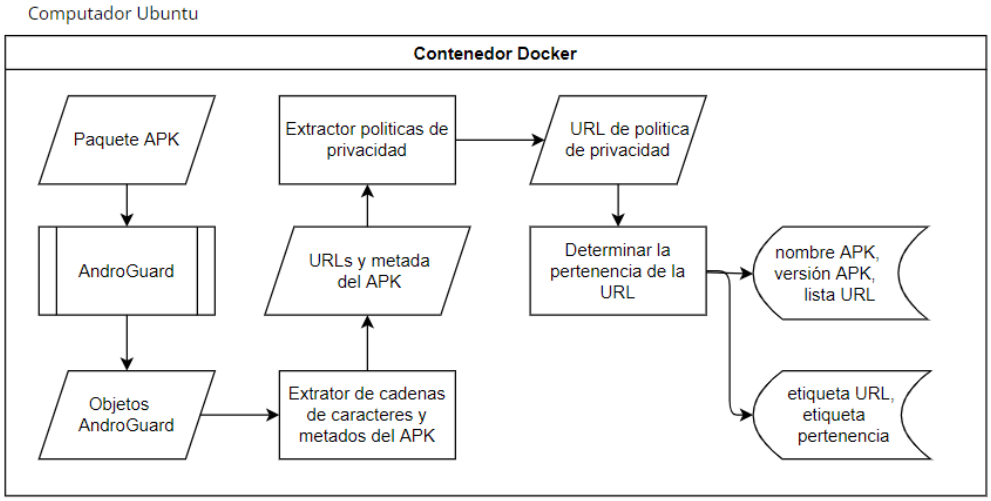


Figura 2.3 Arquitectura del microservicio 2

a. **Computador Ubuntu**: Máquina anfitrión que albergará todos los componentes del microservicio.

b. **Contenedor Docker**: Ambiente instalado en el Computador Ubuntu que permitirá el despliegue y ejecución del microservicio, encapsulado en un contenedor.

c. **Paquete APK:** Paquete de Android que contiene todos los artefactos necesarios para ejecutar una aplicación.

d. **Androguard**: Herramienta de software que permitirá analizar estáticamente un paquete APK de una aplicación.

e. **Objetos Androguard**: Los objetos se crearán con técnicas de análisis estático de la herramienta de software Androguard.

f. **Extractor de cadenas de caracteres y metadatos del APK**: Realizará la extracción de las cadenas de caracteres a partir de métodos, clases y funciones. También llevara a cabo la extracción de metadatos del APK (p.ej. nombre del paquete, versión del código, proveedor).

g. **Extractor las políticas de privacidad**: Realizará la extracción de URLs que hagan referencia a políticas de privacidad contrastándolas con expresiones regulares.

h. **Determinar la pertenencia de la URL:** Realizará la determinación de la pertenencia del dominio de la URL de la política de privacidad cotejando los dominios y subdominios de la URL y los del proveedor de la aplicación.

2.3.2.2 Diseño de actividades del microservicio 2

En la Figura 2.4 se ilustra el diagrama de actividades del microservicio 2.

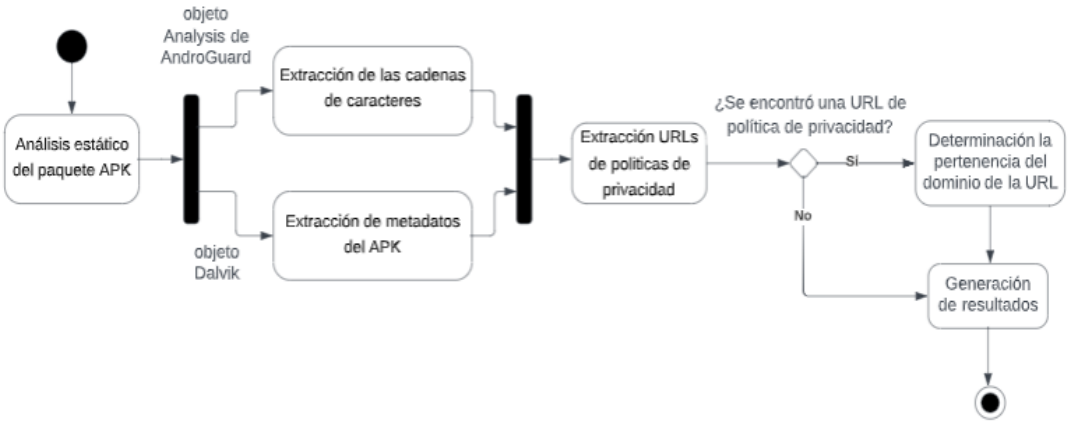


Figura 2.4 Diagrama de actividades del microservicio 2

1. Análisis estático del paquete APK

Este módulo tiene como objetivo el análisis estático[21] del APK de una aplicación. Para ello, se utilizará la herramienta AndroGuard[22]. Utilizando el método: analizeAPK(), se generaran 3 objetos: (i)“*a*” es un objeto APK; (ii)“*d*” es un objeto DalvikVMFormat; y (iii)“*dx*” es un objeto de análisis basado en el objeto “*d*”. Los objetos AndroGuard generados serán utilizados para la extracción de cadenas de caracteres y metadatos.

2. Extracción de las cadenas de caracteres y metadatos del APK

Este módulo tiene como objetivo extraer las cadenas de caracteres embebidas en métodos, funciones y clases del APK; y los metadatos del APK. Para ello, se extraerán los metadatos del objeto Dalvik, como son: nombre del paquete, versión de código y nombre del APP. Se buscará las cadenas de caracteres empleando una expresión regular. Las URLs obtenidas serán utilizadas para la búsqueda políticas de privacidad.

3. Extracción de URLs de políticas de privacidad

Este método tiene como objetivo identificar URLs que potencialmente estén relacionadas con políticas de privacidad. Para ello, se utilizarán palabras claves como: privacidad y política; y sus correspondientes en inglés. Se filtrarán las URLs que contengan campos con las palabras clave. Las URLs obtenidas serán utilizadas para determinar la pertenencia de la URL.

[21] **análisis estático**: descomposición de una aplicación sin ejecutarla.
[22] Esta herramienta se explicó en la subsección 1.4.8.4

4. Determinación de la pertenencia de la URL

Este método tiene como objetivo determinar la pertenencia de la URL. Para ello, se extraerán los dominios y subdominios de la URL, así como los dominios y subdominios del proveedor de la aplicación. Se compararán dichos campos y determinará si la URL pertenece a la aplicación, caso contrario, la URL le pertenecerá a un tercero.

5. Generación de resultados

Los resultados de la ejecución del microservicio 2 son almacenados en formato JSON, nombre del paquete, versión de la aplicación, proveedor de la aplicación, etiqueta de detección de URL de política de privacidad, etiqueta de pertenencia de la URL y la lista de URLs de políticas de privacidad.

2.3.3 DISEÑO DEL MICROSERVICIO 3

Como se ha explicado en la sección 2.2.1, este microservicio permitirá ampliar la descarga de políticas de privacidad a partir de su URL. A continuación, se presenta la arquitectura del microservicio y el diagrama de actividades que modela su comportamiento.

2.3.3.1 Arquitectura del microservicio 3

En la Figura 2.5 se ilustra la arquitectura del microservicio 3. Los componentes principales de la arquitectura son:

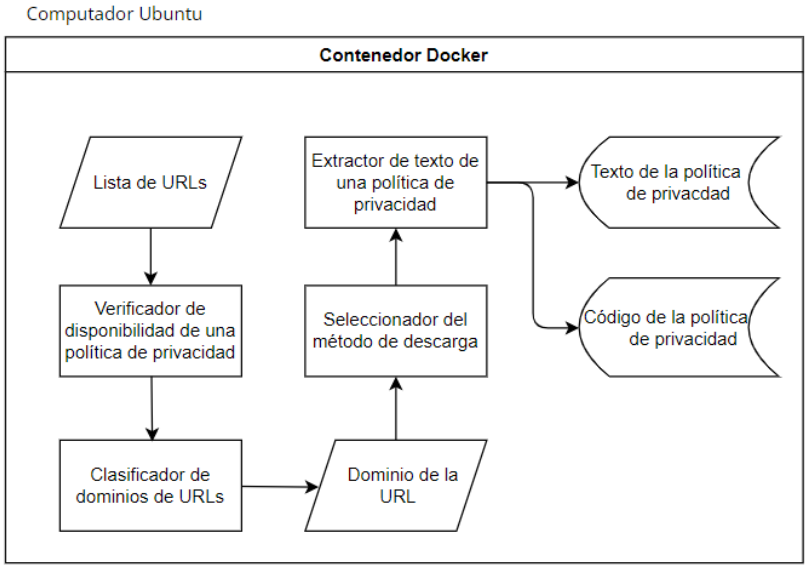


Figura 2.5 Arquitectura del microservicio 3

a. **Lista de URLs**: Archivo con un listado de URLs de políticas de privacidad.

b. **Verificador de disponibilidad de una política de privacidad**: Realizará la verificación el estado de una política de privacidad.
c. **Clasificador de dominios de URLs**: Realizará la identificación del sitio web o formato de documento de una política de privacidad.
d. **Dominios de una URL**: Identificará el proveedor del sitio web que contiene una política de privacidad.
e. **Seleccionador del método adecuado de descarga**: Realizará la comparación del dominio de una política de privacidad con los dominios de sitios web considerados en las soluciones elaboradas.
f. **Extractor del texto de una política de privacidad**: Realizará la extracción el texto de una política de privacidad.

2.3.3.2 Diseño de actividades del microservicio 3

En la Figura 2.6 se ilustra el diagrama de actividades del microservicio 3.

