Universidad de **Valladolid**

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Evaluación y depuración de una aplicación para la toma de flujometrías sonoras

Autor:
Gonzalo Ovejero Alonso
Tutores:
Mario Corrales Astorgano
Alfonso Bahillo Martínez

Ten el coraje de seguir tu corazón y tu intuición. De algún modo saben en qué quieres convertirte.
Steve Jobs

Agradecimientos

Este proyecto pone fin a una etapa muy enriquecedora, donde he vivido numerosas experiencias y vivencias que sin duda me marcarán el camino que me queda por recorrer.

Quiero agradecer en primer lugar, a mis tutores, Alfonso Bahillo y Mario Corrales, que me hayan dado la oportunidad de poder poner mi granito de arena en este proyecto, que sin duda será de gran ayuda para muchas personas. Su inestimable ayuda y su confianza en mí han sido fundamentales en el desarrollo del proyecto.

A mi familia, en especial a mis padres y a mi hermano, porque gracias a ellos he llegado a ser lo que soy, y sin su apoyo incansable y sus continuas enseñanzas no habría alcanzado los desafíos que he ido superando a lo largo de mi vida.

Me gustaría también agradecer a todos mis amigos, que han estado a mi lado a lo largo de estos años, compartiendo risas, momentos felices y celebraciones, y que han sido un pilar fundamental en los momentos difíciles.

Por último, quiero dedicar este proyecto a mis abuelos, a Floren y Marci, y en especial a Aga y a Toya, que aunque no estén ya a mí lado, sé que estarían orgullosos de ver en la persona que me he convertido.

Gracias.

Resumen

Los numerosos y constantes avances tecnológicos, sobretodo en el campo de la medicina, han permitido que podamos tener un mejor control y prevención de posibles patologías, mejorando la calidad de vida de cualquier paciente. En nuestro caso, centrándonos en el campo de la urología, las uroflujometrías son un procedimiento clave, ya que nos permiten examinar el estado del aparato urinario masculino, y propiciar un diagnóstico más adecuado. A partir de esta necesidad, surge la idea de desarrollar una aplicación de uroflujometría, que a través de mediciones sonoras, permite detectar si el paciente esté padeciendo algún problema urinario de una manera sencilla e intuitiva. El objetivo de este TFG será evaluar y depurar esta aplicación, con el fin de que pueda ser implementada en la práctica clínica, facilitando la monitorización continua en cualquier paciente.

Abstract

The numerous and constant technological advances, especially in the field of medicine, have allowed us to have a better control and prevention of possible pathologies, improving the quality of life of any patient. In our case, focusing on the field of urology, urowflowmetry is a key procedure, since it allows us to examine the state of the male urinary system and provide a more accurate diagnosis. From this need, the idea of developing a uroflowmery application arises, which through sound measurements, allows us to detect if the patient is having an urinary problem in a simple and intuituve way. The aim of this FDP will be to evaluate and refine this application, so that it can be implemented in a clinical practice, providing continous monitoring in any patient.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

El auge de la medicina tecnológica[1] ha provocado que en los últimos años se hayan desarrollado numerosas soluciones que permiten mejorar notablemente los servicios sanitarios, así como cuidar la vida de los pacientes y reducir los costes de las pruebas y tratamientos asociados. En este contexto, las aplicaciones móviles toman una alta importancia, ya que permiten poner al alcance de cualquier paciente la posibilidad de tener un diagnóstico de su salud, así como poder prevenir posibles dolencias futuras.

Gracias a uno de los últimos informes realizados por el IQVIA Institute[2], podemos comprobar que en el mercado actual de aplicaciones móviles más importante de Android, Google Play, contamos con más de 1000 aplicaciones relacionadas con el entorno sanitario, que abarcan un amplio rango de especialidades, como se puede observar en el siguiente gráfico (figura 1.1).

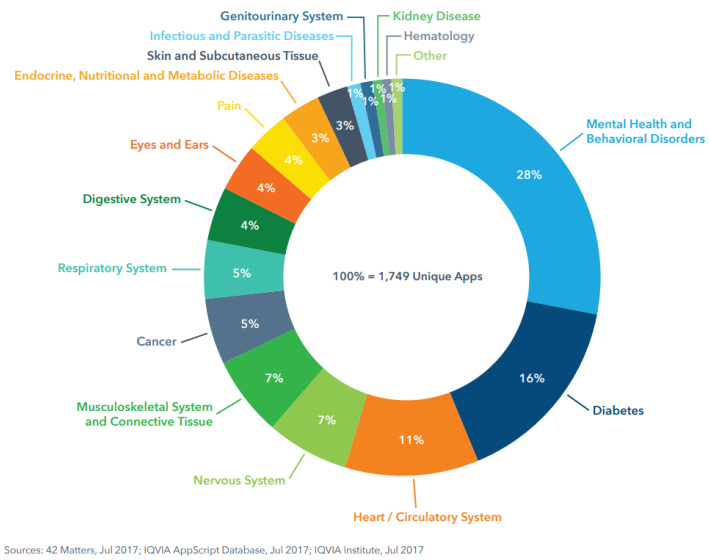


Figura 1.1: Estado actual del mercado de aplicaciones de ámbito médico en Android [2].

En nuestro caso, centraremos el foco en las aplicaciones relacionadas con el sistema genitourinario, dentro del ámbito médico de la urología, y más concretamente con la aplicación Urosound.

Urosound[3] es una aplicación privada nacida en el Instituto de Tecnología de la Universidad de Deusto, la cual permite realizar la toma de uroflujometrías de manera sencilla e intuitiva, con la ayuda de un smartwatch. La aplicación permite realizar una flujometría sonora, que se grabará con el micrófono del dispositivo, y que posteriormente será enviada a un servidor, donde será procesada y analizada gracias a unos algoritmos de machine learning.

Esta grabación junto con el análisis posterior de los algoritmos, permite evaluar el estado de las vías urinarias, y detectar posibles complicaciones como hiperplasia prostática benigna o estenosis de uretra. La aplicación cuenta con una interfaz muy accesible, y con varios modos de funcionamiento, dependiendo de la interacción con el paciente que se desee.

El usuario puede controlar la aplicación de tres maneras distintas:

- A través de la propia interfaz, utilizando los botones táctiles implementados, o los físicos si el smartwatch que se utilice dispusiese de los mismos, para dar comienzo y fin a la grabación.
- A través de un sistema de proximidad basado en la conexión con una baliza BLE (Bluetooth Low Energy), la cual permite enviar pequeñas señales que son recibidas por el smartwatch, que detecta la presencia del paciente y comienza a realizar la grabación. Para la activación de este sistema, sería suficiente con estar en el rango de la baliza, y pronunciar una keyword (palabra clave) para que de comienzo la toma de flujometría.
- A través de un algoritmo de deep learning, el cual es capaz de distinguir si el sonido que está captando el dispositivo por su micrófono se trata de una muestra de orina o no. En caso afirmativo, realizaría la grabación de dicha toma.

Este tipo de aplicaciones suponen un gran avance en este tipo de pruebas, ya que realizarlas de manera convencional es un proceso más complejo y costoso, teniéndose que utilizar un aparato llamado flujómetro, similar al representado en la figura 1.2, que conectado a un ordenador permitirá recoger todos los parámetros relacionados con la orina (cantidad, flujo, tiempo...).

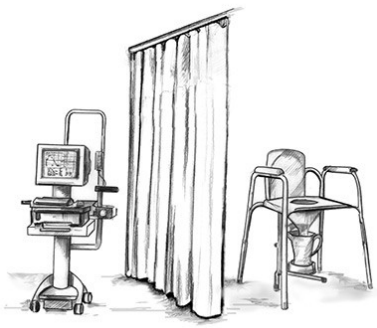


Figura 1.2: Representación gráfica de un flujómetro convencional [4].

Con el uso de la aplicación conectada a un smartwatch, la prueba puede ser realizada en cualquier lugar, y de manera más cómoda para el paciente, además de contar con el importante beneficio de que su prueba podrá ser analizada y estudiada de manera más eficiente gracias al algoritmo alojado en el propio servidor. La idea es que la aplicación pueda estar también disponible en las versiones más nuevas de Wear OS, de manera que pueda ser utilizada por un mayor rango de pacientes, ya que actualmente está implementada en una versión inferior.

1.2. Motivación

La motivación principal para la realización de este Trabajo de Fin de Grado surge de la necesidad de permitir que la aplicación Urosound pueda ser utilizada en numerosos centros hospitalarios, así como en centros médicos donde se lleven a cabo pruebas flujométricas. Como se ha comentado anteriormente, esta aplicación supondría un gran avance tecnológico en el campo de la urología, porque permitiría agilizar todo el proceso, pudiéndose realizar de manera sencilla sin la necesidad de un equipamiento específico y contando con un servidor especifico donde alojar todos los resultados obtenidos.

Además, el lanzamiento de la aplicación servirá de gran ayuda al personal sanitario, permitiendo que puedan tener acceso a los resultados de las pruebas de manera sencilla, facilitando así el diagnóstico y posterior seguimiento de los pacientes.

De igual manera, es importante que la aplicación pueda ser utilizada por un amplio número de smartwatches distintos, por lo que será un reto el poder adaptar el funcionamiento de la aplicación a distintas versiones del sistema, dotando a la aplicación de una gran versatilidad. Actualmente la aplicación trabaja sobre la versión 9 de Android, y realizaremos un trabajo de adecuación a la última versión disponible, que es Android 13.

Por último, uno de los puntos fuertes de la aplicación es su apuesta por ser utilizada de manera autónoma, gracias a la utilización de un algoritmo de detección de micción. Este desafío también deberá de ser afrontado, ya que esto permitirá que la interacción del paciente con el dispositivo sea la mínima, haciendo que la barrera inicial de entrada de los usuarios que utilizan la aplicación sea muy pequeña, y mejorando la integración del algoritmo con la aplicación.

1.3. Objetivos

El objetivo principal de este Trabajo de Fin de Grado se centra en la evaluación y depuración de la aplicación Urosound, donde se buscará mejorar y optimizar su comportamiento, así como realizar las modificaciones necesarias para su correcto funcionamiento. Esto permitirá que la aplicación sea más completa, y que su uso pueda llegar a extenderse a entidades médicas.

Los objetivos específicos del proyecto son los siguientes:

- Evaluar la aplicación, verificando que cumple con los requisitos previstos, así como su correcto funcionamiento con respecto a sus especificaciones. Esto conllevará a depuraciones y optimizaciones que permitan que la aplicación sea más eficiente. Se realizarán diversos tipos de pruebas, incluso en entornos reales, así como distintos análisis de la autonomía y el rendimiento.
- Realizar un trabajo de adaptación de la aplicación a una nueva versión del sistema operativo, que será clave para el desarrollo futuro de la aplicación en los nuevos dispositivos que se están lanzando actualmente en en el mercado.
- Mejorar la implementación e integración del algoritmo de deep learning para la detección de micción, de forma que pueda aumentar su eficiencia y su rendimiento, y permita que la aplicación pueda utilizarse de manera autónoma por el paciente.

Además, se revisará el conjunto de la aplicación para detectar posibles áreas de mejora, de igual forma que se valorará la implementación de los aspectos indicados como trabajo futuro en el desarrollo inicial de la aplicación.

1.4. Metodología

Tras valorar exhaustivamente el proyecto en su conjunto, así como las restricciones que se aplican, se ha optado por implementar una metodología incremental, la cual se ha aplicado a lo largo del proyecto, aunque con algunas mínimas variaciones.

Esta decisión ha sido consensuada junto con los tutores del Trabajo de Fin de Grado, ya que era la más apropiada en el contexto de este proyecto, siendo descartadas metodologías ágiles, debido a que su enfoque está más orientado a grandes proyectos con un grupo amplio de desarrolladores, así como metodologías en cascada, debido a que cuenta con la gran desventaja de que no permite que se realicen modificaciones o cambios en fases anteriores del proyecto.

Esta metodología aplicada cuenta con numerosas ventajas:

- Permite comprobar que la aplicación es funcional de forma regular, de manera que se pueden ir verificando los progresos cada cierto tiempo, así como establecer una mejor planificación desde los primeros momentos del desarrollo.
- Proporciona una mayor flexibilidad, ya que posibilita que se puedan realizar cambios a lo largo de las diferentes iteraciones de desarrollo, así como en distintos intervalos de tiempo.
- Permite una mejora en la calidad del software resultante, ya que cada iteración delimita el proceso de desarrollo a realizar, lo que hace que se tenga mayor control sobre el mismo.

Para este proyecto, se utilizarán iteraciones de una duración media de 4 a 6 semanas, ajustándose de manera progresiva a lo largo del mismo. En la figura 1.3 se puede observar el proceso completo en el que se basa esta metodología, que se detallará de manera más concreta en el capítulo siguiente.

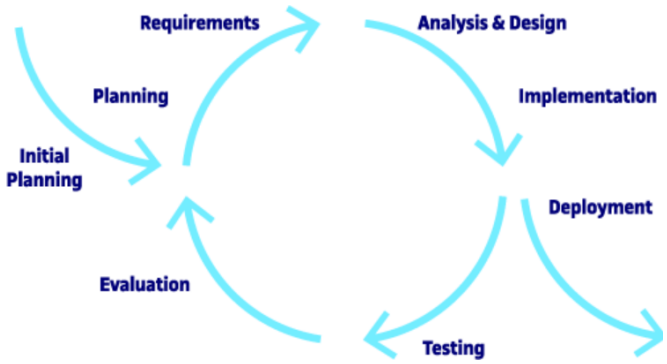


Figura 1.3: Ciclo de vida en una metodología incremental [5].

1.5. Tecnologías utilizadas

En esta sección se describirán las herramientas y diversas tecnologías que se han aplicado a lo largo del proyecto. Al tratarse de un proyecto que ya estaba desarrollado en su gran parte, se ha optado por continuar utilizando las mismas tecnologías en las cuales se sustentaba la implementación original.

En líneas generales, para la evaluación y depuración de la aplicación se ha utilizado Android Studio como entorno de desarrollo principal, que es con el que trabaja Android, todo bajo el lenguaje de programación Kotlin.

1.5.1. Android

Android[6][7] es un sistema operativo orientado para móviles que está basado en el núcleo de Linux. Fue diseñado por la compañía estadounidense Google, que adquirio la compañía de software Android Inc. en 2005. Su objetivo principal es fomentar el uso de un sistema de código abierto, gratuito, seguro y multiplataforma, ya que actualmente podemos encontrarlo en smartphones, tablets o smartwatches.

Figura 1.4: Logo Android [8].

Con respecto a su estructura, como hemos comentado Android cuenta con un kernel Linux, y donde se realizan los servicios más básicos, como la gestión de procesos y memoria, la seguridad o los controladores. A partir de él, Android se compone de 4 capas, que se pueden ver reflejadas en la Figura 1.5:

- *Android Runtime:* aquí podemos encontrar las bibliotecas asociadas a la funcionalidad de Java, así como la instancia de la máquina virtual asociada a cada aplicación, que se llama Dalvik.
- *Libraries (Bibliotecas):* contienen las bibliotecas base de C y C++ donde se sustentan los componentes del sistema.
- *Application Framework (Marco de trabajo de aplicaciones):* está formado por las APIs que se utilizan en las aplicaciones base, y que facilitan su reutilización para el desarrollo de nuevas aplicaciones.
- *Applications (Aplicaciones):* aquí es donde se encuentran las distintas aplicaciones escritas en Java, que vienen preinstaladas en el dispositivo por defecto (Calculadora, Cámara...)

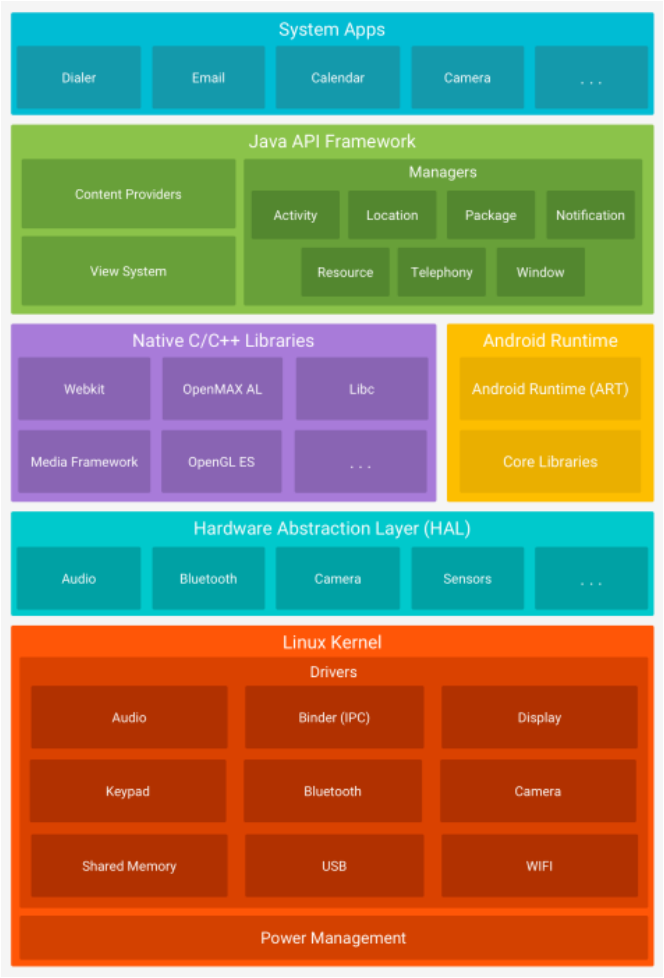


Figura 1.5: Diagrama de capas de Android [9].

1.5.2. Kotlin

Kotlin[10] es un lenguaje de programación de código abierto, que fue creado por JetBrains en 2010, y que es muy utilizado a la hora de realizar el desarrollo de una aplicación Android. Cuenta con un lenguaje de tipado estático, que permite que se pueda utilizar sobre JVM o JavaScript. Sus principales ventajas son:

- **Alta interoperabilidad con el código Java**, que hace que se complemente con su sintaxis, lo que permite que se pueda interactuar entre si.
- **Orientado a objetos**
- **Alta flexibilidad y multiplataforma**

Figura 1.6: Logo Kotlin [11].

1.5.3. Java

Java[12] es una plataforma de software y un lenguaje de programación orientado a objetos que se utiliza en numerosos dispositivos como ordenadores o smartphones. Sus reglas y sintaxis están basados en los lenguajes de C y C++. Sus principales ventajas son:

- **Portabilidad**, que permite que se pueda mover fácilmente de un sistema a otro, así como ejecutarse en diversos sistemas a la vez.
- **Facilidad de aprendizaje**
- **Robustez y seguridad**

Figura 1.7: Logo Java [13]

1.5.4. Wear OS

Wear OS[14][15] es una versión de Android que ha sido diseñada para ser utilizada en smartwatches y otros wearables, que ejecuten como mínimo una versión 6.0 Marsmallow o iOS 10, o versiones más recientes de ambos sistemas operativos.

Fue lanzado en 2014 bajo el nombre de Android Wear, y ha ido evolucionando a lo largo de los años. A día de hoy tiene compatibilidad con todo tipo de conectividades (Bluetooth, Wi-Fi, LTE...) y cuenta con acuerdos con los principales fabricantes de software del mundo.

A nivel funcional cuenta con toda la integración con los servicios de Google, así como todas las ventajas que proporciona la arquitectura de Android, con unos recursos inferiores por el tipo de dispositivos a los que va enfocado.

Figura 1.8: Logo Wear OS [14].

1.5.5. TensorFlow Lite

TensorFlow Lite[16] es una biblioteca de código abierto, desarrollada por Google, que sirve para el aprendizaje automático por tareas, y que es utilizada para satisfacer las necesidades de las nuevas redes neuronales que están surgiendo en la actualidad. Puede ser implementada en dispositivos móviles, así como en microcontroladores y cuenta con muchísimos usos, como la clasificación, reconocimiento o detección de imágenes, audios, gestos o de voz.

Figura 1.9: Logo TensorFlow Lite [17].

1.6. Estructura de la memoria

En cuanto a la memoria, está estructurada en los siguientes capítulos:

- **Introducción:** Se realiza una descripción global del proyecto, donde se comenta el contexto, las motivaciones y los objetivos que se involucran en el proyecto, así como la metodología y tecnologías utilizadas.
- **Planificación:** Se describe la secuencia temporal realizada para el desarrollo de proyecto, analizando los riesgos y costes del mismo. Además, se realiza una comparación entre la planificación estimada y la ejecución real.
- **Estado actual de la aplicación:** Se explica cómo está actualmente el proyecto, para poder verificar si se cumplen o no todos los requisitos previstos.
- **Descripción de las iteraciones:** Se detalla cómo ha sido el desarrollo del proyecto en cada iteración, así como las acciones o cambios realizados en cada uno de ellos.
- **Estado final de la aplicación:** Se describe cómo ha finalizado el proyecto, revisando aspectos como el análisis, diseño y estructura del mismo.
- **Pruebas:** Se incluyen las distintas pruebas que se han realizado para la verificación del funcionamiento de la aplicación, así como los resultados obtenidos.
- **Conclusiones:** Se relatan las conclusiones obtenidas tras la finalización del proyecto.
- **Apéndices:** Se aporta documentación e información general sobre la aplicación para su correcto uso.
- **Bibliografía:** Se indican las referencias bibliográficas utilizadas en el proyecto.

