TRABAJO DE FIN DE GRADO FACULTAD DE INFORMÁTICA DE BARCELONA

Aplicación móvil para ereuse.org: origen y reutilización de dispositivos móviles

Autor: Joel Alarcón Julián
Especialidad: Tecnologías de la Información

Director: Leandro Navarro Moldes
Departamento: Arquitectura de Computadores

Fecha de defensa: 29/04/2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

FIB

Resumen

Aplicación móvil para ereuse.org: origen i reutilización de dispositivos móviles
por Joel Alarcón Julián

Uno de los problemas emergentes en la actualidad es el incremento de la cantidad de WEEE producidos anualmente, y el aumento en la polución que conlleva la producción de recursos que sustituyen estos residuos. Una de las propuestas contra este problema es la economía circular, un sistema económico centrado en convertir recursos que se encuentran al final de su vida útil en recursos para nuevos productos, cerrando así ciclos en el ecosistema industrial y minimizando residuos [1].

Sin embargo, para que la infraestructura de la economía circular funcione, es necesaria la colaboración entre el consumidor y el propietario de las materias primas [2]. eReuse se plantea como el puente entre estos dos actores, realizando la gestión de los dispositivos entre ambos.

Impulsando este ideal, en este proyecto desarrollamos una aplicación móvil que facilita la comunicación entre consumidor y eReuse mediante la recogida de datos y comunicación con la base de datos del proyecto. Los resultados de este proyecto muestran la gran limitación que existe a la hora de adquirir información del dispositivo sin permisos root, así como el obstáculo que esta ausencia de información supone para diseños futuros.

Resum

Aplicació mòbil per ereuse.org: origen i reutilització de dispositius mòbils
por Joel Alarcón Julián

Un dels principals problemes emergents en l'actualitat és l'increment de la quantitat de WEEE produït anualment i l'augment en la pol·lució que implica la producció de recursos que reemplacen aquests residus. Una de les propostes contra aquests problemes és la economia circular, un sistema econòmic enfocat a convertir recursos al final de la seva vida útil en recursos per nous productes, tancant d'aquesta manera cicles en l'ecosistema industrial i minimitzant residus [1].

Tot i així, per a que la infraestructura de la economia circular funcioni, requerim la col·laboració entre consumidor i el propietari de les matèries primes [2]. eReuse es planteja com el pont entre tots dos actors, realitzant la gestió dels dispositius entre tots dos bàndols.

Impulsant aquest ideal, en aquest projecte desenvolupem una aplicació mòbil que facilita la comunicació entre consumidor i eReuse mitjançant la recollida de dades i comunicació amb la base de dades del projecte. Els resultats d'aquest projecte mostren la gran limitació que existeix a l'hora d'obtenir informació del dispositiu sense permisos root, així com l'obstacle que aquesta absència d'informació suposa per a dissenys futurs.

Abstract

Mobile app for ereuse.org: source and reuse of mobile devices
by Joel Alarcón Julián

One of the main emergent issues present is the increase of the quantity of WEEE produced yearly and the rise of pollution due to the production of resources that replaces this waste. One of the proposals to resolve this issue is the circular economy, an economic system focused on turning resources at the end of its life cycle into useful resources, closing this way cycles in the industrial ecosystem and reducing waste [1].

However, for the circular economy's infrastructure to work, it's required a collaboration between consumer and owner of raw materials [2]. eReuse arises as the bridge between both players, doing the management of the devices between both of them.

Boosting this ideal, in this project we develop a mobile application that eases the communication between the consumer and eReuse by gathering data and communicating with the database of the project. The results in this project show the restrictions in regards to gathering data of a mobile device without access to root, and how this lack of information impacts future designs.

Índice

1. Introducción

En la actualidad, múltiples dispositivos de empresas y entidades públicas son desmantelados y reciclados a gran velocidad en lugar de ser reutilizados, reparados o reaprovechados, pese a encontrarse en condiciones perfectas [3].

Una de las investigaciones sobre residuo digital estima que aproximadamente 41.8 millones de toneladas métricas (Mt) de residuo electrónico se generaron en 2014 y que este valor se iba a incrementar a 50 Mt para el 2018 [3], y se estimó como previsión un incremento del 3-5% al año de este residuo [3][4]. Esencialmente, estamos viendo un incremento en el residuo digital generado por año, un residuo que se podría aprovechar para conservar recursos naturales y evitar polución [5].

Una forma de aprovechar esta WEEE es implementar la economía circular, que se basa en convertir recursos que se encuentran al final de su vida útil en recursos para otros, cerrando ciclos en el ecosistema industrial y minimizando residuos [1]. La economía circular implicaría un cambio en la lógica económica dado que sustituye producción con suficiencia: reutilizar lo que se pueda, reciclar lo que no se pueda reutilizar, reparar lo que esté roto y manufacturar lo que no se pueda reparar [1].

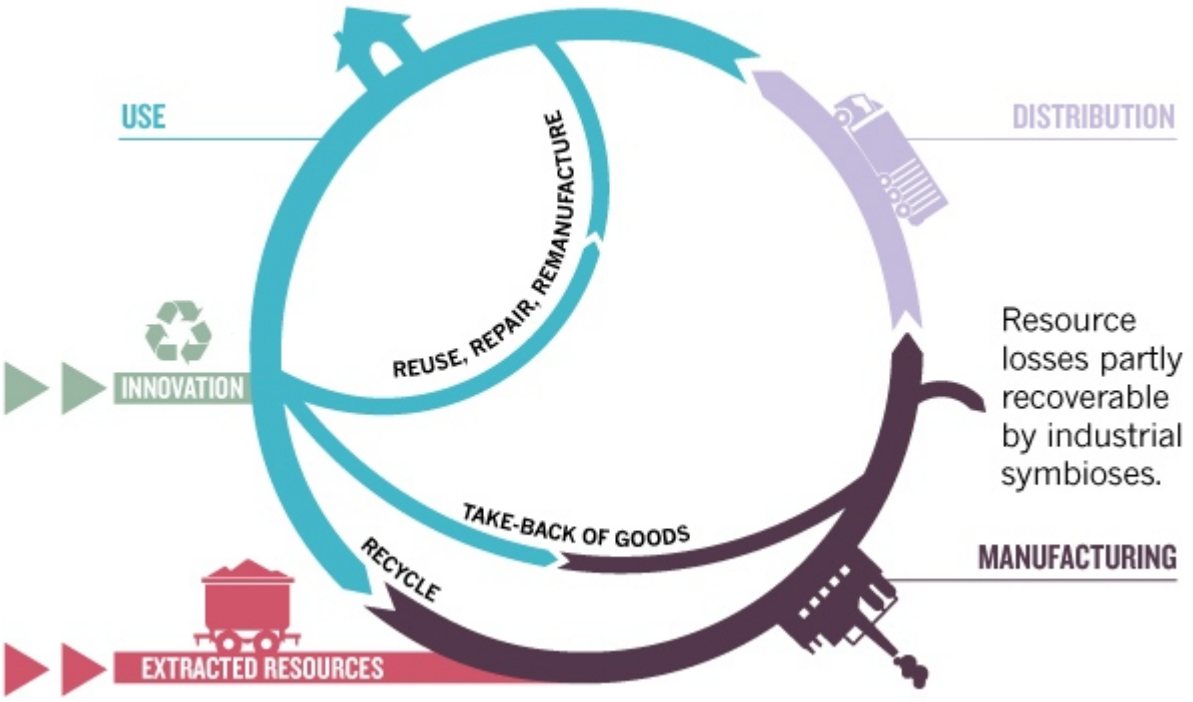

Figura 1: Ciclos cerrados en la economía circular (Fuente: [1])

Sin embargo, para que la infraestructura de la economía circular funcione, es necesaria la colaboración entre el consumidor y el propietario de las materias primas [2]. Una propuesta para conseguir esta colaboración es eReuse.org, una serie de herramientas de código libre, procedimientos, datos libres y servicios en forma de recursos de uso común (CPR) que pretenden aplicar la economía circular en los dispositivos electrónicos mediante la promoción de la reutilización de estos dispositivos y asegurando su trazabilidad hasta que se reciclan o reutilizan [6].

Figura 2: Ciclos de reutilización en la vida de los dispositivos digitales. (Fuente: [6])

Tras la formación de eReuse.org como proyecto en el 2015, se han desarrollado:

- **eReuse Workbench**: conjunto de herramientas que asisten en el registro y seguimiento de dispositivos mediante la captura y gestión de información, así como la instalación de GNU/Linux OS de ser posible [7]. Esta versión del Workbench está diseñada para ordenadores y dispositivos de escritorio y portátiles.

- **eReuse DeviceHub:** un Device Asset Management System ofrecida como una web app y enfocada en la administración del ciclo de los dispositivos, con el objetivo de ofrecer una administración TI para donantes de dispositivos, así como automáticamente recolectar, analizar, procesar y compartir metadatos sobre dispositivos con otras herramientas de eReuse [7].
- **Global Record of Devices:** es un log online que mantiene una lista global de información trazable sobre dispositivos para poder asistir en caso necesario [7].
- **Photobox:** Una máquina que asiste en calificar la apariencia de un dispositivo a través de una foto desde distintas posiciones [7].

Teniendo en cuenta todas las tecnologías desarrolladas, el siguiente paso de eReuse es el de introducir un medio de comunicación entre el usuario de un dispositivo y el DeviceHub. Este proyecto pretende crear este enlace para los usuarios con dispositivos móviles. De esta forma, tenemos una vía de comunicación más directa y descentralizada con el consumidor que el proyecto pretende conectar con la economía circular.

Por tanto, este proyecto propone una solución que fortalecería el sector de reciclado y reutilización en Europa, reduciendo notablemente la WEEE en el sector móvil y, en consecuencia, provocando un impacto social positivo.

Los resultados de este proyecto se han puesto en producción junto al resto de herramientas ofrecidas por eReuse y estarán disponibles al público.

1.1. Objetivo

El objetivo de este proyecto es el desarrollo de una aplicación móvil que nos permite averiguar información del dispositivo donde la aplicación ha sido instalada y, con estos datos, informar sobre las circunstancias del dispositivo y orientar sobre cómo aprovecharlo lo mejor posible, en caso que el usuario no lo requiera más. De esta manera, esta aplicación móvil permite el registro de dispositivos móviles de forma similar a eReuse Workbench para ordenadores.

Para la implementación de la aplicación, se ha tenido que considerar primero las distintas plataformas de desarrollo disponibles basándose tanto en la recogida de datos como en la comunicación con DeviceHub. Una vez escogida la plataforma se ha procedió al estudio del material antes de la implementación.

A continuación, se programó la gestión de permisos del dispositivo para tener acceso a la información del dispositivo y la propia recolección de los datos. Una vez todo lo relacionado con la obtención de información se programó, se creó un JSON donde se guardaron los datos que deseamos enviar y enviamos el JSON mediante un POST al DeviceHub, el cual responderá dependiendo de la situación. Finalmente, gestionamos la respuesta del DeviceHub y se muestra la respuesta al usuario.

1.2. Características

Los principales aspectos que la aplicación móvil debe tener son los siguientes:

- Recogida de información sobre características del dispositivo móvil donde la aplicación se haya instalado.
- La información obtenida debe ser relevante para la reutilización o reciclado del dispositivo móvil donde la aplicación se haya instalado.
- Envío de datos con el DeviceHub de eReuse.
- Acceso a la aplicación desde un amplio perfil de usuarios, incluyendo usuarios no expertos.
- Disponibilidad y transparencia del código y su funcionalidad con el público. Ha de ser un proyecto de código abierto.

1.3. Actores

- **Ciudadanía**: La aplicación de este proyecto va dirigida al ciudadano medio con un dispositivo móvil que ya no utiliza, y desea asegurarse que se utilizará de la mejor forma posible. El perfil que consideraremos es de un ciudadano con nociones básicas de usuario de su dispositivo.

- **Entidades públicas:** Escuelas, ayuntamientos u otras entidades públicas que buscan utilizar dispositivos de segunda mano se benefician de esta aplicación, dado que tendrían más material disponible. También puede aplicarse a la inversa y utilizar la aplicación para realizar donaciones.

- **Plantas de reciclaje:** Los usuarios de la aplicación cuyos dispositivos no se puedan aprovechar serían dirigidos a una planta de reciclaje, asegurando así un mayor número de dispositivos encauzados a estas plantas y que no son exportados ilegalmente a otros países.