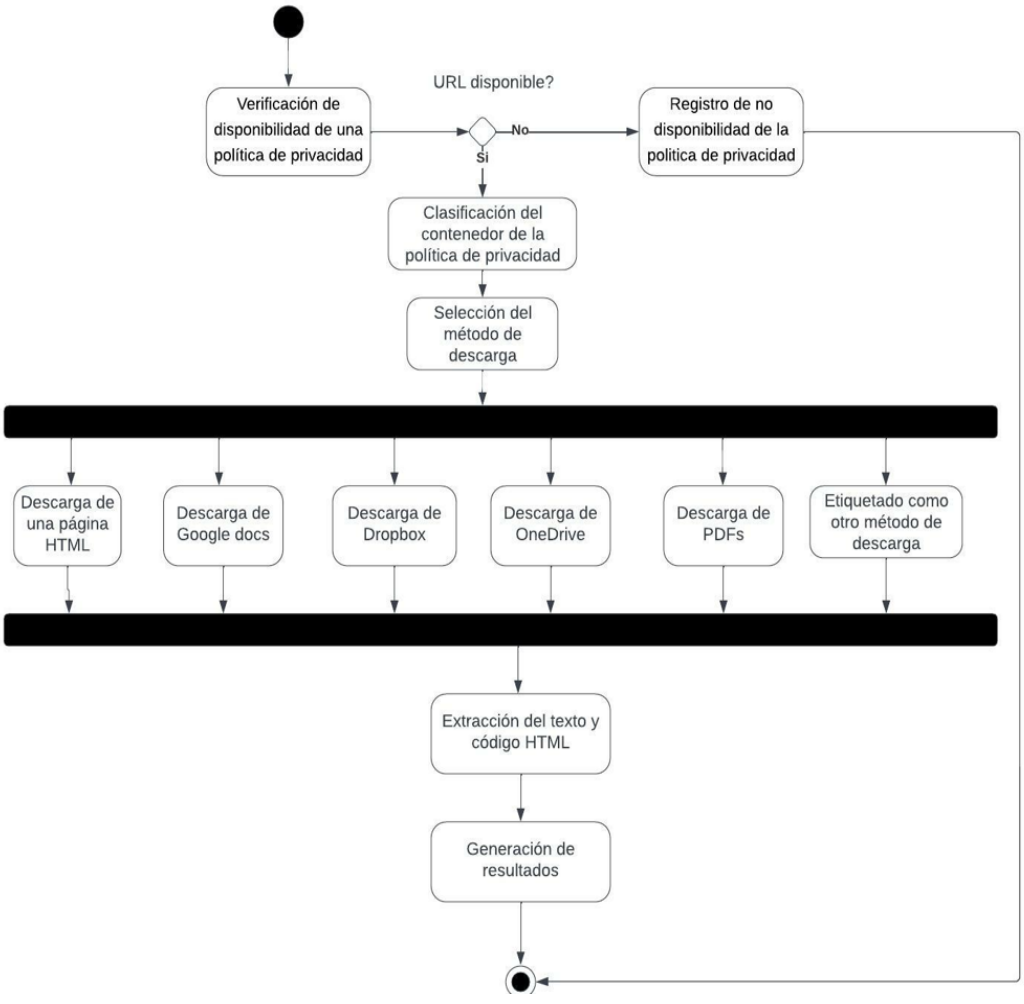


Figura 2.6 Diagrama de actividades del microservicio 3

1. **Verificación la disponibilidad de la política de privacidad**

Este módulo tiene por objetivo verificar la disponibilidad de la política de privacidad. Para ello, se utilizará una petición *HTTP get*. La respuesta de esta petición *HTTP* permitirá identificar el estado de la política de privacidad.

2. **Clasificación del contenedor de una política de privacidad**

Este módulo tiene por objetivo clasificar el contenedor de las políticas de privacidad. Para ello, se extraerá los dominios y subdominios de las políticas de privacidad. Los dominios y subdominios se cotejarán con los dominios de los sitios web contemplados para la descarga de los documentos que contienen políticas de privacidad. Las URLs se etiquetarán para relacionarlas con la política de privacidad del sitio web, que serán utilizadas en la selección del método de descarga.

3. **Selección del método de descarga**

Este método tiene como objetivo seleccionar el método de descarga para una política de privacidad. Para ello, se utilizará la clasificación del contenedor de una política de privacidad. Se podrán utilizar los siguientes métodos de descarga como:

a. Descarga de una página HTML
   Método de descarga general para paginas HTML de contenido estático y dinámico. Descargará la política de privacidad utilizando *Selenium*. Realizara una espera de 30 a 60 segundos para la renderización del código HTML en páginas dinámicas.
b. Descarga de Google docs
   Método de descargar para documentos contenidos en el sitio web Google docs. Descargará la política de privacidad utilizando `requests`.
c. Descarga de OneDrive.
   Método de descargar para documentos contenidos en el sitio web OneDrive. Descargará la política de privacidad de documentos web de formato DOCX, utilizando `requests`. Realizará la búsqueda de la URL de la política de privacidad embebida en el sitio web. Se descargará el documento de la política de privacidad en formato PDF.
d. Descarga de Dropbox
   Método de descargar para documentos contenidos en el sitio web Dropbox. Descargara la política de privacidad de documentos web de formato TXT, HTML, RFT, DOC, DOCX y PDF utilizando `requests`. Para ello, se

realizará la búsqueda de la URL de la política de privacidad embebida en el sitio web. Para documentos de formato RFT, DOC, DOCX, y PDF se descargará el documento de la política de privacidad en formato PDF. En cambio, para documentos TXT y HTML la URL embebida contendrá otra página de una política de privacidad de contenido estático.

e. Descarga de PDFs

   Método de descargar para documentos de formato PDF. Descargara la política de privacidad utilizando `requests`. Para ello, se configurarán ciertas opciones en la descarga de `requests` para especificar el tamaño del flujo de la descarga y la certificación de autenticidad de la página de la política de privacidad.

f. Etiquetado como otro método de descarga

   Método de descargar para descargas parciales, incompletas, vacías o de acceso denegado.

4. Extracción del texto de la política de privacidad

Este módulo tiene como objetivo la extracción del texto de una política de privacidad. Para ello, se extraerán el texto de las políticas de privacidad. Esto dependerá del método asociado a cada política de privacidad.

5. Generación de resultados

Este módulo tiene como objetivo generar archivos con el código HTML y el texto de la política de privacidad. Para ello, se almacenarán localmente los archivos resultados del microservicio 3.

A continuación, se explican las secciones de código relevantes generadas en cada *sprint.*

2.4 IMPLEMENTACIÓN

En esta sección se presentan los *sprints* realizados por el equipo de desarrollo durante la implementación de los artefactos del presente trabajo, junto con las. secciones de código más relevantes.

2.4.1 IMPLEMENTACIÓN DEL MICROSERVICIO 1- SPRINT 1

Este *sprint* incluye la implementación de la funcionalidad para la extracción de permisos solicitados y comparación de estos permisos con una lista de permisos sensibles. También incluye el código para la generación de logs y resultados de la ejecución del microservicio 1.

En la Figura 2.7 se ilustra el diagrama de despliegue del microservicio 1.

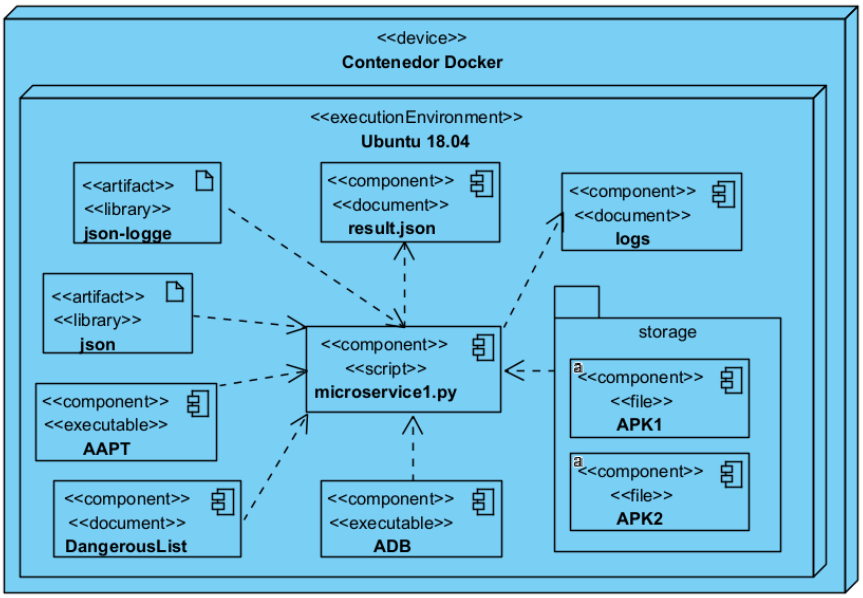


Figura 2.7 Diagrama de despliegue del microservicio 1

Como se muestra en el Código 3.1, se extrae el APK para obtener el archivo AndroidManifest.xml. En la línea 101, se usa *AAPT* para la extracción todas las cadenas de caracteres del fichero AndroidManifest.xml. Dado que el archivo Manifest.xml contiene todos los recursos usados por la aplicación, el script mostrado en las líneas 102 a 105 se extrae solamente los permisos del APK. Finalmente, se retorna el vector *permissions*.

```
96  def aapt_permissions(apk_file):
97      permissions = []
98      try:
99          assert os.path.isfile(apk_file), '%s is not a valid APK path' % apk_file
100         logger.debug('aapt_permissions function has been started')
101         output = aapt_badging(apk_file)
102         if output is not None:
103             lines = output.split('\n')
104             permissions = [x.split('name=')[1].strip("'") for x in lines
105                            if x.startswith('uses-permission:')]
106     except Exception as e:
107         reason = 'aapt_permissions unavailable'
108         logger.error('aapt_permissions failed',
109                      extra={'exception_message': str(e), 'reason': reason})
110     else:
111         logger.debug('aapt_permissions function has been successful')
112         return permissions
```

Código 3.1. Función para la extracción de permisos solicitados por una aplicación

En el Código 3.2 se presenta la función *load_lstPermission()*, que recibe como argumento de entrada: *path_json*, que contiene la ruta de la lista de permisos sensibles. Los permisos sensibles son asignados al vector *lstCheck*. Finalmente, se retorna el vector *lstCheck*.

```
154  def load_lstPermission(path_json):
155      lstCheck = []
156      try:
157          assert os.path.isfile(path_json), '%s is not a valid path of json file' % path_json
158          logger.debug('load_lstPermission function has been started')
159          option = 'r'
160          with open(path_json, option) as lstOutput:
161              lstCheck = json.load(lstOutput)
162      except Exception as e:
163          reason = 'load_lstPermission unavailable'
164          logger.error('load_lstPermission failed',
165                       extra={'exception_message': str(e), 'reason': reason})
166      else:
167          logger.debug('This function load_lstPermission was successful')
168          return lstCheck
```