Capítulo 2

Planificación

En este apartado se describirá cómo se ha planificado el proyecto, así como la aplicación de la metodología descrita en el apartado 1.4 y las tareas y decisiones que se han ido desarrollando a lo largo del trabajo.

2.1. Planificación inicial

2.1.1. Descripción de fases e iteraciones en el marco temporal

El proyecto se acepta a fecha de 19 de octubre de 2023 e inicialmente se estima en 300 horas, que es el mínimo exigible desde la guía docente del propio Trabajo de Fin de Grado.

La primera propuesta realizada consistía en la entrega del proyecto a principios de 2024, debido a la celeridad con la que se requería utilizar el proyecto en el ámbito médico. Esto obligaba a realizar el proyecto en un tiempo estimado de 3 meses, lo cual hacía que la carga fuese demasiado excesiva, teniendo en cuenta que se debían gestionar otras responsabilidades académicas, como asignaturas no finalizadas, que podrían hacer que no se cumpliesen los plazos correctamente. Al cambiar la fecha de finalización del proyecto, la carga semanal se ve reducida, para que se ajuste a la duración total del mismo.

Tras varias reuniones organizativas, y tomando en consideración el estado del proyecto y lo anteriormente expuesto, se decide que se extenderá la realización del Trabajo de Fin de Grado hasta el 30 de Mayo de 2024.

Antes de la descripción de las iteraciones del proyecto, se realizará una primera fase inicial de duración estimada de 3 semanas, en la cual se analizará el funcionamiento y contexto de la aplicación, así como los objetivos, requisitos y recursos disponibles para su realización. Se profundizará además en las tecnologías que se utilizan, así como los aspectos técnicos relacionados con el propio proyecto.

Se decide optar por iteraciones de 4 a 6 semanas de duración, más largas de lo habitual. Esta

elección se debe a que, al tratarse de un trabajo de depuración y evaluación, es esencial realizar un buen análisis de los errores que suceden, así como posibles soluciones a aplicar. Además, existe una mayor posibilidad de que sucedan imprevistos no contemplados, que hace necesario contar con un tiempo de iteración más amplio.

A partir de aquí, el desarrollo del proyecto se divide en las siguientes iteraciones, siguiendo la metodología incremental, ya descrita con anterioridad:

- **1ª Iteración:** En esta iteración trataremos de recopilar la información del trabajo previo, así como organizar y estructurar el trabajo que se irá realizando conforme vaya avanzando el proyecto. Además, se realizarán pruebas para evaluar el estado real del sistema, que serán descritas en el capítulo de pruebas posterior. Tendrá una duración estimada de 4 semanas.
- **2ª Iteración:** Nos centraremos en la evaluación especifica, así como en la depuración de errores del modo físico por botones de la aplicación. Tendrá una duración estimada de 4 semanas.
- **3ª Iteración:** Verificaremos la funcionalidad referida a la utilización de la baliza BLE, que deberá de interactuar con el modo de activación por voz de la aplicación. Se revisará su posible optimización, así como posibles anomalías. Tendrá una duración estimada de 6 semanas.
- **4ª Iteración:** Se implementará un nuevo algoritmo de deep learning de mayor eficiencia, que permita mejorar la detección de micción de manera autónoma. Tendrá una duración estimada de 6 semanas.
- **5ª Iteración:** Se realizarán las pruebas finales, así como se realizarán correcciones finales en el proyecto final, de cara a su entrega final. Esta última iteración tendrá una duración estimada de 3 semanas.

En la figura 2.1 , podemos ver una gráfica donde se muestra el tiempo estimado por iteración:

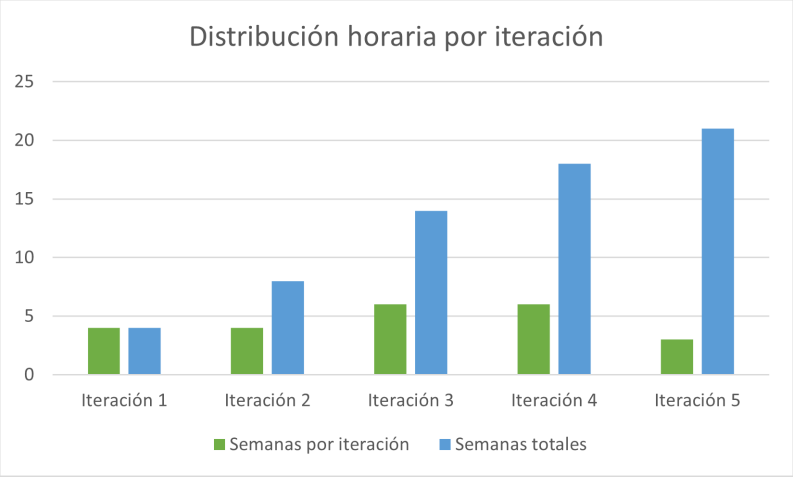


Figura 2.1: Gráfica del tiempo estimado por iteración

2.1.2. Análisis y gestión de riesgos

En esta sección se reflejarán los riesgos que pueden suceder a lo largo del desarrollo del proyecto, así como el impacto que podrían tener y la probabilidad de que ocurran. Es muy importante anticiparse a los mismos, y una buena planificación para como reducirlos al máximo es clave si queremos que el proyecto resulte siendo exitoso. Están descritos en la tabla 2.1:

ID	Riesgo	Descripción	Probabilidad	Retraso estimado (días)
01	Ausencia de tutores asignados	Los tutores sufren una baja en el transcurso del proyecto	0.1	7
02	Planificación deficiente	No se realiza una buena organización para abordar el proyecto	0.25	15
03	Incapacidad médica	El alumno sufre un inconveniente médico	0.1	4
04	Ausencia de material informático	El equipo informático o los dispositivos asociados (smartwatch) pueden no estar disponibles temporalmente	0.1	7
05	Falta de experiencia	El trabajo con nuevas tecnologías puede que requiera de mayor tiempo de adaptación	0.3	15
06	Incompatibilidad con otras asignaturas o trabajo personal	Deben realizarse asignaturas o un trabajo que conlleve a retrasos en el proyecto	0.25	12
07	Cambio en los requisitos técnicos	Se modifican las directrices a medida que el proyecto está en desarrollo	0.2	15

Tabla 2.1: Tabla de análisis y gestión de riesgos.

En la tabla 2.2, por otro lado, se describen las acciones de mitigación y correctivas para que los riesgos afecten en la menor medida posible al proyecto.

ID	Riesgo	Exposición	Acciones de mitigación	Acciones correctivas
01	Ausencia de tutores asignados	1	Solicitar un cambio de tutor académico para planificar de nuevo	Promover una comunicación telemática a través de una herramienta
02	Planificación deficiente	0.25	Dedicar horas extras para reconducir el ritmo de trabajo	Realizar una estimación temporal más ajustada a la realidad
03	Incapacidad médica	0.5	Realizar horas extras, o incluso reprogramar la fecha de entrega en casos graves	Tomar las máximas preocupaciones para no sufrir contagios, así como mantener una vida saludable
05	Falta de experiencia	3	Dedicar horas extras para contrarrestar la falta de experiencia	Dedicar más horas libres al estudio o desarrollo práctico para que el proyecto no sufra retrasos
06	Incompatibilidad con otras asignaturas o trabajo personal	3.5	Reestructuar la planificación para tener en cuenta estas incompatibilidades	Dedicar el tiempo libre a conseguir estar al día con respecto a otras asignaturas o trabajo
07	Cambio en los requisitos técnicos	4.5	Emplear horas extras para ajustarse a los nuevos requisitos, o retrasar el proyecto	Realizar una buena compresión de lo que se solicita, con ayuda de los tutores

Tabla 2.2: Tabla de análisis y gestión de riesgos con las acciones de mitigaciones y corrección.

2.1.3. Entorno de trabajo

En esta sección describiremos las herramientas que se han utilizado para la realización del proyecto:

2.1.3.1. Herramientas de desarrollo utilizadas

- **Android Studio[18]:** es el entorno de desarrollo oficial para el desarrollo de aplicaciones en Android, utilizado para el desarrollo de la aplicación del proyecto.
- **Astah Professional[19]:** es una herramienta de diseño que permite la realización de diagramas UML.
- **GitHub[20]:** es una herramienta online para el almacenamiento del código, así como el control de versiones referentes al proyecto.
- **nRF Connect[21]:** es una aplicación móvil que permite la gestión de las balizas Bluetooth BLE, cuyo software pertenece a Nordic Semiconductors.
- **LightBlue[22]:** es una aplicación móvil que permite la gestión de las balizas Bluetooth BLE, cuyo software pertenece a Punch Through Design.

2.1.3.2. Dispositivos de desarrollo utilizados

Describiremos los equipos que hemos empleado para el desarrollo del proyecto:

Equipo	**MSI Prestige 15 A11SCX**
Sistema Operativo	Windows 11
Arquitectura	64 bits
Procesador	Intel i7-1185G7 @ 3.00GHz
Disco Duro	1 TB SDD
Memoria RAM	16 GB DDR4

Tabla 2.3: Tabla características equipo portátil

Equipo	**OPPO Watch 41 mm (Wi-Fi)**
Sistema Operativo	WearOS
Pantalla	AMOLED 1,91"
Procesador	Snapdragon Wear 3100 de Qualcomm
Disco Duro	8 GB
Memoria RAM	1 GB
Conectividad	WiFI, Bluetooth 4.2, NFC
Sensores	Giroscopio, Frecuencia cardíaca, Altímetro, GPS con GLONASS Acelerómetro de 3 ejes, Óptico de latido, de capacitancia de luz ambiental y de control del sueño.

Tabla 2.4: Tabla características smartwatch Oppo

Equipo	**Samsung Galaxy Watch 4 Classic 4G**
Sistema Operativo	WearOS
Pantalla	AMOLED 1,4"
Procesador	Samsung Exynos W920
Disco Duro	16 GB
Memoria RAM	1.5 GB
Conectividad	WiFI, Bluetooth 4.2, NFC, LTE
Sensores	Frecuencia cardíaca, GPS, Acelerómetro de 3 ejes, Óptico de latido, impedancia de luz ambiental, barómetro, giroscopio y magnético.

Tabla 2.5: Tabla características smartwatch Samsung Watch 4

Además, se utilizan varias balizas, para su posterior utilización con respecto a la detección automática mediante Bluetooth. Las balizas utilizadas son Minew Tech, mostrada en la figura 2.2, así como Global Tag, mostrada en la figura 2.3 . En el desarrollo del proyecto, el uso mayoritario ha sido con la de Global Tag, debido a su alta sencillez para configurarla, así como su rápido funcionamiento.

Ambas balizas cuentan con una pila estándar de litio, la cuales aseguran una larga durabilidad, debido al bajo consumo de este tipo de dispositivos. La baliza de Minew Tech funciona directamente tras retirar el plástico que inhabilita el contacto con la pila, y la de Global Tag cuenta con un interruptor

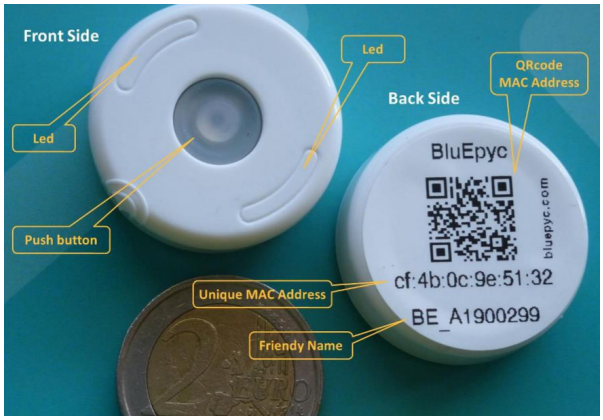


Figura 2.2: Baliza Minew Tech [23].

Figura 2.3: Baliza Global Tag

que permite su encendido y apagado. Con las aplicaciones indicadas [21] [22], se pueden configurar todos sus parámetros internos, como su dirección MAC, tasas de refresco o valores Major y Minor.

2.1.3.3. Otras herramientas de soporte

- **Microsoft Teams[24]:** herramienta utilizada para la comunicación con los tutores sobre aspectos, dudas y avances del proyecto.
- **Overleaf[25]:** es el editor de textos online utilizado para la redacción del proyecto, que utiliza LaTeX como lenguaje de programación orientado a texto.
- **LaTeX[26]:** es un sistema de composición de textos, de software libre, que se utiliza para la realización de artículos y libros orientados al ámbito científico, y sobre el que se ha realizado el proyecto.
- **Notepad++[27]:** es un editor de textos de software libre, que soporta numerosos lenguajes distintos, y que se ha utilizado como bloc de notas y anotaciones sencillas.
- **Notion[28]:** es un software de gestión de proyectos, que permite organizar tareas y objetivos en grandes proyectos o coorporaciones. Se ha utilizado como cuaderno de bitácora del proyecto, reflejando todos los datos relevantes del mismo.
- **Microsoft Excel[29]:** es un software de hojas de cálculo pertenenciente a Microsoft, el cual se ha empleado para la realización de gráficos y representación de datos en el proyecto.

2.1.4. Estimación de costes

Durante esta sección se realizará una estimación del coste que ha supuesto la ejecución del proyecto. Tendremos que tener en consideración el coste humano, así como el referente al equipo informático, y al espacio de trabajo donde se desarrolla.

2.1.4.1. Costes humanos

Para la realización del Trabajo de Fin de Grado, tenemos que tener en cuenta que es realizado por un único desarrollador, el cual es el evaluado por el mismo proyecto, y que no recibe ninguna remuneración económica por el mismo.

Tomamos como referencia el coste medio del salario de un desarrollador junior de Android [30], que en España es de 21.000 anuales, con lo que si aplicásemos una jornada laboral estándar de 40 horas semanales (8 horas diarias), el salario bruto sería de 10.94/hora. De esta manera, como el proyecto está establecido para que su realización sean 300 horas, esto nos daría un coste total del salario del desarrollador de 3281 brutos.

Por último, tendríamos que reflejar el coste de los tutores del proyecto, que ejercen una labor de dirección y supervisión del proyecto. Estimamos un coste de 23 euros por hora, habiendo realizado 5 reuniones, además de una revisión posterior de la memoria de 3 horas. Con todo ello, el coste total aproximado sería de 368.

2.1.4.2. Costes del equipo informático

Con respecto al equipo informático, se indican los precios de los dispositivos utilizados para la realización del proyecto:

- **Equipo portátil:** MSI Prestige 15 A11SCX = 1399
- **Smartwatch 1:** OPPO Watch 41 mm (Wi-Fi) = 289,99
- **Smartwatch 2:** Samsung Galaxy Watch 4 Classic 4G = 369,99

Sin embargo, no podríamos imputar el coste total del dispositivo, ya que se disponía de los dispositivos con anterioridad. Calcularíamos el coste amortizado, dividiendo las horas de uso del material por sus horas de vida totales, multiplicándolo por el coste original. Se estima que el equipo portatil tiene una vida media de 4 años, esto nos arroja:

$$\frac{\text{horasUsoDispositivoProyecto}}{\text{horasVidaUtil}} \times \text{precioTotal}$$

Con lo que en nuestro caso, obtendríamos:

Equipo	Coste amortizado	Coste
MSI Prestige 15 A11SCX	11.97 €	1399 €
OPPO Watch 41 mm (Wi-Fi)	2,48	289,99 €
Samsung Galaxy Watch 4 Classic 4G	3,17	369,99 €
Total	17,62 €	1.998,98 €

Tabla 2.6: Costes amortizados equipo informático

En el caso del equipo cedido por parte de la Universidad, se toman en consideración un tiempo medio de 4 años desde su adquisición, pero no se toman en consideración los costes repercutidos por el uso de licencias de programas que no se distribuyen de manera libre, ya que el coste es asumido por la Universidad de Valladolid.

2.1.4.3. Costes del entorno de trabajo

Por último, tendremos en consideración el coste necesario para el mantenimiento del puesto de trabajo, que serán gastos fijos básicos como el alquiler o los servicios de suministros (luz, agua, internet...). Al realizarse en la propia casa del estudiante, no se reflejará el coste del alquiler. Con respecto al resto, podríamos considerar un coste medio mensual de 45, aplicado las tarifas actuales a un único individuo de manera amortizada.

Además, deberemos considerar el equipo adicional del estudiante, como son los periféricos y una pantalla externa, que conllevará el coste también estimado de manera amortizada de 80.

2.1.4.4. Coste total del proyecto

Tomando en consideración todas las estimaciones que se han descrito con anterioridad, en la siguiente tabla se refleja el coste total del proyecto:

Equipo	Coste (€)
Salario estudiante	3281
Salario supervisores (tutores)	368
Equipo informático	17,62
Entorno de trabajo	125
Total	**3.791,62**

Tabla 2.7: Costes amortizados equipo informático

2.2. Comparativa entre la planificación inicial y el trabajo realizado

En esta sección se detalla cómo ha sido la planificación una vez el proyecto ha finalizado, para verificar cómo ha sido de correcta o no de la estimación.

En la planificación inicial se estimaron 300 horas de trabajo total para la realización del proyecto, y que tras la realización del mismo, ha sido algo superior, sin tener una cantidad de horas exacta, debido a algunos imprevistos no contemplados, como la realización de prácticas curriculares en varias fases del proyecto, así como la contratación posterior del estudiante para un puesto de desarrollador de software junior.

Describiremos comparativamente como han ido completándose las distintas iteraciones:

- **1ª Iteración (4 semanas est.):** se mantuvo la estimación, ya que se trató de realizar pruebas generales, así como una revisión global de la información sobre el proyecto.
- **2ª Iteración (4 semanas est.):** se aumentó en 2 semanas la duración de la iteración, debido a que se observan algunas complicaciones a la hora de depurar el código, debido en parte a la falta de experiencia con Kotlin, que hace que algunos conceptos no se consigan hacer funcionales.
- **3ª Iteración (6 semanas est.):** se aumenta en 1 semana, ya que surgen algunos inconvenientes a la hora de poner en marcha el reconocedor de voz, así como algunas consultas a la Universidad de Deusto en relación a la subida de audios al servidor externo.
- **4ª Iteración (6 semanas est.):** se aumenta el tiempo de la iteración en 3 semanas, debido a que es la que más complejidad tiene. El algoritmo de deep learning sufre inconsistencias, en las cuales se trabaja para su resolución y buen funcionamiento, así como se demora en la espera de un nuevo y actualizado algoritmo por parte de la Universidad de Deusto, que finalmente no llega a tiempo.
- **5ª Iteración (3 semanas est.):** se aumenta en 1 semana, debido a algunos cambios finales en la aplicación, así como para realizar una correcta entrega de la memoria y código de la aplicación.

Finalmente el proyecto es entregado en convocatoria extraordinaria, debido a lo anteriormente comentado. Se ha procurado seguir la planificación en la medida que ha sido posible, así como hacer frente a los distintos requisitos solicitados. Como punto positivo, la aplicación ha sido testada en un entorno real de un centro hospitalario, que se comentará en el capítulo dedicado a ello.

En la figura 2.4 se realiza una comparativa entre la planificación estimada y la llevada a cabo:

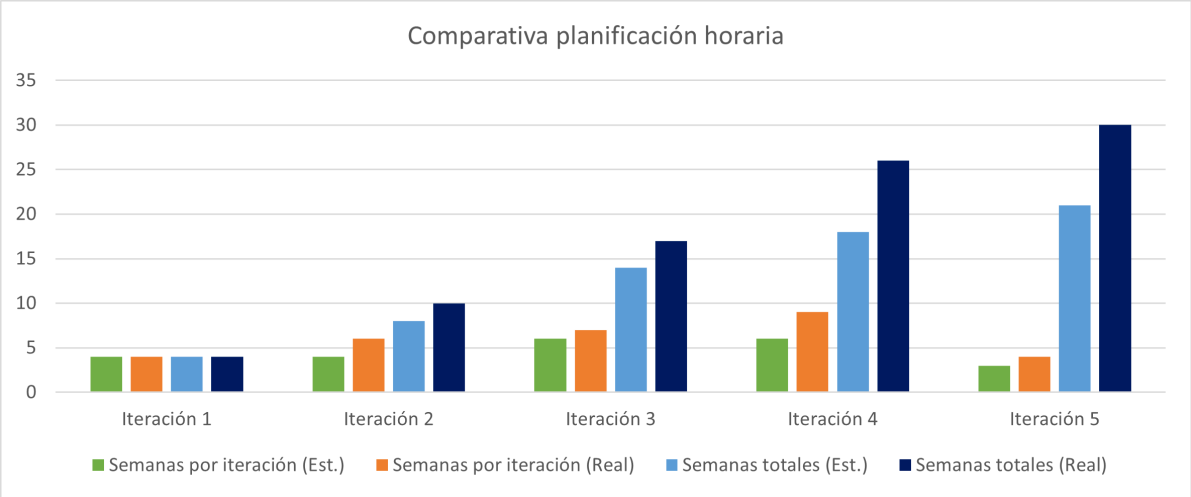


Figura 2.4: Gráfica comparativa entre las iteraciones estimadas y el trabajo realizado.