- **Equipo de eReuse**: Equipo encargado de diseñar, implementar, probar y mantener la aplicación. En el ámbito del desarrollo de la aplicación, estará compuesto por el autor del TFG, que programará la aplicación; el ponente, que se encargará de supervisar el proyecto; los programadores responsables del DeviceHub de eReuse, que se comunicarán con el programador de la aplicación para asegurar que la aplicación se pueda comunicar adecuadamente; el tester, que se encargará de probar la aplicación y que funcione adecuadamente.

- **Gestor de inventario:** Es aquel que guarda un registro de dispositivos móviles disponibles para reutilización. Puede ser desde una empresa que solicita a sus empleados instalar la aplicación para realizar un seguimiento hasta una entidad pública que desea gestionar un conjunto de dispositivos. Esencialmente, son todas aquellas figuras que reciben los datos que la aplicación recoge.

2. Estado del arte

En esta sección abordamos todo el software que se ha considerado antes del desarrollo de la aplicación, así como otros productos en el mercado con similitudes con nuestro proyecto y por qué no resuelven las necesidades que intentamos resolver.

2.1. Software considerado antes el desarrollo

2.1.1. DeviceHub

- **¿Qué es?:** Es un Device Asset Management System ofrecida como una web RESTful y enfocada en la administración del ciclo de los dispositivos, con el objetivo de ofrecer una administración TI para donantes de dispositivos, así como automáticamente recolectar, analizar, procesar y compartir metadatos sobre dispositivos con otras herramientas de eReuse [7].
- **¿Por qué es relevante?:** Es el sistema que utiliza eReuse.org para gestionar toda la información que reciben de donantes y, por tanto, nuestra aplicación debe comunicarse con ella. Dispone de una API RESTful capaz de leer JSONs.
- **¿Se ha utilizado?:** Lo acabamos usando para gestionar la información obtenida. Nuestra aplicación se comunica con la API de DeviceHub. Al ser la comunicación con DeviceHub uno de los requisitos de la aplicación, no se han considerado otras posibles alternativas.

2.1.2. Ionic

- **¿Qué es?:** Ionic es un framework de código abierto enfocado en aplicaciones híbridas para múltiples plataformas utilizando HTML5, CSS3 y Javascript [8].
- **¿Por qué es relevante?:** Nos permite obtener información de varios sistemas operativos, ofreciendo el mayor alcance entre los distintos frameworks y SDKs que se han considerado. Utiliza HTML5, CSS3 y Javascript, que el estudiante sabe utilizar, por lo que reduce el tiempo de aprendizaje para poder desarrollar la aplicación notablemente.
- **¿Se ha utilizado?:** Uno de los principales problemas que tenemos con Ionic es que estamos incluyendo varios sistemas operativos a tener en cuenta en la implementación de la aplicación. Esto es especialmente problemático si tenemos en cuenta que queremos obtener datos sensibles de bajo nivel, y que la documentación relacionada con el dispositivo no es demasiado extensa a primera vista [9].

2.1.3. React Native

- **¿Qué es?:** React Native es un framework de código abierto desarrollado por Facebook basado en React, el cual utiliza JavaScript. Permite crear aplicaciones nativas para Android y iOS [10].

- **¿Por qué es relevante?:** Permite desarrollar aplicaciones nativas tanto para Android y iOS y utiliza JavaScript como lenguaje, el cual el estudiante conoce, reduciendo el tiempo de aprendizaje necesario [10]. Dispone de una documentación bastante extensa y bien definida.
- **¿Se ha utilizado?:** Sufre del mismo problema que Ionic respecto a los objetivos de nuestra aplicación, y es que ofrece pocos recursos y/o alternativas relacionados con información de hardware al ser compatible con varios sistemas operativos. Si bien este problema es menos severo que con Ionic (se puede obtener información como el fabricante o el modelo [11]), hay alternativas que exploran información de bajo nivel con más éxito.

2.1.4. Android Studio

- **¿Qué es?:** Android Studio [12] es el IDE oficial de Android [13], que es un sistema operativo gratuito y se puede instalar sin coste alguno en todo dispositivo móvil que lo requiera. Android Studio basado en IntelliJ, por lo que comparten varias funcionalidades.
- **¿Por qué es relevante?:** Al ser desarrollada con Android en mente, dispone de múltiples herramientas para facilitar y garantizar que la aplicación desarrollada funcione para Android, que es uno de los sistemas operativos más extendidos en la actualidad para dispositivos móviles. Entre estas herramientas destacan la automatización de APKs, un editor visual de la interfaz y, particularmente, el conjunto de herramientas que forman Android SDK, la cual dispone de una documentación muy extensa sobre obtener información de bajo nivel del dispositivo. Soporta el lenguaje de programación orientada a objetos Java [14], el cual el estudiante sabe utilizar y, por tanto, reduce tiempo de aprendizaje.
- **¿Se ha utilizado?:** Dado que la gran mayoría de dispositivos que eReuse maneja utilizan o pueden utilizar Android, y al tener Android Studio herramientas que garantizan la compatibilidad con Android, se acabó utilizando Android Studio como IDE del proyecto. El coste nulo de la herramienta también ha tenido un gran peso.

2.1.5. IntelliJ IDEA

- **¿Qué es?:** IntelliJ [15] es un IDE desarrollado por JetBrains enfocado en el deep learning para ofrecer sugerencias al usuario conforme va desarrollando. Android Studio está basado en IntelliJ, por lo que ambas IDE comparten varias funcionalidades.
- **¿Por qué es relevante?:** Ofrece soporte para el desarrollo de aplicaciones para Android, React Native, Cordova y Ionic en múltiples lenguajes, en especial Java, Kotlin y Scala.
- **¿Se ha utilizado?:** Una de las principales ventajas de IntelliJ es la flexibilidad que tiene al soportar múltiples lenguajes o frameworks durante el desarrollo. Sin embargo, al final se decidió en utilizar Java para desarrollo, el cual Android Studio dispone de

más herramientas siempre actualizadas. No solo eso, para poder utilizar todas las herramientas que IntelliJ IDEA ofrece, es necesario comprarlo. Por esas razones, IntelliJ no se ha utilizado para el desarrollo.

2.1.6. Postman

- **¿Qué es?:** Postman es una plataforma de colaboración enfocada al desarrollo de APIs [16].
- **¿Por qué es relevante?:** Permite comunicarse con facilidad con APIs REST fuera de la aplicación y automatiza la fase de testing del desarrollo, por lo que agiliza establecer el POST que se enviará al DeviceHub.
- **¿Se ha utilizado?:** Dado que las funcionalidades que requerimos de Postman no son muy complejas y Postman es de gran fácil uso, se ha acabado utilizando sin necesidad de considerar alternativas.

2.1.7. Volley

- **¿Qué es?:** Es una librería HTTP que facilita y agiliza el uso de redes en apps para Android [17]. Está enfocado al envío de peticiones de pequeño tamaño de forma sencilla.
- **¿Por qué es relevante?:** Permite a la aplicación comunicarse con APIs de forma lo más personalizada posible. Es una de las librerías con esta funcionalidad para Android más extendidas en la actualidad y es desarrollada por los responsables de Android, por lo que es bastante fiable. También ofrece paralelización, gestión de prioridades de solicitud y herramientas de depuración, entre otras.
- **¿Se ha utilizado?:** Uno de los principales problemas que Volley tiene es que está pensado para la transmisión de información de tamaño pequeño, dado que almacena todas las respuestas en memoria [17]. Pese a que actualmente la aplicación no mueve información muy pesada, futuros desarrollos de la aplicación pueden cambiar esta realidad por completo, por lo que se debería considerar este caso para evitar tener que cambiar la arquitectura de la aplicación por completo en el futuro. Volley sugiere DownloadManager [18] como alternativa en estos casos, pero no se ha encontrado información suficiente sobre el uso de esta clase como para justificar implementar por completo la comunicación con la API de DeviceHub con DownloadManager.

2.1.8. Retrofit

- **¿Qué es?:** Retrofit es una librería que permite convertir una API HTTP en una interfaz de Java [19].
- **¿Por qué es relevante?:** Permite a la aplicación comunicarse con APIs de forma lo más personalizada posible. Al igual que Volley, Retrofit es una de las librerías enfocada a peticiones HTTP para Android más extendidas, y se enfoca en abstraer el código del cliente HTTP OkHttp.

- **¿Se ha utilizado?:** Uno de los principales inconvenientes de Retrofit es la ausencia de algunas herramientas especializadas como gestión de prioridades de llamadas o reintentos automáticos de llamadas. Sin embargo, es capaz de gestionar grandes ficheros y facilita herramientas para generar distintos bodies de la llamada. Esto es particularmente importante para la aplicación, dado que el POST que enviamos a DeviceHub contiene un JSON. Por esas razones Retrofit fue la opción que se decidió utilizar para la aplicación.

2.2. Evaluación de mercado

De todas las funcionalidades que nuestra aplicación ofrece, solo la extracción y visibilidad de la información compite con nuestra aplicación en el mercado, principalmente porque este proyecto está personalizado a las necesidades de eReuse. Por ese motivo, la gran mayoría de aplicaciones disponibles en el mercado con un símil a la nuestra se dedican a la extracción y visualización de los datos del dispositivo móvil. De las aplicaciones que se han visto, tres en particular reflejan de forma más completa esta funcionalidad:

- **Deviceinfo** [20]: Se trata de una aplicación web especializada en extraer información sobre todos los periféricos disponibles en el dispositivo y sobre la llamada HTTP realizada, enfocada a realizar pruebas de seguridad, privacidad y solución de problemas. Funciona tanto con dispositivos móviles como ordenadores como laptops, y es la aplicación que más información ofrece de las que se han podido consultar. Sin embargo, si bien es capaz de definir qué header de una petición HTTP el dispositivo soporta, no ofrece ninguna opción de enviar esta información a una API RESTful o de descargarla. Tampoco ofrece un campo de información que nos permita establecer un identificador único persistente como es el serial, tenga el dispositivo permiso de root o no.

- **Samsung Device Checker** [21]: También llamado *Device Checker *SAM* (Phone and tablet testing)*, Samsung Device Checker es una aplicación disponible en Google Play enfocada en comprobar que los distintos componentes del dispositivo móvil funcionan correctamente. Samsung Device Checker realiza distintos tests para lograr este fin, entre los cuales está el test “27. Device Information Test”, el cual ofrece información del hardware y software del dispositivo. Sin embargo, la información que esta aplicación ofrece cuando el dispositivo móvil no tiene permiso de root es muy limitada y no ofrece la opción de enviar esta información a través de una API, mucho menos la API de DeviceHub, la cual requiere de un token para comunicarse.

Figura 3: Test ”27. Device Information Test” de Samsung Device Checker (Fuente: [21]).

- **Device Info** [22]: Se trata de una aplicación enfocada a la muestra de información a tiempo real del dispositivo, sistema, cpu, batería, red, pantalla, memoria, cámara, temperatura, sensores y aplicaciones. Se encuentra disponible en la tienda de Google Play y es una de las aplicaciones con la mayor cantidad de información disponible al usuario en el mercado, y utiliza un diseño fácil para el usuario. También ofrece información persistente que se podría utilizar como un identificador del dispositivo. También ofrece la opción de exportar la información disponible en un documento para ser enviado posteriormente. Sin embargo, pese a tener todas estas funcionalidades, sigue sin ofrecer la opción de comunicarse con una API RESTful (el cual necesitamos para poder comunicarnos con DeviceHub), necesita permiso de root para poder acceder a la información más sensible y el código no es libre por lo que no podemos saber si se está realizando acciones maliciosas con nuestra información. Este último punto es particularmente relevante al ser una de las aplicaciones que más permisos requiere de las mencionadas.