Código 3.2. Función para obtener la lista de permisos sensibles

Se determinó si la aplicación solicita permisos sensibles al comparar los permisos extraídos con una lista de permisos sensibles predeterminada. Como se observa en el Código 3.3, la función **comparationLst** *()* recibe como argumentos de entrada: el vector *Permissions*, que contiene una lista de permisos sensibles; el vector *Check* que contiene una lista de permisos extraídos de la aplicación. Utilizando un bucle doble en las líneas 199 y 203 se comparan los elementos de los 2 vectores, en caso de encontrar permisos sensibles, se guarda el permiso en el vector *Dangerous* y se modifica la variable *flag* a *True*. Finalmente, se retorna la variable *flag* y el vector *Dangerous*.

```
170  def comparationLst(Permissions, Check):
171      flag = False
172      dangerous = []
173      try:
174          assert Permissions is not None, 'We need initiate a Permissions list'
175          assert Check is not None, 'We need initiate a Checking list'
176          logger.debug('comparationLst function has been started')
177          for element in Permissions:
178              for aux_element in Check:
179                  if element == aux_element['description']:
180                      dangerous.append(x)
181                      flag = True
182      except Exception as e:
183          reason = 'comparationLst unavailable'
184          logger.error('comparationLst failed',
185                       extra={'exception_message': str(e), 'reason': reason})
186      else:
187          logger.debug('comparationLst function has been successful')
188          return [flag, dangerous]
```

Código 3.3. Función para la comparación de permisos solicitados con permisos sensibles

En el Código 3.4 se *muestra la función* `writeJson()`, *utilizada para el registro de resultados del microservicio 1 en formato JSON. Esta función recibe como argumentos de entrada: el vector* `lstResult`, *que contiene de permisos de la aplicación; el vector* `lstDangerous`, *que contiene permisos sensibles de la aplicación;* `version`, *que contiene la versión del código de la aplicación; y* `name`, *que contiene el nombre del paquete.* Todas estas variables con escritas en el archivo `result.json`.

```
202     def writeJson(lstResult, lstDangerous, version, name, flag):
203         lstWrite = []
204         lstApp = []
205         try:
206             logger.debug('The function writeJson was initiate')
207             lstApp.append({'packagename': name,'version': version,
208                            'privacyPolicy': flag})
209             for i in range(0, len(lstResult), 1):
210                 lstWrite.append({'id': i,'name': lstResult[i],
211                                  'description': 'NA'})
212             for j in range(0, len(lstDangerous), 1):
213                 lstWrite.append({'id': j,'name': lstDangerous[j],
214                                  'description': 'Dangerous'})
215             with open('result/results.json', '+a') as fp:
216                 fp.write(',\n'.join(json.dumps(j) for j in lstApp) +
217                     '\n')
218                 fp.write(',\n'.join(json.dumps(i) for i in lstWrite) +
219                     '\n')
220         except Exception as e:
221             reason = 'writeJson unavailable'
222             logger.error('writeJson failed',
223                          extra={'exception_message': str(e), 'reason': reason})
```

Código 3.4. Función para la generación de resultados del microservicio 1

Se documentó la ejecución del microservicio 1 utilizando *logs*. En el Código 3.5 se presenta la configuración de *logs*. En la línea 25 se define la inicialización del agente *log* `init_logger()`. En la línea 28 se define el formato para la presentar los parámetros del *log* y en la línea 33 se define el nivel de severidad mínimo para *logs*. En la línea 36 se define la finalización del agente *log* `stop_logger()`. Considerando que al inicio y final del microservicio se deberá inicial y terminar el uso del agente *log*.

```
24  # log agent initialization
25  def init_logger(file):
26      global handler, logger
27      handler = log.FileHandler(file)
28      format_str = '%(levelname)s%(asctime)s%(filename)s%(funcName)s%(lineno)d%(message)'
29      formatter = jsonlogger.JsonFormatter(format_str)
30      handler.setFormatter(formatter)
31      logger = log.getLogger(__name__)
32      logger.addHandler(handler)
33      logger.setLevel(log.DEBUG)
34      return logger
35  # log agent termination
36  def stop_logger():
37      logger.removeHandler(handler)
38      handler.close()
```

Código 3.5. Función para la configuración del agente log

2.4.2 IMPLEMENTACIÓN DEL MICROSERVICIO 2- SPRINT 2

Este *sprint* incluye la implementación la funcionalidad para la generación de objetos Androguard y extracción de metadatos y cadenas de caracteres a partir de estos. También, se incluye el código para la extracción de políticas de privacidad, así como el código para determinar la pertenencia de la URL. Finalmente se incluye el código para la generación de logs y resultados de la ejecución del microservicio 2.

En la Figura 2.8 se ilustra el diagrama de despliegue microservicio 2.

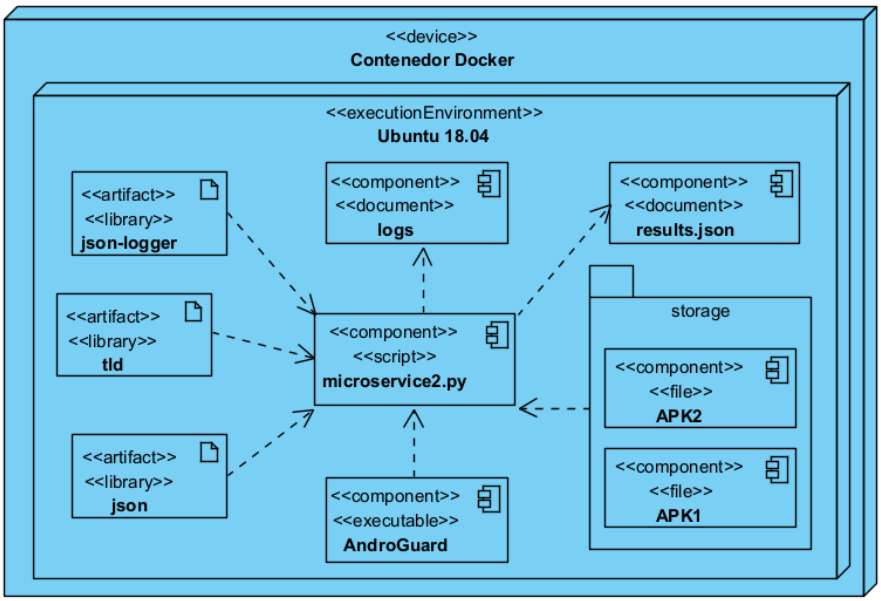


Figura 2.8 Diagrama de despliegue del microservicio 2

En el Código 3.6 se emplea el método `analyzeAPK()`—línea 98— para generar 3 objetos AndroGuard: `a`, `d` y `dx`. El objeto `dx` pertenece a la clase AndroGuard Analysis. El objeto `dx` contiene los métodos, clases, campos y cadenas de caracteres contenidas en uno o varios archivos DEX. El objeto `a` contiene los metadatos de la APK.

```
94  def get_urls_apk(path):
95      url_List = []
96      try:
97          logger.debug('Using AndroGuard')
98          a, d, dx = AnalyzeAPK(path)
99          logger.debug('modulus a, d, dx for ready')
100         app_name = a.get_app_name()
101         app_version = a.get_androidversion_name()
102         app_package = a.get_package()
```

Código 3.6. Fragmento de código para la generación de objetos AndroGuard

En el Código 3.7 se emplea el método *find_strings()*—línea 104— para la búsqueda de cadenas de caracteres empleando un patrón especifico *'http*'*. Obteniendo un iterador del tipo *StringAnalysis*, esta subclase contiene las XREF[23] de las cadenas de caracteres. Se emplea el método *get_value()*—línea 106—para extraer las cadenas de caracteres de las URLs del iterador referenciado.

```
103         logger.debug('Filtering urls from the obtained strings ')
104         regex_result = dx.find_strings('http*')
105         for element in regex_result:
106             url = element.get_value()
107             url_List.append(str(url))
108     except Exception as e:
109         reason = 'get_urls_apk unavailable'
110         logger.error('get_urls_apk failed',
111                      extra={'exception_message': str(e),
112                             'reason': reason})
113     else:
114         logger.debug('Extraction of urls and metadata successful')
115         return url_List, app_name, app_version, app_package
```

Código 3.7. Función para la extracción de URLs a partir de cadenas de caracteres

En el Código 3.8 se presenta la función *detec_of_urls_of_privacy_policy()* para la extracción de política de privacidad, que recibe como argumento de entrada: un vector *url_List* que contiene la o las URLs extraídas previamente. En la línea 121 se le asigna a la vector *pattern* expresiones regulares que emplean palabras clave como: política y privacidad y sus equivalentes en inglés. En la línea 122 se inicia un bucle para comparar cada elemento del vector *pattern*. Entre las líneas 123 a 126 se emplea módulo *re* para compilar y buscar las expresiones regulares en el vector *url_List*. Finalmente, se compara el tamaño del vector *privacypolicy_List* para determinar el valor del booleano *url_Flag*, si el tamaño del vector es diferente de cero se le asigna el valor *True*, caso contrario, se le asigna el valor *False*. Finalmente, esta función retorna el vector *privacypolicy_List* y la variable *url_Flag*.