Capítulo 3

Descripción de las iteraciones

En este capítulo se comentarán las iteraciones realizadas, describiendo las tareas que se han llevado a cabo en cada una de las mismas, así como los inconvenientes detectados y los obstáculos superados. A pesar de estructurar el trabajo en iteraciones, hay algunas tareas que se han terminado de corregir en iteraciones posteriores, algo que contraviene uno de los principios de este tipo de metodología. En cualquier caso, ha sucedido de manera excepcional, y en puntos del proyecto que involucran a distintas partes del desarrollo.

3.1. Primera iteración

A lo largo de la primera iteración la idea principal fue evaluar cómo estaba el proyecto en su fase inicial, para poder decidir que aspectos de la aplicación eran necesarios depurar y verificar para que la aplicación alcanzase el objetivo final. Se había realizado un trabajo previo unas semanas antes, para conocer de manera general y a alto nivel información sobre el proyecto.

Esta iteración tuvo 4 semanas de duración, en las cuales se realizó una evaluación de la aplicación, realizando pruebas de caja negra, es decir, sin tener aún conocimiento muy específico sobre cómo funcionaba el sistema de manera interna, poder comprobar que funcionaba y que no. Se detallarán en un capítulo posterior las pruebas que se realizaron.

En esta primera fase, se verificó que el sistema era funcional, pero tenía algunas inconsistencias, así como un desempeño algo errático, sobretodo a la hora de realizar las grabaciones de las micciones, al igual que con las interacciones con los reconocedores de voz y con el modelo de deep learning. Fue fundamental, en las primeras reuniones con los tutores del trabajo, el poder discutir con D.º Alfonso Bahillo Martínez, uno de los tutores del mismo, aspectos más técnicos sobre la aplicación, ya que él tuvo un importante papel en el desarrollo de la aplicación en sus inicios.

De igual manera, también fue muy valioso compartir conocimientos con el alumno que realizó la automatización de todos estos mecanismos como su trabajo de fin de grado [31] en 2023, Martín Ordóñez Vivó, que sirvió de gran utilidad para la comprensión de conceptos relacionados con el tratamiento y

procesado del audio grabado, así como aspectos internos de la aplicación.

Se inicia la redacción de la memoria, en formato borrador, anotando puntos claves observados y detalles sobre las primeras impresiones percibidas con las pruebas observadas.

3.2. Segunda iteración

3.2.1. Revisión y evaluación modo básico aplicación

Una vez realizadas las pruebas más generales, y conociendo un poco el sistema y su implementación, el objetivo principal para esta iteración fue conseguir que el modo de interacción físico sea completamente funcional.

La idea es que el usuario, de manera básica, pueda interactuar con el reloj para poder realizar la grabación de la micción, así como su posterior almacenamiento en el dispositivo. El paciente, tras acceder al lugar donde va a realizar la micción, deberá acceder a la aplicación, y de manera táctil, o utilizando los botones propios del dispositivo, realizará la grabación de la micción. Una vez haya finalizado, deberá de igual manera parar la grabación, y posteriormente se le mostrará un menú donde se le preguntará si desea guardar o no el registro sonoro realizado.

Es aquí donde nos surge el primer problema, ya que el reloj es capaz de realizar grabaciones, pero no consiguen ser almacenadas en el dispositivo de manera correcta. Los audios presentan algunos errores a la hora de ser procesados una vez que la grabación ha finalizado, lo que obliga a revisar el método de grabación startRecordingAudio() del MainActivity, representado en la figura 3.1.

```
gonovej
private fun startRecordingAudio(forced: Boolean) {
    killAppAfterSavingTimer.cancel()

    if (forced) {
        appState = AppState.RECORDING_WAV_FIXED
        setUiState(AppState.RECORDING_WAV_FIXED)
    } else {
        appState = AppState.RECORDING_WAV
        setUiState(AppState.RECORDING_WAV)
    }

    val dm = getSystemService(DISPLAY_SERVICE) as DisplayManager
    for (display in dm.displays) {
        if (display.state == Display.STATE_OFF) {
            Log.i( tag: "Screen",  msg: "WAKE UP!")
            wakeDevice()
        }
    }

    val audiosFolder = File( pathname: this.filesDir.toString() + File.separator + getString(R.string.app_name))
    var success = audiosFolder.exists()
    // If folder doesn't exist, try to create it
    if (!success) {
        success = audiosFolder.mkdirs()
    }
    // If folder creation failed, return
    if (!success) {
        Toast.makeText( context: this, getString(R.string.folder_creation_failure), Toast.LENGTH_LONG).show()
        return
    }
```

Figura 3.1: Método startRecordingActivity

Finalmente, el problema se detecta, y es causado por una mala comprobación de las variables cuando se crea la carpeta que almacena los audios. La variable audiosFolder origina alguna inconsistencia cuando la aplicación se reinstala, ya que no permite que la nueva carpeta se origine de manera correcta y que causa que el comportamiento se reproduzca en ciertas situaciones concretas, dando error en el método exists()[32].

Más allá de este inconveniente, el modo de uso de mayor interacción con el dispositivo funciona de forma correcta, registrando los audios de manera satisfactoria y en la memoria del reloj. Se deja registrado este cambio en el borrador previamente comentado, para su posterior redacción en la memoria.

3.2.2. Permisos de la aplicación

Los permisos de la aplicación han sido uno de los problemas que más se ha continuado reproduciendo a lo largo del desarrollo de las iteraciones. Debemos de tener en cuenta, los permisos que se requieren, que serán descritos a continuación:

3.2.2.1. Read External Storage (Almacenamiento) [33]

Este permiso nos habilita la aplicación para que pueda leer el almacenamiento externo, así como acceder a los archivos que están alojados en el sistema. Es clave en la aplicación para el almacenamiento de los audios, y donde nacerán muchas complicaciones que veremos en la siguiente iteración.

3.2.2.2. Write External Storage (Almacenamiento) [34]

De manera análoga, este permiso habilita la escritura en el sistema, de forma que se puedan crear y guardar archivos en el almacenamiento interno del dispositivo, que nos sirve de igual manera que lo descrito en el permiso anterior.

3.2.2.3. Record audio (Micrófono) [35]

Este permiso es imprescindible, ya que es el encargado de habilitar el micrófono para realizar una escucha del entorno del dispositivo. Este permiso, junto con el uso la librería Android Wave Recorder [36], que permite la grabación de audios directamente en formato .wav, permiten que la aplicación pueda grabar su entorno.

3.2.2.4. Access Coarse Location (Ubicación) [37]

Este permiso está relacionado con el uso del Bluetooth en la aplicación. A raíz de la utilización de balizas que utilizan BLE (Bluetooth de bajo consumo) [38], Google ha decidido que el escaneo y el

emparejamiento solo pueda ocurrir si el permiso de ubicación está permitido, ya que las propias balizas pueden revelar la ubicación del propio usuario. Lo requerimos ya que de lo contrario la aplicación no funcionaría correctamente. Este permiso nos proporciona una ubicación aproximada, que servirá únicamente para la conexión con la baliza.

3.2.2.5. Access Fine Location (Ubicación) [39]

Vinculado con el anterior permiso, en este caso este permiso nos proporciona el acceso a una ubicación de manera precisa. De igual manera que lo explicado con anterioridad, este permiso está relacionado con una restricción por parte de la empresa propietaria de Wear OS, en relación al acceso con Bluetooth LE.

3.2.2.6. Bluetooth (Bluetooth) [40]

Este permiso es imprescindible, ya que es el encargado de permitir que la aplicación pueda descubrir y emparejarse a otros dispositivos con Bluetooth en su entorno, así como comunicarse con los mismos.

La correcta gestión de los permisos es vital para el correcto funcionamiento de la aplicación, y en esta iteración no generan inconvenientes, tanto por parte del almacenamiento como por parte de la ubicación y Bluetooth.

Con respecto a uno de los trabajos futuros comentados en el Trabajo de Fin de Grado de automatización, que consiste en la ejecución de permisos de tipo root en el smartwatch, que permitan poder aumentar el rendimiento y su autonomía, se realiza un estudio del impacto que podrían tener en el proyecto, así como las implicaciones que esto tendría en la aplicación.

Por situarnos en el contexto, el concepto de rootear permite aprovechar diversos exploits o fallos en el sistema Android para poder tener acceso a restricciones impuestas por los propios fabricantes de los dispositivos, que permiten mejorar el rendimiento, autonomía o funcionalidad del dispositivo. Se puede realizar de diversas maneras, pero inicialmente consisten en la instalación de capas de usuario personalizadas, acceso a funciones ocultas u optimización extrema del dispositivo. Se decide investigar un poco más en profundidad para tomar una decisión en una iteración posterior.

Por último, destacar que la duración de la iteración es de 4 semanas, que posteriormente se incrementó en 2 semanas adicionales, en parte debido a la pronunciada curva de aprendizaje que se observa con respecto al lenguaje Kotlin. El estudiante sigue las acciones de reducción previstas en el análisis de riesgos, para que esto afecte en menor manera posible al desarrollo del proyecto.

3.3. Tercera iteración

3.3.1. Revisión y evaluación modo reconocedor de voz

A lo largo de esta iteración, de manera análoga a la 2ª iteración, la idea es revisar por completo la implementación de los 2 reconocedores de voz con los que cuenta la aplicación, así como detectar posibles fallas o errores en el código que no permitan que se realice de manera satisfactoria esta detección.

La idea principal es que gracias al uso de la baliza BLE, que estará ubicada en el lugar donde el paciente vaya a realizar la micción, el reloj detecte que está próximo a ella, y de esta manera cargará el reconocedor de voz seleccionado, que puede ser el nativo de Android [41], o el de Vosk [42], decisión tomada tras el exhaustivo estudio que se realizó en el anterior trabajo de automatización y que fueron los que mejores resultados ofrecían en nuestro contexto particular.

En primer lugar, tras las pruebas realizadas con los reconocedores, mayormente con el de Vosk, los resultados obtenidos eran poco prometedores. El reconocedor, como se puede observar en la figura 3.2, obtenía un error cuando se cargaba el modelo de idioma que utiliza el propio reconocedor para establecer la comparación entre la entrada de voz y sus propios diccionarios.

```
I/MainActivity:: 2024-01-31T20:03:12.492 100.0% onCreate, PATIENT_IN_WC
V/StorageService: Making directory /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model/model-es
V/StorageService: Copy model-es/README to /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model
V/StorageService: Making directory /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model/model-es/am
V/StorageService: Copy model-es/am/final.mdl to /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model
A/libc: Fatal signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x5b8 in tid 5668 (ech.urosoundapp), pid 5668 (ech.urosoundapp)
I/ech.urosoundap: Background concurrent copying GC freed 4334(377KB) AllocSpace objects, 1(20KB) LOS objects, 55% free, 1231KB/2MB, paused 6.094ms total 286.152ms
V/StorageService: Making directory /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model/model-es/conf
V/StorageService: Copy model-es/conf/mfcc.conf to /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model
V/StorageService: Copy model-es/conf/model.conf to /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model
V/StorageService: Making directory /storage/emulated/0/Android/data/es.deustotech.urosoundapp/files/model/model-es/graph
```

Figura 3.2: Error reconocedor de voz Vosk

Esta problemática se tardó en resolver, ya que no se conseguía dar con el fallo que se provocaba, y no parecía un problema a nivel de código. La carga del modelo del reconocedor se realizaba de manera estándar por defecto, y al no activarse el propio reconocedor no se podía determinar si era funcional o no.

Se testearon diferentes aproximaciones, como la implementación de una nueva forma de cargar el modelo, utilizando distintas funciones de la clase StorageService, que es nativa del propio reconocedor Vosk, así como el uso de la interfaz nativa de Android StorageManager [43], que no obtuvo resultados satisfactorios.

Esto propició a pensar, en una reunión promovida con los tutores del proyecto, a que pudiese ser un problema que afectase al propio smartwatch como tal, bien por haberse quedado obsoleto, o achacándolo a un problema de almacenamiento. Es por esto que se decide solicitar otro modelo de reloj, para verificar si el funcionamiento sigue originándose de la misma manera.

Finalmente, la solución que se llevó a cabo tras numerosas revisiones, ya que en el nuevo reloj, el cual detallaré en el próximo apartado, el error seguía siendo el mismo, fue la de actualizar el modelo de Vosk a una versión más actualizada, ya que esto podría hacer que los archivos se regenerasen de

manera satisfactoria, y esto fue lo que resolvió la problemática inicial.

Una vez que el propio reconocedor se podía utilizar en el smartwatch, surgió una nueva problemática. El reconocedor se iniciaba, y tras detectar la palabra clave, no se iniciaba correctamente la grabación del audio, obteniendo el error reflejado en la figura 3.3.

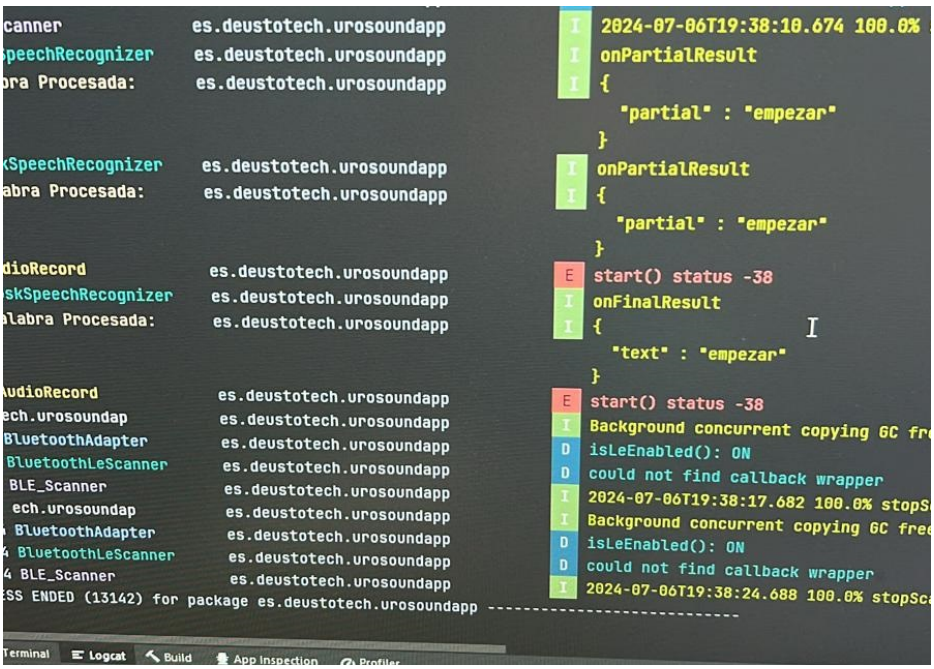


Figura 3.3: Error inicio grabación reconocedor de voz Vosk

Este problema era originado por la propia implementación del algoritmo de reconocimiento de Vosk, que accede en repetidas ocasiones al mismo método, y esto hace que la grabación se inicié de manera duplicada, lo cual conduce a un error en el propio sistema de grabación. Esto fue corregido, realizando una adaptación de la detección final de la palabra clave, que permitió que se subsanase esta falla, como se muestra en la figura 3.4.

Finalmente, el reconocedor de voz era capaz de detectar palabras pronunciadas por el usuario, y en caso de que la palabra coincida con la palabra clave que ejecuta la grabación automática, el propio smartwatch comienza su grabación, sin requerir de tanta interacción por parte del usuario.

3.3.2. Nuevo reloj para la ampliación de pruebas

Con respecto al nuevo reloj, el modelo recibido fue un Samsung Galaxy Watch 4, como se ha incorporado en el análisis de costes correspondiente. Como he comentado anteriormente, la solicitud se realizó con motivo del error en la carga de los modelos del reconocedor.

Finalmente no fue un asunto relacionado con el propio reloj inteligente proporcionado, con lo que desde esta iteración se trabaja paralelamente para que la aplicación sea funcional en ambos smartwatches.

Esto supone un reto, ya que los relojes cuentan con versiones de Android diferentes. Por un lado, el

```
private fun stopVoskSpeechRecognition() {
    speechService?.let { it: SpeechService
        it.stop()
        it.shutdown()
    }
}

gonovej *
private fun processSpeechResult(result: String) {
    Log.i( tag: "Palabra Procesada: ", result)

    if (result.contains(keywordSpeechRecognition)) {
        stopVoskSpeechRecognition()
    }
}

new *
private fun processSpeechFinalResult() {
    startRecordingAudio( forced: false)
}
```

Figura 3.4: Nuevo método procesamiento reconocedor Vosk

Oppo cuenta con una versión de Wear OS 2.45, basada en Android 9, que data de 2018, y el Samsung Watch consta de la versión 4.0 de Wear OS, que está basada en Android 13, del año 2023, como se muestra respectivamente en las figuras 3.5 y 3.6

Figura 3.5: V. Wear OS smartwatch Oppo

Figura 3.6: V. Wear OS smartwatch Samsung

Es evidente que es un cambio importante, ya que hay una gran diferencia de versiones, y numerosas variaciones entre las mismas. Esto conllevó a un cambio en los requisitos del proyecto, ya que se

creyó conveniente que la aplicación pudiese ser adaptada a la última versión de Android, para que de esta manera fuese funcional en más dispositivos, así como permitir que la aplicación estuviese lo más actualizada posible al contexto tecnológico actual.

A lo largo de las iteraciones se ha ido refactorizando el código con respecto a lo estimado para cada una de las mismas, de manera que se iba realizando una adaptación escalonada en función de la iteración a resolver. La adaptación ha resultado compleja, ya que Google ha reformulado los permisos, con el fin de proteger la privacidad de los usuarios, y eso ha generado algunas inconsistencias en la aplicación al ser testada en el Samsung Watch.

3.3.3. Revisión y nueva implementación de permisos

Como se ha comentado en el apartado anterior, la incorporación al proyecto de una nueva versión de Android, ha generado más inconvenientes en el funcionamiento de la aplicación. El más costoso de corregir, ha sido el referente al almacenamiento, así como actualizar los permisos que se refieren al Bluetooth.

Los permisos comentados en la primera iteración, *READ EXTERNAL STORAGE* y *WRITE EXTERNAL STORAGE* no tienen efecto en Android 13, con lo que es necesario reevaluar cómo permitir que los audios se puedan almacenar en el almacenamiento externo del dispositivo, ya que la propia aplicación cuenta con una funcionalidad que permite realizar una copia de seguridad a los audios, en el propio almacenamiento del dispositivo. Se pospone para una iteración posterior su implementación, por no ser prioritaria como función básica.

Con respecto a los permisos de Bluetooth, en Android 13 el permiso *BLUETOOTH* no está en uso, por lo que es necesario definir 2 nuevos permisos:

- **BLUETOOTH SCAN [44]:** este permiso es obligatorio para que el dispositivo pueda reconocer y emparejarse con los dispositivos Bluetooth que estén dentro de su alcance.
- **BLUETOOTH CONNECT[45]:** este permiso permite la comunicación entre los dispositivos que están emparejados. Es fundamental para la configuración correcta de la baliza.

La nueva implementación de los permisos quedaría reflejada en la figura 3.7, donde se verifica la versión sobre la que se ejecuta la aplicación, para solicitar al paciente los respectivos permisos:

3.3.4. Inconvenientes de la implementación de código en tiempo de ejecución

Uno de los aspectos más importantes a la hora de realizar una aplicación en Android, es la simultaneidad de la aplicación. La simultaneidad [46] es una propiedad básica dentro de una aplicación, que permite que la aplicación realice varias tareas a la vez.

Figura 3.7: Método que gestiona los permisos en el código

Es muy útil, ya que permite que la ejecución sea más rápida, así como proporcionar una experiencia muy buena para el usuario. El principal inconveniente es que es necesario tener un buen control sobre la misma, porque de lo contrario puede ocasionar errores no deseados.

Este comportamiento fue detectado en las pruebas realizadas a la aplicación mientras se utilizaba el Samsung Watch, con Android 13. El problema que se ocasionaba era que, la aplicación ejecutaba algunos trozos de código antes incluso de solicitarle al usuario que aceptase los permisos necesarios para que la aplicación funcionase correctamente.

De esta forma, una de las soluciones que propone Kotlin para evitar este tipo de problemáticas es la utilización de corrutinas, que permiten suspender la ejecución de bloques de código, y reanudarlas cuando sea necesario. Tras evaluar el estado de la aplicación y cómo esto afectaba en el código, introducir corrutinas supondría que la experiencia de usuario fuese algo menos positiva, y se podía alcanzar una buena solución, con una ligera reorganización del código que respetase los tiempos de ejecución.

Tras aplicar estos cambios, sobretodo en el bloque de código referente a la solicitud de permisos (figura 3.7), la aplicación no originaba mayores inconvenientes cuando se realizaba una ejecución completa.