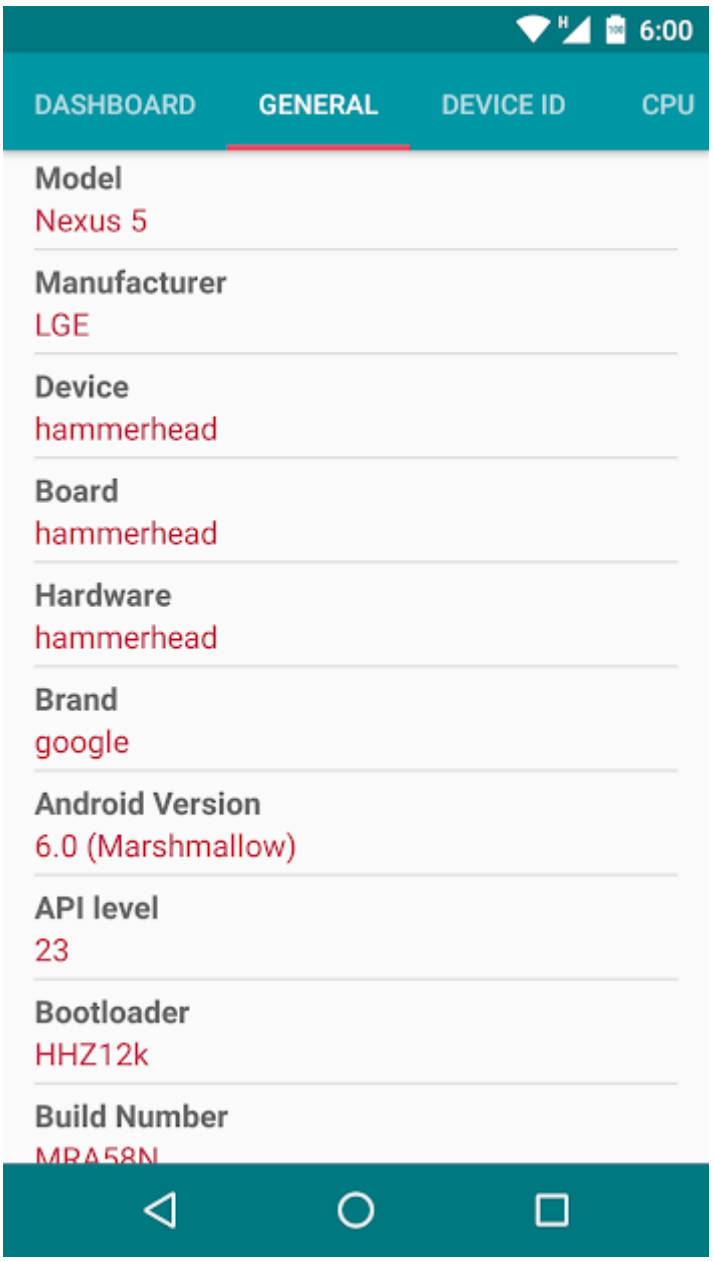

Figura 4: Aplicación Device Info en funcionamiento (Fuente: [22]).

3. Metodología y seguimiento

3.1. Método de trabajo

El desarrollo del proyecto se realizó utilizando los principios de la metodología Scrum.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente y obtener el mejor resultado posible de un proyecto. En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija, con cada ciclo del proyecto acercándose en incrementos al producto final [23]. Sprint es el nombre que va a recibir cada uno de los ciclos o iteraciones que vamos a tener dentro del proyecto [24].

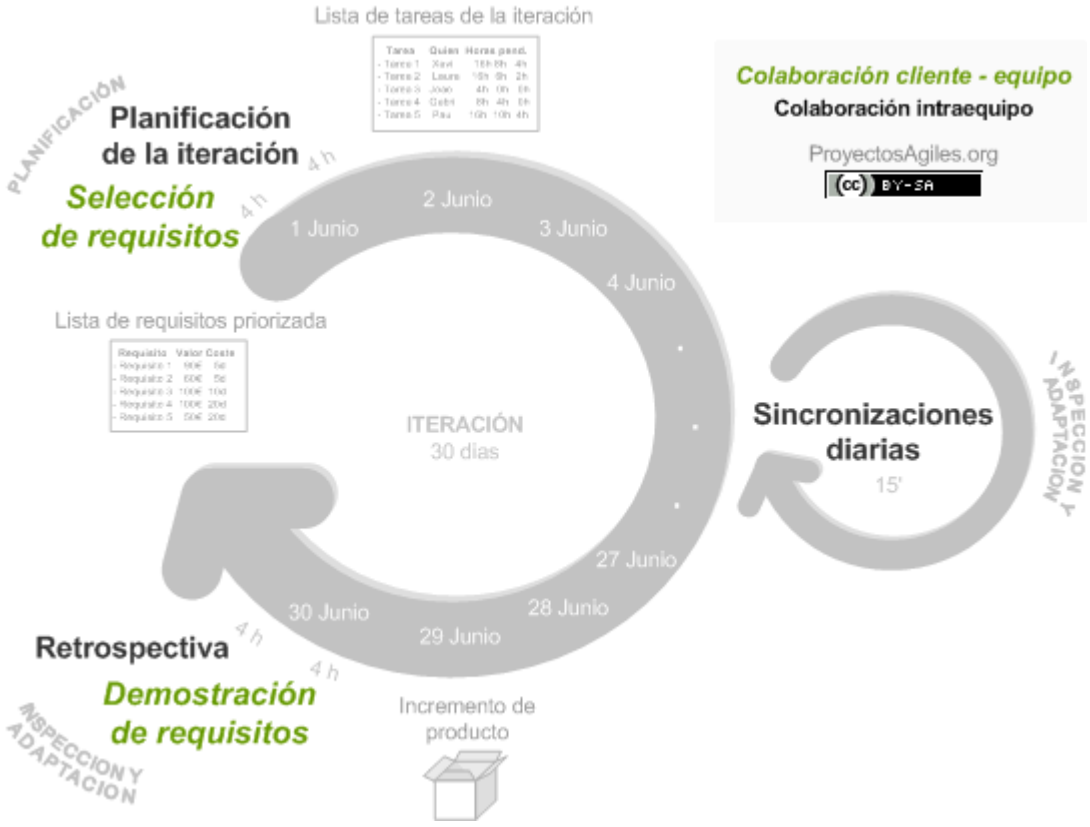

Figura 5: Representación de las iteraciones de un proyecto Scrum (Fuente: [23]).

Hemos dividido este proyecto en 4 sprints, representando cada funcionalidad principal de la aplicación y el proceso de aprendizaje para poder realizar el propio proyecto. Se determina un período de tiempo entre sprints de aproximadamente 1 de duración entre ellos, dando margen para cambios en caso de feedback o contratiempos.

Debido a que es sumamente importante mantener una buena comunicación con la entidad que solicita el proyecto (en este caso, eReuse), se realizaron reuniones periódicas a lo largo del

sprint, especialmente al final y principio de las mismas, para así asegurarse que el proyecto progresa como se espera.

Los Sprints presentados en esta memoria son aquellos que reflejan el proceso real que ha tenido el proyecto, por lo que ya se han considerado las actualizaciones en la planificación original tras cada iteración.

3.2. Herramientas de seguimiento

Para el buen desarrollo de la aplicación, se utilizó *Git* como control de versiones del proyecto, asegurando así una buena comunicación con el resto del equipo de eReuse y manteniendo un historial de las versiones realizadas. Se utilizó un solo repositorio alojado en *Github* dado que no se puso la aplicación en producción hasta la última iteración del desarrollo.

También utilizamos Trello como gestor de proyectos, organizando así los puntos a resolver del proyecto para cada sprint.

3.3. Planificación temporal

A continuación se mostrará la planificación final del proyecto a lo largo del tiempo disponible.

Ante todo, este proyecto se inició el **10/02/20**, una vez se inscribió el proyecto y la gestión por parte de secretaría concluyó, con una reunión entre el estudiante y el ponente. El proyecto se terminó el **22/04/2021**, considerando las fechas disponibles de lectura del TFG.

Se han asumido **490h totales dedicadas al proyecto**, de las cuales **145h se dedicaron a la gestión del proyecto** y el resto **(340h) al desarrollo.**

3.3.1 Definición de tareas

A continuación definiremos las distintas tareas que forman este proyecto y las distribuiremos en función de su realización en el espacio temporal disponible.

T1: Gestión del proyecto

T1.1: Reuniones

Todas las reuniones realizadas con el ponente a lo largo del TFG. Se realizó reuniones con el director los lunes y jueves por la tarde, así como reuniones con el equipo de eReuse mensuales por las mismas fechas para informar sobre el progreso del proyecto.

Tiempo: **20h**

Dependencias: -

T1.2: GEP

Incluye la redacción del alcance del proyecto, la planificación temporal, el presupuesto y el informe de sostenibilidad.

Tiempo: **29h**

Dependencias: **T1.1**

T1.3: Memoria

Tanto la redacción del documento principal del TFG a lo largo del desarrollo del proyecto, como las valoraciones de los usuarios que lo prueben, como la inclusión de T1.2 y el refinado previo a su entrega.

Tiempo: **60h**

Dependencias: **T1.2**

T1.4: Lectura del TFG

Tanto la preparación para la lectura (desarrollo de las diapositivas a utilizar, practicar la presentación) como la propia lectura en sí entran dentro de esta tarea.

Tiempo: **8h**

Dependencias: **T1.3**

T2: Trabajo previo

T2.1: Consulta de software disponible

Analizar los programas realizados previamente por eReuse y determinar qué herramientas se usarán finalmente para el proyecto y si se reutiliza código..

Tiempo: **15h**

Dependencias: **T1.1**

T2.2: Solicitud y comparación de material

Debido a que estamos desarrollando una aplicación que debe funcionar para múltiples dispositivos, y para realizar pruebas con mayor agilidad, se solicitó, recogió y comparó una serie de dispositivos con los que probar la aplicación. También incluye la comparación del estado del arte del software disponible y determinar qué herramientas se utilizaron al final.

Tiempo: **25h**

Dependencias: **T1.1**

T2.3: Aprendizaje del material

Engloba el aprendizaje de la tecnología software utilizada por primera vez.

Tiempo: **50h**

Dependencias: **T1.1, T2.2**

T3: Implementación inicial

T3.1: Lectura del modelo del dispositivo

Generar una primera versión de la aplicación que acceda a la información del dispositivo donde se encuentre instalado.

Tiempo: **41h**

Dependencias: **T2.3**

T3.2: Gestión de permisos de Android

Para acceder a información sensible a través de una aplicación en un dispositivo Android, es necesario solicitar estos permisos al usuario y considerar tanto si son concedidos como si no.

Tiempo: **15h**

Dependencias: **T2.3, T3.1**

T3.3: Compatibilidad con distintos modelos 1

Comprobación que la aplicación es operativa en los dispositivos recibidos en igualdad de condiciones, y depurar la aplicación en caso contrario. Esta tarea fue particularmente volátil, pudiendo ser resuelta en un par de horas o tener que dedicar múltiples días en la resolución de la misma ante cualquier modificación de la aplicación.

Tiempo: **15h**

Dependencias: **T2.2, T3.1, T3.2**

T3.4: Feedback 1

Realizar cambios necesarios en función del primer ciclo de feedback de eReuse.

Tiempo: **10h**

Dependencias: **T1.1, T3.3**

T4: Implementación principal

T4.1: Comunicación con DeviceHub de eReuse

Comunicación a través de una API RESTful con el DeviceHub para enviar en un JSON la información del dispositivo. Incluye también que la comunicación realizada no tuviese errores o que reciba estos mensajes de error y que el JSON enviado sea legible para la API.

Tiempo: **60h**

Dependencias: **T3.1, T3.2**

T4.2: Respuestas al usuario

En función de la información aportada por la aplicación, responder al usuario de forma adecuada. Esta tarea se centra en la gestión del feedback enviado por el DeviceHub.

Tiempo: **15h**

Dependencias: **T4.1**

T4.3: Compatibilidad con distintos modelos 2

Comprobación que la aplicación es operativa en los dispositivos recibidos en igualdad de condiciones, y depurar la aplicación en caso contrario. Esta tarea fue muy volátil, pudiendo ser resuelta en un par de horas o tener que dedicar múltiples días en la resolución de la misma tras una modificación en el código.

Tiempo: **35h**

Dependencias: **T4.1, T4.2**

T4.4: Feedback 2

Mostrar la aplicación resultante a eReuse y realizar cambios necesarios en función del segundo ciclo de feedback.

Tiempo: **25h**

Dependencias: **T1.1, T4.3**

T5: Implementación final

T5.1: Realizar cambios y comunicación para incluir la app en F-droid

Comunicarse con la plataforma F-droid y realizar cambios acordes en la aplicación para poder incluirla en su plataforma.

