Universidad de **Valladolid**

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Automatización de toma de flujometrías sonoras

Autor:
Martín Ordóñez Vivó
Tutores:
Mario Corrales Astorgano
Alfonso Bahillo Martínez

Le ofrezco a Dios mi trabajo para el bien y la salud de las personas del mundo.

Agradecimientos

Le agradezco a D.º Mario Corrales y a D.º Alfonso Bahillo la oportunidad de trabajar en este TFG, y a todos los que me han apoyado todo este tiempo.

Resumen

La uroflujometría es un procedimiento médico del ámbito de la urología que permite medir características de la orina del paciente o detectar patologías relacionadas al tracto urinario. La uroflujometría sonora por su parte, permite tomar las mismas mediciones utilizando sólamente el sonido de la orina. La diferencia con la uroflujometría convencional es que ésta requiere equipamiento especializado y no permite la libertad de tomar las mediciones en casa. El objetivo de este TFG es automatizar una aplicación de uroflujometría sonora para que no necesite ningún tipo de entrada por parte del paciente, o en su defecto la ínfima entrada necesaria, permitiendo así que mejore su usabilidad, especialmente en el caso de personas poco familiares con los dispositivos electrónicos.

Abstract

Uroflowmetry is a medical procedure in the field of urology that allows measuring characteristics of the patient's urine or detecting pathologies related to the urinary tract. Sound-based uroflowmetry, on the other hand, allows taking the same measurements using only the sound of urine. The difference with conventional uroflowmetry is that it requires specialized equipment and does not allow the freedom to take measurements at home. The objective of this TFG is to automate a sound-based uroflowmetry application so that it does not need any type of input from the patient, or failing that, the minimal input necessary, thus allowing it to improve its usability, especially in the case of people unfamiliar with electronic devices.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

La idea de este proyecto es automatizar la aplicación privada Urosound del instituto de tecnología de la Universidad de Deusto en Bilbao, DeustoTech. Esta aplicación permite hacer grabaciones de audio en un smartwatch mediante una interfaz muy simple con un botón y también permite enviar los audios grabados a un servidor para su procesamiento, desde un panel de control oculto para el usuario. El propósito de esta aplicación es la toma de flujometrías sonoras en el ámbito médico de la urología, es decir, grabar el sonido de la orina de los pacientes impactando contra el inodoro para medir características de la orina o detectar posibles patologías en el tracto urinario. Esta detección se hace mediante algoritmos de machine learning y deep learning en el servidor de Deustotech.

La uroflujometría convencional se lleva a cabo con un aparato similar a un orinal, que consta de un embudo conectado a un ordenador, a través del cual paciente orina (figura 1.1). Con este sistema se puede recoger información como el volúmen y velocidad del vaciado, así como el patrón que sigue, para detectar anomalías. Gracias a esta técnica, pueden diagnosticarse enfermedades como la Hiperplasia Prostática Benigna (HBP), la Incontinencia Urinaria, la Estenosis Uretral o la Vejiga Hiperactiva. La uroflujometría es comúnmente utilizada junto con otras pruebas, como la cistometría (medición de la presión en la vejiga) y el análisis de orina, para obtener una evaluación completa de la función del tracto urinario.

La propuesta inicial del TFG es hacer una serie de implementaciones al código de la app Urosound para reducir al mínimo imprescindible la interacción del usuario con la aplicación. Estas propuestas son tres: implementar un mecanismo de activación por voz de la grabación, de manera que cuando el paciente diga en voz alta una “palabra clave”, se active la toma flujométrica, eliminando la necesidad de pulsar un botón; implementar un sistema de detección de proximidad del paciente al inodoro mediante el uso de una baliza BLE (Bluetooth Low Energy), que permita ahorrar batería en el dispositivo como mecanismo de “detección constante” de la presencia del paciente en el baño; e implementar un algoritmo de detección de micción de inteligencia artificial en tiempo real para cerciorarse de que está sucediendo un evento de micción una vez el paciente está próximo al inodoro y no está sonando un gri-

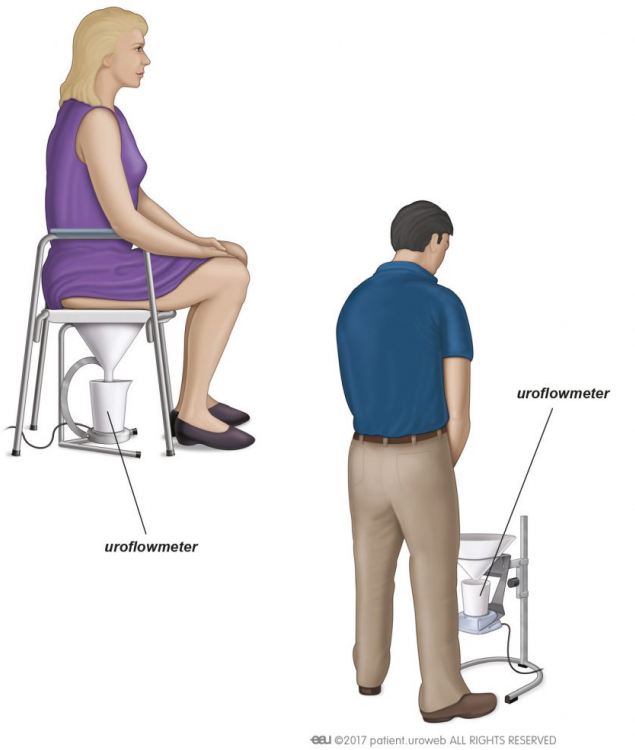


Figura 1.1: Uroflujometría convencional [1]

fo o la ducha, por ejemplo. Esto último también permite liberar un poco de carga de procesamiento del servidor, ya que el procesamiento del audio para detectar patologías se sigue realizando en el servidor.