[23] **XREF**: Referencias cruzadas generadas para clases, métodos, campos y cadenas.

```
117  def detect_urls_of_policy_privacy(url_List):
118      privacypolicy_List = []
119      try:
120          logger.debug('Beginning to detect privacy policies from urls')
121          pattern = ['policy', 'privacy', 'politica', 'privacidad']
122          for regex in pattern:
123              aux_regex = 'r' + regex
124              aux_pattern = re.compile(aux_regex)
125              privacypolicy_urls = [x for x in url_List if re.search(regex, x)]
126              privacypolicy_List.extend(privacypolicy_urls)
127          if len(privacypolicy_List) != 0:
128              logger.info('Privacy policy url has been found')
129              url_Flag = False
130          else:
131              logger.info('Privacy policy url has not been found')
132              url_Flag = True
133
134      except Exception as e:
135          reason = 'detect_urls_of_policy_privacy unavailable'
136          logger.error('detect_urls_of_policy_privacy failed',
137                       extra={'exception_message': str(e), 'reason': reason})
138      else:
139          logger.debug('URL detection process was successful')
140          return privacypolicy_List, url_Flag
```

Código 3.8. Función para la extracción de políticas de privacidad

En el Código 3.9 se presenta la función *inform_url_belong()* para determinar la pertenencia de una URL, que recibe como argumento de entrada: el vector *url_List*, que contiene la o las URLs de las políticas de privacidad y la variable *package*, que contiene campos del *packagename*. Se extraen los dominios y subdominios del *packagename* en la variable *package_tokens*. En la línea 148 se inicia un bucle para obtener cada elemento del vector *url_List*. A continuación, se extraen los dominios y subdominios de la variable *url* en el vector *url_tokens*. Se compararon los elementos del vector *package_tokens* con el vector *url_tokens*, en caso de encontrar campos en común se asigna el valor *True* a la variable *belong_Flag* y se agrega la URL al vector *app_list*, caso contrario, la variable *belong_Flag* obtiene el valor *False* y se agrega la URL al vector *third_list*.

```
142     def inform_url_belong(urlList, package):
143         app_List = []
144         third_List = []
145         try:
146             logger.debug('starting url and packet token matching')
147             package_tokens = get_bag_of_package_domains(package)
148             for url in urlList:
149                 url_tokens = get_bag_of_url_domains(url)
150                 common_elements = set(package_tokens) & set(url_tokens)
151                 if len(common_elements) > 0:
152                     logger.debug('There are domains in common')
153                     app_list.append(url)
154                     belong_Flag = True
155                 else:
156                     logger.debug('There are no domains in common')
157                     third_list.append(url)
158                     belong_Flag = False
159         except Exception as e:
160             reason = 'inform_url_belong unavailable'
161             logger.error('inform_url_belong failed',
162                          extra={'exception_message': str(e), 'reason': reason})
163         else:
164             logger.debug('Match tokens were successful')
165             return app_list, third_list
```

Código 3.9. Función para determinar la pertenencia de una URL

En el Código 3.10 se presenta la función *writeJson()* para escribir resultados del microservicio 3. Utilizando el vector *lstApp* se almacenan los metadatos del APK y 2 booleanos que etiquetan la pertenencia y existencia de políticas de privacidad en formato JSON. En la línea 191 se escribe el resultado de la ejecución del microservicio 2 en el fichero *result.json*.

```
185 def writeJson(version, name, url_flag, url_belong):
186     lstApp = []
187     try:
188         logger.debug('The function was initiate')
189         lstApp.append({'name': name,'version': version,
190                        'belong url': url_belong,'empty url': url_flag})
191         with open(result_dir+'results.json', '+a') as fp:
192             fp.write(',\n'.join(json.dumps(line) for line in lstApp) +
193                 '\n')
194     except Exception as e:
195         reason = 'writeJson unavailable'
196         logger.error('writeJson failed',
197                      extra={'exception_message': str(e), 'reason': reason})
```

Código 3.10. Función para el almacenamiento de resultados del microservicio 2

En el Código 3.11 se presenta la función *writeURLjson()* para el almacenamiento de URLs de políticas de privacidad del microservicio 2. Utilizando el vector *lstWrite* se almacenan las políticas de privacidad en formato JSON. En la línea 208 se escribe el contenido en el fichero *result.json*.

```
199  def writeURLjson(lstURL):
200      lstWrite = []
201      try:
202          for i in range(0, len(lstURL), 1):
203              lstWrite.append({'n': i,'url': lstURL[i]})
204          with open(result_dir+'results.json', '+a') as fp:
205              fp.write(',\n'.join(json.dumps(i) for i in lstWrite) +
206                       '\n')
207      except Exception as e:
208          reason = 'writeURLjson unavailable'
209          logger.error('writeURLjson failed',
210                       extra={'exception_message': str(e), 'reason': reason})
```

Código 3.11. Función para el almacenamiento URLs de políticas de privacidad

2.4.3 IMPLEMENTACIÓN DEL MICROSERVICIO 3 - SPRINT 3

Este *sprint* incluye la implementación: del código para la verificación de disponibilidad de la política de privacidad. También incluye el código para la clasificación de la URLs y selección del método de descarga de políticas de privacidad y las funciones generadas para la descarga exitosa desde diversas tecnologías web. Finalmente, se presentan detalles de implementación para la generación de archivos que contiene el texto y código de las políticas de privacidad.

En la Figura 2.9 se ilustra el diagrama de despliegue microservicio 3:

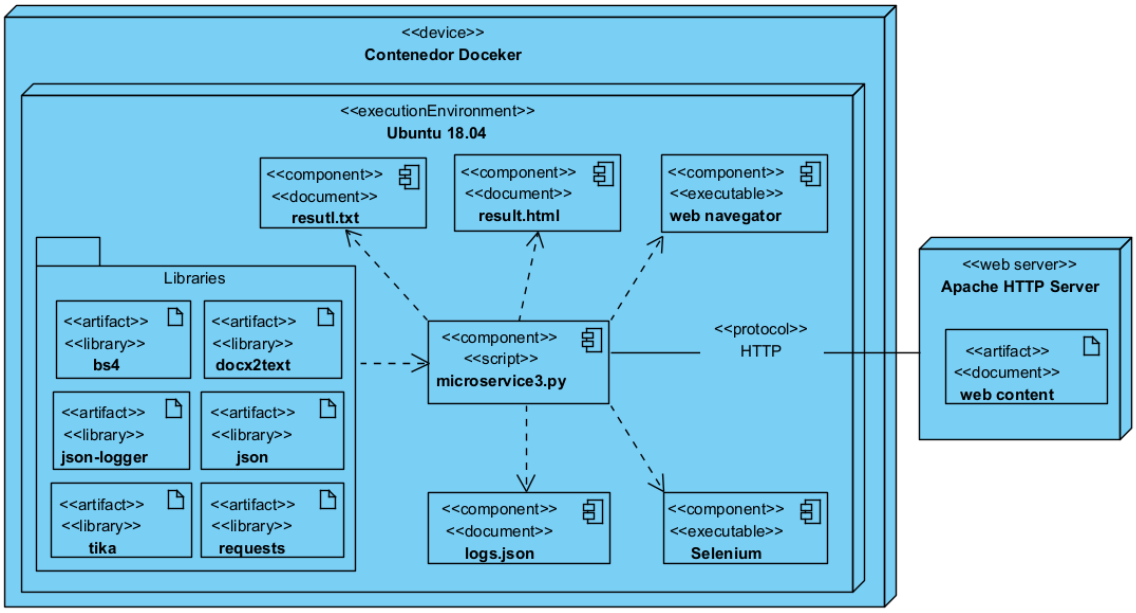


Figura 2.9 Diagrama de despliegue del microservicio 3

Se verifico la disponibilidad de la política de privacidad. Utilizando el método *`urllib.request.urlopen().read()`*— linea 151 —, se obtiene una respuesta del estado por parte del servidor que aloja la política de privacidad. En caso de no presentar ninguna excepción se devuelve un código HTTP 200 y el valor *`True`*, como se observa en el Código 3.12.

```python
def get_status_code(url):
    headers = {'User-Agent':
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 '
    '(KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36'}
    try:
        req = urllib.request.Request(url, data=None, headers=headers)
        urllib.request.urlopen(req, timeout=120)
    except urllib.error.HTTPError as e:
        if e.code == 404:
            reason = 'Privacy policy unavailable'
            logger.error("Privacy policy download failed",
                         extra={'exception_message': str(e),
                                'reason': reason, 'exit_code': e.code})
            return False, e.code
        else:
            return True, e.code
    except urllib.error.URLError as e:
        reason = 'Cannot connect to the domain server'
        logger.error("Privacy policy download failed",
                     extra={'exception_message': str(e),
                            'reason': reason, 'url': url})
        return False, e.reason
    except Exception as e:
        reason = 'Timeout in urllib.request.urlopen'
        logger.error("Privacy policy download failed", extra={
            'exception_message': str(e), 'reason': reason, 'url': url})
        return False, str(e)
    else:
        return True, 200
```

Código 3.12. Función para la verificación de la disponibilidad de una política de privacidad

Se selecciono el método de las políticas de privacidad. Utilizando el nombre de sitios web con el dominio o subdominio de la política de privacidad, como se observa en el Código 3.13.

```
104  def url_selector(url):
105      try:
106          csf_html = False
107          # To download pdf from common web page
108          csf_pdf = is_pdf_web(url)
109          # To download the docs format document from google docs website
110          token_docs = ['docs']
111          csf_docs = url_matching(url, token_docs)
112          # To download the txt format file from drive website
113          token_drive = ['drive']
114          csf_drive = url_matching(url, token_drive)
115          # To download the txt, html, doc, docx and pdf format file from drive website
116          token_dropbox = ['dropbox']
117          csf_dropbox = url_matching(url, token_dropbox)
118          # To download the docx format document from onedrive website
119          token_onedrive = ['onedrive', 'live']
120          csf_onedrive = url_matching(url, token_onedrive)
121          # if it is not any of the special cases, it is considered a web page of
122          if csf_pdf == False and csf_onedrive == False and \
123                  csf_drive == False and csf_dropbox == False and csf_docs == False:
124              csf_html = True
125      except Exception as e:
126          reason = 'url_selector unavailable'
127          logger.error('url_selector failed',
128                       extra={'exception_message': str(e), 'reason': reason})
129      else:
130          logger.debug('The function was successful')
131          return csf_pdf, csf_docs, csf_drive, csf_html, csf_dropbox, csf_onedrive
```

Código 3.13. Función para la selección del método de descarga adecuado

En el Código 3.14 se presenta la función `download_general_text()` que es utilizado para la descarga de una página HTML utilizando *Selenium*, su herramienta *Selenium Webdriver*, mediante el navegador web Chrome que extrae el contenido de la política de privacidad contenido en una página web.