Tiempo: **25h**

Dependencias: **T4.1,T4.2**

T5.2: Compatibilidad con distintos modelos 3

Comprobación que la aplicación es operativa en los dispositivos recibidos en igualdad de condiciones, y depurar la aplicación en caso contrario. Esta tarea fue muy volátil, pudiendo ser resuelta en un par de horas o tener que dedicar múltiples días en la resolución de la misma tras una modificación en el código.

Tiempo: **25h**

Dependencias: **T5.1**

T5.3: Feedback final

Realizar cambios necesarios en función del tercer ciclo de feedback de eReuse.

Tiempo: **20h**

Dependencias: **T1.1, T5.2**

T5.4: Publicación del APK de la app

Una vez realizada la aplicación, publicar la APK bajo el nombre de eReuse para que el público tenga acceso a ella.

Tiempo: **5h**

Dependencias: **T5.3**

3.3.2 Gestión de riesgos

Estos son los posibles riesgos de nuestro proyecto que se definieron al inicio, cómo de grande sería el impacto en caso de producirse y, más importante, cómo podríamos mitigarlos:

Riesgo	Se produce una nueva actualización del SDK de Android que produce problemas de incompatibilidad con la applicación.
Probabilidad	Muy baja (trabajamos con la versión más reciente para evitarlo)
Impacto	Alta
Resolución	Restar tiempo de las funcionalidades finales para asegurarnos que el núcleo del proyecto (obtención de información y comunicación con el servidor DeviceHub) funciona correctamente.

Figura 6. Riesgo de la API.

Riesgo	La aplicación no funciona correctamente en ciertos dispositivos Android.
Probabilidad	Alta
Impacto	Baja
Resolución	Para dispositivos muy antiguos, añadir un disclaimer en la propia aplicación para que el usuario actualice el sistema operativo del dispositivo o directamente sugerir que lo recicle debido a antigüedad. Para dispositivos recientes, añadir en el código ajustes para ese caso concreto.

Figura 7. Riesgo de malfuncionamiento.

Riesgo	El software utilizado es más complejo de lo previsto
Probabilidad	Media
Impacto	Alta
Resolución	Al igual que con el riesgo de una nueva versión de la API, restar tiempo destinado a las funcionalidades finales para asegurarnos que el núcleo del proyecto funciona correctamente.

Figura 8. Riesgo de complejidad.

3.4. Obstáculos

La planificación del proyecto ha sido el principal aspecto del proyecto que más ha sido modificado, viéndose afectada por dos factores fuera del alcance del estudiante.

El primer factor fue la pandemia producida por COVID-19 y el impacto que tuvo a lo largo de todo 2020, el cual produjo varios confinamientos y circunstancias que dificultaron el seguimiento original del proyecto, especialmente durante el primer semestre de 2020.

El segundo factor fue el cambio en la dificultad de la extracción de campos con información sensible de los dispositivos móviles debido a la política de privacidad de Google [25] y, en particular, establecer un identificador persistente para el dispositivo, el cual tuvo un impacto relevante en el desarrollo durante el segundo semestre de 2020.

Teniendo en cuenta ambos factores, se decidió alargar el proyecto hasta el primer semestre de 2021, el cual supuso un cambio radical en la gestión final. También se decidió establecer un mayor énfasis en la aplicación como un producto fácil de mantener de cara a nuevas versiones del SDK de Android, dado que involucra notoriamente la eficacia de la aplicación.

3.5. Diagrama de Gantt del proyecto

Este es el diagrama de Gantt de la planificación inicial del proyecto. Si bien refleja adecuadamente la distribución de tareas y la cantidad de horas que cada tarea realiza, dada la naturaleza de la metodología ágil Scrum y los cambios de planificación causados el 2020 como consecuencia del COVID-19, algunas de las tareas se han cambiado y los últimos sprints se realizaron entre finales del 2020 y principios del 2021, dado que por los cambios de planificación mencionados anteriormente el desarrollo se tuvo que posponer.

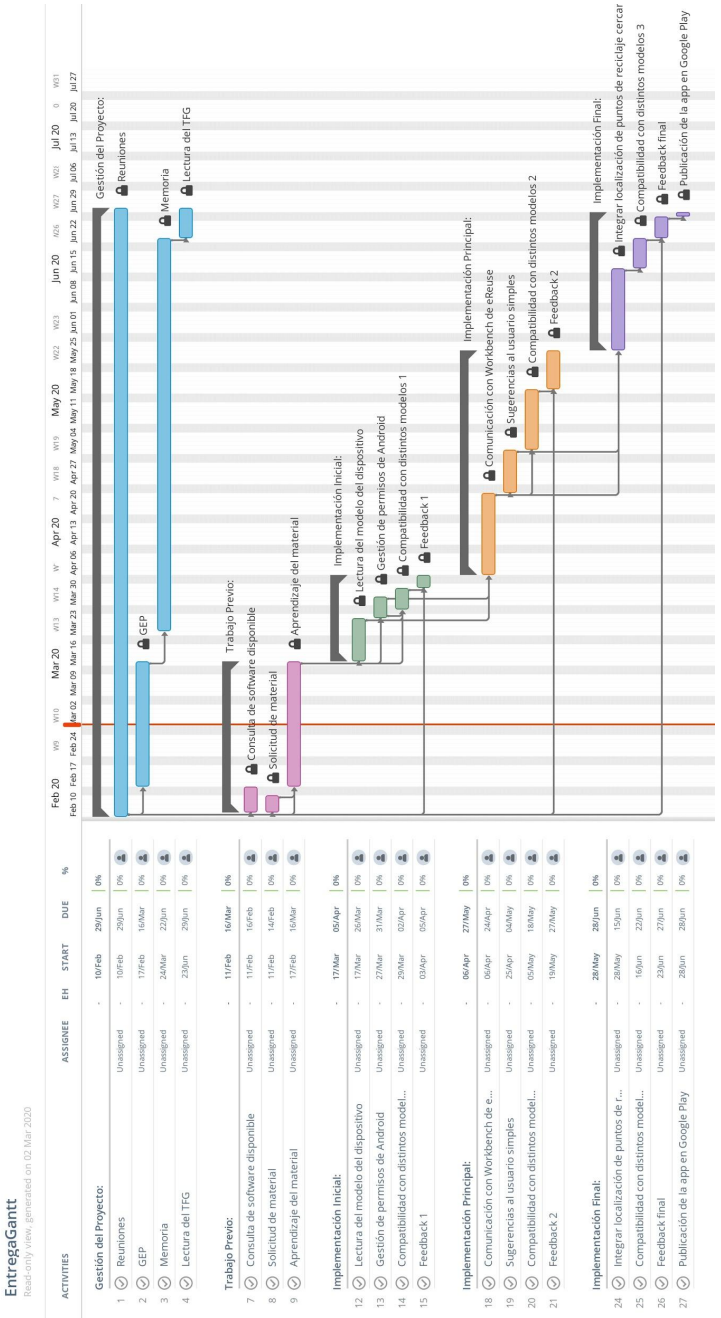

Figura 9: Diagrama de Gantt del proyecto.

4. Gestión Económica

4.1. Identificación de los costes

4.1.1. Costes por tarea

Para determinar el coste por tarea del proyecto, se han establecido unos roles orientativos para las personas implicadas en el proyecto y cuál es el salario promedio de los mismos en España. Las cifras utilizadas son extraídas de Payscale [26]:

Rol	**Sueldo/hora (€ brutos)**
Gestor de proyectos IT	25.60 €/h
Gestor de operaciones	23.70 €/h
Desarrollador de software 1	15.35 €/h
Desarrollador de software 2	15.35 €/h
Tester	11.37 €/h

Figura 10. Tabla de los sueldos por hora de los roles involucrados durante el desarrollo.

Una vez sabemos el salario de cada rol, tenemos que identificar cuántas horas dedica cada uno:

	Gest. IT	**Gest. Op.**	**Des. soft.1**	**Des. soft.2**	**Tester**
Horas (h)	30 h	129 h	264 h	60 h	15 h
Total (€)	768.00 €	3057.30 €	4052.40 €	921.00 €	170.55 €
	8969 €				

Figura 11. Tabla de distribución de las horas entre los roles involucrados.

4.1.2. Costes de hardware

Para el desarrollo de la aplicación se ha hecho uso tanto de varios dispositivos móviles de prueba, como de un portátil donde se desarrolló la propia aplicación.

Elemento	Precio	Vida útil	Amortización
Portátil Lenovo V130-15’	650 €	6 años	108 €
Xiaomi Pocophone F1	290 €	4 años	72 €
Samsung Galaxy s4 mini	79 €	2 años	39 €
Asus Zenfone 2	100 €	2 años	50 €
Total	1119 €		

Figura 12. Tabla de los costes de los dispositivos utilizados para el proyecto.

4.1.3. Costes de software

Al tratarse de una aplicación de código abierto y al utilizar software con licencia libre o perteneciente a eReuse, no tenemos coste alguno en cuanto a software se refiere.

4.1.4. Costes genéricos

Entre los costes de este proyecto, gran parte viene destinado al suministro de los dispositivos hardware mencionados previamente. No es necesario destinar gastos en transporte o desplazamiento debido a que se realiza gran parte del proyecto en una misma localización.

Elemento	Precio	Vida útil
Internet	128 €	4 meses
Electricidad	290 €	4 meses
Total	418 €	

Figura 13. Tabla de los costes genéricos del proyecto.

4.2. Control de gestión

Como se ha visto previamente, son tres los principales riesgos en el proyecto, y en todos ellos la solución a nivel de coste se basa en dedicar un mayor número de horas por parte del desarrollador de software, es decir, un incremento en los recursos humanos del proyecto.
Para reflejar todos estos imprevistos en los costes, hemos asumido que cada uno requerirá un número de horas equivalente a la tarea con la que está relacionada (T4.3 y T2.3):

Imprevisto	Horas	Coste	Probabilidad	Coste total
Actualización SDK	**35 h**	**537 €**	**1%**	**5€**
Incompatibilidad con dispositivos	**1 h**	**15 €**	**50%**	**7 €**
Software complejo	**25 h**	**383 €**	**25%**	**96 €**
Total				**109 €**

Figura 14. Tabla de imprevistos.

Por otro lado, los costes hardware y software apenas se verán variados a lo largo del proyecto, debido a que no se verían afectados en caso de requerir el proyecto un mayor o menor número de horas. En el caso de los costes genéricos, cubre la totalidad de los meses del proyecto, por lo que una diferencia de 25-35h no tendría impacto alguno en su coste.

4.3. Contingencias

Una vez determinados los costes, aplicamos un 15% al coste total para considerar cualquier imprevisto que pueda surgir a lo largo del proyecto debido a los riesgos previamente establecidos.

Coste por tareas	8969 €
Costes de hardware	1119 €
Costes de software	0 €
Costes genéricos	418 €
Coste Total	**10506 €**
Coste Final (+10%)	**11557 €**
Imprevistos	**109 €**
Coste Final con Imprevistos	**11666 €**

Figura 15. Tabla del coste final.

5. Implementación

Esta aplicación se ha basado en la versión del 5 de Julio de 2020 de Android 10, correspondiente al nivel 29 de la API [27]. Durante los últimos pasos del desarrollo del proyecto Google lanzó Android 11 con un nuevo nivel de la API que introdujo múltiples modificaciones [28]. Sin embargo, como estos cambios no afectan al funcionamiento de la aplicación, se ha decidido no actualizarlo para evitar posibles complicaciones futuras. Teniendo estas condiciones presentes, a continuación explicamos todo el proceso de desarrollo de la app, qué decisiones se tomaron y por qué se tomaron, así como el resultado final del desarrollo.

5.1. Software de terceros

A lo largo de la implementación ha sido necesario utilizar software de terceros. En esta sección explicamos qué se ha utilizado y porqué.

En primer lugar se tuvo que escoger el kit de desarrollo o SDK que a utilizar. Los principales factores fueron el alcance que tienen los sistemas operativos que el SDK soporta y la complejidad para acceder a la información del hardware. Se acabó escogiendo el SDK de Android, particularmente el de Android 10 (nivel de SDK 29) al ser el más estable cuando se terminó el desarrollo de la aplicación. La razón principal por la que se escogió este SDK es que nos garantiza obtener la información hardware para Android siempre que se cumplan las especificaciones establecidas en la documentación y, en caso que una actualización de Android rompa una de las funcionalidades de la aplicación, podremos cambiarlo desde el día 1 de la actualización, dado que el SDK está ligado al desarrollo de Android.