3.3.5. Dificultades subida archivos a servidor remoto

Otro de los errores que se ha solventado, era el referente al envío de los audios grabados en el dispositivo hacia el servidor. En la versión actual, era necesario registrar un paciente, que se realiza mediante la app, asignándole un nombre y un código de paciente, como se refleja en la figura 3.8.

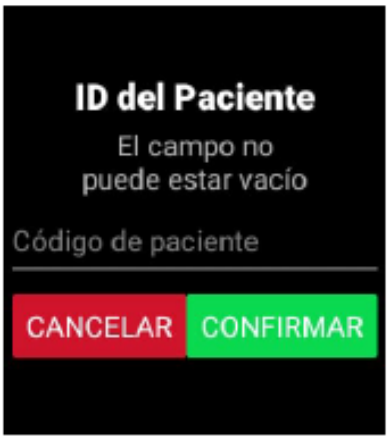


Figura 3.8: Interfaz registro paciente

Tras registrarlo, cuando se intentaba acceder a la opción que permite el envío de los audios al servidor, se obtenía un error, reflejado en la figura 3.9.

```
D/OpenGLRenderer: endAllActiveAnimators on 0x93c30f00 (RippleDrawable) with handle 0x93ce3100
D/NetworkSecurityConfig: No Network Security Config specified, using platform default
E/Volley: [365] NetworkUtility.shouldRetryException: Unexpected response code 400 for http://watchapp.dev.mobility.deustotech.eu/api/patientAuth/registerAnonymous
D/ViewRootImpl[SettingsActivity]: changeCanvasOpacity: opaque=false
I/Process: Sending signal. PID: 6080 SIG: 9
```

Figura 3.9: Error al realizar la subida del paciente

Esto obligó a contactar con la Universidad de Deusto, donde se aloja el servidor, para solicitar información sobre si se había realizado algún cambio en la aplicación web que gestiona los envíos de los audios.

Finalmente, se actualizó el endpoint al que se llamaba desde el propio método serverRegisterAnonymous(), así como se ajustaron los valores del JSON que es enviado para la creación del paciente registrado en el reloj.

3.3.6. Valoración ejecución modo superusuario (root) en el dispositivo

Por último, tras realizar una investigación en lo propuesto por parte de Martín Ordóñez en su trabajo futuro a implementar, y revisando la información que él ha aportado, se realiza una valoración global sobre si es o no interesante la posibilidad de rootear el dispositivo.

La aplicación inicialmente está concebida con el objetivo de poder realizar pruebas flujométricas, que siempre estarían solicitadas por el centro método o entidad sanitaria a la cual el paciente está adscrito, con la supervisión de su médico. En una primera implantación, la idea es que la aplicación se pueda utilizar de manera ocasional en los distintos centros médicos o hogares de los pacientes a los que se les pueda ordenar realizarlas.

No se considera prioritario, ya que como se observará en un apartado posterior, la autonomía del dispositivo cumple con el objetivo impuesto por parte de la aplicación, siendo este uno de los motivos más definitorios para proceder con su desbloqueo.

Además, rootear el dispositivo puede conllevar a grandes fallos de seguridad, que no serían admisibles en una aplicación donde se tratan datos médicos sensibles, así como a fallos en la funcionalidad del dispositivo si no se realiza de la manera adecuada.

También es interesante comentar, que cabe la posibilidad de que en un futuro la aplicación pueda evolucionar y aumente su funcionalidad, permitiendo que puedan realizarse otros tipos de pruebas, o que los pacientes puedan tener acceso más libre a la instalación de la aplicación, a modo de autoexploración sanitaria. No sería lógico que el buen rendimiento del proyecto dependiese de que el dispositivo donde se va a instalar esté o no rooteado, por lo que carece de coherencia plantearse su implantación.

Esta iteración se estima en 6 semanas, que finalmente se retrasará 1 semana debido a lo anteriormente expuesto, con la espera de comunicaciones con la Universidad de Deusto, así como la fase de adaptación al manejo del nuevo smartwatch recibido, así como la revisión de los requisitos propuestos, para añadir la nueva tarea a realizar. De igual manera que en el resto de iteraciones, se siguen reflejando todos los aspectos fundamentales en un borrador, que posteriormente se redactará de manera correcta para la realización de la memoria.

3.4. Cuarta iteración

3.4.1. Revisión y evaluación modo detección automática de micción

En esta iteración se tratará el último de los modos de funcionamiento de la aplicación, como es la utilización de un algoritmo de deep learning, en este caso TensorFlow Lite [16], que permitirá directamente detectar que se está realizando una micción, para que de esta manera se realice la grabación de audio, teniendo una interacción completamente inexistente con el smartwatch, que es objetivo final de la aplicación.

Inicialmente se trata del proceso más complejo, y ha sido el que más revisiones ha requerido, ya que no realiza las grabaciones de los audios de manera correcta. Nos centraremos en investigar y verificar toda la información asociada a la implementación de este sistema, para poder corregir su funcionamiento, y de esta manera que funcione de manera exitosa.

Se realiza una primera aproximación, donde se intentan corregir ciertos aspectos de la clase SoundRecorder, que es la que se encarga de procesar el audio que se ha recibido para verificar si se trata o no de una micción, pero al ser un proceso complejo y con un alto requerimiento de comprensión del código existente, se decide pausar hasta la próxima iteración, donde se retomará este desarrollo.

3.4.2. Lanzamiento versión de pruebas en hospital

Por otro lado, se envía una versión limitada en funciones para que se pruebe en un entorno real, en este caso en el Hospital Clínico Universitario de Valladolid, con el apoyo de uno de los tutores del proyecto, D.º Alfonso Bahillo Martínez. La idea es que se pueda probar in situ el modo más básico de la

aplicación, que a pesar de que es el que más interacción por parte del paciente requiere, a nivel de uso es el más sencillo de probar y verificar. Se comentarán los resultados de esta prueba en un capítulo posterior.

Se realiza la modificación del tiempo máximo de micción, de 90 segundos a 120 segundos, ya que se valoran posibles errores o tardanzas a la hora de accionar la grabación con el smartwatch en las pruebas reales.

3.4.3. Revisión y evaluación modo reconocedor de voz Android Nativo

Como se comentó en la iteración anterior, la aplicación cuenta con 2 reconocedores de voz incorporados, como son el nativo de Android, y el de Vosk. En la iteración 3 se trató lo referente al reconocedor de voz de Vosk, realizando las pruebas adecuadas y depurando el código de la manera necesaria, y en esta iteración revisaremos el uso y funcionamiento del reconocedor nativo de Android.

Se prueba en ambos relojes, obteniendo resultados consistentes en el Oppo, con una buena tasa de reconocimiento de las palabras que se le indican, aunque siempre con la gran desventaja de que es necesario que esté conectado a Internet, aspecto que en el de Vosk no es necesario.

Con respecto a las pruebas en el Samsung, el reconocedor nativo de Android ha dejado de tener soporte para SpeechService, que es el que se venía utilizando hasta Android 13, con lo que hace imposible esta ejecución [47]. Ante esta problemática, se valora el migrar el reconocedor actual a la solución que propone Google, que obligaría a implementar la aplicación de Servicios de Voz sobre Urosound.

Tras investigar con mayor profundidad, y teniendo en cuenta que los resultados arrojados por parte del reconocedor de voz de Vosk, así como el consenso alcanzado con los tutores, se decide prescindir de ese reconocedor, dejando unicamente la implementación del reconocedor de Android Nativo.

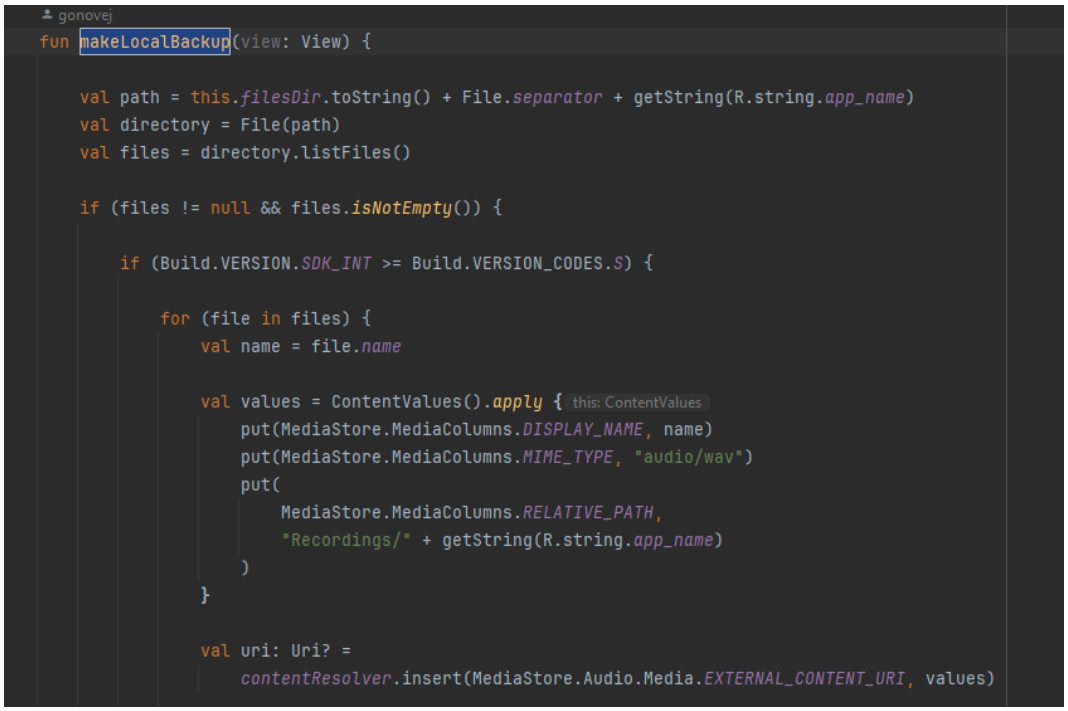
3.4.4. Resolución problema permiso almacenamiento en Android 13

Para finalizar, revisaremos el problema que pospusimos para esta iteración, con respecto a la implementación de los permisos de almacenamiento para la realización de la copia de seguridad de los audios en Android 13.

Tras valorar las posibles opciones, se opta por implementar la solución con MediaStore [48], que consiste en un proveedor de contenidos que permite a las aplicaciones acceder y gestionar contenido multimedia en el dispositivo. Lo que hace es mantener una base de datos indexada de todo el contenido, y facilita la recuperación del contenido tomando en cuenta sus metadatos, que se almacenan en las propias columnas de la base de datos.

De esta manera, no es necesario solicitar permisos en tiempo de ejecución, ya que MediaStore únicamente trabaja con las referencias a esos archivos. En la figuras 3.10 y 3.11 se detalla como se ha

realizado la implementación del MediaStore.



```
gonovej
fun makeLocalBackup(view: View) {

    val path = this.filesDir.toString() + File.separator + getString(R.string.app_name)
    val directory = File(path)
    val files = directory.listFiles()

    if (files != null && files.isNotEmpty()) {

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {

            for (file in files) {
                val name = file.name

                val values = ContentValues().apply { this: ContentValues
                    put(MediaStore.MediaColumns.DISPLAY_NAME, name)
                    put(MediaStore.MediaColumns.MIME_TYPE, "audio/wav")
                    put(
                        MediaStore.MediaColumns.RELATIVE_PATH,
                        "Recordings/" + getString(R.string.app_name)
                    )
                }

                val uri: Uri? =
                    contentResolver.insert(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, values)
```

Figura 3.10: Implementación MediaStore 1a parte

```
        if (uri != null) {
            try {
                copyAudiosToWatch(file, uri)
                file.delete()

            } catch (e: IOException) {
                val sw = StringWriter()
                val pw = PrintWriter(sw)
                e.printStackTrace(pw)
                Log.i( tag: "SettingsActivity: ", sw.toString())
                if (logsEnabled) appendLog(
                    applicationContext,
                     text: getTimestamp() + " " + batteryLevel(applicationContext) + "% " + "makeLocalBackup: " + sw.toString()
                )
                Toast.makeText(
                     context: this, getString(R.string.backup_failure), Toast.LENGTH_LONG).show()
            }
        }
    }

    Toast.makeText( context: this, "Audio backup completed successfully", Toast.LENGTH_LONG).show()
```

Figura 3.11: Implementación MediaStore 2a parte

Se recogen el nombre del audio como el tipo del mismo, para que sean incorporados a la base de datos de MediaStore, de forma que posteriormente se realiza la copia de los archivos tomando como referencia la Uri que ha generado la base de datos de MediaStore.

De esta manera, se puede seguir realizando un backup de los audios a la memoria del dispositivo, para que luego puedan ser procesados o utilizados de manera local o en caso de no disponer de una conexión de internet estable para su envío al servidor remoto. En versiones inferiores a Android 13, el backup se realizará de manera estándar, guardandolo en la carpeta DCIM del dispositivo, y en versiones superiores a Android 13, se realizará utilizando MediaStore como se indica con anterioridad.

Esta iteración se estima en una duración de 6 semanas, y se retrasa en 3 semanas, en parte por lo

que se ha comentado con respecto al código referente al algoritmo de deep learning, que es bastante complejo en su implementación, y que requiere de un estudio de mayor profundidad. Se continua con la práctica habitual de recoger información para su posterior transcripción a la memoria.

3.5. Quinta iteración

3.5.1. Implementación modo detección automática de micción

En esta última iteración, nos centraremos en la correcta implementación y funcionamiento del algoritmo de detección automática, que ya había sido previamente desarrollado por parte del alumno Martín Ordóñez Vivó en su anterior trabajo de automatización, pero que debido al corto tiempo que tuvo para su realización el algoritmo no quedó del todo implementado, y fue uno de los trabajos futuros a mejorar que él mismo menciona en su memoria.

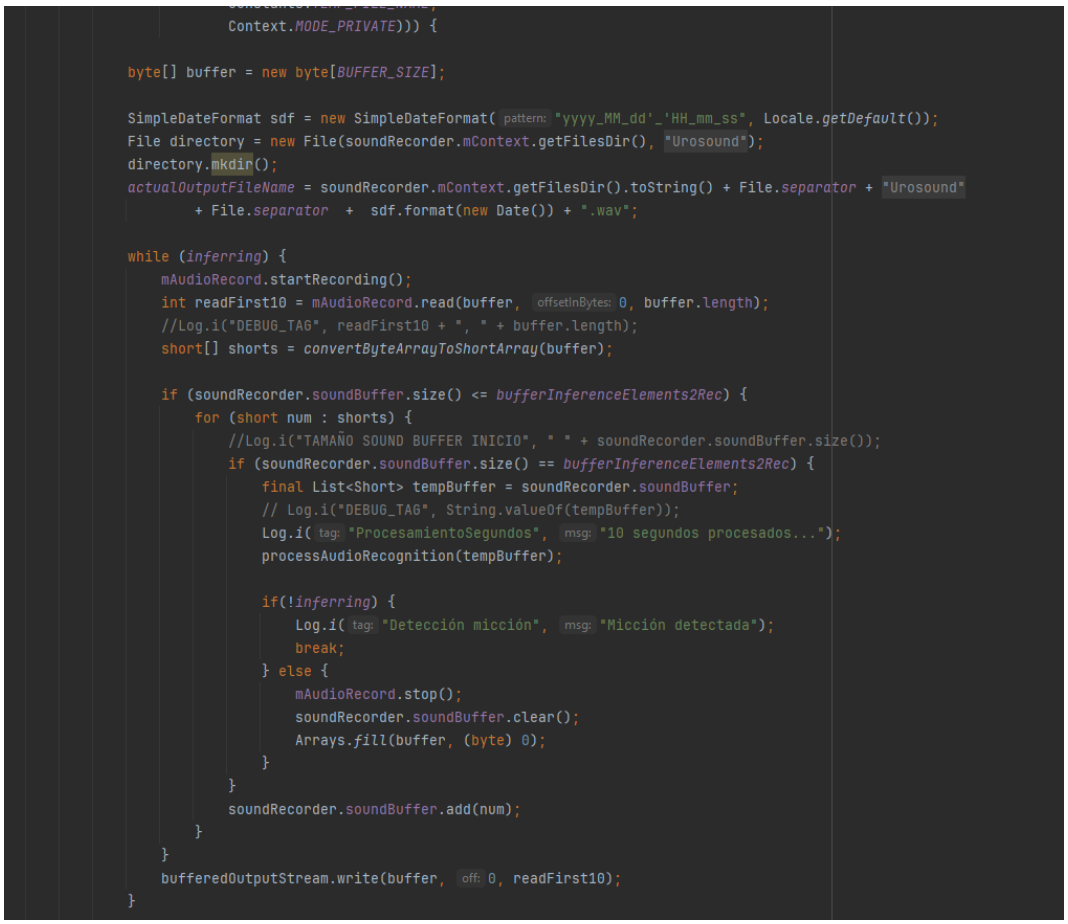
Conceptualmente, la idea es que el smartwatch pueda procesar tomas de audio ambiente de una duración de 10 segundos. Esto es debido, como ha sido comentado en el anterior trabajo, a las limitaciones de la carga de procesamiento del algoritmo, que impiden que la escucha activa pueda tener más duración.

En estas escuchas de 10 segundos, la idea es que si se detecta micción, que se puede verificar si el algoritmo resulta en un void en caso afirmativo, o un no void en caso contrario, la uroflujometría continué grabándose hasta una duración total de 120 segundos, que es el tiempo medio estimado para una micción estándar.

Para ello, ha sido fundamental el uso de la clase AudioRecord[49], que se encarga de manejar los recursos de audio en la aplicación, para poder grabar audio con el propio hardware del dispositivo. Todo ello se combina con el uso de AsyncTask, que es la encargada de ejecutar toda la implementación del algoritmo, mientras que el usuario espera a que la micción sea detectada.

También es fundamental el uso correcto de los buffers, que son elementos claves en cualquier implementación de audio, para el correcto almacenamiento del audio. En nuestro caso, debido a la implementación del servidor remoto al que se enviarán las tomas flujométricas para su procesamiento, es necesario que los audios se graben en formato *.wav*. Debido a esta limitación, es necesario que para que AudioRecord funcione de manera óptima, se realice una grabación siguiendo los parámetros indicados en el propio artículo de los autores de la aplicación[], de manera que luego se pueda convertir el formato inicial *.pcm* al necesario para su subida al servidor.