Se implemento un `timer` de 30 a 60 segundos para solventar el intercambio de peticiones con el servidor y el navegador web, evitando así la descarga parcial del contenido de políticas de privacidad y permitir a través de JavaScript la modificación del contenido renderizado de la página web.

```
181  def download_general_text(url):
182      policy_text = None
183      policy_html = None
184      TIMEOUT = 60
185      TIMERSLEEP = 30
```

Código 3.14. Fragmento de código para la implementación de un timer

En el Código 3.15, se observa las opciones de configuración del Webdriver.

- - -no-sandbox: deshabilita el sandbox para todos los procesos, usado en casos de prueba, omitiendo el sistema de seguridad de SO.
- - -enable-javascript: permite JavaScript
- - -headless: abre el navegador web sin interfaz gráfica.
- - -disable-dev-shm-usage: deshabilita la partición /dev/shm, a veces demasiado pequeña en ciertos entornos VM.

```
187     # Define option for the navegator
188     chromeOptions.add_argument("--no-sandbox")
189     chromeOptions.add_argument("--enable-javascript")
190     chromeOptions.add_argument("--headless")
191     chromeOptions.add_argument('--disable-dev-shm-usage')
```

Código 3.15. Fragmento de código para la configuración de opciones Webdriver

El código HTML presenta una estructurado en forma de árbol. La estructura DOM[24] se compone de una colección anidada de objetos. En la línea 205 se identifica la etiqueta *<html>* y en la línea 207 haciendo uso del atributo de DOM *innerText* se recuperó y estableció el contenido como el texto sin formato de una política de privacidad, como se observa en el Código 3.16.

```
195     driver = webdriver.Chrome(executable_path=r'{}'
196                               .format(chromedriver_path), options=chromeOptions)
197     try:
198         logger.debug('The webdriver was being started')
199         WebDriverWait(driver, TIMEOUT).until(EC.presence_of_element_located(
200             (By.TAG_NAME, "html")))
201         # Get the HTML code from the page
202         driver.get(url)
203         time.sleep(TIMERSLEEP)
204         # Get the HTML code from the page
205         element = driver.find_element_by_tag_name('html')
206         # Extract text from the attribute innerText
207         policy_text = element.get_attribute('innerText')
```

Código 3.16. Fragmento de código para la extracción de texto y código de una política de privacidad

Se generó un archivo con el contenido de texto de la política de privacidad y otro archivo con el código HTML de la política de privacidad. En el Código 3.17 se presenta la función *store_text()* que recibe como argumento de entrada: *policytxt*, que tiene el texto

[24] Document Object Model: Conjunto de utilidades diseñadas para manejar documentos XML y HTML

de la política de privacidad; *policyhtml*, que contiene el código HTML de la política de privacidad y *title* que tiene el título de la página de la política de privacidad. Se establece el nombre del archivo y se escribe el contenido tanto del archivo de extensión txt y html.

```
507 def store_text(policytxt, policyhtml, title):
508     try:
509         logger.debug('store_text function has been started')
510         file = open(result_dir + title + ".txt", "w")
511         file.write(policytxt)
512         file.close()
513         file = open(result_dir + title + ".html", "w")
514         file.write(policyhtml)
515         file.close()
516     except Exception as e:
517         reason = 'store_text function unviable'
518         logger.error("store_text failed",
519                      extra={'exception_message': str(e), 'reason': reason})
520     else:
521         logger.info('store_text function has been successful')
```

Código 3.17. Función para el almacenamiento del texto y código de una política de privacidad

El sitio web Google Docs contiene documentos web que albergan políticas de privacidad, específicamente documentos de formato DOCX. Se utiliza el código HTML extraído con el método *requests.get()*. En el Código 3.18 se utiliza el método *soup.find_all()* que se emplea para buscar etiquetas <*script*> del tipo *'text/javascript'*. Se extrae el texto todo el texto utilizando una expresión regular utilizando el método *re.findall()*.

```
237         js_text_lst = soup.find_all('script', type='text/javascript')
238         for js_text in js_text_lst:
239             js_text = str(js_text)
240             # Splitting and filtering the text matching with [XXXXX].
241             for text in re.findall("\[.+\]", js_text):
```

Código 3.18. Fragmento de código para la selección de cadenas de caracteres contenidas en un Google Doc

En el Código 3.19 se empleado un condicional para evaluar cadenas de caracteres que cumplan con el patrón: *"ty":"is"* y se evalúa el texto extraido.

```
244                 if text is not None and '"ty":"is"' in text:
245                     text = text.replace('true', 'True')
246                     text = text.replace('false', 'False')
247                     text = text.replace('null', 'None')
248                     policy_text += ast.literal_eval(text)[0]['s']
```

Código 3.19. Fragmento de código para la selección de texto de un Google doc

En el Código 3.20 se presenta la función *download_pdf()* que recibe como argumento de entrada: la variable *url*, que tiene la política de privacidad. Se comienza la descarga empleando el método *requests.get()*, configurando las opciones *stream=True,* estableciendo una flujo de descarga y *verify=False,* evitando problemas en la comprobación del certificado SSL. Finalmente, en la escritura del documento, se estableció modo de escritura configura en wb(*write bytes*), estableciendo un tamaño de fragmento de 1024 bytes.

```
259  def download_pdf(url):
260      try:
261          logger.debug('The download pdf was start')
262          n_ram = random.randrange(10, 100, 4)
263          pdf_name = 'privacyPolicy' + str(n_ram)
264          responde = requests.get(url, stream=True, verify=False)
265          file = open(result_dir + pdf_name + '.pdf', 'wb')
266          for chunk in responde.iter_content(chunk_size=1024):
267              if chunk:
268                  file.write(chunk)
269      except Exception as e:
270          reason = 'Error while downloading pdf documento from the web'
271          logger.error("download_pdf download failed",
272                       extra={'exception_message': str(e), 'reason': reason})
273      else:
274          logger.info('The pdf download was successful')
275          return pdf_name
```

Código 3.20. Función de descarga directa de documentos PDF

Se extrajo el texto del documento PDF utilizando el método *raw* perteneciente a la bilbioteca *tika*. Se establece en nombre del archivo y se escribe el contenido tanto del archivo de extensión TXT, como se observa en el Código 3.21.

```
400  def pdf2text(file_name):
401      try:
402          logger.debug('Text extraction from PDF document started')
403          raw = parser.from_file( result_dir + file_name+'.pdf')
404          content = raw['content']
405          file = open( result_dir + file_name + ".txt", "w")
406          file.write(content)
407          file.close()
408      except Exception as e:
409          reason = 'Error while extract text from pdf document'
410          logger.error("download_pdf download failed",
411                       extra={'exception_message': str(e), 'reason': reason})
412      else:
413          logger.debug('Extraction of text from a PDF document was successful')
```

Código 3.21. Función de extracción de texto de documentos PDF

El sitio web OneDrive contiene documentos de políticas de privacidad, específicamente documentos de formato DOCX. Se descargo el código HTML de la página utilizando el método *requests.get()*. Se utilizo el código HTML de la política de privacidad para buscar la URL del contenedor del texto de la política de privacidad mediante expresiones regulares. Se busco una línea que comience con la expresión *"var $Config="* y se buscar un *tag* especifico, cuyo contenido es la URL del contenido de la política de privacidad, como se observa en el Código 3.22.

```
293         tag = '"FileGetUrl"'
294         lines = policy_html.split('\n')
295         aux = [x for x in lines if x.startswith('var $Config=')]
296         url_text = aux[0].split(tag)[1].split('"')[1]
```

Código 3.22. Fragmento de código para la búsqueda de la URL del contenido de una política de privacidad

Se descargo el documento de la política de privacidad directa desde la URL del contenido de la política de privacidad, estableciendo la extensión del archivo como DOCX. Finalmente, se extrajo el contenido del documento DOCX que tiene una estructura XML utilizando la bilbioteca *docx2txt*, como se observa en el Código 3.23.

```
304         aux_response = requests.get(url_text, stream=True, verify=False)
305         file = open(result_dir + file_name + '.docx', 'wb')
306         for chunk in aux_responde.iter_content(chunk_size=1024):
307             if chunk:
308                 file.write(chunk)
309         file.close()
310         policy_text = docx2txt.process(result_dir + file_name + '.docx')
```

Código 3.23. Fragmento de código para la descarga y extracción de una política de privacidad

El sitio web Dropbox contiene documentos que alojan políticas de privacidad. Específicamente documentos de formato TXT, HTML, PDF, DOC y DOCX. Se descargo el código HTML de la política de privacidad. Para ello, se utiliza el método *requests.get()*. En el Código 3.24 se presenta un fragmento de código para la selección del método de descarga de un documento alojado en Dropbox. Se evalúa mediante condicionales el formato del archivo para ejecutar funciones adecuadas dependiendo del formato del documento.

```
429         tag = '"preview_url"'
430         url_text = OD_filtrado(policyhtml, tag)
431         if formato == 'txt':
432             OD_html_store(file_name, url_text)
433         elif formato == 'html':
434             OD_text_store(file_name, url_text)
435         elif formato == 'docx' or formato == 'doc' or \
436                 formato == 'rtf'or formato == 'pdf':
437             OD_pdf_store(file_name, url_text, url)
438             pdf2text(file_name)
```

Código 3.24. Fragmento de código para la selección del método de descarga de una política alojada en el sitio Dropbox

En el Código 3.25 se presenta el código *OD_filtrado()* que recibe como argumento de entrada: la variable *policyhtml*, que contiene el código HTML de la política de privacidad; la variable *tag*, que contiene el patrón para la búsqueda y extracción de la URL del contenido de la política de privacidad.

```
362 def OD_filtrado(policyhtml, tag):
363     try:
364         logger.debug('OD_filtrado function has been started')
365         lines = policyhtml.split('\n')
366         aux = [x for x in lines if x.startswith('InitReact.mountComponent')]
367         url_text = aux[0].split(tag)[1].split('"')[1]
368     except Exception as e:
369         reason = 'Error while filter url from dropbox'
370         logger.error("download_pdf download failed",
371                      extra={'exception_message': str(e), 'reason': reason})
372     else:
373         logger.debug('OD_filtrado function has been successful')
374         return url_text
```

Código 3.25. Función para la extracción de una URL del contenido de una política de privacidad