En segundo lugar fue necesario decidir qué IDE se iba a utilizar. Los principales factores durante esta decisión son la facilidad con la que se puede actualizar y mantener la aplicación con las herramientas que el IDE ofrece, el tiempo requerido de aprendizaje para que el autor de este TFG pueda utilizar las herramientas mencionadas y qué SDK soporta. Teniendo en cuenta estos factores, el entorno escogido fue Android Studio [12] por las siguientes razones:

- Es el IDE oficial de Android [13], que es un sistema operativo gratuito y que se puede instalar sin coste alguno en todo dispositivo móvil que lo requiera. También soporta el SDK oficial de Android, por lo que estamos garantizando que la aplicación sea accesible para Android al implementarlo con este IDE.
- Al ser el IDE oficial de Android, también nos garantiza que la optimización una vez compilado el código sea la adecuada, reduciendo así los errores que nos podamos encontrar [29].

- Soporta Java, eliminando la necesidad para el estudiante responsable de este proyecto de aprender un lenguaje de programación nuevo.

En tercer lugar están los paquetes que la aplicación utiliza para sus distintas funcionalidades. En este caso se han priorizado aquellos paquetes que son fiables basándonos en sí son ampliamente utilizadas por entidades o otros proyectos:

- **java.utils:** Contiene clases e interfaces variadas relacionadas con la colección de elementos, fecha y tiempo y otras utilidades misceláneas [30]. Es uno de los paquetes básicos de java y, en nuestro caso, se utiliza principalmente para la creación de arrays y comparación de atributos.
- **android.os:** Provee servicios básicos del sistema operativo, así como envío de mensajes y comunicación entre procesos del dispositivo móvil [31]. Lo utilizamos para extraer información del hardware del dispositivo.
- **android.widget:** Contiene elementos de la interfaz de usuario para ser utilizados en la pantalla de la aplicación [32]. Lo utilizamos para gestionar toda la información e interacción con los botones, textos e información que se muestran en la pantalla.
- **android.view.WindowManager:** Se encarga de comunicar la aplicación con el Window Manager [33]. Lo utilizamos para extraer las dimensiones de la pantalla del dispositivo móvil.
- **androidx.appcompat.app.AppCompatActivity:** Clase para actividades que quieren utilizar funciones de plataformas más recientes en dispositivos Android antiguos [34]. Lo utilizamos para mejorar la compatibilidad de la aplicación con dispositivos más antiguos.
- **com.google.gson.Gson:** Librería para convertir Objetos Java en JSON o un string JSON en un Java Object [35]. Lo utilizamos para convertir el Java Object con la información que hemos recolectado en el JSON que enviaremos a DeviceHub.
- **retrofit2:** Convierte una HTTP API en una interfaz de Java [19]. Lo utilizamos para enviar el JSON con la información del dispositivo mediante un POST a la API Restful de DeviceHub. Utilizamos Retrofit debido a su simplicidad al abstraer OkHttp, uno de los clientes HTTP más predominantes del mercado [36].

En cuarto lugar está la API con la que la aplicación se comunica. En este caso solo había una opción, siendo esta la API de DeviceHub [37], dado que la razón por la que se desarrolla esta aplicación es para poder enviar la información del dispositivo móvil donde la aplicación está instalada al DeviceHub, y realizar un POST a la API de DeviceHub es la forma más ágil para lograrlo.

Finalmente, utilizamos la plataforma Postman [16] para asegurarnos que el POST que queremos enviar con la aplicación a DeviceHub se leerá correctamente y saber qué respuesta recibirá la aplicación cuando realice la petición HTTP.

5.2. Funciones principales del Workbench Android

5.2.1. Recolección de información

La primera funcionalidad que la aplicación requería es la extracción de información del dispositivo donde se instala. Considerando que esta información tiene como objetivo delimitar la antigüedad y funcionalidad del dispositivo móvil, se ha enfocado en aquella información relacionada con el hardware.

Pese a querer extraer información hardware, no son muchos los permisos requeridos para extraer los datos que utilizamos actualmente. El principal motivo de esto es la ausencia de disponibilidad de permisos que nos den acceso a información sensible y persistente. Gran parte de estos permisos se encuentran protegidos bajo permiso de root [38], el cual Android no da acceso al usuario promedio y cuya única forma de obtenerlo es a través de un proceso complicado y que anula la garantía del dispositivo [39]. Finalmente, la política de Android está firmemente asentada en reducir en todo lo posible el acceso a información y potenciales identificadores persistentes por motivos de seguridad [40].

Esta situación planteó un problema para el desarrollo de la aplicación, dado que buscábamos tanto información hardware y, por tanto, persistente, como ser lo más accesibles posible para el usuario promedio, por lo que procedimientos para acceder a root no se podían considerar. Esto nos llevó a extraer la menor cantidad de información necesaria para poder crear un perfil del dispositivo en DeviceHub, y que todo dato imprescindible no accesible a través de android.os.Build [41] requiriese o encontrar una solución o sustituto a través de llamadas a la terminal de Android, o directamente ser descartado.

Teniendo en cuenta lo mencionado anteriormente, se decidió que los datos que la aplicación recolecta son:

- Información sobre el fabricante y características del dispositivo. Esta información ha de ser suficiente como para identificar exactamente el modelo.
- Información sobre el almacenamiento y memoria del dispositivo móvil.
- Un identificador único persistente relacionado con el dispositivo móvil. Este debe mantenerse incluso en el caso del dispositivo siendo restaurado de fábrica.

La mayoría de la información del fabricante y del dispositivo la obtenemos a través de android.os.Build [41]. Tanto el fabricante como el modelo del dispositivo son funciones en Build que no requieren de permisos adicionales y, además, hay una gran probabilidad que sigan estando disponibles en futuras versiones de la API de Android, dado que se implementaron en las primeras versiones de la misma (versión 1 de la API para el modelo y

versión 4 para el fabricante) y en ningún momento se ha tratado como información sensible a través de un permiso.

La única excepción en cuanto a información del dispositivo que se encuentra disponible en Build se refiere es el tamaño de la pantalla del dispositivo. En este caso utilizamos el display de WindowManager, que se trata de la interfaz que las aplicaciones utilizan para la gestión de la pantalla [42]. Dado que la función que nos devuelve el tamaño de la pantalla en píxeles, antes de mostrarlo lo convertimos en pulgadas.

El siguiente paso en la obtención de información es determinar la capacidad del dispositivo móvil. En este caso obtenemos tanto el tamaño del disco duro como de la RAM. Determinamos el tamaño del disco a través del tamaño del sistema de ficheros gracias a StatFs [43] y Environment [44], mientras que el tamaño de la RAM la obtenemos con ActivityManager, el cual se encarga de distribuir información relacionadas con servicios y actividades [45]. Dado que estos campos son sensibles, se encuentran sujetos a cambios de privacidad por parte de Google, por lo que son dos campos que, pese a ser obtenibles con relativa facilidad, es importante que se vayan revisando y manteniendo con mayor frecuencia comparado con los otros campos de información obtenidos en la aplicación.

Finalmente, obtenemos un identificador único para el dispositivo. Originalmente este iba a ser el número Serial del dispositivo, dado que todos los dispositivos tienen uno y su propósito es identificar. Sin embargo, desde la versión 26 de la API es requerido permiso root para poder acceder al Serial, dado que la nueva función que se utiliza para obtenerlo requiere el permiso android.Manifest.permission.READ_PRIVILEGED_PHONE_STATE, el cual solo se puede obtener si tienes privilegios de administrador [46]. Inicialmente, se bajó el nivel de la API al 26 para poder acceder a la función original para obtener el Serial [47]. Sin embargo, la ausencia de compatibilidad con versiones posteriores dada la sensibilidad de la información determinó que era una mejor alternativa utilizar un identificador completamente distinto.

El identificador que se acabó utilizando es la dirección MAC, el cual se puede acceder a través de la función ipconfig. Dado que se trata de un identificador único de una tarjeta de red, y dado que la aplicación requiere de una conexión a Internet para comunicarse con DeviceHub, la dirección MAC cumplía con el criterio establecido. Para poder acceder a la información de la tarjeta de red es necesario que el usuario de permiso para acceder al estado de red del móvil con android.Manifest.permission.ACCESS_WIFI_STATE [48].

5.2.2. Muestra de información en pantalla

Una vez recogida la información, se consideró necesario que el usuario fuese capaz de visualizarla, tanto para que sea consciente de la información de su dispositivo móvil y que sepa que se va a enviar a DeviceHub, así como para que reciba cualquier feedback relacionado con la interacción entre la aplicación y el usuario.

El primer paso en mostrar la información es establecer una actividad. Una actividad en android proporciona la ventana en la que la app dibuja la interfaz de usuario. Por lo general, esta ventana llena la pantalla, pero puede ser más pequeña y flotar sobre otras ventanas [49].

En nuestro caso buscamos la mayor simplicidad posible en el diseño de la aplicación dado que nos interesa que sea lo más accesible posible para toda clase de público. Dado que se trata también de un prototipo, no nos enfocamos en la estética de la interfaz en un intento de no sacrificar la estructura.

Teniendo en cuenta ambos puntos, la interfaz consiste en una sola actividad que contiene la información previamente adquirida por la aplicación sobre el dispositivo. Esta información se estructuró como una tabla de dos columnas donde la primera columna contiene el nombre del campo y la segunda la susodicha información del dispositivo. Finalmente, la actividad dispone de un botón que ocupa la zona inferior de la aplicación y con unas dimensiones horizontales iguales a las de la pantalla. De esta forma el botón es mucho más cómodo de pulsar al encontrarse cerca del área donde el pulgar suele encontrarse al sujetarse con una mano, y reduciendo el tiempo necesario para determinar dónde presionarlo en el eje horizontal.

Una vez se determinó la actividad y sus elementos, se implementó la actividad en la aplicación. Por defecto Android Studio utiliza el formato xml para definir las actividades y sus distintos elementos. En nuestro caso, definimos la actividad principal como activity_main.xml en app/src/main/res/layout con la estructura de la actividad, y la relacionamos con los distintos atributos en app/src/main/res/values. Una especial mención la tiene strings.xml, dado que contiene el texto que se muestra en actividad y es este archivo en el que se actualiza la información del dispositivo desde la función principal en java, una vez obtenidos estos datos.

Figura 16. Actividad de la aplicación visualizada en pantalla.

El botón en la zona inferior de la pantalla se utiliza para indicar que el usuario está listo para enviar el POST a DeviceHub. Este botón ha pasado por varias iteraciones, originalmente llamando desde el botón definido en activity_main.xml a una función llamada sendSnapshot. Al final esta decisión se sustituyó por una función en MainActivity.java que escucha toda la actividad en caso que un botón se pulse. Una vez un botón se pulse, sin importar cual sea, la función se ejecuta y actúa de una forma u otra dependiendo del identificador del botón. De esta forma, en caso que en el futuro se decida implementar más funcionalidades que requieran añadir complejidad a la actividad, la implementación y gestión de más botones será mucho más simplificada.

5.2.3. Construcción del JSON

Una vez disponibles los datos y la petición del usuario para enviar la información del dispositivo a DeviceHub, fue necesario estructurar la información de tal forma que la API de DeviceHub fuese capaz de leerla. La API de DeviceHub es RESTful, que se refiere a una interfaz de programación de aplicaciones que se ajusta a los límites de la arquitectura REST. REST no es un protocolo ni un estándar, sino que se trata de un conjunto de principios de arquitectura [50]. Esencialmente, debe de ser capaz de enviar la información por HTTP en un formato preestablecido. DeviceHub utiliza la estructura JSON para la transmisión de datos, por lo que la aplicación debe de ser capaz tanto de crear JSONs, rellenarlos, enviarlos con un POST a la API de DeviceHub, que el JSON esté bien estructurado para que la API lo pueda leer correctamente, recibir la respuesta de la API y leer el JSON que la API envíe.

Para poder crear el JSON primero es necesario saber qué estructura debe tener. En el caso de este proyecto primero se estableció el JSON teniendo en cuenta la información que la aplicación es capaz de leer y los parámetros que DeviceHub requiere para poder procesar la petición:

```
{
    "type":"Snapshot",
    "uuid":"74827f74-ceb0-4c84-bc8a-8be85c58ea03",
    "software":"WorkbenchAndroid",
    "version":"0.0.1",
    "device":{
        "type":"Mobile",
        "model":"GT9000-test",
        "manufacturer":"Samsung",
        "serialNumber":"00:00:00:00:00:00",
        "ramSize":2048,
        "dataStorageSize":16384,
        "displaySize":5
    }
}
```

Figura 17. Estructura del JSON que la aplicación envía a la API RESTful de DeviceHub.