1.2. Motivación

La motivación de una automatización de este estilo puede parecer innecesaria, ya que la interfaz de la aplicación sólo tiene un botón para iniciar/parar la grabación de audio y luego dos botones de sí/no para decidir si se desear guardar el audio o descartarlo, además de poder operar la app al completo mediante los botones físicos del smartwatch. Sin embargo, estas mejoras propuestas tienen el fin de ayudar a personas muy poco familiarizadas con los dispositivos electrónicos, como niños o ancianos, e intentar evitar errores por parte del paciente, ya que en caso de incontinencias, micciones nocturnas u olvidos, se lleven a cabo las grabaciones correspondientes para garantizar que el procedimiento se lleva a cabo con la mayor eficacia posible, al tiempo que se evita la incomodidad de estar pendiente de interactuar con el reloj inteligente, y tratando de ser lo menos intrusivo posible en la intimidad de la micción.

1.3. Estado de la cuestión

Actualmente se pueden hacer flujometrías convencionales con equipamiento médico en clínicas de urología, pero también se pueden hacer flujometrías sonoras tanto en las mismas clínicas o en casa mediante aplicaciones que lo permitan. Existen unas cuantas alternativas con unos miles de descargas para teléfonos Android, pero no se han encontrado aplicaciones para smartwatch (Wear OS) que cumplan esa misma función. Para la búsqueda se ha utilizado tanto la Play Store genérica como la Play Store específica del smartwatch y también se ha buscado de forma genérica en internet. Con respecto a las aplicaciones de Android, las más relevantes han sido descritas en las siguientes secciones.

1.3.1. MenHealth

MenHealth [2] (fig. 1.2), desarrollada por BE Technologies, Inc., es una aplicación con acceso limitado, ya que según la descripción de Google Play:

Para usar la aplicación móvil Uroflowmetry, comuníquese con su médico para recibir una invitación. Si usted es médico, contáctenos en support@myuroflow.com

Además de esto, está centrada sobre todo en los varones, ya que la "puntuación" que da la aplicación tras realizar los cálculos correspondientes va en base a la probabilidad de padecer Hiperplasia o Hipertrofia Benigna de Próstata (HBP). A pesar de esto, parece tener una interfaz descriptiva y con alto grado de usabilidad.

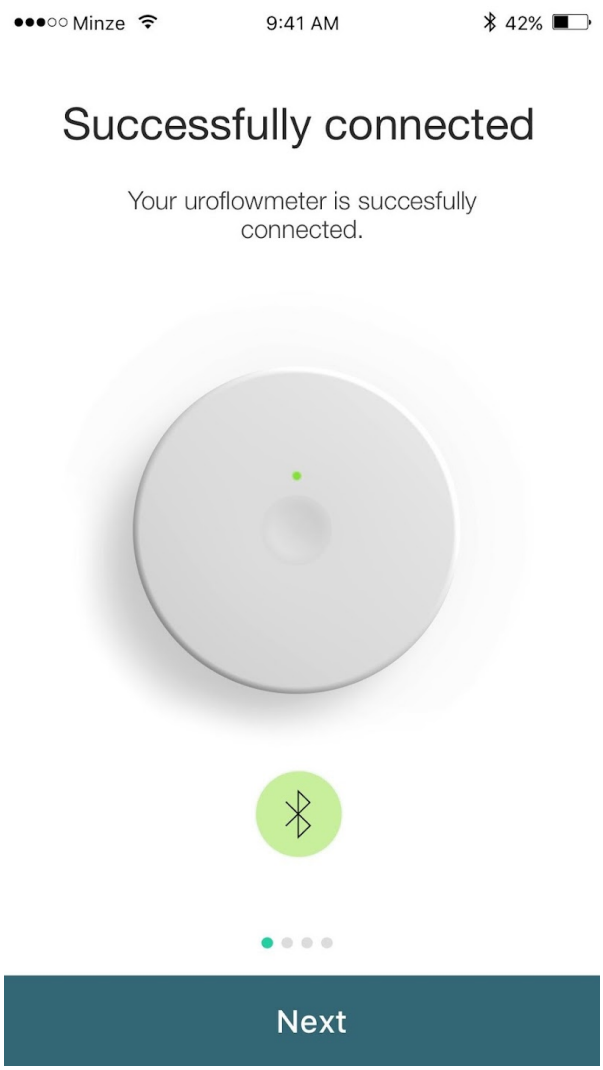
(a) Diario de micción

(b) Resultados tras el vaciado

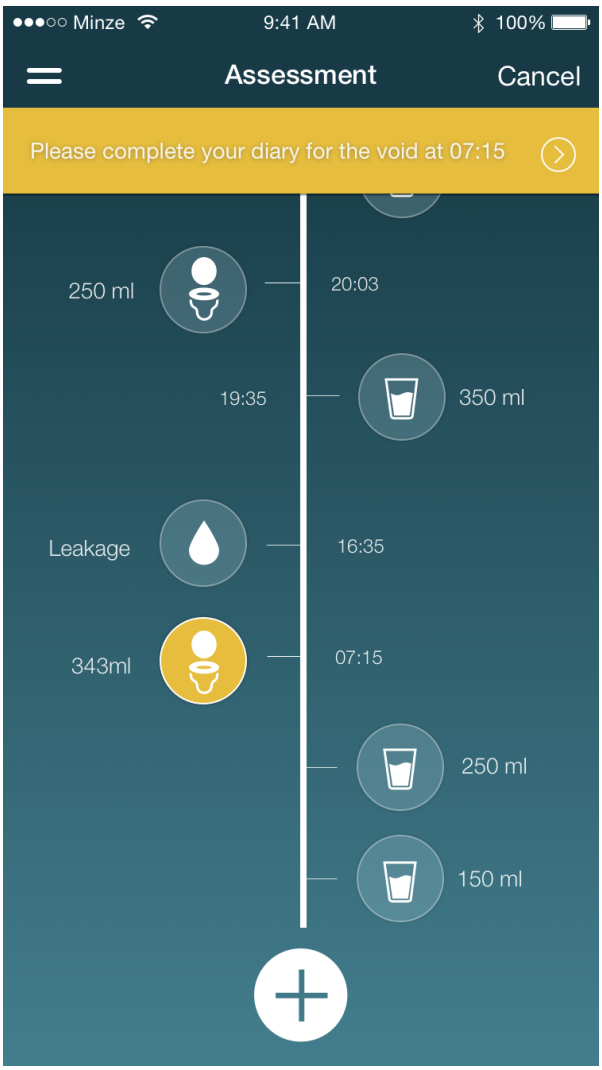
Figura 1.2: MenHealth

1.3.2. Minze Flow

Minze Flow [3] (fig. 1.3), desarrollada por Minze Health, permite el despliegue de un sistema de uroflujometría privativo de la marca Minze controlado por Bluetooth. No utiliza la uroflujometría sonora, por lo que requiere de equipamiento profesional extra.