Obtenida la URL del contenido de la política de privacidad. Esta URL alberga una página web de contenido estático para el caso de los documentos de formato TXT y HTML. Se utilizó la función *OD_text_store()* que a partir de una URL del contenido de la política de privacidad se extrae y almacena el texto y código de la política de privacidad, como se observa en el Código 3.26.

```
343     def OD_text_store(file_name, url_text):
344         try:
345             logger.debug('The download txt document from dropbox was start')
346             response = requests.get(url_text)
347             html = response.content
348             soup = BeautifulSoup(html, 'html.parser')
349             policy_html = (str(soup))
350             policy_text = soup.find('body').pre.text
351             file = open( result_dir + file_name + '.txt', "w")
352             file.write(policy_text)
353             file.close()
354             file = open( result_dir + file_name + '.html', "w")
355             file.write(policy_html)
356             file.close()
```

Código 3.26. Fragmento de código para el almacenamiento de un documento TXT de una política de privacidad

En el Código 3.27 se presenta la función *OD_html_store()* que a partir de una URL del contenido de la política de privacidad se extrae y almacena el texto y código de la política de privacidad.

```
321 def OD_html_store(file_name, url_text):
322     try:
323         logger.debug('The download html document from dropbox was start')
324         response = requests.get(url_text)
325         html = response.content
326         soup = BeautifulSoup(html, 'html.parser')
327         policy_html = soup.find('body').pre.text
328         soup = BeautifulSoup(policy_html, 'html.parser')
329         policy_text = soup.find('body').text
330         file = open(result_dir + file_name + '.txt', "w")
331         file.write(policy_text)
332         file.close()
333         file = open(result_dir + file_name + '.html', "w")
334         file.write(policy_html)
```

Código 3.27. Fragmento de código el almacenamiento de un documento HTML de una política de privacidad

Para documentos de formato PDF, DOC, DOCX y RFT se obtiene una URL del contenido de la política de privacidad que permitió la descarga directa del contenido de la política de privacidad en formato PDF, como se observó en el Código 3.20. Finalmente se extrae el texto del documento PDF, como se observó en el Código 3.21.

2.4.4 CREACIÓN DE CONTENEDORES DOCKER - SPRINT 4

Este *sprint* incluye la planeación, creación y ejecución de un contenedor Docker para cada uno de los microservicios.

La planificación y construcción de una imagen Docker comprende la elección del sistema operativo, herramientas, bibliotecas y programas necesarios para la ejecución de cada uno de los microservicios. La creación de una imagen Docker tiene como base un documento denominado Dockerfile. En la Figura 2.10 se observa el ambiente de ejecución del microservicio 1.

```
#Select the SO
FROM ubuntu:18.04

#Dates of developer
LABEL mantenedor="juliomir1994@gmail.com"
LABEL version="1.0"
LABEL description="Imagen Microservicio 1"

#software and dependences
RUN apt update && \
apt -y upgrade && \
apt install -y aapt && \
apt install -y openjdk-11-jdk && \
apt install -y git && \
apt install -y python3 && \
apt install -y python3-pip && \
pip3 install  python-json-logger
```

Figura 2.10 Bibliotecas y utilidades del microservicio 1

En el ANEXOS II se visualiza las URLs a los contenedores Docker de los microservicios.

2.5 RESULTADOS Y DISCUSIÓN

En esta sección se presentan los resultados y discusión del funcionamiento del microservicio.

2.5.1 RESULTADOS

Para la evaluación de los resultados obtenidos de los microservicios 2 y 3 se utilizó una matriz de confusión [31], que, a su vez, permite el cálculo de las siguientes métricas:

$$Exactitud = \frac{Verdaderos\ positivos + Verdaderos\ negativos}{Total\ de\ Casos\ de\ Prueba} \quad (3.1)$$

$$Sensibilidad = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ negativos} \quad (3.2)$$

$$Especificidad = \frac{Verdaderos\ negativos}{Verdadero\ negaticos + Falsos\ positivos} \quad (3.3)$$

$$Precisión = \frac{Verdaderos\ positivos}{Verdadero\ positivos + Falsos\ positivos} \quad (3.4)$$

2.5.1.1 Microservicio 1

Se realizaron pruebas con 20 aplicaciones móviles para la validación del microservicio 1. De esta forma, se pretende ver los resultados del microservicio ante una variedad de aplicaciones móviles. Los resultados de cada APP se detallan en el Anexo III.

El rendimiento del microservicio 1 se observa en la Tabla 3.1.

Tabla 3.1 Resultados del microservicio 1

	Exitosa	Fallida
Determinación de permisos sensibles solicitados.	100%	0%

Se evaluó un total de 20 aplicaciones y se corroboró manualmente que en todos los casos (100%) se detectó correctamente los permisos sensibles solicitados por estas aplicaciones. Esta fiabilidad se debe a que los permisos son declarados por los desarrolladores en el fichero Manifest, que es un documento semiestructurado que puede ser fácilmente analizado mediante procesadores automáticos.

Por otro lado, en la evaluación de las 20 aplicaciones se detectaron 19 aplicaciones que solicitan permisos sensibles. De ellas, dos aplicaciones no presentan una política de privacidad en Play Store, por lo que podría caer en potenciales incumplimientos.

2.5.1.2 Microservicio 2

Este microservicio fue validado a través de la evaluación de 30 aplicaciones móviles. Los resultados de cada aplicación evaluada se detallan en el Anexo IV. El resumen del rendimiento del microservicio 2 se observa en la Tabla 3.2

Tabla 3.2 Resultados del microservicio 2

Exactitud	Sensibilidad	Especificidad	Precisión
63.33%	60,00%	70,00%	80,00%

Habiendo evaluado un total de 30 APPs y luego de verificar manualmente si en verdad se trataban de URLs de políticas de privacidad, el microservicio 2 es capaz de detectar correctamente 12 de 15 URLs que realmente son políticas de privacidad. Los 3 casos de falsos positivos se podrían atribuir al uso de una expresión regular no lo suficientemente precisa en el filtrado de URLs. Los 8 de falsos negativos se podrían atribuir a la falta de

más palabras clave para identificar las URLs de políticas de privacidad. Así también, las URLs de las políticas de privacidad podrían estar embebidas en otro componente del APK. Se discuten más a fondo los resultados del microservicio 2 en la subsección 2.5.2.2

2.5.1.3 Microservicio 3

La validación del microservicio 3 se realizó a través de la evaluación de 500 URLs. Los textos de estas 500 URLs no pudieron ser descargadas con una versión del módulo de descarga previo, así que determina el porcentaje de mejora de la versión actual. Los resultados detallados se muestran en el Anexo V. El rendimiento del microservicio 3 se observa en la Tabla 3.3

Tabla 3.3 Resultados del microservicio 3

	Exitosa	Parcial	Fallida
Descarga	88,00%	2,60%	8,40%

Habiendo evaluado un total de 500 URLs, se puede observar que la versión actual del microservicio ha mejorado en un 88% respecto a su versión previa. Todavía existe un margen de mejora, ya que hay un 8,40% de descargas fallidas de políticas de privacidad y un 2,60% de descargas parciales del texto de políticas de privacidad.

2.5.2 DISCUSIÓN

A continuación, se presenta una discusión sobre los resultados de los microservicios desarrollados.

2.5.2.1 Utilidad del microservicio

1) El microservicio 1 provee insumos relevantes para detectar posibles incumplimientos del requisito de transparencia especificados en la LOPDP. Así, una aplicación que solicita permisos sensibles y no divulga una política de privacidad o protección de datos, podría estar incumpliendo el artículo 67 de la LOPDP, que en su literal 3 define *"No mantener disponibles políticas de protección de datos personales afines al tratamiento de datos personales*" [4] .
2) Algunas organizaciones publican una misma política de privacidad tanto en la aplicación como en la consola de Play Store. No obstante, en el contexto de la evaluación de cumplimiento, se produce un inconveniente (i) cuando una política de privacidad solo se publica dentro de la aplicación o (ii) cuando se publica en ambos lugares, pero hay diferencias en el texto de las políticas de privacidad. En el primer caso, para la evaluación del cumplimiento, si solo se revisa la disponibilidad

de una política de privacidad en la Play Store, se podría caer en un falso positivo (falso incumplimiento), ya que la organización podría haber publicado la política de privacidad dentro de la aplicación. En el segundo caso, sería deseable extraer las políticas de privacidad tanto de la aplicación como de Play Store, para de esta forma poder compararlos y detectar potenciales inconsistencias. En este contexto, el microservicio 2 es muy útil ya que permite descubrir una política de privacidad (su URL) embebida en el interior de un APK, y entonces contribuir a solventar los inconvenientes antes mencionados.

3) El microservicio 3 resulta muy relevante para extraer el texto de políticas de privacidad desde sitios, que presentaban inconvenientes en una versión anterior del microservicio. Sin el texto de las políticas de privacidad, no se pueden evaluar ciertos requisitos de transparencia. Por lo tanto, este microservicio ayuda a mejorar la cobertura de evaluación de aplicaciones.