Los parámetros del POST son:

- **Type:** El tipo de petición para que DeviceHub lo pueda procesar de forma adecuada. En el caso de la aplicación siempre se trata de un Snapshot o captura del estado de un dispositivo, por lo que es un parámetro estático.

- **UUID:** Identificador único de la Snapshot para evitar duplicados de una misma petición. Este identificador lo genera la aplicación automáticamente cuando se envía la petición.
- **Software:** Cuál es el software que está enviando este JSON. Actualmente el nombre temporal de la aplicación móvil en DeviceHub es Workbench Android, por lo que ese es el valor de este parámetro. Está sujeto a cambios.
- **Version:** Versión de la aplicación. Al final del desarrollo de este proyecto la aplicación se encontraba en la versión 0.0.2, y este valor se irá incrementando conforme la aplicación se vaya desarrollando en trabajos futuros.
- **Device:** Contiene todos los datos que la aplicación obtiene sobre el dispositivo móvil. Dos parámetros de especial interés son "type", que define qué clase de dispositivo es, y "serialNumber", que realmente contiene el identificador único del dispositivo, en este caso la dirección MAC.

Una vez definido el JSON se comprobó que su estructura es correcta enviando un post a DeviceHub con el JSON mediante Postman. Una vez comprobado que el JSON era correcto se implementó la creación del JSON en la aplicación.

Una de las librerías más populares para la creación de strings JSON en Java es la librería Gson, que convierte Objetos Java en su equivalente en JSON [35]. En el caso de la aplicación, esta funcionalidad nos permite generar una clase con los valores del vector device en el JSON como variables de la clase. De esta forma, una vez llamado el constructor de la clase encargado en asignar la información obtenida a las variables, utilizamos el objeto User en una llamada a Gson y obtenemos el JSON que se enviará a DeviceHub.

Utilizamos la librería Gson por dos razones principales: La primera, nos permite modificar sin dificultad la estructura del JSON, dado que está atada a la estructura de la clase User.java, disponiendo de esta forma un método para expandir las funcionalidades de la aplicación sin dificultad. La segunda, se trata de una librería desarrollada y mantenida por Google, por lo que nos cercioramos que será actualizada con relativa frecuencia, otorgándole una vida útil larga.

5.2.4. Comunicación con la API de DeviceHub

Una vez generado el JSON que enviamos a DeviceHub, fue necesario generar el POST que se enviará con cada llamada HTTP. El primer paso es determinar el contenido de la cabecera de la llamada HTTP que realizaremos. Afortunadamente, la cabecera que requiere DeviceHub es muy simple, necesitando únicamente el campo Authorization con un token que nos autorice el acceso a DeviceHub y que no se rechace la conexión, y el campo Content-Type determinando qué contiene el Body de la llamada.

Para conseguir el token necesario para completar el campo Authorization hizo falta realizar una llamada por separado a DeviceHub. Esta llamada no fue necesaria ser realizada en la propia aplicación, dado que solo requerimos de un token. Esta llamada genera el token dependiendo de si contiene acceso a un usuario del DeviceHub. En nuestro caso disponemos de este usuario, permitiéndonos obtener así el token.

Como se ha establecido previamente, la llamada para obtener el token no está ligada a la aplicación, por lo que se utilizó la siguiente función para obtenerla (con los datos del usuario censurados por motivos de seguridad):

```
curl --location --request POST
'https://api.testing.usody.com/users/login/' \
--header 'Content-Type: application/json' \
--data-raw '{
    "email": "xxxx@dhub.com",
    "password": "xxxx"
}'
```

Figura 18. POST para obtener el token para DeviceHub.

Una vez tenemos el token, pudimos establecer el POST que la aplicación realizará:

```
curl --location --request POST
'https://api.usody.com/usody/actions/' \
--header 'Authorization: Basic
Yzc5YzI5MTItOWJhNi00NjU4LTg1MTAtNWNlOWYyZTBmNzk2Og==' \
--header 'Content-Type: application/json' \
--data-raw '{
    {
      "type":"Snapshot",
      "uuid":"74827f74-ceb0-4c84-bc8a-8be85c58ea03",
      "software":"WorkbenchAndroid",
      "version":"0.0.1",
      "device":{
          "type":"Mobile",
          "model":"GT9000-test",
          "manufacturer":"Samsung",
          "serialNumber":"00:00:00:00:00:00",
          "ramSize":2048,
          "dataStorageSize":16384,
          "displaySize":5
      }
  }
}'
```

Figura 19. POST para realizar la Snapshot del dispositivo en DeviceHub.

Utilizamos Retrofit2 para crear la llamada HTTP en la aplicación. Retrofit es una librería que permite convertir una API HTTP en una interfaz de Java [19], necesaria dado que nuestra aplicación se ha desarrollado con Java. La principales razones por las que utilizamos Retrofit son su simplicidad al abstraer OkHttp y, al ser uno de los clientes HTTP más predominantes del mercado [36], hay disponible documentación muy extensa sobre el funcionamiento de Retrofit, facilitando el aprendizaje del estudiante y asegurándonos que posible desarrollos posteriores sean lo más ágiles posibles [51] [52].

A la hora de implementar la llamada, primero se creó una instancia de Retrofit, la cual actúa de cliente REST y realiza las llamadas a la url establecida (en nuestro caso, DeviceHub). Una vez establecido el cliente REST, se creó una instancia del Endpoint de la API (AppClient.java) donde se envía el JSON y se realiza contacto con la API de DeviceHub. Una vez realizada la llamada, recogemos la respuesta en función de si la conexión ha sido exitosa y respondemos al usuario.

5.2.5. Generación y distribución de la aplicación final

Una vez finalizada la implementación de todas las características de la aplicación, el último paso a realizar fue generar el binario de la aplicación, establecer una licencia de uso y, finalmente, decidir en qué plataformas distribuirla.

Dado que se utilizó Android Studio como IDE durante el desarrollo del proyecto, una de las herramientas a nuestra disposición es la de generar automáticamente un APK del proyecto. Es una opción encontrada en la pestaña **Build > Build Bundle/APKs > Build APK** de Android Studio, y nos ha permitido establecer varias versiones de la aplicación en producción.

La licencia de la aplicación es la **GNU Affero General Public License v3.0** [53], que condiciona a que el código completo de la aplicación y sus derivados estén disponibles al público. Todos los derivados de esta aplicación deben encontrarse bajo la misma licencia, pero no impide el uso comercial, la modificación, uso de patentes o el uso privado de la misma.

Finalmente, son dos las plataformas en las que se distribuye la aplicación:

- **GitHub** [54]: En GitHub se encuentra tanto el código de la aplicación como los APKs de las últimas versiones de la aplicación. Es el primer lugar donde se reflejan las actualizaciones de la aplicación, dado que se encuentra en directo control del equipo de eReuse y es donde se realiza el control de errores de la aplicación.

- **F-droid** [55]: En esta plataforma web de aplicaciones móvil de código abierto se distribuye la última release de la aplicación una vez actualizado el GitHub. Para poder subir la aplicación es necesario confirmar que no se utiliza software propietario o que no se tienen fines lucrativos de cualquier tipo. Una vez establecidos estos puntos se debe realizar una issue en el repositorio de la plataforma en GitLab y esperar a que lo acepten, realicen un merge del pull request y actualicen la propia plataforma. Implementar los metadatos de la aplicación en el formato fastlane agiliza el proceso de actualización de la aplicación en f-droid. Actualmente se está esperando a que se apruebe el pull request para la segunda release del APK de la aplicación.

5.3. Testing

A lo largo del desarrollo de la aplicación se han utilizado varias tecnologías, técnicas y estrategias para asegurar un buen funcionamiento de las funcionalidades implementadas:

- **Postman:** Plataforma de colaboración enfocada al desarrollo de APIs [16]. Se ha utilizado múltiples veces para comprobar que tanto el POST establecido por el equipo de eReuse como el POST que la aplicación genera reciben la misma respuesta por parte de DeviceHub, aislando errores relacionados con la implementación de Retrofit.

- **Debug:** A lo largo de la implementación utilizamos el depurador de Android Studio para establecer interrupciones en el código Java y examinar variables y evaluar expresiones en el tiempo de ejecución [56]. Gran parte de las compilaciones realizadas en el proyecto fueron utilizando la pestaña Debugger, y una de las herramientas para determinar errores más utilizada a lo largo del proyecto fueron las interrupciones que Android Studio dispone en la ventana Breakpoints.

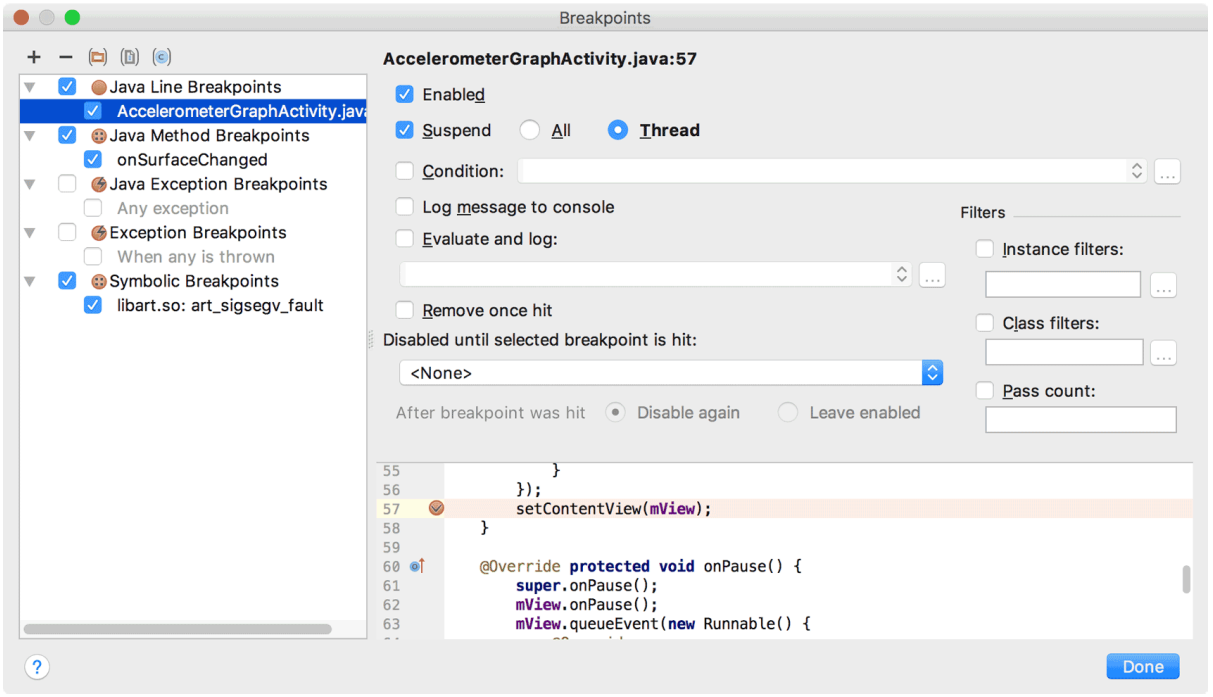

Figura 20. Ventana de Breakpoints de Android Studio (Fuente: [56]).

- **Código:** Durante la implementación de la aplicación se introdujo código para comprobar la estructura del JSON antes de ser enviado al DeviceHub en la aplicación y para interceptar los logs de la comunicación entre APIs en caso que el error no se encuentre en la estructura del POST.

- **Git:** Utilizamos Git como plataforma de control de versiones del código. De esta forma podemos gestionar el desarrollo de la aplicación de forma acorde a los Sprints de la planificación y asegurarnos que el código que se introduce en la rama en distribución se ha validado.

- **F-droid:** Una de las plataformas en las que se quiere introducir la aplicación es F-droid, una página con un repositorio de aplicaciones libres y de código abierto [55]. Para poder añadir la aplicación de este proyecto es necesario realizar un pull request al repositorio de F-droid y cumplir los requisitos establecidos en su página, los cuales son comprobados nada más realizar la petición [57]. Estos requisitos están establecidos para asegurarse que las aplicaciones que se suban son de código abierto y que no utilizan software de terceros con ánimo de lucro o cerradas.