(a) Conexión del dispositivo Minze



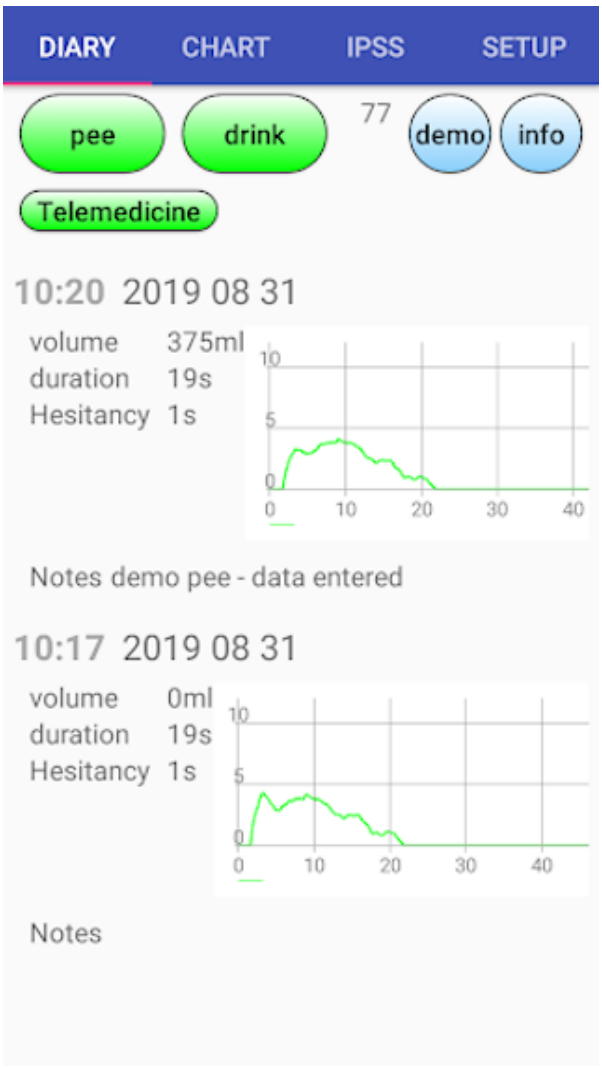
(b) Histórico de micciones e ingestas de líquidos

Figura 1.3: Minze Flow

1.3.3. TeleSonoUroflow

TeleSonoUroflow [4] (fig. 1.4), desarrollada por John Brohan, es gratuíta y permite realizar la toma de uroflujometrías sonoras sin requisitos adicionales, pero tiene una interfaz bastante pobre y sólamente genera una gráfica simple. Es la alternativa no profesional más viable que he encontrado.

(a) Pantalla principal



(b) Pantalla de resultados

Figura 1.4: TeleSonoUroflow

1.3.4. Urosound

Urosound (fig. 1.5), desarrollada por Deustotech, es la aplicación compatible con Wear OS que se pretende automatizar. Es privada, y dispone de una pantalla principal (a, b, c) desde la que el paciente graba los audios, y un panel de control (d) oculto al paciente desde el que el doctor puede introducir el código del paciente y enviar los audios al servidor. Al pulsar el botón lateral (a, marcado en verde), la aplicación se abrirá y comenzará la grabación automáticamente. Tras finalizar la grabación, el usuario podrá decidir (b) si se desea guardar la grabación o eliminarla. Tras esto, se volverá a la pantalla principal (c), desde donde se podrá repetir el proceso pulsando otra vez en el botón.

Figura 1.5: Urosound [5]

1.4. Tecnologías empleadas

Esta sección tiene el objetivo de exponer las principales tecnologías utilizadas en el desarrollo del proyecto.

Cabe destacar que al partir de una aplicación ya existente, el lenguaje de programación utilizado, Kotlin, ya estaba predefinido. Si bien es cierto que al principio no sabía programar en Kotlin, me ha resultado sencillo adaptarme ya que sólo es una “capa de abstracción” de Java que permite interoperabilidad entre ambos lenguajes y funciona de forma nativa en Android, además de proporcionar soporte para iOS como se describe más adelante.

Además de esto, al tratarse de una aplicación para smartwatch, el sistema operativo empleado es Wear OS, la versión de Android para wearables.

1.4.1. Android

Figura 1.6: Logo de Android

El sistema operativo móvil Android [6] [7] (fig. 1.6) está basado en el kernel de Linux y es de código abierto. Es propiedad de Google y éste lo desarrolla en conjunto con el consorcio Open Handset Alliance. Es compatible con móviles, tabletas, wearables, televisiones inteligentes y sistemas integrados de los automóviles, entre otros. El sistema operativo se divide en las siguientes capas (fig. 1.7):

- **Aplicaciones** Es la capa a más alto nivel. Todas las aplicaciones están escritas en Java, y tanto las aplicaciones base como las de terceros tienen acceso a la misma API y pueden tener acceso a otras aplicaciones respetando las reglas del framework.

- **Framework de Aplicaciones** Incluye características muy útiles para el desarrollo, como un sistema de vistas enriquecido con componentes para crear las interfaces; un administrador de recursos, que pueden ser strings localizadas (útiles para traducciones), archivos gráficos o componentes complejos de diseño; un sistema de notificaciones para que las aplicaciones muestren alertas personalizadas en la barra de estado; un administrador de tareas que gestiona el ciclo de vida [8] de las aplicaciones y la pila de actividades de las mismas; proveedores de contenido para comunicación entre aplicaciones y gestión de servicios, entre otros.