2.5.2.2 Análisis de Resultados

1) El microservicio 1 obtuvo una tasa de éxito del 100% atribuyéndoselo al correcto desempaquetado y extracción de permisos desde el fichero Manifest, que es un documento semiestructurado y que, por lo tanto, es relativamente simple de procesar. Los resultados obtenidos en el microservicio 1 permiten proveer un insumo para determinar un potencial incumplimiento a la LOPDP.
2) El microservicio 2 presenta una sensibilidad del 60,00% y una especificidad del 70,00%. Los 3 casos de falsos positivos que afectan a la especificidad del microservicio se le atribuyen al uso de la expresión regular “http*”, que incluye tantas páginas web http y htttps, pero en algunas aplicaciones hacen uso de código que utilizan métodos como “*http.google.privacy”*, provocando los falsos positivos en la identificación de URLs de políticas de privacidad. Los 8 casos de falsos negativos que afectan a la sensibilidad se le atribuyen a la falta del uso de más palabras clave para la búsqueda de URLs de políticas de privacidad embebidas en el APK. Así también, las URLs de las políticas de privacidad que podrían estar embebidas en otros componentes del APK (p.ej., Resource, assets).
3) El microservicio 3 ha mejorado la descarga exitosa de políticas de privacidad en un 88% el índice de descargas exitosas. Entre el 12% de descargas fallidas restante, el 8,40% ha sido identificado como URLs de políticas de privacidad con problemas durante la verificación de disponibilidad de la política de privacidad, entre los mensajes de error de los logs se mencionan los problemas: *timeout* durante la verificación de la disponibilidad de la política de privacidad, error durante la

verificación del certificado SSL y el uso de software de seguridad para evitar ataques de bots identificado como Bootstrap. El 2,60% se identifican como descargas parciales de políticas de privacidad, atribuyéndoselo principalmente el uso de *frames* en su gran mayoría. También, se identificó el uso bibliotecas y marcos front-end como: React, ZURB Foundation y Uikit, que podría ser abordadas en trabajos futuros

2.5.2.3 Limitaciones de los microservicios desarrollados.

1) El microservicio 1 está limitado por la lista de permisos sensibles; es decir, debido a las constantes actualizaciones de Android, podría quedar obsoleta. Así, se podrían generar falsos negativos. No obstante, el microservicio posee un fichero JSON externo para actualizar esta lista fácilmente.
2) El microservicio 2 depende principalmente de la efectividad de Androguard, una biblioteca de Python utilizado para ingeniería inversa. La evaluación de esta biblioteca está fuera del alcance del presente trabajo. Otra limitación es el algoritmo utilizado para detectar la pertenencia de una URL a un dominio. Actualmente, se están empleando tokens[25] para determinar su pertenencia, y una organización podría no contener campos en común con la URL de su política de privacidad y, por lo tanto, no se podría detectar su pertenencia.
3) Existe una cantidad muy grande y diversas de tecnologías web que utilizan las organizaciones para divulgar sus políticas de privacidad. En este trabajo no hemos podido tratarlas a todas, sino que nos hemos centrado en las más prioritarias. En función de evaluaciones previas, se identificó que ciertas organizaciones emplean principalmente webs estándar, Drive, OneDrive, Dropbox; siendo estas las casuísticas abordadas en el desarrollo del componente y dejando otras casuísticas para trabajos futuros.

3 CONCLUSIONES Y RECOMENDACIONES

El presente capítulo presenta las conclusiones y recomendaciones obtenidas en el desarrollo del presente trabajo de integración curricular.

3.1 CONCLUSIONES

A continuación, se presentan las conclusiones formuladas a partir del desarrollo de este trabajo de integración curricular.

[25] tokens: referencia a un identificador

- Los fundamentos teóricos estudiados durante la realización del presente trabajo me permitieron adentrarme en un tema novel en nuestro país, para principalmente entender el contexto en el cual se evalúan los sistemas de software en general, incluyendo a la privacidad, datos personales, protección de datos personales y políticas de privacidad.
- La evaluación de la privacidad en las aplicaciones móviles es algo novedoso en Ecuador, además de obligatorio una vez entrada en rigor la LOPDP en mayo del 2021. Así, este trabajo de integración curricular contribuye a la evaluación de cumplimiento de las aplicaciones móviles a través de la provisión de los tres microservicios desarrollados.
- Los microservicios del componente fueron desarrollados guiándose en las buenas prácticas definidas en Scrum. Esta facilitó una constante interacción con el equipo Scrum, desarrollo iterativo del componente y facilidad en la planificación de los microservicios.
- El uso de contenedores Docker para el despliegue de los microservicios facilita la ejecución de los microservicios independientemente del entorno físico que ejecute los microservicios.
- Mediante las pruebas de funcionalidad se verificó que los microservicios cumplen con los requerimientos funcionales planteados desde el inicio por el *product owner*. No obstante, hay que reconocer que el Microservicio 3, que propone la extracción de textos de una página web, aún podría ser mejorado, abordando nuevas casuísticas, que por cuestiones de tiempo y alcance del presente trabajo no han sido consideradas.
- Con base en base los resultados obtenidos de la evaluación de los microservicios, se reconoce que los microservicios requieren funcionalidades adicionales durante el desarrollo del componente. En este sentido, durante el desarrollo de los microservicios se tuvo presente buenas prácticas de programación, incluyendo la legibilidad del código, reutilización de funciones y organización modular del código. Facilitando que los microservicios sean extendidos por terceras personas que requieran implementar nuevas funcionalidades.

3.2 RECOMENDACIONES

En esta sección se indican las recomendaciones formuladas a partir del desarrollo de este trabajo de integración curricular.

- Para el microservicio 1 se sugiere abordar el muestreo de otro componente del archivo AndroidManifest.xml, como por ejemplo los elementos intent.
- Para el microservicio 2 se sugiere mejoras en la utilizando una mejor expresión regular para el filtrado de URLs de cadenas de caracteres utilizando Androguard.
- Para el microservicio 2 se sugiere mejorar la utilización de palabras clave mediante un archivo JSON para su fácil modificación sin tocar el código fuente. Así mismo, se podría enfocar en el análisis de otro componente del APK (p.ej., Resources, assets).
- Para el microservicio 3 se sugiere abordar otras casuísticas durante la descarga del texto de políticas de privacidad de la gran variedad de tecnologías web utilizadas por los proveedores de las aplicaciones para publica sus políticas, enfocándose principalmente en la utilización de *frames*.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] S. Kemp, “DIGITAL 2021: ECUADOR,” Feb. 11, 2021. https://datareportal.com/reports/digital-2021-ecuador (accessed Dec. 22, 2021).

[2] “Mobile Operating System Market Share Ecuador.” https://gs.statcounter.com/os-market-share/mobile/ecuador/#monthly-202004-202104-bar (accessed Dec. 22, 2021).

[3] J. M. del Alamo, D. Guaman, B. Balmori, and A. Diez, “Privacy Assessment in Android Apps: A Systematic Mapping Study,” *Electronics 2021, Vol. 10, Page 1999*, vol. 10, no. 16, p. 1999, Aug. 2021, doi: 10.3390/ELECTRONICS10161999.

[4] Asamblea Nacional del Ecuador, *Ley Orgánica de Protección de Datos Personales*. 2021. Accessed: Dec. 22, 2021. [Online]. Available: https://www.telecomunicaciones.gob.ec/wp-content/uploads/2021/06/Ley-Organica-de-Datos-Personales.pdf

[5] C. Troncoso, “Privacy & Online Rights Knowledge Area,” Oct. 2019. Accessed: Feb. 18, 2022. [Online]. Available: https://www.cybok.org/media/downloads/Privacy__Online_Rights_issue_1.0_FNULPeI.pdf

[6] Organización de las Naciones Unidas, “Declaración Universal de los Derechos Humanos,” *1948*, Accessed: Dec. 06, 2021. [Online]. Available: www.lexis.com.ec

[7] Biblioguias - Biblioteca CEPAL, “Protección de los datos - Gestión de datos de investigación,” Dec. 18, 2020. https://biblioguias.cepal.org/c.php?g=495473&p=4398118 (accessed Dec. 22, 2021).

[8] G. Pisanu and E. Massé, “Protección de datos: ¿por qué es importante y cómo debes hacerlo?,” May 07, 2018. https://www.accessnow.org/proteccion-de-datos-es-importante/ (accessed Dec. 22, 2021).

[9] Aqua Social Media, “¿Qué es y para qué sirve la Política de Privacidad?,” Mar. 31, 2016. https://www.aquasocialmedia.com/blog-dynamic/91-que-es-y-para-que-sirve-la-politica-de-privacidad (accessed Apr. 05, 2022).

[10] J. M. del Alamo, D. S. Guaman, B. García, and A. Diez, “A systematic mapping study on automated analysis of privacy policies,” *Computing*, vol. 104, no. 9, pp. 2053–2076, May 2022, doi: 10.1007/S00607-022-01076-3/FIGURES/5.

[11] Asamblea Nacional del Ecuador, “Constitución de la República del Ecuador,” Oct. 2008, Accessed: Dec. 06, 2021. [Online]. Available: www.lexis.com.ec

[12] Chakray, “¿Qué son los microservicios? Caraterísticas y ventajas.” https://www.chakray.com/es/que-son-los-microservicios-definicion-caracteristicas-y-ventajas-y-desventajas/ (accessed Feb. 15, 2022).

[13] S. Lázaro, “Análisis del entorno Android: aplicaciones y el sistema de permisos,” 2017. Accessed: Feb. 18, 2022. [Online]. Available: https://zaguan.unizar.es/record/61407/files/TAZ-TFG-2017-037.pdf

[14] Android Developers, “Permisos en Android.” https://developer.android.com/guide/topics/permissions/overview (accessed Dec. 08, 2021).

[15] Android Developers, “Manifest.permission.” https://developer.android.com/reference/android/Manifest.permission (accessed Dec. 25, 2021).

[16] Cl. Weeks, “Guía sobre los permisos de las aplicaciones de Android y cómo usarlos debidamente | AVG,” Jan. 28, 2018. https://www.avg.com/es/signal/guide-to-android-app-permissions-how-to-use-them-smartly (accessed Apr. 05, 2022).

[17] K. Kendall and C. McMillan, “Practical Malware Analysis”.

[18] Editorial Etecé, “Página Web - Concepto, tipos y para qué sirve,” Aug. 05, 2021. https://concepto.de/pagina-web/ (accessed Feb. 16, 2022).

[19] M. López, “Páginas web estáticas vs páginas web dinámicas | OpenWebinars,” Feb. 01, 2021. https://openwebinars.net/blog/paginas-web-estaticas-vs-paginas-web-dinamicas/ (accessed Apr. 06, 2022).

[20] F. Penalba, “Qué es una página web: partes, estructura y contenido,” Feb. 04, 2021. https://raiolanetworks.es/blog/partes-de-una-pagina-web-estructura-elementos-y-contenido/ (accessed Jan. 16, 2022).

[21] B. Traynor, “Android aapt,” Jun. 22, 2010. https://elinux.org/Android_aapt (accessed Feb. 22, 2022).

[22] K. Reitz, “Requests: HTTP for Humans.” https://docs.python-requests.org/en/latest/ (accessed Feb. 16, 2022).

[23] B. Muthukadan, “Selenium with Python,” 2018. https://selenium-python.readthedocs.io/ (accessed Dec. 01, 2021).

[24] G. Sánchez, “Selenium WebDriver en un Ambiente de Pruebas Continuas.” https://sg.com.mx/revista/54/selenium-webdriver-un-ambiente-pruebas-continuas (accessed Feb. 16, 2022).