5.3.1. Feedback de los usuarios

A lo largo del desarrollo se ha puesto a prueba la aplicación ante distintos perfiles para recibir feedback sobre el estado de la aplicación y qué cambios se deberían realizar para mejorar la experiencia y eficiencia de la misma. Son tres los perfiles a los que se ha consultado sobre la aplicación a lo largo del proyecto:

- **Familiares y amistades**: En este perfil se encuentran aquellos usuarios con un conocimiento escueto o nulo sobre diseño de aplicaciones móvil y dispositivos móviles, limitado a un uso doméstico de un dispositivo móvil y sin mucho interés en el reciclado de dispositivos móviles. El principal feedback que dio este perfil fue sobre cómo se sentía utilizar la aplicación y cómo pensaban que iba a ser tras dar una explicación breve de lo que hace. Del feedback recibido se decidió poner un **fondo oscuro** para no afectar la vista de aquellos con ojos cansados y se **agrandó la fuente del texto** de la aplicación.

- **Usuarios de eReuse**: En este perfil se encuentran aquellos usuarios que utilizan los servicios de eReuse y buscan activamente darle una segunda vida a sus dispositivos. El conocimiento sobre dispositivos móviles de este perfil es promedio, el suficiente como para saber cómo funciona un dispositivo móvil. El principal feedback recibido a lo largo del proyecto por este perfil fueron los campos de información que quizás se debería añadir a la aplicación y por qué, así como errores encontrados al utilizar dispositivos con versiones de Android particularmente antiguas.

- **Equipo de eReuse**: En este perfil se encuentra el equipo de eReuse, el cual ha trabajado con el estudiante para el desarrollo de la aplicación y tiene un alto conocimiento tanto de uso del dispositivo como de diseño. Este perfil dio feedback sobre posibles identificadores persistentes para la aplicación, el tamaño y posición de los botones de la aplicación y cómo responder al usuario tras una interacción

6. Sostenibilidad y Compromiso Social

6.1. Dimensión Económica

El impacto económico de este proyecto en la sociedad del bienestar es muy notorio. Dado que el resultado de este proyecto facilita la gestión de dispositivos móviles a través de eReuse para determinar si se deben reciclar o reutilizar, estamos reintroduciendo dispositivos móviles en el mercado con un precio reducido y, por tanto, no solo estamos alargando la esperanza de vida de los dispositivos móviles, también estamos reduciendo el coste de entrada para que un usuario tenga acceso a un dispositivo.

· **¿Coste estimado para la realización del proyecto?**

Como se ha mencionado en el apartado 4.3., se estima un coste final de **11666 €.** Sin embargo, hay que tener presente que parte de ese coste en realidad no se verá reflejado en el proyecto, debido a que los realiza el propio estudiante sin coste alguno.

· **¿Cómo se resuelven actualmente los aspectos de costes del problema que quieres abordar (estado del arte)?**

Generalmente, la población guarda sus dispositivos móviles en un cajón y los deja en el olvido, o los recicla de forma incorrecta. Por tanto, el coste del problema actual es altísimo (no se reintroducen los dispositivos obsoletos y sus componentes al mercado) y este proyecto forma parte de uno mayor con el objetivo de mejorar ese estado del arte.

· **¿En qué mejoraría económicamente (costes…) tu solución respecto a los existentes?**

Al impulsar el estado del arte de la accesibilidad de reutilización y reciclado de dispositivos (en este caso, móviles) para la población general, estamos reintroduciendo materiales y dispositivos en el mercado que, de otra forma, se perderían, reduciendo así tanto costes en materia prima, como coste del producto final si se adquiere dispositivos de segunda mano.

6.2. Dimensión Ambiental

El impacto ambiental de este proyecto es muy positivo para el mercado de los dispositivos móviles. Al facilitar la gestión de dispositivos móviles, estamos aumentando cuántos dispositivos móviles se reutilizan o reciclan, reduciendo la necesidad de introducir nuevos dispositivos en el mercado y, finalmente, reduciendo la cantidad de dispositivos creados que requieren materia prima. Esencialmente, el éxito de este proyecto y el éxito de eReuse tendrá un impacto directo en la reducción de la huella ambiental del sector electrónico.

· **¿Has estimado el impacto ambiental que tendrá la realización de tu proyecto?**
El impacto ambiental de este proyecto es uno de los pilares del mismo, dado que facilitar la reutilización y reciclado de dispositivos móviles implica una reducción de la materia prima requerida y, por tanto, menores emisiones de CO2, menor residuo electrónico, etc.

· **¿Te has planteado minimizar el impacto?**
Minimizar el impacto de los recursos es uno de los puntos clave de este proyecto, dado que a la hora de diseñar la aplicación nos enfocamos en que sea lo más intuitiva y clara posible para que el ciudadano promedio pueda entender cómo funciona y le dé un uso adecuado, maximizando así el número de dispositivos que podemos reutilizar/reciclar.

· **¿Cómo se resuelve actualmente el problema que quieres abordar? ¿En qué mejoraría ambientalmente tu solución respecto al resto?**
Actualmente hay una gran ausencia de proyectos ajenos a eReuse que hagan de puente entre la población y las plantas de reciclaje o entidades que adquieran productos de segunda mano, y usualmente los dispositivos móviles que dejan de estar en uso no se les da una segunda vida o no se reciclan adecuadamente. Este proyecto pretende evitar ese escenario, suponiendo una mejora ambiental notable.

6.3. Dimensión Social

Este proyecto facilita la reintroducción de dispositivos que ya no se utilizan pero que todavía se les puede dar un uso. Como se ha mencionado en el apartado 6.1. reintroducir dispositivos de segunda mano reduce el coste de entrada de los dispositivos móviles, aumentando la cantidad de usuarios que pueden costear un dispositivo. Esto tiene un impacto social enorme, dado que otorga mayor igualdad de condiciones en la sociedad al lograr que un mayor porcentaje de la población pueda utilizar un smartphone y a las múltiples aplicaciones y sensores que esto conlleva.

· **¿Qué crees que aportaría a nivel personal la realización de este proyecto?**
Más allá de la satisfacción moral que supone colaborar en un proyecto con un objetivo ético tan claro, este proyecto me aporta la posibilidad y el tiempo para aprender sobre un problema cada día más vigente en nuestra sociedad, aprender nuevas tecnologías y, finalmente, realizar una herramienta que yo mismo he encontrado en falta más de una ocasión.

· **¿Cómo se resuelve actualmente el problema que quieres abordar? ¿En qué mejoraría socialmente (calidad de vida) tu solución?**
Más allá de la reducción de CO2 y reducción de costes, este proyecto proporciona tanto a entidades como al ciudadano promedio una herramienta que asegura un proceso de reciclado con la garantía de mantener la privacidad (se trata de una de la políticas de eReuse el garantizar la protección de datos de los dispositivos donados).

· **¿Existe una necesidad real para este proyecto?**
La hay, y con el tiempo con mayor urgencia, dado que este proyecto intenta facilitar la integración de la economía circular en la población.

7. Competencias técnicas

7.1. CTI1.3

Seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización con los criterios de costo y calidad identificados.

Se han calculado todos los gastos necesarios para la realización de este proyecto. Cabe decir, que al tratarse de un proyecto de código abierto sin ánimo de lucro, todas las tecnologías presentes utilizan licencias muy poco o nada restrictivas, reduciendo notablemente el coste de este proyecto en comparación con uno con fines lucrativos y, por lo tanto, reduciendo los gastos en todo lo posible.

Las funcionalidades necesarias para que la aplicación cumpliese con el objetivo original fueron establecidas en la primera fase del proyecto. Por lo tanto, la calidad del proyecto puede medirse en tanto que los objetivos originales hayan sido alcanzados. El producto final cumple con las especificaciones establecidas por eReuse en relación a la transmisión de información, y actualmente la aplicación tiene una versión en producción disponible para todo el público.

Respecto a la calidad del proyecto, se estableció en una fase temprana que el objetivo del mismo es la creación de una aplicación con unas funcionalidades específicas. En este caso se han gestionado la recolección y la transmisión de información mediante una API facilitada por eReuse a través de peticiones HTTP.

7.2. CTI1.4

Seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de costo y calidad adecuados.

A lo largo del proyecto se ha evaluado qué datos del dispositivo móvil son indispensables para el proceso de recolección de información de la app, definidos en el apartado 5.2.1. Además, durante la implementación de la aplicación se determinó qué tecnologías se utilizan. Estas tecnologías han sido en su totalidad de código abierto o con licencias libres, evitando así costes. El único coste material considerable ha sido la adquisición de dispositivos móviles para realizar los tests de la aplicación, y aún así ese coste ha sido mitigado al tratarse de dispositivos de segunda mano.

7.3. CTI3.1

Concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación ubicua.

A lo largo del proyecto se ha concebido una aplicación que interactúa con el usuario mediante una interfaz multimedia que le permite interactuar con los servidores backend mediante peticiones HTTP y una API RESTful. Se ha establecido un énfasis en la simplificación de la aplicación y escalabilidad de estos servicios de cara a futuros desarrollos de la aplicación y las funcionalidades que estos futuros desarrollos puedan conllevar. También se ha implementado en la aplicación la visualización de mensajes dirigidas al usuario correspondientes a cómo la API del DeviceHub ha gestionado las peticiones HTTP realizadas.

8. Conclusiones

El desarrollo y distribución de esta aplicación nos ha permitido llegar a unas conclusiones en relación a la planificación del proyecto, a unas conclusiones a nivel personal del estudiante y qué trabajo se podría realizar por otros estudiantes. Estas conclusiones son sobre el desarrollo de una aplicación que recoge información para identificar un dispositivo y sus características para un posterior reciclado.

8.1. Objetivos y características cumplidas

En la introducción del proyecto se estableció que el objetivo era el desarrollo de una aplicación que obtuviese información esencial del dispositivo móvil, mostrar esta información en pantalla al usuario y dar la opción de informar sobre las circunstancias del dispositivo a eReuse.

Una vez acabado el proyecto, podemos afirmar que los objetivos establecidos se han cumplido, dado que la aplicación recoge una serie de datos que eReuse considera como vitales (apartado 5.2.1), los muestra en pantalla (apartado 5.2.2.) y da la opción en enviarlo a DeviceHub a través de una petición HTTP (apartado 5.2.3. y 5.2.4.).

La aplicación se puede descargar y consultar en las siguientes plataformas:

```
//Repositorio de GitHub
https://github.com/eReuse/eReuse-NativeAndroid-app
//Lista de APKs publicadas de la app en GitHub
https://github.com/eReuse/eReuse-NativeAndroid-app/releases
//Issue para añadir la aplicación en el repositorio de F-droid
https://gitlab.com/fdroid/rfp/-/issues/1710
```

Por otro lado, las características que se consideraron vitales (apartado 1.2.) son:

1. Recogida de información sobre características del dispositivo móvil en el que la aplicación se haya instalado.
2. La información obtenida debe ser relevante para la reutilización o reciclado del dispositivo móvil donde la aplicación se haya instalado.
3. Envío de datos con el DeviceHub de eReuse.
4. Acceso a la aplicación desde un amplio perfil de usuarios, incluyendo usuarios no expertos.
5. Disponibilidad y transparencia del código y su funcionalidad con el público. Ha de ser un proyecto de código abierto

Estas características se ven reflejadas en el resultado final de la siguiente forma:

1. El dispositivo recoge la información del fabricante, modelo, dirección mac, tamaño del display, tamaño de la ram y tamaño de almacenamiento del dispositivo.
2. Recogemos esta información para enviársela a DeviceHub, que se encarga de gestionar los ciclos del dispositivo para determinar si el dispositivo debe ser reutilizado o reciclado.
3. Utilizamos una API HTTP creada con Retrofit y realizamos un POST a la API RESTful de DeviceHub con un JSON conteniendo los datos del dispositivo móvil.
4. La aplicación está disponible en GitHub [54] y está en proceso de ser aceptado en F-droid [55], ambas plataformas son libres y accesibles para todo el mundo con acceso a internet. Si bien no todo el mundo conoce ambas plataformas, la aplicación también se podría subir a Google Play si eReuse tuviese una cuenta de desarrollador.
5. El código completo se encuentra disponible en GitHub [54] y utiliza la licencia GNU Affero General Public License v3.0 que establece que tanto el código de la aplicación como el de sus derivados ha de estar disponible al público [53].