- **Bibliotecas** Escritas en C y C++, algunos ejemplos son la biblioteca C estándar, bibliotecas de medios, bibliotecas de gráficos, 3D (como OpenGL) y SQLite, entre otras. Dan soporte al framework de aplicaciones.

- **Runtime de Android** Es un entorno de ejecución sobre el que se ejecutan máquinas virtuales para cada aplicación. Utiliza archivos ejecutables en formato .dex y originalmente se usaba Dalvik como máquina virtual, pero a partir de la versión 5.0 de Android (Lollipop), se utiliza ART (Android RunTime).

- **Kernel Linux** Una sólida opción de código abierto que permite flexibilidad y transparencia para Android.

Figura 1.7: Arquitectura Android [9]

1.4.2. Wear OS

Wear OS [10] [11] (fig. 1.8) es un sistema operativo desarrollado por Google diseñado específicamente para relojes inteligentes y otros wearables. Fue lanzado inicialmente en 2014 bajo el nombre de Android Wear y fue posteriormente renombrado como Wear OS en 2018. Su interfaz de usuario está optimizada para ser utilizado en dispositivos con pantallas pequeñas y táctiles, y suele ser asistida por comandos de voz. Esta es una de las razones detrás de la implementación del reconocimiento de voz en este trabajo.

Admite conectividad Bluetooth y Wi-Fi para interactuar con otros dispositivos y acceder a Internet. Además, los smartwatches con Wear OS suelen estar equipados con algunos sensores diferentes a los de un móvil, como monitores de frecuencia cardíaca, además de acelerómetros, giroscopios y GPS, entre otros. Estos sensores permiten a las aplicaciones realizar un seguimiento de la actividad física, medir el ritmo cardíaco, registrar la ubicación o la posición del usuario, etc. En este trabajo se aprovechan estas capacidades para detectar la posición relativa del usuario mediante una baliza Bluetooth Low Energy, como se explica más adelante.

Google ha trabajado constantemente en el desarrollo y mejora de Wear OS. Sin embargo, el sistema operativo sigue teniendo problemas en términos de rendimiento y duración de la batería. Aunque ha habido avances en estas áreas, los usuarios todavía pueden encontrar variaciones en la experiencia según el dispositivo específico que utilicen. Por esta razón se ha intentado optimizar la autonomía de la batería al utilizar la aplicación Urosound, como se detalla en capítulos posteriores.

Figura 1.8: Logo de Wear OS

1.4.3. Java

Java [12] [13] (fig. 1.9) es un lenguaje de programación de alto nivel y orientado a objetos que fue desarrollado por Sun Microsystems en la década de 1990 y posteriormente adquirido por Oracle Corporation. Es uno de los lenguajes más populares y ampliamente utilizados en la industria del software.

Una de las características más destacadas de Java es su portabilidad, ya que puede ejecutarse en diferentes plataformas sin necesidad de realizar cambios significativos. Esta característica se basa en el principio de “write once, run anywhere”. Esto es posible gracias a la máquina virtual de Java (JVM), que interpreta el código intermedio (bytecode) Java y lo ejecuta en cualquier sistema operativo sin necesidad de recompilación. Esto ha permitido que Java sea utilizado en una amplia variedad de

dispositivos, desde ordenadores de escritorio hasta dispositivos móviles y embebidos.

Java utiliza un recolector de basura (garbage collector) para administrar automáticamente la memoria asignada a los objetos. Esto significa que los programadores no necesitan preocuparse por liberar manualmente la memoria utilizada por los objetos que ya no se necesitan, lo que facilita la programación y reduce los errores relacionados con la administración de memoria.

Figura 1.9: Logo de Java

1.4.4. Kotlin

Kotlin [14] [15] (fig. 1.10) es un lenguaje de programación moderno y conciso que se ejecuta en la máquina virtual de Java. Fue desarrollado por JetBrains y se presentó públicamente en 2011. Está diseñado para ser interoperable con Java, lo que significa que se puede utilizar código Kotlin en proyectos Java existentes y viceversa.

Kotlin se ha vuelto muy popular en el desarrollo de aplicaciones para Android. En 2017, Google anunció el soporte oficial para Kotlin en Android, lo que llevó a un aumento significativo en su adopción por parte de los desarrolladores. Desde entonces, se ha convertido en un lenguaje preferido para el desarrollo de aplicaciones Android junto con Java.

Una de las principales ventajas de Kotlin es su sintaxis concisa y expresiva. Está diseñado para reducir el código repetitivo y aumentar la legibilidad. También incorpora muchas características modernas de lenguajes de programación, como la inferencia de tipos, las funciones de orden superior, las extensiones de funciones y las clases de datos. Cabe destacar su sistema de tipado, que distingue entre valores nulos y no nulos de forma predeterminada, lo que ayuda a prevenir errores de referencia nula en tiempo de compilación.

Kotlin Multiplatform permite soporte parcial de Kotlin para iOS. Por ejemplo, se puede implementar la interfaz de usuario y las capas de acceso a datos específicas de iOS utilizando Swift y las API nativas de iOS, mientras que la lógica de negocio central puede estar escrita en Kotlin y compartida entre Android e iOS.

Figura 1.10: Logo de Kotlin

1.5. Objetivos

Los objetivos del proyecto se pueden definir como los siguientes:

- Implementar un **sistema de reconocimiento de voz**, mediante el cual el paciente pueda decir una palabra clave en voz alta que inicie la grabación de la micción.
- Implementar un **sistema de detección de proximidad** del paciente al inodoro mediante una baliza Bluetooth Low Energy (BLE).
- Implementar un **algoritmo de inteligencia artificial de detección de micción** en tiempo real.

También se harán mejoras y refactorizaciones al código existente y se explorará la posibilidad de desactivar servicios del smartwatch para aumentar la autonomía del dispositivo.