Cada 10 segundos, el audio grabado será enviado al método processAudioRecognition(), que será el encargado de ejecutar la inferencia del algoritmo, y devolverá un *void* o un *no void* en función del audio detectado, como se puede observar en las siguientes figuras 3.12 y 3.13.



```
                Context.MODE_PRIVATE))) {

        byte[] buffer = new byte[BUFFER_SIZE];

        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy_MM_dd'_'HH_mm_ss", Locale.getDefault());
        File directory = new File(soundRecorder.mContext.getFilesDir(), "Urosound");
        directory.mkdir();
        actualOutputFileName = soundRecorder.mContext.getFilesDir().toString() + File.separator + "Urosound"
                + File.separator  +  sdf.format(new Date()) + ".wav";

        while (inferring) {
            mAudioRecord.startRecording();
            int readFirst10 = mAudioRecord.read(buffer,  offsetInBytes: 0, buffer.length);
            //Log.i("DEBUG_TAG", readFirst10 + ", " + buffer.length);
            short[] shorts = convertByteArrayToShortArray(buffer);

            if (soundRecorder.soundBuffer.size() <= bufferInferenceElements2Rec) {
                for (short num : shorts) {
                    //Log.i("TAMAÑO SOUND BUFFER INICIO", " " + soundRecorder.soundBuffer.size());
                    if (soundRecorder.soundBuffer.size() == bufferInferenceElements2Rec) {
                        final List<Short> tempBuffer = soundRecorder.soundBuffer;
                        // Log.i("DEBUG_TAG", String.valueOf(tempBuffer));
                        Log.i( tag: "ProcesamientoSegundos",  msg: "10 segundos procesados...");
                        processAudioRecognition(tempBuffer);

                        if(!inferring) {
                            Log.i( tag: "Detección micción",  msg: "Micción detectada");
                            break;
                        } else {
                            mAudioRecord.stop();
                            soundRecorder.soundBuffer.clear();
                            Arrays.fill(buffer, (byte) 0);
                        }
                    }
                    soundRecorder.soundBuffer.add(num);
                }
            }
            bufferedOutputStream.write(buffer,  off: 0, readFirst10);
        }
```

Figura 3.12: Implementación detección micción 1a parte

```
        //Log.i("PARAMS", Arrays.toString(params));

        if (max > PREDICTION_THRES) {
            // Get label and confidence
            final String prediction = labels[argmax];
            final String confidence = String.format("%,.2f", max);

            // Send prediction back to MainActivity
            Intent broadcastIntent = new Intent();
            String data = prediction + "," + confidence + "," + LocalTime.now() + "," + db(sData) + "," + recordTime;
            Log.i( tag: "TEST", data);


            if(prediction.equals("void")) {
                inferring = false;
                broadcastIntent.setAction("mBroadcastSoundPrediction");
                broadcastIntent.putExtra( name: "INFERENCE_DATA", data);
                mContext.sendBroadcast(broadcastIntent);

                Log.i( tag: "INFERENCIA",  msg: "Audio detectado correctamente");
                recordingDetection = true;
            }
        }
    }
```

Figura 3.13: Implementación detección micción 2a parte

En la figuras 3.14 y 3.15, podemos observar cómo se ha resuelto la implementación del resto de la toma flujométrica. Con el uso de las flags inferring y recording, podemos conocer el estado del algoritmo, y verificar en qué fase se encuentra. Una vez que la inferencia ha resultado correcta, se procederá a la

grabación de los 110 segundos restantes para completar la toma flujométrica.

```
                Context.MODE_PRIVATE))) {

            byte[] buffer = new byte[BUFFER_SIZE];

            SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy_MM_dd'_'HH_mm_ss", Locale.getDefault());
            File directory = new File(soundRecorder.mContext.getFilesDir(), "Urosound");
            directory.mkdir();
            actualOutputFileName = soundRecorder.mContext.getFilesDir().toString() + File.separator + "Urosound"
                    + File.separator  +  sdf.format(new Date()) + ".wav";

            while (inferring) {
                mAudioRecord.startRecording();
                int readFirst10 = mAudioRecord.read(buffer,  offsetInBytes: 0, buffer.length);
                //Log.i("DEBUG_TAG", readFirst10 + ", " + buffer.length);
                short[] shorts = convertByteArrayToShortArray(buffer);

                if (soundRecorder.soundBuffer.size() <= bufferInferenceElements2Rec) {
                    for (short num : shorts) {
                        //Log.i("TAMAÑO SOUND BUFFER INICIO", " " + soundRecorder.soundBuffer.size());
                        if (soundRecorder.soundBuffer.size() == bufferInferenceElements2Rec) {
                            final List<Short> tempBuffer = soundRecorder.soundBuffer;
                            // Log.i("DEBUG_TAG", String.valueOf(tempBuffer));
                            Log.i( tag: "ProcesamientoSegundos",  msg: "10 segundos procesados...");
                            processAudioRecognition(tempBuffer);

                            if(!inferring) {
                                Log.i( tag: "Detección micción",  msg: "Micción detectada");
                                break;
                            } else {
                                mAudioRecord.stop();
                                soundRecorder.soundBuffer.clear();
                                Arrays.fill(buffer, (byte) 0);
                            }
                        }
                        soundRecorder.soundBuffer.add(num);
                    }
                }
                bufferedOutputStream.write(buffer,  off: 0, readFirst10);
            }
```

Figura 3.14: Implementación algoritmo detección micción completo 1a parte

```
        //Log.i("PARAMS", Arrays.toString(params));

        if (max > PREDICTION_THRES) {
            // Get label and confidence
            final String prediction = labels[argmax];
            final String confidence = String.format("%.2f", max);

            // Send prediction back to MainActivity
            Intent broadcastIntent = new Intent();
            String data = prediction + "," + confidence + "," + LocalTime.now() + "," + db(sData) + "," + recordTime;
            Log.i( tag: "TEST", data);


            if(prediction.equals("void")) {
                inferring = false;
                broadcastIntent.setAction("mBroadcastSoundPrediction");
                broadcastIntent.putExtra( name: "INFERENCE_DATA", data);
                mContext.sendBroadcast(broadcastIntent);

                Log.i( tag: "INFERENCIA",  msg: "Audio detectado correctamente");
                recordingDetection = true;
            }
        }
    }
```

Figura 3.15: Implementación algoritmo detección micción completo 2a parte

Es importante comentar, que es necesario realizar un ajuste en los buffers de grabación, ya que la longitud del audio no es la misma que en la primera fase de la ejecución. Una vez que se ha grabado el

resto de la toma, se procede a la detención de la grabación, y de la conversión de formatos de audio.

De esta manera, se puede completar el flujo de trabajo del algoritmo de detección de micción, que difiere con el representado en el informe inicial de los autores[], debido a las limitaciones físicas del dispositivo donde se ejecuta, pero que ha sido adaptado y sigue cumpliendo de manera satisfactoria.

3.5.2. Pruebas finales y informe de la autonomía de la batería

Además, en esta última iteración se realizarán algunas pruebas finales, sobretodo enfocadas en el rendimiento final del modo de detección automática de micción, que en una primera instancia no son del todo satisfactorias, y que se detallarán en un capítulo posterior. Por este motivo, tras discutirlo con los tutores, se opta por comunicarse con la Universidad de Deusto solicitando una nueva versión del algoritmo, que mejore las prestaciones actuales del mismo.

Finalmente, la nueva versión no llega a tiempo, debido a problemas logísticos y de fechas por parte de la Universidad de Deusto, con lo que la aplicación permanece con la versión actual, a la espera de una posible actualización en el futuro.

3.5.3. Optimizaciones y realización de la memoria

Por último, se revisa el código realizando pequeñas mejoras en cuanto a la legibilidad y organización del mismo, prescindiendo del código que se ha eliminado, así como haciendo que sea más sencillo de comprender. Se realiza también la redacción final de la memoria, partiendo como base de todas las anotaciones que se han ido consultado a lo largo de las iteraciones realizadas.

Esta última iteración se estima en 3 semanas, que finalmente se ampliará en 1 más, debido a la espera por la nueva versión, que finalmente no se puede implementar, así como algunas dificultades surgidas en la implementación final del modo de detección automática.

Capítulo 4

Estado final de la aplicación

En esta sección detallaremos cual es el estado final de la aplicación, dando una visión del proyecto a alto nivel, para poder conocer cual es su arquitectura, cómo está estructurado del proyecto, y alguna información significativa.

Este proyecto tiene como objetivo la depuración y evaluación de una aplicación ya existente, Urosound, que ha sido desarrollada por parte del instituto de Tecnología de la Universidad de Deusto, en primera instancia, y automatizada posteriormente en otro Trabajo de Final de Grado [31], por parte del alumno Martín Ordóñez Vivó en el año 2023.

Es por ello que, en las primeras semanas del proyecto, fue fundamental la realización de un proceso de ingeniería inversa, donde se analizó la estructura y la funcionalidad existente en la aplicación, de forma que se asimile de manera correcta cómo opera la aplicación internamente.

4.1. Análisis

En la fase de análisis determinaremos y recopilaremos los requisitos del cliente. Es necesario que los objetivos del proyecto, así como los usuarios finales, los casos de uso y requisitos del sistema puedan ser comprendidos en manera satisfactoria. En este apartado no se comentarán todos los aspectos de la aplicación, ya que hay funcionalidad y requisitos en los cuales no se ha intervenido, por lo que se utilizarán como referencia los aportados por el alumno Martín Ordóñez Vivó, que serán debidamente referenciados.

Con respecto a los requisitos funcionales, que es la funcionalidad que se quiere representar, y los requisitos no funcionales, que únicamente consisten en los aspectos y restricciones aplicadas sobre los funcionales, se ha realizado la tabla X, donde se representan los que han sido intervenidos en este proyecto. De igual manera, también se incorporaran los requisitos funcionales de información, donde se detallan cómo se va a procesar el manejo de la información y los datos en el sistema.

Requisitos Funcionales	
RF01	La aplicación permitirá el uso de controles táctiles y físicos para grabar y parar la grabación
RF02	La aplicación permitirá la activación automática de micción mediante un reconocedor de voz integrado
RF03	La aplicación permitirá la ejecución del algoritmo de detección de micción de manera satisfactoria
RF04	La aplicación permitirá el envío de los datos recogidos a un servidor externo
Requisitos No Funcionales	
RNF01	La aplicación realizará una escucha activa de 10 segundos para poder detectar si se produce micción o no.
RNF02	La aplicación debe permitir que tras la activación por detección de micción, grabe 120 segundos como micción estándar.
RNF03	La aplicación podrá ejecutarse en la versión más actualizada de WearOS, basada en Android 13
Requisitos Funcionales de Información	
RFI01	La aplicación guardará los audios en formato WAV para su análisis en el servidor externo

Tabla 4.1: Tabla de Requisitos Funcionales, No Funcionales y Funcionales de Información

4.1.1. Casos de uso

El diagrama de casos de uso representa la funcionalidad y visión de alto nivel del sistema, indicando las acciones realizan cada uno de los actores involucrados en la aplicación. Se utilizará el diagrama indicado en el trabajo de automatización descrito como referencia, en la figura 4.1.

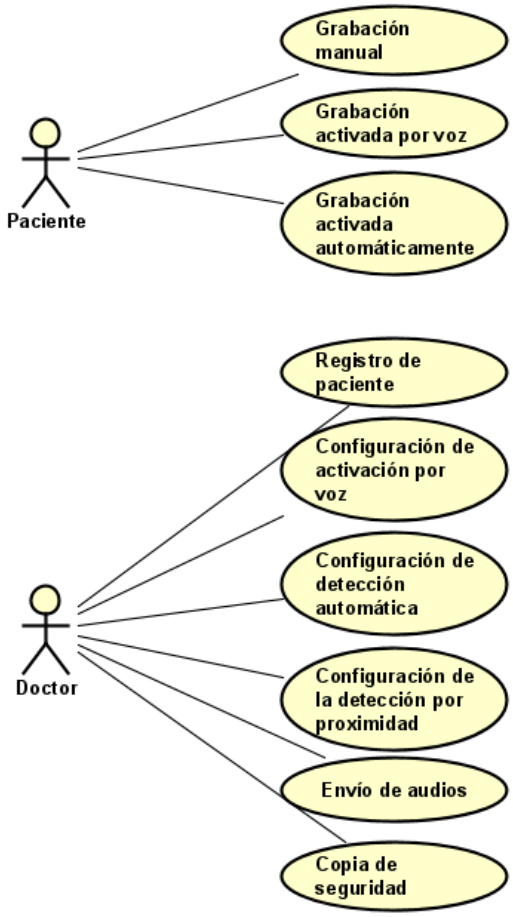


Figura 4.1: Diagrama de casos de uso de la aplicación [31].

4.2. Diseño

4.2.1. Diagrama de paquetes

El diagrama de la figura 4.2 permite visualizar los distintos paquetes de la aplicación, así como la dependencia y relaciones entre los mismos. Se utiliza como referencia el propuesto en el anterior trabajo de automatización.

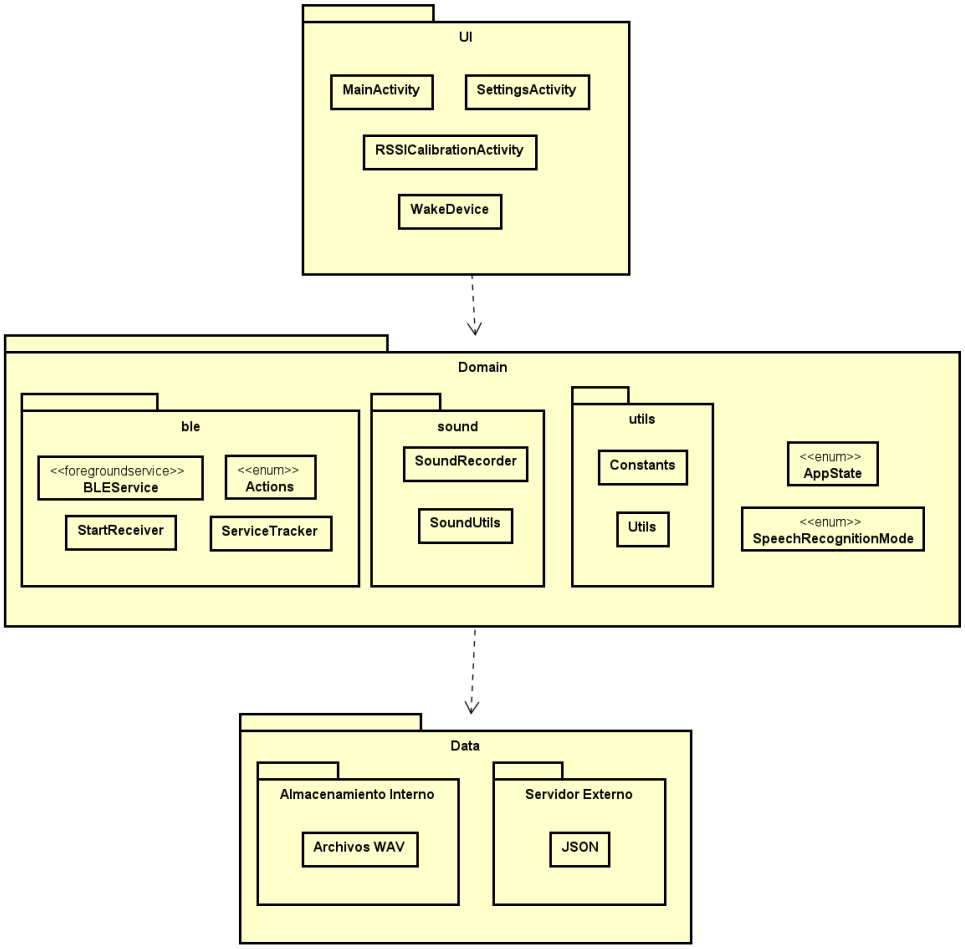


Figura 4.2: Diagrama de paquetes de la aplicación [31].

4.2.2. Diagrama de despliegue

El diagrama de despliegue es una representación de cómo se distribuye en un entorno real los dispositivos físicos, con respecto a los entornos donde se ejecutan. Se puede ver en la figura 4.3.

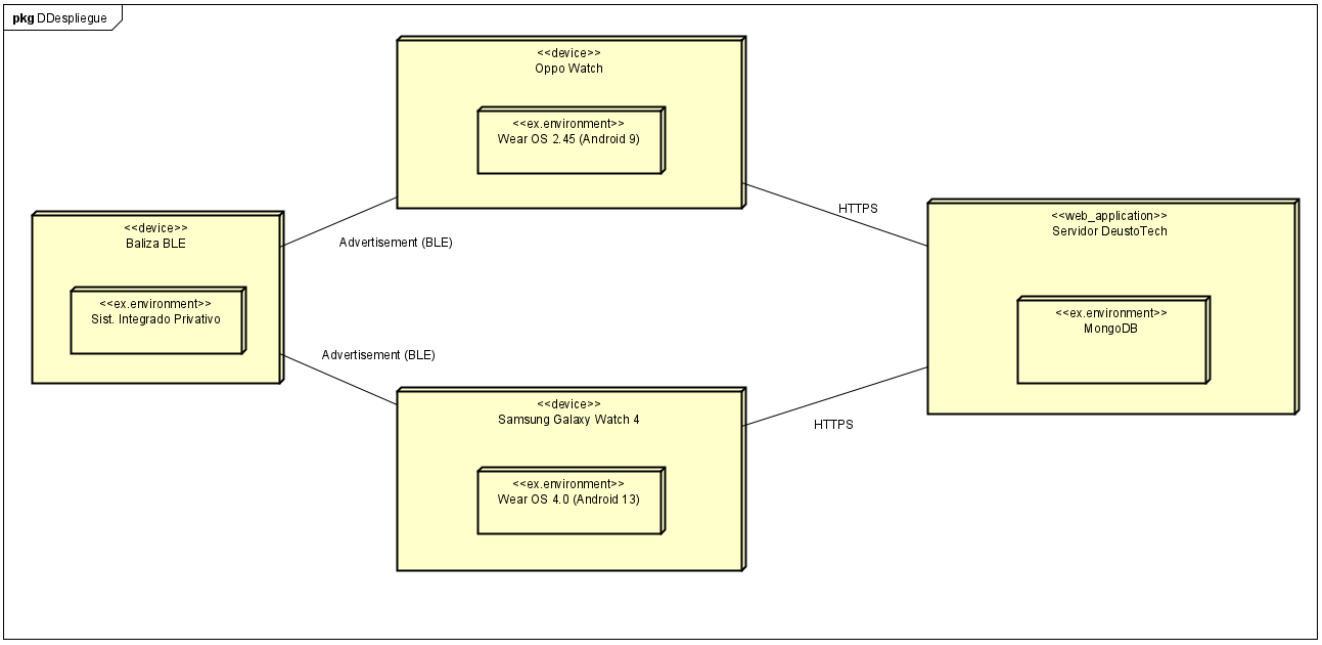


Figura 4.3: Diagrama de despliegue de la aplicación [31].

4.3. Arquitectura de la aplicación

Urosound consiste en una aplicación Android, la cual permite la realización de tomas flujométricas desde un dispositivo wearable, que se trata de un smartwatch. De esta manera, la aplicación está compuesta por numerosos componentes, que describiremos a continuación:

- **Actividades:** son el componente principal de la interfaz gráfica, que se encarga de la representación la interfaz al usuario. En el caso de nuestra aplicación, contamos con 3 actividades.
- **Servicios:** componentes que se ejecutan en segundo plano, y que se encargan de controlar acciones ocultas para el usuario. En Urosound contamos con un servicio que se encarga del control de las balizas BLE que se conectan con la aplicación.
- **Receptores de emisión:** componente destinado a detectar o reaccionar ante eventos que se generan en el sistema. El comportamiento del Bluetooth, así como el reconocimiento de señales transmitidas por la baliza BLE.

La aplicación graba el audio de la micción mediante el smartwatch, y una vez almacenado en el dispositivo, es enviado vía Wi-Fi a un servidor remoto, donde se almacenará en un base de datos MongoDB. Para la correcta visualización y análisis de las muestras recogidas, se utiliza una aplicación web. Podemos observar el proceso en la figura 4.4, obtenida de uno de los informes de los autores de la aplicación original [3].

Aunque la arquitectura no ha sido descrita de manera exacta en la documentación del proyecto, podemos determinar que se trata de una arquitectura de tipo clean [50] [51], debido a su estructura de

Figura 4.4: Diagrama de flujo de la aplicación [3].

capas, que se han reflejado en la figura 4.2. Este tipo de arquitectura es una de las bases a la hora de realizar una implementación de una aplicación en Android, y trasciende más allá del propio concepto.

Consiste en la utilización de 3 capas básicas, Domain, Data y Presentation, y cuya finalidad principal es ocultar los detalles de la implementación a la lógica de dominio de la aplicación. Así, se mantiene la lógica aislada, lo que hace que la aplicación en su conjunto sea más mantenible y escalable en el tiempo. Podemos ver un esquema de capas más amplio de este tipo de arquitecturas en la figura 4.5.

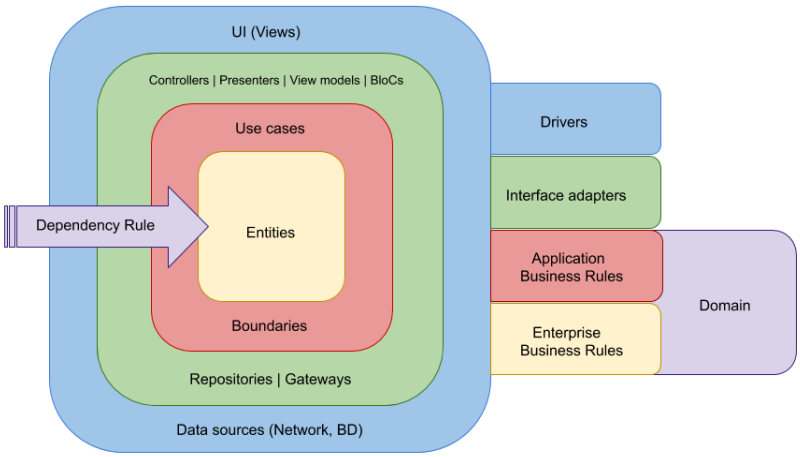


Figura 4.5: Diagrama de capas de la arquitectura clean [51].

Debe cumplir con la regla de dependencia, la cual nos restringe que las capas internas de la aplicación puedan conocer detalles de las capas exteriores. Sin embargo, las capas exteriores si pueden conocer aspectos concretos de las capas interiores. Esto ocurre porque el dominio es algo que se va a mantener con menos cambios, mientras que la implementación si que puede evolucionar en el tiempo.

Como principales ventajas, destacan el uso independiente de frameworks, que pueden ser utilizadas de manera sencilla, así como su alto grado de testing. La clave es que se pueden alterar las interfaces, o la implementación de la base de datos, sin que eso afecte al funcionamiento global de la aplicación, debido a la regla de dependencia que hemos comentado con anterioridad.

4.4. Estructura del proyecto

Con respecto a la estructura del proyecto, se ha respetado la configuración por defecto que nos ofrece el entorno de desarrollo utilizado, Android Studio, y la cual está representada en la figuras 4.6, 4.7, 4.8 . Describimos los aspectos más relevantes de esta estructura:

- **Manifiesto Android:** es el archivo XML que contiene la definición sobre los aspectos más importantes de la aplicación, como son su identificación, sus componentes, o los permisos de ejecución.
- **Código fuente de la aplicación:** son los archivos que describen la lógica y el funcionamiento de toda la aplicación. En nuestro caso, están escritos mayoritariamente en Kotlin conjunto con algún uso puntual de Java.
- **Directorio assets:** aquí se encuentran los archivos asociados con herramientas externas que se implementan en la aplicación, como son las referencias del reconocedor de voz Vosk, y el modelo de TfLite.
- **Directorio resources:** es donde se encuentran todos los archivos de tipo visual y gráfico de la aplicación. Se describen distribuidas en distintas carpetas, en función del tipo de archivo que se almacena, como por ejemplo, la carpeta "drawable", que es donde se encuentran los iconos y recuersos visuales de la aplicación.
- **Módulo models:** aquí se reflejan los archivos relacionados con la ejecución del reconocedor de voz de Vosk, tanto en idioma castellano como en idioma inglés.
- **Scripts Gradle:** son los archivos que contienen la información necesaria para la compilación del proyecto. Pueden convivir varios a la vez, que harán referencia a los distintos niveles de proyecto o de módulo.