[25] L. Richardson, “Beautiful Soup Documentation.” https://www.crummy.com/software/BeautifulSoup/bs4/doc/#translating-this-documentation (accessed Feb. 16, 2022).

[26] A. Desnos, “Androguard Documentation,” Oct. 18, 2019.

[27] C. Aneiro, “¿Conoces Docker? Descubre como automatizar el despliegue de aplicaciónes,” Jul. 09, 2021. https://ivorysoluciones.com/blog/docker-y-el-desarrollo-web/ (accessed Dec. 28, 2021).

[28] I. de Fez, P. Arce, R. Belda, and J. C. Guerri, “Uso de contenedores Docker en el entorno educativo,” *Congreso In-Red 2021*, Jul. 15, 2021. https://riunet.upv.es/bitstream/handle/10251/175650/deArceBelda%20-%20Uso%20de%20contenedores%20Docker%20en%20el%20entorno%20educativo.pdf?sequence=1 (accessed Jan. 01, 2022).

[29] E. Abellán, “Metodología Scrum: qué es y cómo funciona,” Mar. 05, 2020. https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html (accessed Mar. 30, 2022).

[30] K. Schwaber and J. Sutherland, “The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game,” Nov. 2020.

[31] J. Barrios, “La matriz de confusión y sus métricas,” Jul. 26, 2019. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/ (accessed Apr. 06, 2022).

5 ANEXOS

Esta sección presenta un resumen de los anexos que fueron necesarios para este trabajo.

ANEXO I. Lista de Permisos Sensibles

ANEXO II. Enlaces hacia los contenedores Docker de los microservicios

ANEXO III. Resultados del Microservicio 1

ANEXO IV. Resultados del Microservicio 2

ANEXO V. Resultados del Microservicio 3

ANEXO I. Lista de Permisos Sensibles

N	Tipo de dato personal	Descripción	Nombre del permiso	Nivel de protección
1	**Mensajes y llamadas**	SMS, MMS, llamadas, emails	RECEIVE_SMS	Peligrosa
2			READ_SMS	Peligrosa
3			RECEIVE_MMS	Peligrosa
4			PROCESS_OUTGOING_CALLS	Peligrosa
5			RECEIVE_WAP_PUSH	Peligrosa
6			USE_SIP	Peligrosa
7			CALL_PHONE	Peligrosa
8			ANSWER_PHONE_CALLS	Peligrosa
9			ACCEPT_HANDOVER	Peligrosa
10	**Audio y video**	Multimedia	CAMERA	Peligrosa
11			RECORD_AUDIO	Peligrosa
12			ADD_VOICEMAIL	Peligrosa
13	**Sensores**	Acceso a sensores personales o ambientales	ACTIVITY_RECOGNITION	Peligrosa
14			BLUETOOTH_ADVERTISE	Peligrosa
15			BODY_SENSORS	Peligrosa
16			BODY_SENSORS_BACKGROUND	Peligrosa
17			BLUETOOTH_CONNECT	Peligrosa
18	**Localización**	Localización fina (GPS), localización gruesa (ej. Inferida de información de redes Wi-Fi)	ACCESS_COARSE_LOCATION	Peligrosa
19			ACCESS_FINE_LOCATION	peligrosa
20			ACCESS_BACKGROUND_LOCATION	Peligrosa
21			ACCESS_WIFI_STATE	Normal
22			ACCEES_NETWORK_STATE	Normal
23			ACCESS_LOCATION_EXTRA_COMMANDS	Normal
24			ACCESS_MEDIA_LOCATION	Peligrosa
25	**Almacenamiento externo**	Almacenamiento externo	WRITE_EXTERNAL_STORAGE	Peligrosa
26			READ_EXTERNAL_STORAGE	Peligrosa
27	**Contacto**	Lista de contactos	READ_CONTACTS	Peligrosa
28			WRITE_CONTACTS	Peligrosa
29	**Uso/Historial**	Preferencias de usuario, llamadas, registros de uso	READ_CALL_LOG	Peligrosa
30			WRITE_CALL_LOG	Peligrosa
31	**Calendario**	Calendario del usuario	READ_CALENDAR	Peligrosa
32			WRITE_CALENDAR	Peligrosa
33	**Identificadores del terminal**	Identificadores que pueden servir para identificar al usuario	READ_PHONE_STATE	Peligrosa
34			READ_PRECISE_PHONE_STATE	peligrosa
35			READ_BASIC_PHONE_STATE	peligrosa
36			READ_PHONE_NUMBER	Peligrosa
37			GET_ACCOUNTS	Peligrosa

38	**Credenciales**	Contraseñas, PINS	USE_BIOMETRIC	Normal
39			USE_FINGERPRINT	Normal
40	**Comunicaciones**	Conectividad	INTERNET	Normal

ANEXO II. Enlaces hacia los contenedores Docker de los microservicios

- Microservicio 1
  - Imagen: https://hub.docker.com/layers/rashec/ubuntuservice1/1.0-i/images/sha256-0dd499adfafa84398ca9b78da89192381547b512566f39823844ca035c6d258f?context=repo
  - Dockerfile: https://epnecuador-my.sharepoint.com/:t:/g/personal/julio_miranda01_epn_edu_ec/ESStjJ-2GJJLoCNa2-zZhqIBG9ME9S5Q68bnkh8nhvu2Og?e=VgFwuj
  - Despliegue: docker pull rashec/ubuntuservice1:1.0-i
- Microservicio 2
  - Imagen: https://hub.docker.com/layers/rashec/ubuntuservice2/1-b/images/sha256-b624432869c0cb381dcf5a524b27c8a2aa43ceb32e2c95f52ec077b58f2b66c8?context=repo
  - Dockerfile: https://epnecuador-my.sharepoint.com/:t:/g/personal/julio_miranda01_epn_edu_ec/ETe7Y3dkIFpIi7-1U3Hz_P8BUXeTlsFmeW20c4ywGdfHrA?e=8IaCvP
  - Despliegue: docker pull rashec/ubuntuservice2:1-b
- Microservicio 3
  - Imagen: https://hub.docker.com/layers/rashec/ubuntuservice3/1.0-c/images/sha256-3de7012d9e98014f0aad5cae99b26f4354e1dc90035221363bbc393c22fd08a5?context=repo
  - Dockerfile: https://epnecuador-my.sharepoint.com/:t:/g/personal/julio_miranda01_epn_edu_ec/EbDB5pYaVfVHnD2AQdtcbhMBzH57hkcQDgnxQYjuDCJWXA?e=G9PCLn
  - Despliegue: docker pull rashec/ubuntuservice3:1.0-c

ANEXO III. Resultados del Microservicio 1

N	Aplicación	Versión	Solicitud de permisos sensibles (verificación manual)	Solicitud de permisos sensibles (extracción automática)	¿La APP requiere una política de privacidad?
1	Draughts	2,11,1	Sí	Sí	Sí
2	Asamblea Ecuador	2,0,6	Sí	Sí	Sí
3	RealCalc	2,3,1	Sí	Sí	No
4	Compass	3,3,3	Sí	Sí	Sí
5	Speedtest	4,6,13	Sí	Sí	Sí
6	Translator	1,5	Sí	Sí	Sí
7	Metal	1,6,0	Sí	Sí	Sí
8	Banca móvil	7,2,0	Sí	Sí	Sí
9	8 Ball Pool	5,6,1	Sí	Sí	Sí
10	Banco Guayaquil	8,0,2	Sí	Sí	Sí
11	Ch Farina	1,0,9	Sí	Sí	Sí
12	Multicines Ecuador	3,7,4	Sí	Sí	Sí
13	Glovo	5,149,0	Sí	Sí	Sí
14	BdP	7,2,02	Sí	Sí	Sí
15	Produbanco	5,0,0	Sí	Sí	Sí
16	PATIOTuerca	3,1,3	Sí	Sí	Sí
17	Tinder	13,0,0	Sí	Sí	Sí
18	ToyotaGo	4,0,1	Sí	Sí	Sí
19	El Hornero	1,1,4	Sí	Sí	Sí
20	Servientrega	1,0,7	Sí	Sí	Sí

ANEXO IV. Resultados del Microservicio 2

N	Aplicación	Versión	Disponibilidad de una política de privacidad (extracción automática)	Disponibilidad de una política de privacidad (verificación manual)
1	Marine Weather	6,7	Sí	Sí
2	Compass	3,3,3	Sí	Sí
3	Whatsapp	2,22,2,73	Sí	Sí
4	be Produbanco	4,1,4	Sí	Sí
5	Sticker Maker	0,0,3-20	Sí	Sí
6	Toddler Games	25	Sí	No
7	CPN	7,1,3	No	No
8	ETAFASHION	1,0,0	No	Sí
9	Banco Guayaquil	8,0,2	Sí	Sí
10	BdP	7,2,02	No	No
11	Speedtest	4,6,17	Sí	Sí
12	Glovo	5,149,0	No	No
13	Ch Farina	1,0,9	No	Sí
14	Mercado Libre	10,195,2	Sí	Sí
15	Epson iPrint	7,7,1	No	Sí
16	Gob.EC	2,1,2	No	Sí
17	Fing	11,6,0	Sí	Sí
18	JEP Móvil	14,1	No	No
19	ToyotaGo	4,0,1	No	Sí
20	mychevrolet	5,9,0	No	Sí
21	Cinemark Ecuador	5,2,6	Sí	Sí
22	KFC APP	2,6,8	No	No
23	SRI Móvil	1,17,5	No	Sí
24	PATIOTuerca	3,1,3	Sí	No
25	Salud EC	1,29	No	No
26	Servientrega	1,0,7	No	No
27	Drink Water Reminder	16,0	Sí	Sí
28	8 Ball Pool	5,6,1	Sí	Sí
29	Telegram	8,4,4	No	Sí
30	Tinder	13,0,0	No	Sí

ANEXO V. Resultados del microservicio 3

El enlace de los resultados del microservicio 3 se puede visualizar en el enlace: https://epnecuador-my.sharepoint.com/:x:/g/personal/julio_miranda01_epn_edu_ec/EeQDgPQms61EvNXeMaf-Dl0BilItZHALl_pcXAMDnu-Kow?e=KLES0k