1.6. Metodología

Dadas las características del proyecto y otros aspectos relevantes, como la estabilidad de los requisitos, la posibilidad de no llegar a cumplir con los objetivos y el hecho de ser un proyecto llevado a cabo por una única persona (bajo la tutela de Dº Mario Corrales y Dº Alfonso Bahillo), se ha tomado la decisión de emplear una metodología iterativa e incremental. En este caso, no es buena idea utilizar metodologías ágiles, ya que están diseñadas para grupos de varias personas.

El uso de esta metodología tiene varias ventajas, como el hecho de que se puedan obtener resultados significativos y utilizables desde las primeras iteraciones, permitiendo cambiar el rumbo del proyecto sobre la marcha si fuera necesario, a diferencia de las metodologías en cascada. Tambien evita la posibilidad de caer en el “gold-plating” o en el perfeccionismo, teniendo objetivos claros para cada iteración. Además es más fácil retomar el proyecto desde la última iteración en caso de que se dejara de trabajar en ello temporalmente.

En la Figura 1.11 se muestran los pasos que se siguen durante el desarrollo iterativo e incremental. En el Capítulo 3 se abordará en detalle el trabajo llevado a cabo en cada iteración del proyecto.

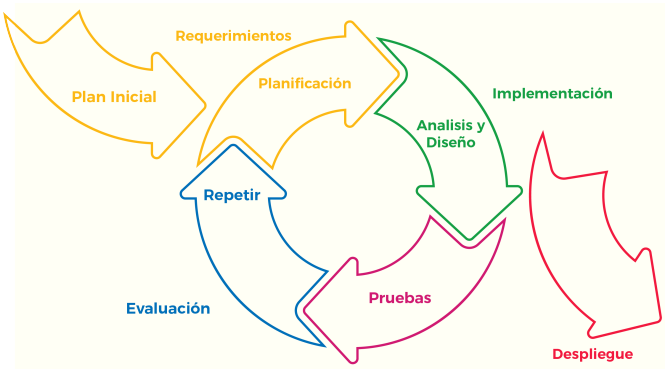


Figura 1.11: Metodología Iterativa e Incremental [16]

1.7. Estructura de la memoria

La memoria se divide en los siguientes capítulos:

- **Introducción:** Presentación a alto nivel de abstracción del proyecto. Incluye el contexto, motivación, un breve estado de la cuestión, las tecnologías empleadas en el desarrollo, los objetivos del proyecto, la metodología utilizada y la estructura del proyecto (esta sección).
- **Planificación:** Planificación inicial, descripción del entorno de trabajo, estimación de costes del proyecto, análisis de riesgos y planificación final.
- **Descripción de las iteraciones:** Explicaciones detalladas del desarrollo por iteraciones del trabajo; permite ver la evolución en el desarrollo del proyecto y los obstáculos encontrados.
- **Estado final de la aplicación:** Descripción completa del estado final de la aplicación, diagramas e información pertinente.
- **Conclusiones:** Compilación final del trabajo realizado y aprendizaje obtenido a alto nivel de abstracción, y el posible trabajo futuro.
- **Apéndices:** Documentación extra como el manual de despliegue de la aplicación, el manual de uso de la aplicación y otros datos.
- **Bibliografía:** Lista de las referencias más relevantes para el desarrollo del proyecto.

Capítulo 2

Planificación

2.1. Planificación inicial

En esta sección se detalla la planificación inicial, tanto temporal como de trabajo estimado según las iteraciones descritas en la sección de Metodología 1.6.

El desarrollo del Trabajo de Fin de Grado debe durar un total de al menos 300 horas según la normativa de la Universidad de Valladolid. Para cumplir con este objetivo, se ha dividido el tiempo de trabajo en 3 bloques a lo largo del curso. El TFG se aceptó el día 10 de noviembre del 2022 y se finalizará el 3 de julio de 2023. Los bloques serán los siguientes:

- **1er Bloque:** Del 10 de noviembre del 2022 al 12 de febrero del 2023. Durante este período se emplearán 12 horas para documentarse sobre la temática y preparar el entorno de trabajo, así como realizar la planificación, requisitos y diagramas iniciales. En este período estaré ocupado con las asignaturas del primer cuatrimetre.
- **2do Bloque:** Del 13 de febrero del 2023 al 14 de mayo del 2023. Durante este período se emplearán 120 horas para trabajar en el proyecto, invirtiendo 10 horas totales en los fines de semana durante 12 semanas, excluyendo Semana Santa. En este período estaré haciendo las prácticas en empresa y una asignatura.
- **3er Bloque:** Del 15 de mayo del 2023 al 3 de julio del 2023. Se utilizará este período para finalizar el proyecto, empleando 24 horas por semana durante 7 semanas, haciendo un total de 168 horas. Si fuera necesario, invertiré más tiempo por semana durante este período para poder finalizar en la fecha acordada, en caso de que surgiera cualquier imprevisto o retraso en el desarrollo.

El contenido del trabajo por su parte, está dividido en 4 iteraciones de trabajo:

- **1ª Iteración.** Activación por voz: Se implementará un sistema de activación de la grabación de la micción por comandos de voz, como alternativa a la pulsación del botón de encendido del

smartwatch o al botón de inicio de grabación de la interfaz de la aplicación. Se estiman 70 horas para el trabajo de esta iteración.

- **2ª Iteración.** Activación por proximidad BLE: Se implementará un sistema de activación de la grabación mediante proximidad a una baliza Bluetooth Low Energy (BLE), para aumentar la autonomía del dispositivo y mejorar la precisión de la activación. Se estiman 100 horas para el trabajo de esta iteración.
- **3ª Iteración.** Detección de micción en tiempo real: Se tratará de conseguir la implementación de los algoritmos de Machine Learning y Deep Learning para su ejecución en tiempo real en el smartwatch, de modo que se pueda detectar si el paciente ha comenzado la micción una vez se encuentre cerca del inodoro. Se estiman 80 horas para el trabajo de esta iteración.
- **4ª Iteración.** Integración en Urosound: Se integrarán los sistemas implementados en la aplicación Urosound y se realizarán los cambios en la estructuración del código que procedan. Se estiman 50 horas para el trabajo de esta iteración.