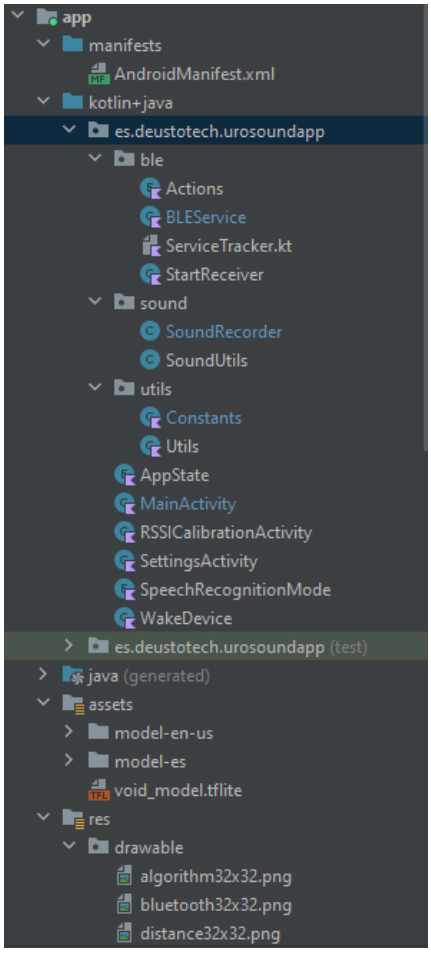


Figura 4.6: Estructura del proyecto pt.1

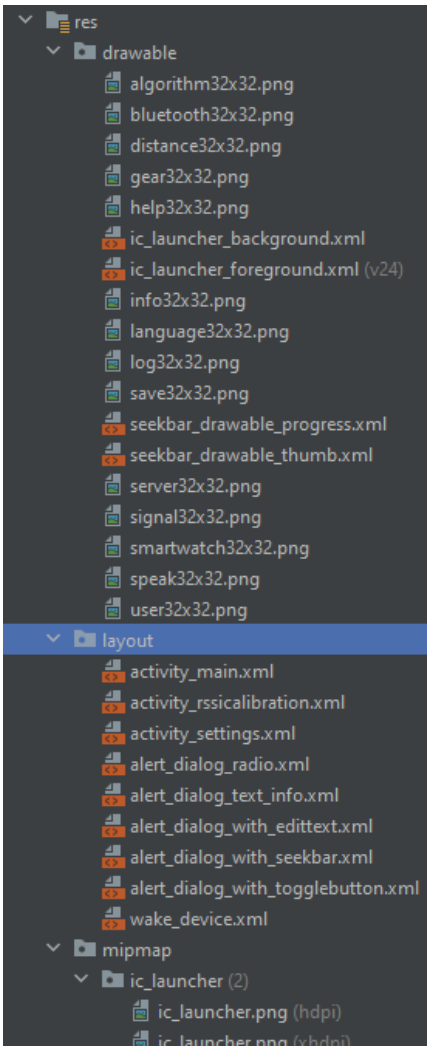


Figura 4.7: Estructura del proyecto pt.2

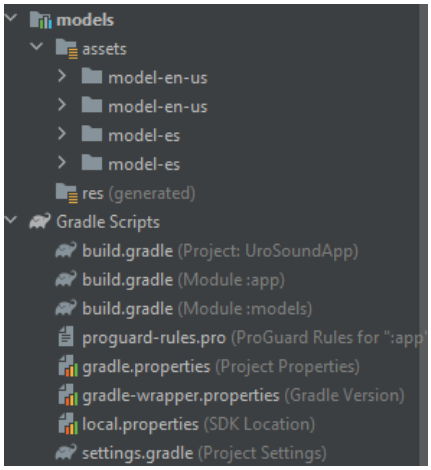


Figura 4.8: Estructura del proyecto pt.3

Capítulo 5

Pruebas

En este capítulo se describirán las pruebas realizadas, para verificar que el funcionamiento del proyecto es el correcto, así como si se cumplen los requisitos de calidad adecuados para su aceptación.

Las pruebas de software [52] son el proceso que evalua y verifica que una aplicación software funciona de la manera que debería de hacerlo. Esto permite que se puedan prevenir errores, así como mejorar el estado general de la aplicación y de su funcionamiento.

Existen muchos tipos de pruebas [53], pero nos centraremos en describir las que más relevantes con respecto a un sistema software genérico:

- **Pruebas funcionales:** comprueba que las funciones del software cumplen con el comportamiento del sistema, sin tener en cuenta su implementación. Se centran en aspectos de usabilidad y enfoque hacia el mercado.
- **Pruebas unitarias:** verifican que cada mínima unidad de software implementada funciona como debería. Son de las primeras en realizarse, ya que debían de hacerse cada vez que se implementa un bloque de código.
- **Pruebas de aceptación:** sirven para verificar que el proyecto funciona de la manera planificada. Se deben definir junto con el equipo responsable antes del inicio del proyecto.
- **Pruebas de integración:** comprueba que los diferentes componentes son capaces de operar juntos correctamente. Es complejo, ya que puede generar numerosos errores al realizarse con todos los componentes acoplados, y deben realizarse tras las pruebas unitarias.
- **Pruebas de estrés:** enfocadas en verificar en que momento exacto o bajo que condiciones el sistema se satura y colapsa.
- **Pruebas de rendimiento:** verifican la respuesta del software ante cargas de trabajo exigentes, y sobre condiciones reales, midiendo aspectos como la estabilidad o la velocidad de la aplicación.

- **Pruebas de usabilidad:** determinan la facilidad de uso de una aplicación software, verificando aspectos como si es intuitiva o la comodidad en el uso y manejo.

- **Pruebas de regresión:** comprueban que los cambios en un componente software provocan reacciones inesperadas o fallos en otros puntos de código.

5.1. Pruebas enfocadas en una aplicación para dispositivo móvil

En el caso de Urosound, al tratarse de una aplicación enfocada a ser utilizada en un dispositivo wearable, muchas de las pruebas que se realizan de manera estándar no son aplicables, ya que se trata de un contexto de utilización distinto, donde ciertos tipos de pruebas no tienen demasiada razón de aplicarse.

Primeramente, **las pruebas unitarias** carecen de sentido en nuestra aplicación, ya que partimos de que ya está implementada al completo, por lo que no sería viable ir probando cada pequeño trozo funcional del código. De igual manera, las **pruebas de aceptación** irían enfocadas a una implementación futura, con vistas a que la aplicación saliese al mercado y se realizase en una fase de producción, que no es la situación de este proyecto.

Tampoco se tendrán en consideración **pruebas de estrés**, ya que el objetivo prioritario es buscar una funcionalidad correcta, sin descuidar el rendimiento pero sin forzar al máximo la aplicación.

En nuestro caso, se realizarán **pruebas funcionales**, buscando que el sistema opere de forma correcta con toda la funcionalidad aplicada. La idea es realizarlas de **caja gris** [54], ya que mezclaremos pruebas de caja blanca, donde se conoce el sistema que se está probando, así como pruebas de caja negra, donde no se conoce cómo funciona el sistema en su interior, ni su implementación. Por otro lado, también se detallarán algunas **pruebas no funcionales**, relacionadas con el rendimiento y la usabilidad, así como un análisis de la autonomía de la batería con el uso de los distintos modos.

Por último, aunque no se profundizará con detalle, se realizarán **pruebas de regresión** que evaluarán si los cambios que se han aplicado sobre el sistema, y en funcionalidades que no son correctas permiten que se realicen de manera satisfactoria.

Se detallarán a continuación las pruebas realizadas, así como su resultado y su conclusión. En caso de que la conclusión sea negativa o el desempeño de la prueba haya sido incorrecto, la corrección para cada problemática obtenida ha sido descrita en el capitulo 3, en su respectiva iteración.

5.2. Pruebas preliminares

Se realizan unas pruebas preliminares sobre aspectos comunes de una aplicación móvil, para verificar que la aplicación cumple con la funcionalidad mínima.

Nombre	Instalación de la aplicación
Descripción	Se realiza la instalación de la aplicación, siguiendo el manual de despliegue descrito en la memoria del trabajo. Es posible visualizar el icono de la aplicación.
Resultado	Se realiza de manera satisfactoria
Conclusión	No existe ningún problema con respecto a la instalación, siempre que se siga con el manual de despliegue previsto.

Tabla 5.1: Prueba de instalación de la aplicación

Nombre	Comportamiento inicial de la aplicación
Descripción	Se evalúa el comportamiento de la aplicación tras su instalación, verificando que es posible interactuar por los distintos menús sin problemas.
Resultado	El comportamiento de la aplicación es correcto, pudiendo desplazarse por ella de manera exitosa. Es algo dificultoso en ocasiones acceder al menú oculto, pero no es nada importante.
Conclusión	La aplicación se comporta de manera satisfactoria, con la salvedad de lo comentado con respecto al menú oculto, que no es problemático, con lo que no se realizan acciones al respecto.

Tabla 5.2: Prueba del comportamiento de la aplicación

Nombre	Validación de permisos en tiempo de ejecución
Descripción	Se evalúa el comportamiento de la aplicación en función de si se aceptan o no los permisos que se solicitan al abrir por primera vez la aplicación. En concreto • **Smartwatch Oppo:** se solicitan permisos de almacenamiento, micrófono y ubicación. En caso de tener el Bluetooth desactivado, también el de Bluetooth. • **Smartwatch Samsung Galaxy Watch 4:** se solicitan permisos de micrófono y ubicación. En caso de tener el Bluetooth desactivado, también el de Bluetooth.
Resultado	La validación se realiza de forma correcta. En la situación, en la cual no se aceptan los permisos, se accede a la aplicación con normalidad, pero con las funciones limitadas. No se espera que el paciente inicie la aplicación por primera vez sin aceptar los permisos.
Conclusión	La aplicación se comporta de manera satisfactoria cuando se aceptan los permisos, en caso contrario, con funciones limitadas y algún cierre inesperado (no se espera esta actuación)

Tabla 5.3: Prueba de la validación de los permisos

Nombre	Desinstalación de la aplicación
Descripción	Se realiza el cierre de la aplicación y posteriormente se desinstala, verificando que no aparece en el dispositivo, y que la memoria de la misma se libera.
Resultado	La aplicación se desinstala de manera correcta
Conclusión	No hay ningún problema a la hora de desinstalar la aplicación.

Tabla 5.4: Prueba desinstalación de la aplicación

5.3. Pruebas funcionales

5.3.1. Pruebas de los modos de la aplicación

Se realizan unas pruebas donde se pueda verificar el funcionamiento exacto del sistema, con sus respectivos modos. La idea es observar en el que punto está la aplicación en su primera fase, para aplicar la refactorización y corrección que sea necesaria para su correcto funcionamiento.

Nombre	Modo básico botones
Descripción	Se realiza una grabación de una micción interactuando con el reloj. Existen 2 maneras de interactuar con el dispositivo: • Mediante la propia interfaz de la aplicación, de manera táctil. • Mediante el uso de los botones, cuando se ha accedido a la aplicación, con una pulsación sobre el botón de función comienza la grabación, y con otra pulsación en el mismo botón se detiene la grabación.
Resultado	En el smartwatch Oppo, se realiza la grabación pero se obtiene un error a la hora de guardar el audio, mismo comportamiento que se sucede en el Samsung Watch 4.
Conclusión	Es necesaria la revisión de la función de guardado de las micciones realizadas con el smartwatch, sobretodo en el aspecto relacionado con los permisos.

Tabla 5.5: Prueba modo básico botones

Nombre	Modo reconocedor de voz
Descripción	Se realiza una grabación de un audio de manera semi-autónoma, con la detección de la baliza interactuando con el reloj. La idea es que la baliza, situándose encendida y colocada en el lugar donde se va a realizar la micción (aseo), detecte la presencia del reloj cuando esté próximo a él, y de esta manera habilite el reconocedor de voz. Esto permitirá que tras pronunciar la palabra clave, se active la grabación de la micción.
Resultado	En ninguno de los 2 relojes el reconocedor carga correctamente, tanto el de Vosk como el de Android Nativo.
Conclusión	Se necesaria la revisión de la implementación del reconocedor de voz, en busca de posibles fallas o puntos de ruptura.

Tabla 5.6: Prueba modo reconocedor voz

Nombre	Modo algoritmo deep-learning
Descripción	Se realiza una grabación de un audio de manera autónoma, con un algoritmo de detección deep learning que es capaz de procesar la muestra de audio que se está grabando a través del micrófono, y reconocer si es o no una micción. Para activar el algoritmo, se realizaría de igual manera que en el modo anterior, mediante proximidad a la baliza previamente encendida y colocada.
Resultado	En una primera prueba, simplemente no es funcional por los fallos de los permisos que se han comentado en los apartados anteriores.
Conclusión	Se necesaria la revisión de los permisos concedidos, así como del algoritmo propiamente dicho.

Tabla 5.7: Prueba modo algoritmo deep learning

Estas pruebas nos permiten tener una primera aproximación de cuales pueden ser los problemas a solucionar, y que posteriormente podamos analizar y realizara los cambios que sean necesarios.

5.3.2. Pruebas de las opciones de la aplicación

Además, se realizan pruebas al menú de opciones para verificar que cambios se pueden realizar en cuanto a las opciones de la aplicación:

Nombre	Id del paciente
Descripción	Se introduce el ID del paciente, de forma que en el servidor remoto se creará el perfil, para que se puedan almacenar sus micciones de manera controlada.
Resultado	Se obtiene un error al intentar registrar el ID de un paciente, en ambos relojes.
Conclusión	Es necesario revisar cómo se realiza el registro del paciente en el servidor externo, así como si sigue siendo accesible.

Tabla 5.8: Prueba de registro de paciente

Nombre	Reconocimiento de voz
Descripción	Permite cambiar el reconocedor de voz entre las 3 opciones disponibles: Deshabilitado, Android (Online) y Vosk (Offline)
Resultado	Se realiza el cambio de manera satisfactoria en ambos relojes.
Conclusión	Se puede realizar el cambio de manera satisfactoria, y se aplica correctamente.

Tabla 5.9: Prueba de cambio de reconocedor de voz

Nombre	Palabra para el reconocimiento de voz
Descripción	Permite cambiar la palabra clave que activará la grabación de la micción cuando el paciente esté próximo a la baliza.
Resultado	Se realiza el cambio de manera satisfactoria en ambos relojes.
Conclusión	Se puede realizar el cambio de manera satisfactoria, y se aplica correctamente.

Tabla 5.10: Prueba de cambio de palabra clave

Nombre	Filtrar por dirección MAC
Descripción	Permite filtrar la dirección MAC de la baliza a la que conectase. Si se introdujese, solo se podría conectar a esa dirección física, y en caso de dejarla vacía, permitirá conectarse a cualquier dispositivo sin limitaciones.
Resultado	Se puede filtrar exitosamente según la dirección MAC del dispositivo escogido.
Conclusión	No existe ningún problema con respecto al filtrado de MAC.

Tabla 5.11: Prueba de filtrado por dirección MAC

Nombre	Filtrar por Major
Descripción	De manera análoga al filtrado por MAC, permite filtrar el valor Major de la baliza a la que conectase. Si se introdujese, solo se podría conectar a ese dispositivo y en caso de dejarla vacía, permitirá conectarse a cualquier dispositivo sin limitaciones.
Resultado	Se realiza de manera satisfactoria en ambos relojes
Conclusión	No existe ningún problema con respecto al filtrado por Major

Tabla 5.12: Prueba de filtrado por Major

Nombre	Filtrar por Minor
Descripción	De manera análoga al filtrado por Major, permite filtrar el valor Minor de la baliza a la que conectase. Si se introdujese, solo se podría conectar a ese dispositivo y en caso de dejarla vacía, permitirá conectarse a cualquier dispositivo sin limitaciones.
Resultado	Se realiza de manera satisfactoria en ambos relojes
Conclusión	No existe ningún problema con respecto al filtrado por Minor

Tabla 5.13: Prueba de filtrado por Minor

Nombre	Distancia de activación BLE
Descripción	Permite calibrar la distancia de activación de la baliza BLE, en un rango mínimo de 0.5 metros a 5 metros. Cuanto más bajo sea el valor, eso implicará que será necesaria una menor distancia entre el smartwatch y la baliza para activarse.
Resultado	Se verifica que al ir alterando los valores en ambos relojes, activandose no en función de la distancia a la que nos ubiquemos de la baliza.
Conclusión	No existe ningún problema con respecto a la calibración de la distancia con la baliza

Tabla 5.14: Prueba de calibración de la distancia a la baliza BLE

Nombre	Calibrar RSSI a 1 metro
Descripción	Permite calibrar el RSSI (Received Signal Strength Indicator), que es la potencia de señal que se recibe de la baliza. En función de la distancia a la que estemos ubicados de la baliza, esa potencia puede ser modificada para realizar cambios en el comportamiento de la baliza y el smartwatch
Resultado	Se muestran el RSSI promedio de los últimas 10 medidas tomadas, así como la distancia en tiempo real que se obtiene hasta la baliza, sin ningún fallo, en ambos relojes
Conclusión	No existe ningún problema con respecto a la calibración del RSSI a 1 metro

Tabla 5.15: Prueba de calibración del RSSI a 1 metro

Nombre	Factor medioambiental
Descripción	Permite ajustar el valor del factor medioambiental, el cual está comprendido entre 2 y 4. Por recomendación del tutor D.º Alfonso Bahillo Martínez el valor por defecto es 2.4
Resultado	Se permite el cambio del valor del factor medioambiental, en ambos relojes
Conclusión	No existe ningún problema con respecto al cambio del valor del factor medioambiental

Tabla 5.16: Prueba de modificación de factor medioambiental

Nombre	Algoritmo de reconocimiento de micción
Descripción	Se permite cambiar el algoritmo de reconocimiento de micción, entre TFLite, que es el que se utiliza por defecto, así como Yamnet.
Resultado	Es posible cambiar entre ambos reconocedores, de manera sencilla, en ambos relojes
Conclusión	No existe ningún problema con respecto al cambio de algoritmo de reconocimiento de micción

Tabla 5.17: Prueba de cambio de algoritmo de reconocimiento de micción

Nombre	Activar pantalla si el Bluetooth se suspende
Descripción	Permite que la pantalla se active en caso de que el Bluetooth se suspenda, que solo se recomienda en casos en los que el comportamiento del Bluetooth en el dispositivo sea errático.
Resultado	Se puede activar la opción sin errores en ambos relojes
Conclusión	No existe ningún problema con respecto al cambio en la activación de la pantalla

Tabla 5.18: Prueba de activación de la pantalla por suspensión de Bluetooth

Nombre	Enviar audios al servidor
Descripción	Se realiza el envío de los audios almacenados en el almacenamiento del dispositivo, con respecto al paciente que los ha realizado.
Resultado	En ambos relojes, al intentar realizar la subida, en caso de que el paciente haya grabado los audios sin registrarse en el smartwatch, obtendremos un aviso para que lo hagamos. No es funcional ya que el registro del paciente no funciona correctamente.
Conclusión	Es necesario revisar el registro del paciente en la aplicación web de Urosound, así como verificar si el acceso es correcto.

Tabla 5.19: Prueba de subida de audios al servidor

Nombre	Crear copia de seguridad local
Descripción	Se realiza la copia de seguridad de los archivos de audio de la memoria de la aplicación a la memoria del dispositivo. En caso de que no existise, se obtiene un mensaje informativo.
Resultado	Se realiza de manera satisfactoria en el smartwatch Oppo, pero en el smartwatch Samsung no funciona debido a un problema de permisos de almacenamiento
Conclusión	Es necesario revisar la gestión de los permisos con respecto al smartwatch de Samsung

Tabla 5.20: Prueba de copia de seguridad local

Nombre	Habilitar/Deshabilitar logs
Descripción	Se permite habilitar o deshabilitar los logs, de manera que se pueda obtener más informacion a la hora de depurar.
Resultado	Se realiza de manera satisfactoria en ambos dispositivos, dejando activar o desactivar esta opción.
Conclusión	No existe ningún problema con respecto a habilitar o deshabilitar logs

Tabla 5.21: Prueba de habilitar o deshabilitar los logs

Nombre	Idioma
Descripción	Se permite realizar un cambio en el idioma de la aplicación entre el inglés y el español al realizar un cambio en el idioma del dispositivo
Resultado	Se realiza de manera satisfactoria si se cambia el idioma a ambos dispositivos al inglés
Conclusión	No existe ningún problema con respecto al cambio de idioma

Tabla 5.22: Prueba de cambio de idioma

Nombre	Ayuda
Descripción	Se muestra una breve descripción de la aplicación y su funcionalidad
Resultado	Se visualiza de manera satisfactoria en ambos dispositivos
Conclusión	No existe ningún problema con respecto a la opción de ayuda

Tabla 5.23: Prueba de opción de ayuda

Nombre	Restaurar valores por defecto
Descripción	Permite realizar una restauración de los valores por defecto de la aplicación, si se realiza un borrado de la caché y los datos de la aplicación.
Resultado	Se realiza de manera satisfactoria en ambos dispositivos
Conclusión	No existe ningún problema con respecto a la instalación, siempre que se siga con el manual de despliegue previsto.