Teniendo en cuenta que todas las características establecidas se han cumplido en su totalidad, podemos afirmar que el proyecto ha cumplido con su propósito.

8.2. Conclusiones personales

Como conclusiones personales, he notado a lo largo del proyecto que existe un conflicto entre proteger datos sensibles del dispositivo móvil y facilitar el reciclado del propio dispositivo. Conforme progresan los años y las versiones de la API de Android avanzan, Google ha ido reemplazando funciones que acceden a información del dispositivo por versiones que requieren de permisos usualmente fuera del alcance del usuario promedio, en un intento de evitar que aplicaciones maliciosas den un mal uso de estos datos. Sin embargo, esta misma información es necesaria para determinar la antigüedad del dispositivo y, finalmente, determinar si es justificable darle una segunda vida y, a la larga, reducir el gasto de recursos naturales.

Esta oposición por parte de Android a la hora de obtener datos persistentes se ha percibido en este proyecto particularmente cuando se buscó un identificador único en el dispositivo y persistente pese a restauraciones de fábrica. Este identificador originalmente iba a ser el Serial del dispositivo, pero desde el nivel 26 de la API es necesario permisos de root para poder acceder a esta información. Desafortunadamente, para conseguir permisos de root es necesario realizar un proceso que el usuario promedio no sabrá o querrá ejecutar, por lo que ha sido un verdadero dilema determinar qué datos útiles seguirían siendo inaccesibles en un año o dos.

Por otro lado, he observado que, a diferencia de la obtención de datos, es relativamente sencillo mantener la compatibilidad entre dispositivos dado que utilizamos el SDK de Android. El único aspecto a tener en cuenta es el rango de las versiones de Android que los dispositivos capaces de ser reutilizados podrán llegar a tener, y considerar los distintos casos a la hora de obtener los datos.

Finalmente, se llega a la conclusión que, para poder aplicar la economía circular al mundo de los dispositivos móviles, es necesaria una comunicación estrecha entre los desarrolladores del SDK del sistema operativo objetivo (principalmente Android al ser el más extendido) y los responsables de gestionar el inventario de los dispositivos móviles a reciclar y mantener la aplicación responsable de obtener y realizar la ficha de los datos de los dispositivos. En caso contrario, será necesario encontrar alternativas para conseguir permisos de root, ya sea por ejemplo facilitando el proceso de root de un dispositivo móvil o sus consecuencias para que el usuario promedio lo realice.

8.3. Trabajo a futuro

Como trabajo a futuro será necesario ir actualizando el SDK que la aplicación utiliza a la versión más reciente y comprobar que no aumenten las restricciones para la obtención de datos. Actualmente, la aplicación se encuentra en el nivel 29 de la API, pero a finales del 2020 salió Android 11, y con él el nivel 30 de la API. Dado que la aplicación se encontraba en las últimas etapas de su desarrollo, se decidió no actualizar el SDK para evitar posibles errores en la aplicación y al ver que el identificador persistente que utilizamos del dispositivo, la @MAC de la tarjeta de red del dispositivo, no se veía afectada. De todas formas, pese a que no afecta gravemente a la aplicación, queda como trabajo pendiente actualizar el SDK.

Aparte de actualización y mantenimiento, se podría ampliar la aplicación ofreciendo al usuario sugerencias en base a la edad del dispositivo, ya sea obteniendo los ciclos del dispositivo anotados en DeviceHub como respuesta a la petición a la API, o determinando la fecha de fabricación del dispositivo y compararla con la fecha actual. Un aspecto muy interesante para determinar la antigüedad del dispositivo es calcular la capacidad actual y original de la batería, pero el SDK de Android carece de las herramientas para determinarlo, por lo que se tendría que averiguar de otra forma.

Finalmente, se podría ampliar los datos obtenidos a aquellos no considerados esenciales siempre y cuando no tenga un impacto relevante al mantenimiento de la aplicación. Si se tiene presente todos estos aspectos y se publican nuevas versiones de la aplicación, el producto final tiene el potencial de aumentar notablemente la cantidad de dispositivos móviles que reciben una segunda vida y, por tanto, reducir la cantidad de móviles que hacen falta ser fabricados, impactando a su vez a la materia prima extraída y reduciendo la huella ambiental.

9. Lista de figuras

10. Glosario

- **WEEE:** Waste Electrical and Electronic Equipment.
- **EC**: Economía Circular
- **CPR:** Common Pool Resources
- **TI:** Tecnologías de la Información
- **JSON:** JavaScript Object Notation
- **SDK:** Software Development Kit
- **IDE:** Integrated Development Environment
- **API:** Application Programming Interface
- **HTTP:** Hypertext Transfer Protocol
- **MAC:** Media Access Control
- **REST:** Representational State Transfer
- **UUID:** Universally Unique Identifier
- **URL:** Uniform Resource Locator
- **WI-FI:** Wireless Fidelity
- **SDK:** Software Development Kit

11. Bibliografía

[1] Stahel, Walter R. "The circular economy." Nature News 531.7595 (2016): 435.

[2] Franquesa, David, and Leandro Navarro. "eReuse: pensamiento circular e interacciones sobre la reutilización de dispositivos digitales."

[3] Franquesa Griso, David, et al. "Breaking barriers on reuse of digital devices ensuring final recycling." 29th International Conference on Informatics for Environmental Protection and the 3rd International Conference ICT for Sustainability (EnviroInfo & ICT4S 2015): proceedings of a meeting held 7-9 September 2015, Copenhagen, Denmark. Atlantis Press, 2015.

[4] GRAMATYKA, Paweł, et al. Recycling of waste electrical and electronic equipment. *Journal of Achievements in Materials and Manufacturing Engineering*, 2007, vol. 20, no 1-2, p. 535-538.

[5] Abdelbasir, Sabah M., et al. "Status of electronic waste recycling techniques: a review." Environmental Science and Pollution Research 25.17 (2018): 16533-16547.

[6] Franquesa, David, Leandro Navarro, and Xavier Bustamante. "A circular commons for digital devices: tools and services in ereuse.org." Proceedings of the Second Workshop on Computing within Limits. 2016.

[7] eReuse.org. *Software.* URL: https://www.ereuse.org/software/

[8] Docs, *Ionic Framework,* URL: https://ionicframework.com/docs

[9] Docs, *Extended Device Information*, URL: https://ionicframework.com/docs/native/extended-device-information

[10] React Native, *React Native. Learn once, write anywhere.* URL: https://reactnative.dev/

[11] React Native, *Platform,* URL: https://reactnative.dev/docs/platform#constants

[12] Android Developers, *Download Android Studio and SDK Tools*, URL: developer.android.com/studio.

[13] Android Developers, *Introducción a Android Studio,* URL: https://developer.android.com/studio/intro

[14] Java, *¿Qué es Java?*, URL: https://www.java.com/es/about/whatis_java.jsp

[15] Jet Brains, *IDE para JVM eficaz y ergonómico,* URL: https://www.jetbrains.com/es-es/idea/

[16] Postman, *The Collaboration Platform for API Development,* URL: https://www.postman.com/

[17] Android Developers, *Descripción general de Volley,* URL: https://developer.android.com/training/volley

[18] Android Developers, *android.app.DownloadManager,* URL: https://developer.android.com/reference/android/app/DownloadManager

[19] Retrofit, *A type-safe HTTP client for Android and Java,* URL: https://square.github.io/retrofit/

[20] Device Info, *Device Info,* URL: https://www.deviceinfo.me/

[21] Google Play, *Device Checker *SAM* (Phone and tablet testing),* URL: https://play.google.com/store/apps/details?id=trunghieu.tnt.samsungdevicetest

[22] Google Play, *Device Info,* URL: https://play.google.com/store/apps/details?id=com.alphabetlabs.deviceinfo

[23] Proyectos Ágiles, *Qué es SCRUM,* URL: https://proyectosagiles.org/que-es-scrum/

[24] OpenWebinars, *Qué es un Sprint,* URL: https://openwebinars.net/blog/que-es-un-sprint-scrum/

[25] developers. *Cambios en la privacidad de Android 10.* URL: https://developer.android.com/about/versions/10/privacy/changes

[26] Payscale. *Spain.* URL: https://www.payscale.com/research/ES/Country=Spain/Salary

[27] Android Developers, *Android 10 para desarrolladores,* URL: https://developer.android.com/about/versions/10/highlights

[28] Android Developers, *Android 11 para desarrolladores,* URL: https://developer.android.com/studio/releases/platforms#11

[29] Android Developers, *Todo lo que necesitas para realizar compilaciones en Android,* URL: https://developer.android.com/studio/features

[30] Oracle, *java.util (Java Platform SE 8),* URL: https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html

[31] Android Developers, *android.os,* URL: https://developer.android.com/reference/android/os/package-summary

[32] Android Developers, *android.widget,* URL: https://developer.android.com/reference/android/widget/package-summary

[33] Android Developers, *android.view.WindowManager,* URL: https://developer.android.com/reference/android/view/WindowManager

[34] Android Developers, *androidx.appcompat.app.AppCompatActivy,* URL: https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity

[35] GitHub, *google/gson,* URL: https://github.com/google/gson

[36] OkHttp, *Overview*, URL: https://square.github.io/okhttp/

[37] GitHub, *eReuse/devicehub-teal,* URL: https://github.com/ereuse/devicehub-teal

[38] xdadevelopers, *How to Root Any Device,* URL: https://www.xda-developers.com/root/

[39] Android Developers, *android.os.Build#getSerial(),* URL: https://developer.android.com/reference/android/os/Build#getSerial()

[40] Android Developers, *Recomendaciones para identificadores únicos,* URL: https://developer.android.com/training/articles/user-data-ids

[41] Android Developers, *android.os.Build,* URL: https://developer.android.com/reference/android/os/Build

[42] Android Developers, *android.view.WindowManager,* URL: https://developer.android.com/reference/android/view/WindowManager

[43] Android Developers, *android.os.StatFs,* URL: https://developer.android.com/reference/android/os/StatFs

[44] Android Developers, *android.os.Environment,* URL: https://developer.android.com/reference/android/os/Environment

[45] Android Developers, *android.app.ActivityManager,* URL: https://developer.android.com/reference/android/app/ActivityManager

[46] Android Developers, *android.os.Build#getSerial(),* URL: https://developer.android.com/reference/android/os/Build#getSerial()

[47] Android Developers, *android.os.Build#SERIAL*, URL: https://developer.android.com/reference/android/os/Build#SERIAL

[48] Android Developers, *android.Manifest.Permission#ACCESS_WIFI_STATE,* URL: https://developer.android.com/reference/android/Manifest.permission#ACCESS_WIFI_STATE

[49] Android Developers, *Introducción a las actividades,* URL: https://developer.android.com/guide/components/activities/intro-activities

[50] Red Hat, *¿Qué es una API de REST?,* URL: https://www.redhat.com/es/topics/api/what-is-a-rest-api

[51] CodingSonata, *Retrofit Tutorial in Android - Part 1 Introduction,* URL: http://codingsonata.com/retrofit-tutorial-android-part-1-introduction/

[52] Future Studio, *Retrofit 2 - Basics of API Description,* URL: https://futurestud.io/tutorials/retrofit-2-basics-of-api-description

[53] GitHub, *eReuse/eReuse-NativeAndroid-app,* URL: https://github.com/eReuse/eReuse-NativeAndroid-app/blob/master/LICENSE.txt

[54] GitHub, *eReuse/eReuse-NativeAndroid-app,* URL: https://github.com/eReuse/eReuse-NativeAndroid-app

[55] F-Droid, *About*, URL: https://f-droid.org/en/about/

[56] Android Studio, *Cómo depurar tu app,* URL: https://developer.android.com/studio/debug

[57] F-Droid, *Submitting to F-Droid Quick Start Guide,* URL: https://f-droid.org/en/docs/Submitting_to_F-Droid_Quick_Start_Guide/

[58] Android Developers, *Manifest.permission,* URL: https://developer.android.com/reference/android/Manifest.permission#READ_PHONE_STATE

[59] xatakandroid, *Cómo crear y publicar una app en Google Play,* URL: https://www.xatakandroid.com/aplicaciones-android/como-crear-publicar-app-google-play