En la figura 2.1 se pueden ver las horas estimadas para cada iteración.

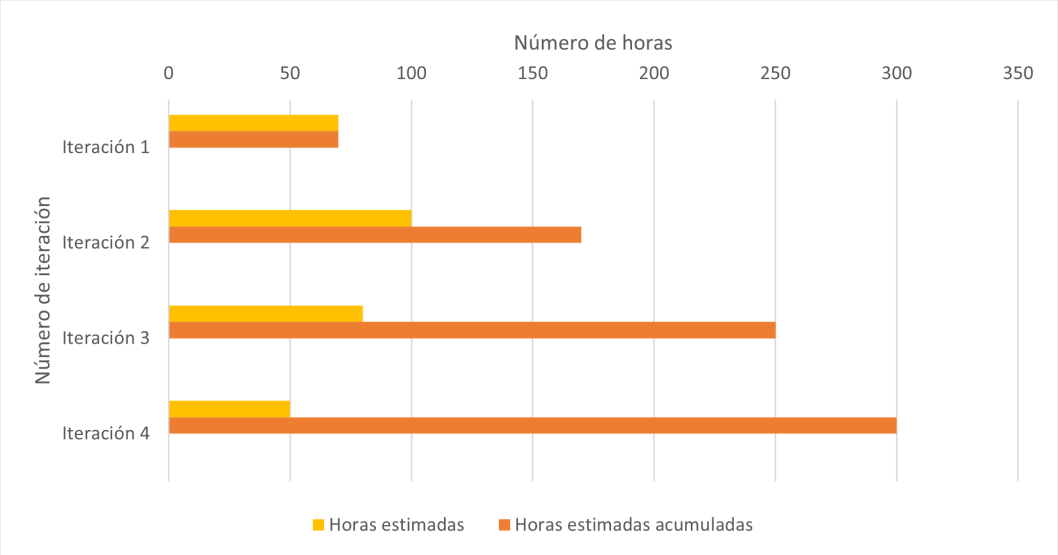


Figura 2.1: Horas estimadas iniciales por iteración

Además de esto, si el tiempo lo permite, se explorará la obtención de permisos de superusuario en el smartwatch para deshabilitar servicios innecesarios del sistema operativo y realizar optimizaciones, cuyo fin es aumentar la autonomía del smartwatch. También se harán mejoras de usabilidad a la aplicación existente y refactorizaciones de código.

2.2. Entorno de trabajo

Esta sección tiene como objetivo de describir las herramientas del entorno de trabajo y su preparación y configuración.

Para el desarrollo de este trabajo, se ha utilizado un portátil Lenovo ideapad 330 [17] (tab. 2.1) operando sobre Windows 10, un móvil LG K40 [18] (tab. 2.2) con Android 9 (API 28 [19]), un smartwatch Oppo Watch modelo OW19W6 [20] (tab. 2.3) versión europea con Wear OS 2.21 (API 28 de Android) y dos beacons BLE, uno de Global Tag [21] y otro de Minew Tech [22].

Lenovo Ideapad 330	
Procesador	Intel Core i5-8300H
Memoria	8 GB
Disco HDD	1 TB
Disco SSD	250 GB
Tarjeta gráfica	Nvidia Geforce GTX 1050
Resolución	1920 x 1080
Sistema operativo	Windows 10
Arquitectura del sistema	64 bits

Tabla 2.1: Especificaciones del portátil de trabajo

LG K40	
Procesador	Octa-core 2.0 GHz Cortex-A53
Memoria	2 GB
Almacenamiento	32 GB
Tarjeta gráfica	PowerVR GE8320
Resolución	720 x 1440
Sistema operativo	Android 9 (Pie)
Batería	3000 mAh

Tabla 2.2: Especificaciones del móvil de trabajo

Oppo Watch 41mm (OW19W6EU)	
Procesador	Qualcomm Snapdragon Wear 3100, Apollo 3
Memoria	1 GB
Almacenamiento	8 GB
Resolución	320 x 360
Sistema operativo	Wear OS 2.21
Batería	300 mAh

Tabla 2.3: Especificaciones del smartwatch

Para el montaje y preparación del entorno de desarrollo se ha seguido la documentación oficial [23] [24] [25] [26] [27] [28]. De forma resumida, el proceso es la instalación y configuración de Android Studio en el ordenador de desarrollo, el inicio y configuración inicial del smartwatch como lo haría cualquier usuario, y la configuración de desarrollo del smartwatch. El procedimiento para esta última es ir al menú de configuración, luego a Sistema, luego a Información, presionar 7 veces el número de versión de compilación, volver al menú de configuración, ir al nuevo menú Opciones para desarrolladores y habilitar la opción Depuración ADB. De este modo, se puede conectar el smartwatch al pc a través de

la plataforma de carga por USB y ejecutar desde Android Studio la aplicación que se desee, así como acceder al almacenamiento interno, etc. También se conectó el smartwatch al móvil [29] para hacer depuración, pero esto no es necesario para un entorno real.

Con respecto a la configuración de las balizas (ver figura 2.2), he utilizado mayoritariamente la de Global Tag debido a que ésta tiene documentación y manual de usuario, a diferencia de la de Minew Tech. En ambos casos se desmontan las carcasas de las balizas y se inserta la pila de botón correspondiente, en particular una pila de tipo CR2032 3V de litio. En general pueden pasar varios meses o un par de años hasta que haga falta cambiar la pila. En el caso de la de Minew Tech, se enciende quitando un pequeño plástico transparente que evita que se cierre el circuito, y se apaga volviéndolo a poner. Por otro lado, en la parte inferior del beacon de Global Tag se encuentra el botón de encendido. Se enciende manteniéndolo pulsado durante 3 segundos, y se confirma visualmente al encenderse brevemente un LED de color azul. Análogamente, se apaga manteniéndo el botón pulsado 3 segundos y en este caso la luz el roja. Ambas balizas tienen impresa su dirección MAC por defecto en la parte anterior de las mismas para que la configuración sea más sencilla. Tras esto las balizas ya se pueden utilizar, pero si se desea cambiar la dirección MAC, los números Major y Minor u otros parámetros como por ejemplo la frecuencia de advertising o la potencia de la señal, se debe hacer uso de una aplicación externa [30] [31] para conectarse a la baliza y asignar los parámetros que se deseen.