Tabla 5.24: Prueba de restauración de valores por defecto

Nombre	Acerca de
Descripción	Se muestra la información de la aplicación
Resultado	Se visualiza de manera satisfactoria en ambos dispositivos
Conclusión	No existe ningún problema con respecto a la opción de acerca de

Tabla 5.25: Prueba de opción de acerca de

5.4. Pruebas no funcionales

Se realizan pruebas no funcionales sobre la aplicación, que permitirán verificar aspectos que no entran dentro de los requisitos exigidos por el sistema. Aquí nos centraremos en comentar aspectos relacionados con el rendimiento y la usabilidad del sistema.

5.4.1. Pruebas de rendimiento

En cuanto a las pruebas de rendimiento, no se ha realizado un análisis exhaustivo, ya que la búsqueda de un rendimiento máximo de la aplicación no se considera un objetivo primordial con respecto al proyecto. Los tiempos de carga, así como los de espera de las detecciones de la baliza y de la ejecución del algoritmo de deep learning están dentro de los valores estándar para un aplicación desarrollada para un entorno de smartwatch.

De igual manera, la aplicación es capaz de realizar notificaciones y mensajes mostrando el estado de las operaciones que está realizando, de manera precisa y rápida. En el apartado de conexiones, la conexión mediante Bluetooth en ambos dispositivos ha sido siempre efectiva y sin errores, así como la conexión a internet requerida para la subida de archivos al servidor, que al tratarse de la del propio estudiante, no ha experimentado problemas.

Por otro lado, se ha evaluado el rendimiento del algoritmo de detección de micción, de manera menos exhaustiva que la realizada en el informe preliminar de los tutores de la aplicación básica [55]. En total, se han realizado un total de 50 audios de prueba sobre el algoritmo, que detallaremos en la tabla 5.26.

Tipo de audio aplicado	Fuente	# samples
Silence	Aplicación propia	18
Pump (liquid)	Audioset	2
Sink (filling or washing)	Audioset	2
Squish	Audioset	2
Splash, splatter	Audioset	2
Toilet Flush	Audioset ECS50	2
Fill (with liquid)	Audioset	2
Pour	ECS50	2
Slosh	Audioset	2
Drip	Audioset	2
Stir	Audioset	2
Spray	Audioset	2
Water tap	Audioset	2
Boiling	Audioset	2
Bathtub	Audioset	2
water drops	ESC-50	2
brushing teeth	ESC-50	2

Tabla 5.26: Tabla de ejemplos de clases de audio, sus fuentes y número de muestras

Para el uso de muestras de audio, se han utilizado 2 herramientas, como son AudioSet y ESC-50. AudioSet [56] es una ontología que consiste en la colección de más de 600 clases de eventos de audio y más de 2 millones de clips de audio de 10 segundos obtenidos de Youtube, de los cuales se extraen algunas muestras aleatorias para realizar pruebas. Con respecto a ESC-50 [57], se trata de una colección de 50 clases de 2000 muestras de audio ambiental de 5 segundos de duración. Todas son extraídas de un proyecto de libre acceso, Freesound.org [58]. De igual manera, con respecto a los audios en silencio, se han utilizado los propios smartwatches para grabar dichos eventos, y luego han sido ejecutados con el algoritmo de detección. En la tabla 5.27 se pueden comprobar los resultado obtenidos tras la ejecución de los audios en el algoritmo de detección:

	Exactitud	# samples erróneos
Exactitud total audios	76 %	12
Falsos positivos	66 % de los erróneos	8
Falsos negativos	33 % de los erróneos	4

Tabla 5.27: Tabla de ejemplos de clases de audio, sus fuentes y número de muestras

Con respecto a los resultados obtenidos, han sido bastante satisfactorios, dentro de que la muestra realizada es pequeña debido a que la eficiencia del algoritmo de detección no corresponde a una de los objetivos del estudiante, que solo se centra en su correcta implementación en la aplicación. Se puede observar un pequeño resumen de las pruebas realizadas. Si que se pueden observar más problemas a la hora de detectar correctamente los silencios, pero el resto de audios procesados en ambos relojes obtienen unos valores aceptables. Hay mayor número de falsos positivos, es decir, que el algoritmo detecta que hay micción cuando no la hay, que falsos negativos.

Cabe indicar que el proceso de evaluación del algoritmo de detección no se realizado en un entorno

con las condiciones ideales como podrían ser las de un laboratorio especializado en acústica o un sala de sonido especial. Al haber sido realizadas en un entorno más cotidiano, es posible que los resultados obtenidos difieran de los examinados previamente, afectando en la precisión o exactitud del algoritmo.

A continuación comentaremos algunos aspectos de la autonomía del dispositivo con respecto al uso de la aplicación.

5.4.2. Análisis de la autonomía de la batería

Con los distintos modos previamente detallados, se realizarán unas pruebas para determinar el impacto de cada uno de ellos, en el Oppo Watch y en el Samsung Galaxy Watch 4.

Para ello, detallaremos la horas de uso aproximadas de cada dispositivo en modo standby, con un uso normal y con las conexiones de Wi-Fi y Bluetooth activadas continuamente, un brillo de pantalla medio:

- **Oppo Watch (300mAh de batería):** aproximadamente 18-20 horas de autonomía
- **Samsung Galaxy Watch 4 (361 mAh de batería):** aproximadamente 28-30 horas de autonomía

5.4.2.1. Pruebas Oppo Watch

Nombre	Modo básico botones
Descripción	Se realizan 6 micciones de una duración de 120 segundos espaciadas a lo largo de 20 horas de medición. Se interactúa con normalidad en el reloj y se graban las flujometrías con el modo básico táctil o botones. Se realizan alternativamente las 2 maneras: • Mediante la propia interfaz de la aplicación, de manera táctil. • Mediante el uso de los botones, cuando se ha accedido a la aplicación, con una pulsación sobre el botón de función comienza la grabación, y con otra pulsación en el mismo botón se detiene la grabación. Se realiza la verificación del porcentaje de batería cada hora para observar la autonomía, que se comparará con el uso normal del dispositivo en ese intervalo de tiempo.
Resultado	• Batería uso normal: 20 horas (0 %) • Batería usando modo grabación: 18 horas (0 %)
Conclusión	Se observa que la duración de la batería al utilizar el modo de grabación básico ha disminuido en 2 horas con respecto al normal, valor aceptable ya que no se requiere de conexión con la baliza ni de utilización de ningún algoritmo de detección.

Tabla 5.28: Análisis batería a lo largo del tiempo modo básico

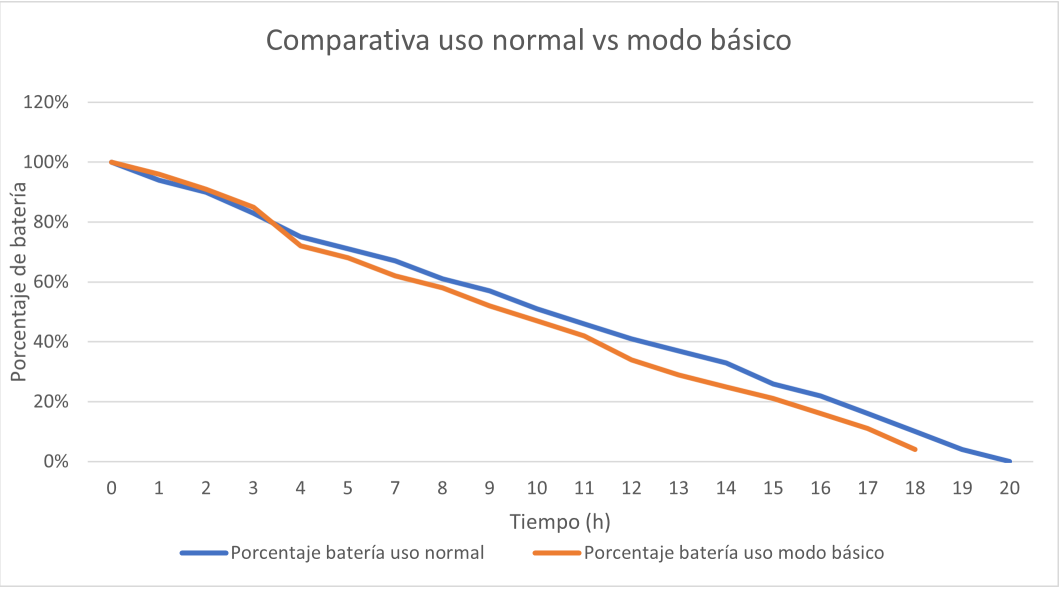


Figura 5.1: Gráfica análisis batería reloj Oppo modo básico.

Nombre	Modo reconocedor de voz
Descripción	Se realizan 6 micciones de una duración de 120 segundos espaciadas a lo largo de 20 horas de medición. Se interactúa con normalidad en el reloj y se graban las flujometrías de manera semi-autónoma, con la detección de la baliza interactuando con el reloj.
Resultado	• Batería uso normal: 20 horas (0 %) • Batería usando modo grabación: 15 horas (0 %)
Conclusión	Se observa que la duración de la batería al utilizar el modo de grabación del reconocedor ha disminuido en 5 horas la autonomía estándar. Sigue tratándose de un valor aceptable, ya que el reloj constantemente busca a través de Bluetooth la conexión con la baliza, por tanto hay un mayor consumo a la hora de ejecutar el reconocedor de voz.

Tabla 5.29: Análisis batería a lo largo del tiempo modo reconocedor voz

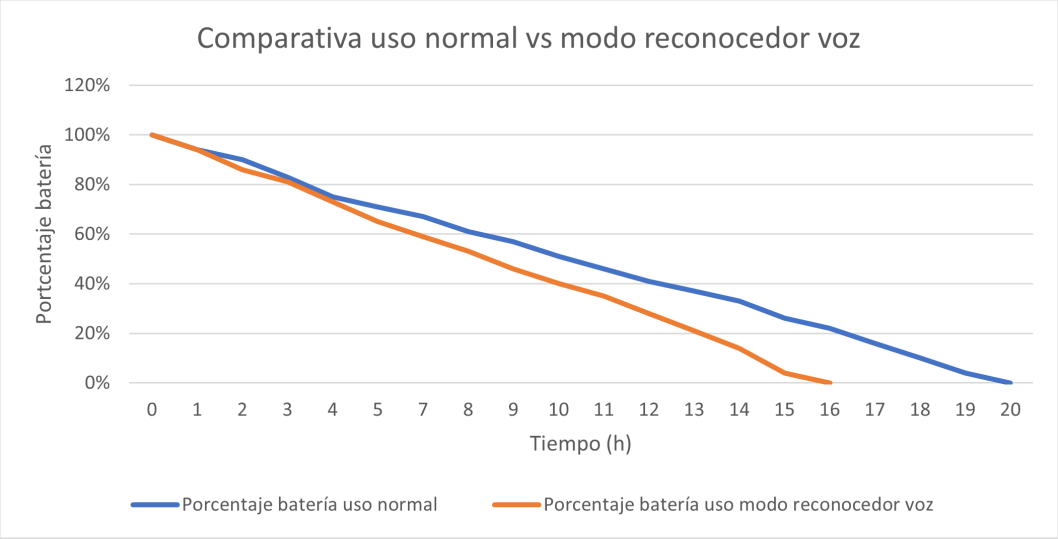


Figura 5.2: Gráfica análisis batería reloj Oppo modo reconocedor voz.

Nombre	Modo algoritmo deep-learning
Descripción	Se realizan 6 micciones de una duración de 120 segundos espaciadas a lo largo de 20 horas de medición. Se realiza la flujometría de manera autónoma, con un algoritmo de detección deep learning que es capaz de procesar la muestra de audio que se está grabando a través del micrófono, y reconocer si es o no una micción. Para activar el algoritmo, se realizaría de igual manera que en el modo anterior, mediante proximidad a la baliza previamente encendida y colocada.
Resultado	• Batería uso normal: 20 horas (0 %) • Batería usando modo grabación: 13 horas (0 %)
Conclusión	Se observa que la duración de la batería al utilizar el modo de grabación del reconocedor ha disminuido en 7 horas la autonomía estándar. La bajada de autonomía es bastante considerada, tratándose del 35 % de la autonomía total. A pesar de ello, se trata de un modo de autonomía completa, por lo que merece la pena la disminución de la autonomía ganando en experiencia de uso.

Tabla 5.30: Analisis batería modo algoritmo deep learning

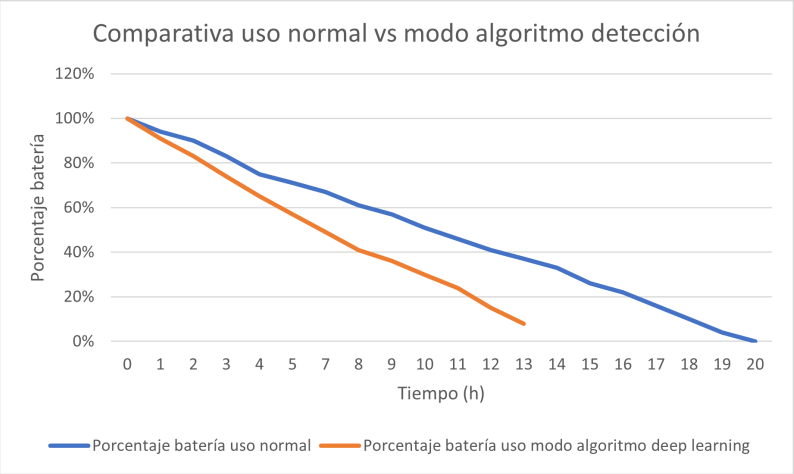


Figura 5.3: Gráfica análisis batería reloj Oppo modo algoritmo deep learning.

Podemos concluir que, los resultados de la autonomía con la aplicación implementada en el smartwatch Oppo son muy satisfactorios, ya que la autonomía se mantiene en un rango aceptable de horas, a pesar de que el consumo se eleva mucho sobretodo en los 2 últimos modos de uso debido al uso de sensores y algoritmos implementados.

5.4.3. Pruebas Samsung Galaxy Watch 4

Nombre	Modo básico botones
Descripción	Se realizan 6 micciones de una duración de 120 segundos espaciadas a lo largo de 20 horas de medición. Se interactúa con normalidad en el reloj y se graban las flujometrías con el modo básico táctil o botones. Se realizan alternativamente las 2 maneras: • Mediante la propia interfaz de la aplicación, de manera táctil. • Mediante el uso de los botones, cuando se ha accedido a la aplicación, con una pulsación sobre el botón de función comienza la grabación, y con otra pulsación en el mismo botón se detiene la grabación. Se realiza la verificación del porcentaje de batería cada hora para observar la autonomía, que se comparará con el uso normal del dispositivo en ese intervalo de tiempo.
Resultado	• Batería uso normal: 20 horas (32 %) • Batería usando modo grabación: 20 horas (17 %)
Conclusión	Se observa que la duración de la batería al utilizar el modo de grabación básico se ha mantenido en 20 horas, aunque ha bajado la batería al 17 %. Se trata de una situación muy favorable, ya que se sigue manteniendo la autonomía del dispositivo como si lo utilizásemos de manera normal.

Tabla 5.31: Análisis batería a lo largo del tiempo modo básico

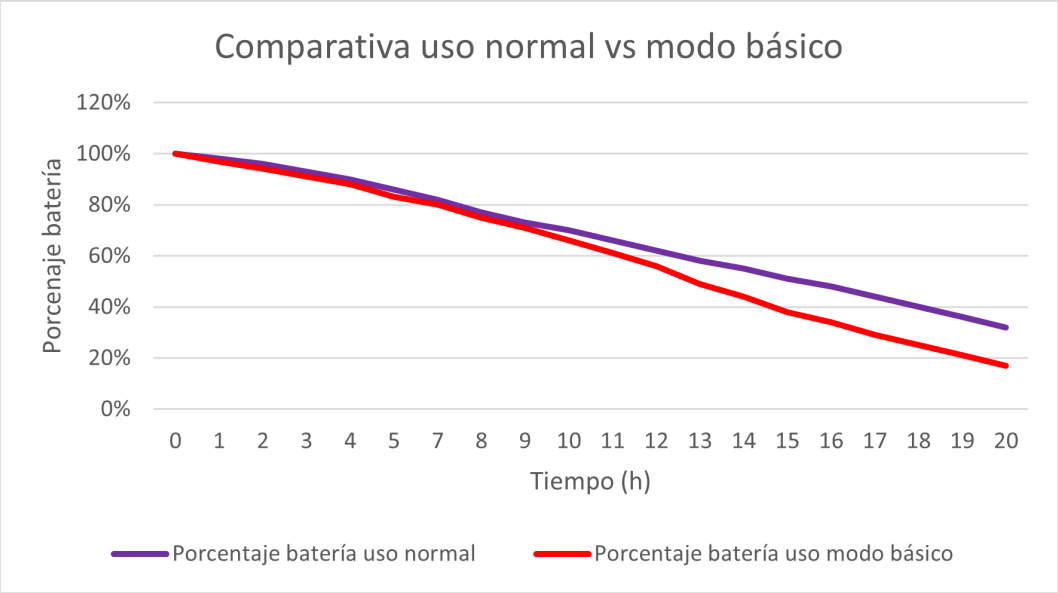


Figura 5.4: Gráfica análisis batería reloj Samsung modo básico.

Nombre	Modo reconocedor de voz
Descripción	Se realizan 6 micciones de una duración de 120 segundos espaciadas a lo largo de 20 horas de medición. Se interactúa con normalidad en el reloj y se graban las flujometrías de manera semi-autónoma, con la detección de la baliza interactuando con el reloj.
Resultado	• Batería uso normal: 20 horas (32 %) • Batería usando modo grabación: 18 horas (0 %)
Conclusión	Se observa que la duración de la batería al utilizar el modo de grabación del reconocedor de voz ha disminuido en 2 horas con respecto a la autonomía estándar. Se trata de un valor asumible, ya que el reloj constantemente busca a través de Bluetooth la conexión con la baliza, así como experimenta el mayor consumo a la hora de ejecutar el reconocedor de voz.

Tabla 5.32: Análisis batería a lo largo del tiempo modo reconocedor voz

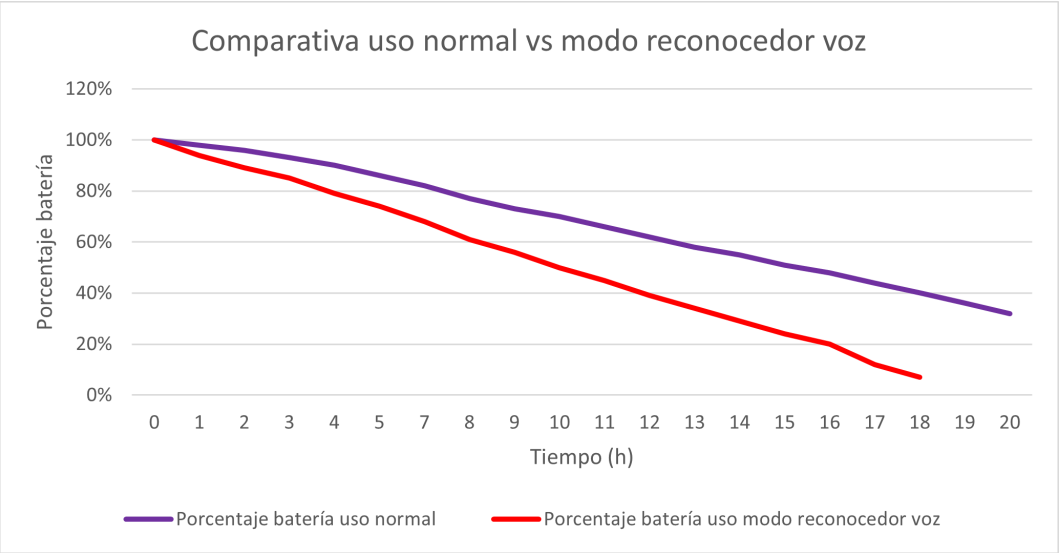


Figura 5.5: Gráfica análisis batería reloj Samsung modo reconocedor voz.