(a) Balizas cerradas (b) Balizas abiertas (c) Vista por el otro lado

Figura 2.2: Balizas BLE utilizadas en el proyecto, arriba Minew Tech, abajo Global Tag

En cuanto al software utilizado, el más relevante se describe a continuación:

- **Android Studio** [32] Entorno de desarrollo oficial de Android desarrollado por Google.

- **Firefox** [33] Navegador web multipropósito de código abierto desarrollado por Mozilla.
- **Notepad++** [34] Editor de texto multipropósito con soporte sintáctico para múltiples lenguajes.
- **SourceTree** [35] Herramienta para la gestión eficiente de repositorios Git, con una GUI limpia.
- **Gitlab** [36] Repositorio remoto de la escuela de Ingeniería Informática de la UVa, donde está alojado el código del proyecto.
- **Teams** [37] Plataforma de comunicación con los tutores, desarrollada por Microsoft. Permite el almacenamiento de archivos y conversaciones por texto, voz y vídeo.
- **Overleaf** [38] Editor web del sistema de composición de textos LaTeX, utilizado para escribir esta memoria.
- **Astah** [39] Software privativo de creación de diagramas UML. Utilizado bajo la licencia de la UVa.
- **LightBlue** [30] Aplicación utilizada en el smartphone para configurar las balizas BLE desarrollada por Punch Through Design.
- **nRF Connect** [31] Aplicación utilizada en el smartphone para configurar las balizas BLE desarrollada por Nordic Semiconductor ASA.

2.3. Análisis de riesgos

Todo proyecto por definición tiene un grado de incertidumbre sobre el éxito o fracaso del mismo. En esta sección se presenta el análisis de los riesgos más comunes previstos (tab. 2.4). En la tabla se puede ver el nombre del riesgo, una breve descripción del mismo, la probabilidad estimada de que el riesgo se materialice y el retraso estimado en días de trabajo que provocaría la materialización de dicho riesgo. De la misma manera, se presenta también una tabla con el plan de acción desarrollado para tales riesgos (tab. 2.5). Esta tabla se compone del riesgo en cuestión, la exposición al mismo calculado como el producto entre su probabilidad y el retraso que conllevaría, el plan de reducción de la probabilidad de que el riesgo se materialice y el plan de mitigación que se llevaría a cabo si se materializara el riesgo.

ID	Riesgo	Descripción	Probabilidad	Retraso (Días)
1	Enfermedad	Contraigo una enfermedad que me incapacite para el desarrollo	0.1	3
2	Falta de experiencia	La falta de experiencia con alguna de las tecnologías de desarrollo implica emplear más tiempo	0.25	7
3	Planificación incorrecta	Una mala planificación implica emplear más tiempo recuperando el tiempo perdido	0.3	15
4	Inutilización de la estación de trabajo	Si el portátil o algún material se perdiera o rompiera, habría que reponerlo	0.05	7
5	Cambio de los requisitos del proyecto	Si los requisitos cambian después de la implementación significará que se ha empleado tiempo y recursos en trabajo sin utilidad	0.2	15
6	Problemas o incapacidad para implementar	Si hubiera alguna funcionalidad difícil, imposible o simplemente inviable, se habría perdido tiempo en ello	0.25	15
7	Pérdida de datos	Retraso debido a la pérdida accidental de avances no guardados del proyecto	0.1	4

Tabla 2.4: Análisis de riesgos

ID	Riesgo	Exposición	Plan de reducción	Plan de mitigación
1	Enfermedad	0.3	Tener cuidado con los contagios y llevar una vida saludable	No trabajar mientras estoy enfermo y descansar al máximo, así como emplear horas extras los siguientes días tras la enfermedad
2	Falta de experiencia	1.75	Dedicar el tiempo libre a familiarizarse con las tecnologías empleadas en lugar de hacerlo durante el desarrollo	Dedicar horas extras para cubrir la falta de experiencia
3	Planificación incorrecta	4.5	Estimar los tiempos de forma pesimista	Dedicar horas extras
4	Inutilización de la estación de trabajo	0.35	Tener cuidado con el material utilizado	Comprar material nuevo o intentar reparar el estropeado
5	Cambio de los requisitos del proyecto	3	Hacer el trabajo según la planificación y por orden de importancia, así como tener buena comunicación con los tutores para asegurarse de que no se trabaja en vano	Emplear horas extras
6	Problemas o incapacidad para implementar	3.75	Informarse cuanto antes de la sistemática e implementación de los módulos, así como sus limitaciones, para verficar si hay algo no viable	Recuperar el tiempo perdido haciendo horas extras
7	Pérdida de datos	0.4	Guardar a menudo y mantener una copia de seguridad en la nube actualizada	Utilizar tiempo adicional para rehacer lo perdido

Tabla 2.5: Plan de acción

2.4. Estimación de costes del proyecto

En esta sección se hará la estimación de costes del proyecto, suponiendo que fuera un proyecto financiado. Los costes se podrían dividir en 4 ámbitos diferentes, costes de hardware, costes de recursos humanos, costes del espacio de trabajo y costes de licencias. Los costes de hardware (tab. 2.6) se pueden considerar como coste amortizado, ya que se disponía de ellos de antemano. La excepción a esto son las balizas y el smartwatch, proporcionados por D Alfonso Bahillo. El coste amortizado se calcula dividiendo las horas de uso del material por sus horas de vida total y multiplicando por su coste original. Sabiendo que el proyecto debe durar al menos 300 horas, podemos prever hasta un 20 % más de este tiempo como margen, es decir, unas 360 horas. Considerando que el portátil costó 750€ en su momento y tiene una vida media aproximada de 6 años, el coste amortizado sería el siguiente:

$$\left(\frac{360}{6 \times 8760}\right) \times 750€ = 5.14€$$

De igual manera se puede calcular el coste amortizado del móvil suponiendo un 25 % de uso en el proyecto, una vida media de 4 años y un coste original de 150€:

$$\left(\frac{90}{4 \times 8760}\right) \times 150€ = 0.39€$$

Material	Coste amortizado	Coste
Lenovo Ideapad 330	5.14 €	750 €
LG K40	0.39 €	150 €
Oppo Watch 41mm	-	200 €
Beacon Global Tag	-	2 €
Beacon Minew Tech	-	1 €
Coste total	208.53 €	1103 €