Nombre	Modo algoritmo deep-learning
Descripción	Se realizan 6 micciones de una duración de 120 segundos espaciadas a lo largo de 20 horas de medición. Se realiza la flujometría de manera autónoma, con un algoritmo de detección deep learning que es capaz de procesar la muestra de audio que se está grabando a través del micrófono, y reconocer si es o no una micción. Para activar el algoritmo, se realizaría de igual manera que en el modo anterior, mediante proximidad a la baliza previamente encendida y colocada.
Resultado	• Batería uso normal: 20 horas (32 %) • Batería usando modo grabación: 15 horas (0 %)
Conclusión	Se observa que la duración de la batería al utilizar el modo de grabación del reconocedor ha disminuido en 5 horas la autonomía estándar. Se trata de una disminución de la batería a tener en cuenta, pero bajo las condiciones en las que se está realizando, es bastante realista con respecto al desempeño que tiene este modo y su alto consumo.

Tabla 5.33: Análisis batería modo algoritmo deep learning

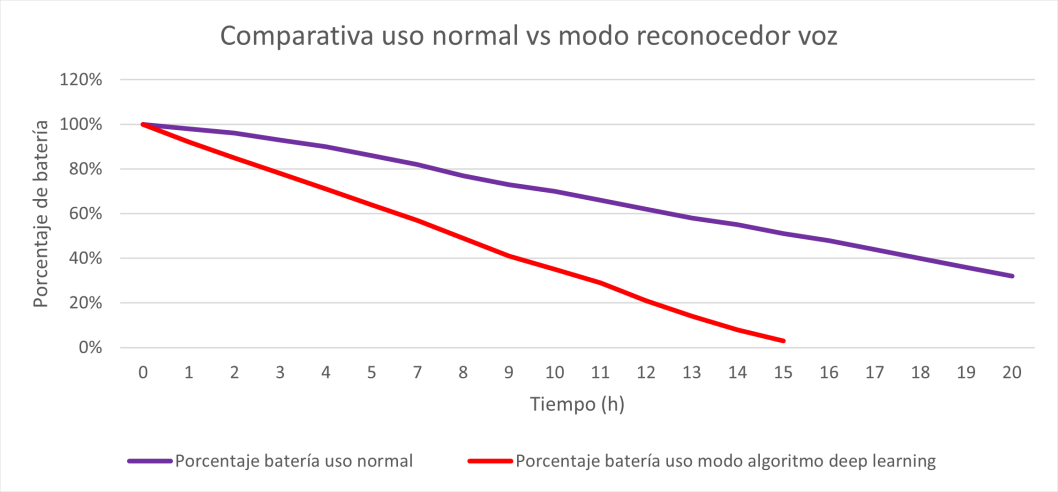


Figura 5.6: Gráfica análisis batería reloj Samsung modo algoritmo deep learning.

Por parte del smartwatch Samsung Galaxy Watch 4, los resultados obtenidos son sorprendentes, ya que mejoran la autonomía que comentabamos anteriormente en el Oppo Watch. Esto sin duda es debido a que su batería es de mayor capacidad, así como una mejor optimización del sistema en general.

Los resultados muy buenos, y nos permiten conocer cómo funcionan los consumos y el rendimiento de los diferentes modos de uso de la aplicación. Es muy útil conocer, que ambos relojes cuentan con un tiempo de carga inferior a 3 horas, lo cual permite que se puedan realizar un gran numero de flujometrías, así como reducir los tiempos en los que los relojes pueden estar disponibles.

5.4.4. Pruebas de usabilidad

Con respecto a la usabilidad, es uno de los aspectos más importantes a la hora de evaluar una aplicación móvil, ya que va a determinar en gran parte que la aplicación sea eficiente y sencilla de usar para cualquier paciente, sobretodo teniendo en consideración que el tipo de pacientes que utilizarán está aplicación va enfocada a personas de mediana y avanzada edad, y es fundamental que el uso sea lo más sencillo posible.

Es por ello que se decide, como se ha comentado en la iteración 4, llevar a cabo una pequeña prueba en el Hospital Clínico Universitario de Valladolid, de la mano de uno de los tutores responsables del proyecto, D.º Alfonso Bahillo Martínez.

La prueba duró un par de semanas, en las cuales se estuvieron realizando numerosas flujometrías, con el modo básico de la aplicación, y los pacientes tuvieron la oportunidad de realizar sus propias muestras con el uso del reloj.

Los pacientes destacaron la facilidad de uso de la interfaz, que era concisa y con los botones muy visibles para que fuese sencillo el inicio de la grabación. Con respecto al funcionamiento, no hubo problemas, con la mayoría de situaciones grabadas de manera correcta. Es cierto que hay que comentar, que es cierto que hubo algún paciente que no recordó pulsar el reloj, con lo cual la flujometría no se llevó a cabo, así como hubo otras personas que también olvidaron no guardar el audio resultante, con lo cual la muestra de flujometría tuvo que volver a repetirse, debido a los nervios a la hora de realizar la prueba.

Por parte del responsable médico, destacan que el menú oculto es de gran utilidad, ya que les permite ajustar parámetros relacionados con la aplicación, así como registar el paciente para que el audio pueda ser procesado correctamente por el servidor. No sé llegó a utilizar, debido a la deficiencia de conexión del centro médico donde se realizó la prueba, pero si que se guardaron de manera local en el dispositivo, haciendo uso

El resultado final de la prueba fue muy satisfactorio, y se pudieron obtener mucho feedback positivo, que será de gran utilidad para el desarrollo e implementación final futura de la aplicación. Se valorará en un futuro el desarrollo de funciones que permitan a pacientes de mayor edad la importancia de pulsar el botón de la interfaz para grabar, así como pulsarla para guardar el audio. Se valorará el uso de señales sonoras como avisos o pitidos para que estas acciones puedan ser llevadas a cabo.

Capítulo 6

Conclusiones

En este capítulo final, se hace un resumen del proyecto, así como aportar una visión global del trabajo realizado, destacando los objetivos conseguidos, así como un análisis del impacto y utilidad del proyecto que se ha implementado.

Este proyecto es una continuación de un proceso de automatización, que ha permitido que la aplicación alcance nuevas formas de usarse, junto con la idea de alcanzar la mayor autonomía a nivel de interacción con el proyecto. Nos encontramos ante un proyecto pionero, ya que no existen apenas soluciones a día de hoy que permitan realizar este tipo de pruebas, de manera sencilla y al alcance de un dispositivo tan común como un smartwatch.

Me siento realmente satisfecho con el trabajo que he desarrollado, el cual ha permitido corregir ciertos aspectos de la aplicación que no funcionaban de manera tan eficiente, ya que la automatización ha sido realizada en un lapso muy breve, y es normal que no se haya dedicado el suficiente tiempo al análisis de pruebas con respecto al código desarrollado. Creo sinceramente, que la labor del tester o analista de QA, que ha sido el nuevo rol al que me he enfrentado en este proyecto, está muy infravalorada y es una labor invisible, pero de gran importancia y que marca el desarrollo futuro de la aplicación.

A nivel personal, el proyecto ha sido realmente valioso, ya que me ha permitido adentrarme de lleno en el mundo de las aplicaciones Android, y poder tener la oportunidad de aprender un nuevo lenguaje de programación, Kotlin, que en el futuro será clave para que pueda desarrollar nuevas soluciones con respecto a las aplicaciones móviles. Quiero agradecer a los tutores, que hayan confiado en mí, así como el gran soporte que me han ofrecido a lo largo de este proyecto.

Por último, no me cabe ninguna duda de que a este proyecto aún le quedan un gran margen de mejora, que hará que siga en constante evolución y que alcance un mayor reconocimiento a nivel médico. A pesar de las dificultades, la gran curva de aprendizaje inicial y la falta de tiempo en el proyecto, considero que he cumplido los objetivos marcados en el inicio del proyecto, así como me ha permitido desarrollar mis conocimientos y mi aprendizaje a un nivel superior.

6.0.1. Trabajo futuro

En esta sección, destacaremos algunos puntos que no han quedado del todo resueltos, como posible trabajo futuro en caso de que se continué con el proyecto:

- Implementar la nueva versión del algoritmo de detección de micción, que está en la fase final de producción por parte de la Universidad de Deusto, y que en este proyecto no ha sido posible implementar por falta de disponibilidad del mismo. La idea es que mejore más aun si cabe las prestaciones, así como su tasa de exactitud a la hora de detectar cuando se realiza una micción o no.
- Realizar más pruebas en entornos reales, que permitan conocer de manera más exacta cuales son los problemas o posibles mejoras que son necesarias en la aplicación para que la usabilidad de la misma sea lo mejor posible. Sería muy importante que se pudiesen recabar un buen número de experiencias desde distintos centros médicos u hospitales, para hacer que la aplicación sea lo más eficiente posible.
- Como posibles mejoras finales, se podría realizar una adaptación de la aplicación a otros idiomas, sobretodo en la fase final de la misma, con la idea de buscar nuevos lugares donde utilizar la aplicación, así como posibles exploraciones de uso de otros sistemas de reconocimiento de micción, que puedan dotar a la aplicación de una mayor robustez con respecto a la toma de uroflujometrías.

Apéndices

Apéndice A

Funcionalidad y código implementado

El código de la aplicación se encuentra en un respositorio GitHub habilitado para ello. El enlace es el siguiente:

`https://github.com/gonzaovej/urosound`

Adjunto también enlaces a la primera versión publicada de la aplicación, así como la aplicación posterior realizada por Martín Ordóñez Vivó:

`https://github.com/DeustoTech/UroSound` (Primera versión de Urosound)

`https://gitlab.inf.uva.es/marordo/urosound-automatization` (Automatización de UroSound)

Apéndice B

Manual de despliegue

En este apartado, se explica brevemente como realizar el despliegue de la aplicación o su instalación. El código está indicado en el Apéndice A, en el siguiente enlace:

`https://github.com/gonzaovej/urosound`

Los pasos a seguir para la instalación de la aplicación son los siguientes:

1. Descargar e instalar Android Studio, que es el entorno de desarrollo donde se ha realizado la aplicación, y configurarlo al gusto personal si se desea.
2. Descargar el código del repositorio indicado, y descomprimirlo en la ruta que se desee, aunque también se puede realizar una clonación desde el propio entorno.
3. Descargar el código del repositorio indicado, y descomprimirlo en la ruta que se desee, aunque también se puede realizar una clonación desde el propio entorno.
4. Una vez que el proyecto se ha abierto en Android Studio, acceder a la opción "Build Project"para su compilación.
5. Instalar la aplicación en el smartwatch que del que se disponga. Para configurar el reloj, es necesario activar las opciones de desarrollador. Esto se realiza accediendo a los ajustes del smartwatch, a la opción Sistema, luego a Información, y presionar 7 veces el apartado donde se indica el Número de versión del compilador. De esta manera, se activará el menú de opciones del desarrollador, y se podrá activar desde ahí la opción Depuración ADB. Una vez que el smartwatch se conecte al ordenador vía USB, el entorno de desarrollo lo reconocerá. En el caso de smartwatches que no dispongan de conexión USB, se realizará de manera análoga, pero habrá que activar la depuración inalámbrica, para realizar la conexión entre Android Studio y la aplicación vía Wi-Fi.
6. Configurar y encender la baliza, en caso de que se quiera utilizar la funcionalidad de la detección por proximidad.

Apéndice C

Manual de uso

En este apéndice se explica de manera breve el manual de uso de la aplicación. Para comenzar, se debería haber realizado el despliegue explicado en el apéndice B. Para que la aplicación funcione es necesario que tanto el Bluetooth como la ubicación estén activas. La ubicación, como se ha explicado con anterioridad, se utilizará para una mejor conexión con la baliza.

Al abrir la aplicación, se mostrará el menú principal, donde se pueden realizar grabaciones utilizando los controles táctiles, así como los botones físicos del reloj. Antes de la primera ejecución, se solicitarán los permisos necesarios, que es obligatorio aceptar, para el correcto funcionamiento de la aplicación.

En el resto de modos, es necesaria la conexión con la baliza, la cual deberá estar encendida y configurada correctamente. Se mostrará una notificación persistente, informando de que el servicio que detecta la baliza está operativo. En el modo de reconocimiento de voz es necesario que se pronuncie la palabra clave asignada, que por defecto es “empezar”, pero que se puede cambiar desde el propio menú de la aplicación, que luego detallaré.

Con respecto al último modo, el del algoritmo de detección, simplemente detectará si se está produciendo o no la micción, una vez que se encuentre en el rango de la baliza. Cabe destacar, que la eficacia del algoritmo no es algo que fuese un objetivo para el estudiante, por lo que no se tienen controladas todas las situaciones que puedan suceder.

En el resto de la interfaz, se dispone de un botón para cerrar la misma, y si se pulsa 7 veces sobre el logo inferior de UroSound, se accederá al menú oculto de configuración. No se recomienda pulsar más de 7 veces o muy rápido, ya que puede ocasionar la pulsación de alguna opción del panel de control. Esto se debe a que tarda unos segundos en abrirse ya que está deteniendo el servicio que se encarga de la búsqueda de la baliza.

La idea es que desde este menú se puedan configurar parámetros por parte del equipo médico encargado, como aspectos de la baliza, reconocedor deseado, envío de audios al servidor, o copia de seguridad local, entre otros.

Lo ideal sería que el responsable médico registrase el ID del paciente, y a partir de ahí ya podría

realizar todas las opciones previstas sin problemas. Para el envío de audios al servidor remoto, es necesario que el paciente esté registrado. De igual manera, se puede realizar la copia de seguridad de los audios, para que se almacenen en la memoria del dispositivo en caso de que ocurra algún problema.

Por último, como recomendaciones generales, se recomienda no habilitar opciones de ahorro de batería en el reloj, así como desactivar aplicaciones que puedan tener relación con la misma, ya que esto puede derivar en fallos inesperados en el comportamiento de la aplicación. La manera ideal de detener y cerrar la aplicación es pulsar en la opción de “Cerrar App”, en el límite superior del reloj.

Bibliografía

[1] Medicina tecnológica: beneficios y tendencias del e-health. Visitado: 2023-12-13. [Online]. Available: https://telefonicatech.com/blog/medicina-tecnologica-beneficios-y-tendencias-ehealth

[2] M. Aitken, B. Clancy, and D. Nass, "The growing value of digital health," IQVIA Institute, Tech. Rep., 2017, visitado: 2023-12-13. [Online]. Available: https://www.iqvia.com/insights/the-iqvia-institute/reports-and-publications/reports/the-growing-value-of-digital-health

[3] L. Arjona, L. Díez, A.Bahillo, and A. Arruza-Echevarría, *UroSound: A Smartwatch-based Platform to Perform Non-Intrusive Sound-based Uroflowmetry*. IEEE, 2022, visitado:. [Online]. Available: https://ieeexplore.ieee.org/document/9670631

[4] Uroflowmetry. Visitado:. [Online]. Available: https://www.urologyhealth.org/urology-a-z/u/uroflowmetry

[5] Modelo iterativo - design toolkit. Visitado:. [Online]. Available: https://design-toolkit.recursos.uoc.edu/es/iterativo/

[6] Características android. Visitado:. [Online]. Available: https://androidos.readthedocs.io/en/latest/data/caracteristicas/

[7] £qué es android? Visitado:. [Online]. Available: https://www.arimetrics.com/glosario-digital/android#Principales_caracteristicas_de_Android

[8] Logo android. Visitado:. [Online]. Available: https://es.wikipedia.org/wiki/Android#

[9] Diagrama de capas de android. Visitado:. [Online]. Available: https://multimedia.codeandcoke.com/_detail/apuntes:android-stack_2x.png?id=apuntes%3Aandroid

[10] Kotlin. Visitado:. [Online]. Available: https://www.plainconcepts.com/es/kotlin-android/#Que_es_Kotlin

[11] Logo kotlin. Visitado:. [Online]. Available: https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n)

[12] Java. Visitado:. [Online]. Available: https://www.ibm.com/es-es/topics/java

[13] Logo java. Visitado:. [Online]. Available: https://en.wikipedia.org/wiki/Java_(programming_language)

[14] Wear os (wikipedia). Visitado:. [Online]. Available: https://es.wikipedia.org/wiki/Wear_OS

[15] Wear os página oficial. Visitado:. [Online]. Available: https://wearos.google.com/intl/es_es/

[16] Tensorflow lite. Visitado:. [Online]. Available: https://www.tensorflow.org/lite?hl=es-419

[17] Logo tensorflow lite. Visitado:. [Online]. Available: https://es.wikipedia.org/wiki/TensorFlow

[18] Android studio. Visitado:. [Online]. Available: https://developer.android.com/studio?hl=es-419

[19] Astah professional. Visitado:. [Online]. Available: https://astah.net/products/astah-professional/

[20] Github. Visitado:. [Online]. Available: https://github.com/

[21] Aplicación nrf connect (baliza). Visitado:. [Online]. Available: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=es_419

[22] Aplicación lightblue (baliza). Visitado:. [Online]. Available: https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=es

[23] M. M, T. J. Kevin, P. Raveendran, R. S*, and A. D. S, *Student Attendance Manager Using Beacons and Deep Learning*. IOP Publishing, 2020, visitado:. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1706/1/012153/pdf1

[24] Microsoft teams. Visitado:. [Online]. Available: https://www.microsoft.com/es-es/microsoft-teams/log-in

[25] Overleaf. Visitado:. [Online]. Available: https://es.overleaf.com/

[26] Latex. Visitado:. [Online]. Available: https://es.wikipedia.org/wiki/LaTeX

[27] Notepad. Visitado:. [Online]. Available: https://notepad-plus-plus.org/

[28] Notion (wikipedia). Visitado:. [Online]. Available: https://es.wikipedia.org/wiki/Notion

[29] Microsoft excel. Visitado:. [Online]. Available: https://www.microsoft.com/es-es/microsoft-365/excel

[30] Sueldo medio de un desarrollador android. Visitado:. [Online]. Available: https://www.glassdoor.es/Sueldos/desarrollador-android-junior-sueldo-SRCH_KO0,28.htm

[31] M. O. Vivó, *Automatización de toma de flujometrías sonoras*. Universidad de Valladolid, 2023. [Online]. Available: https://uvadoc.uva.es/handle/10324/63015

[32] Kotlin check if a directory exists. Visitado:. [Online]. Available: https://kotlinandroid.org/kotlin/kotlin-check-if-a-directory-exists/

[33] Permiso read external storage (almacenamiento). Visitado:. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#READ_EXTERNAL_STORAGE

[34] Permiso write external storage (almacenamiento). Visitado:. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#WRITE_EXTERNAL_STORAGE

[35] Permiso record audio (almacenamiento). Visitado:. [Online]. Available: https://developer.android.com/media/platform/mediarecorder?hl=es-419

[36] Android wave recorder. Visitado:. [Online]. Available: https://github.com/squti/Android-Wave-Recorder

[37] Permiso access coarse location (ubicación). [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#ACCESS_COARSE_LOCATION

[38] Bluetooth ble (bajo consumo). [Online]. Available: https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview?hl=es-419

[39] Permiso access fine location (ubicación). [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#ACCESS_FINE_LOCATION

[40] Permiso bluetooth. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#BLUETOOTH

[41] Reconocedor nativo android. [Online]. Available: https://developer.android.com/training/wearables/user-input/voice?hl=es-419#kotlin

[42] Reconocedor vosk. [Online]. Available: https://alphacephei.com/vosk/

[43] Storage manager. [Online]. Available: https://developer.android.com/reference/kotlin/android/os/storage/StorageManager

[44] Permiso bluetooth scan (bluetooth). [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#BLUETOOTH_SCAN

[45] Permiso bluetooth connect (bluetooth). [Online]. Available: https://developer.android.com/reference/kotlin/android/os/storage/StorageManager

[46] Simultaneidad en android. [Online]. Available: https://developer.android.com/codelabs/basic-android-kotlin-compose-coroutines-kotlin-playground?hl=es-419#0

[47] Speech service en android 13. [Online]. Available: https://developer.android.com/about/versions/13/behavior-changes-all?hl=es-419#speech-service

[48] Mediastore (android). [Online]. Available: https://developer.android.com/reference/android/provider/MediaStore

[49] Audiorecord (android). [Online]. Available: https://developer.android.com/reference/android/media/AudioRecord

[50] Tipos arquitectura android. [Online]. Available: https://alainnicolastello.medium.com/cual-es-la-mejor-arquitectura-para-android-con-kotlin-477b30c20e2a

[51] Arquitectura clean (android). [Online]. Available: https://xurxodev.com/por-que-utilizo-clean-architecture-en-mis-proyectos/

[52] Pruebas de software (ibm). [Online]. Available: https://www.ibm.com/es-es/topics/software-testing

[53] Tipo de pruebas de software). [Online]. Available: https://www.unir.net/ingenieria/revista/pruebas-software/

[54] Pruebas de caja gris. [Online]. Available: https://www.checkpoint.com/es/cyber-hub/cyber-security/what-is-gray-box-testing/

[55] L. Arjona, N. G, S. Hernandez, A.Bahillo, and P. S, *Towards Autonomous Sound-based Uroflowmetries with Machine Learning at the Edge*. UbiCompLab, 2022. [Online]. Available: https://ubicomplab.cs.washington.edu/pdfs/UroSound_EMBC_2022_Final.pdf

[56] Colección audioset. [Online]. Available: https://research.google.com/audioset/dataset/index.html

[57] Colección esc-50. [Online]. Available: https://github.com/karolpiczak/ESC-50

[58] Freesound. [Online]. Available: https://freesound.org/