Tabla 2.6: Costes de hardware

Como se puede ver, el hardware costaría 1103€ desde cero si fuera completamente nuevo o 208.53€ contando con el equipamiento inicial. Con respecto a los costes de recursos humanos, se considera el sueldo medio de un desarrollador junior Android en España en torno a los 12.50€ [40] [41] la hora, por lo que si el tiempo estimado del proyecto son 360 horas, el coste de recursos humanos será 4500€. El coste del espacio de trabajo incluye la luz y el internet, ya que se trabaja en el domicilio personal. Estos se pueden considerar 40€ al mes, durante los 5 meses más activos del proyecto, para un total de unos 200€. En cuanto a las licencias, la única relevante es la del Astah, pero la paga la UVa. Su coste real sería de 57€ [42]. En la tabla 2.7 se puede ver el coste total del proyecto y el coste real si no se dispusiera de ningún equipo de antemano.

Tipo de gasto	Coste	Coste desde cero
Hardware	208.53 €	1103 €
Recursos humanos	4500 €	4500 €
Espacio de trabajo	200 €	200 €
Licencias	0 €	57 €
Coste total	**4908.53 €**	**5860 €**

Tabla 2.7: Costes del proyecto

2.5. Planificación final

En esta sección se detalla la planificación final, tanto temporal como de trabajo realizado, y las modificaciones que se han dado con respecto a la planificación inicial tras la realización del proyecto.

No se ha llevado un seguimiento estricto de las horas realizadas, pero se estiman superiores a 300, posiblemente alrededor de 360, dados los obstáculos encontrados descritos en el capítulo 3. En la figura 2.3 se puede apreciar la comparativa entre las horas horas estimadas originalmente y las horas reales invertidas.

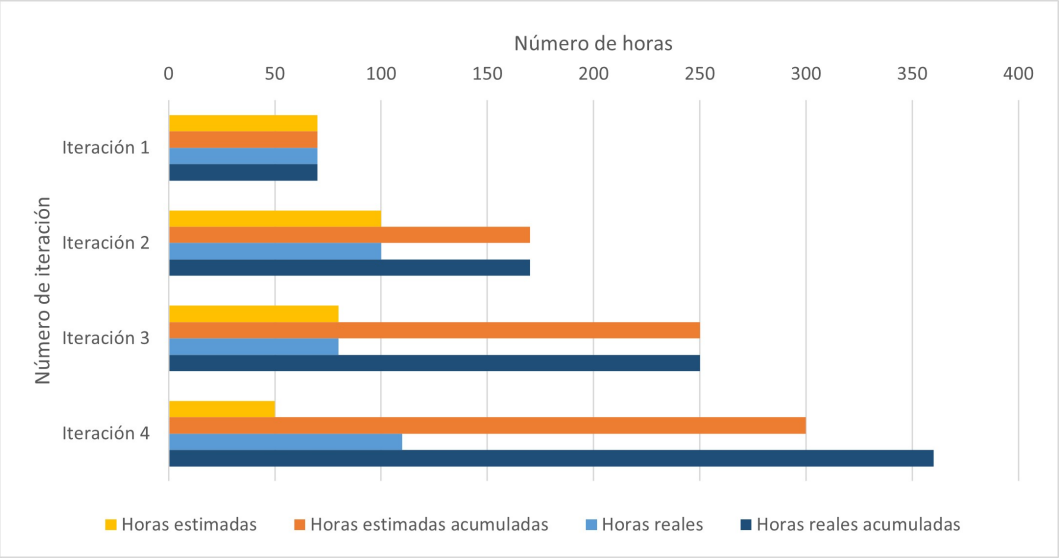


Figura 2.3: Comparativa de horas estimadas vs horas reales invertidas

El tiempo empleado en el 1er bloque se ha cumplido, ya que las tareas a realizar eran sencillas y no supusieron mayor problema. El tiempo empleado en el 2do bloque también se ha cumplido, ya que tanto las prácticas de empresa como la asignatura restante resultaron sencillas y no entorpecieron el tiempo planificado. En contraste, el tiempo empleado en el 3er bloque ha sido mayor a las 168 horas planteadas debido a los problemas encontrados. El TFG se ha entregado en la convocatoria extraordinaria más tarde del 3 de julio y por falta de tiempo no se han podido pulir ciertos detalles. Sin embargo considero que he hecho bastante más de lo que los objetivos iniciales pedían, por lo que lo tomo como un trabajo exitoso. Existe la posibilidad de finalizar las áreas menos refinadas de forma externa a la docencia de la Universidad de Valladolid más adelante en el verano de 2023.

Capítulo 3

Descripción de las iteraciones

En este capítulo se explican las iteraciones realizadas y los problemas encontrados al realizarlas. Cabe destacar que el planteamiento por iteraciones no fue del todo óptimo, pues esta metodología implica finalizar el trabajo de una iteración antes de proceder con la siguiente, y en este caso no fue así. Como se explicará más adelante, se pausó el trabajo de algunas iteraciones para completarlo en instancias más avanzadas de proyecto.

3.1. Primera iteración

3.1.1. Reconocimiento de Voz

A lo largo de la primera iteración se trató de implementar un sistema de reconocimiento de voz que escuchara constantemente el audio ambiental captado por el micrófono del smartwatch y que detectara el habla del paciente, de modo que al decir en alto una palabra clave, por ejemplo “test”, “rock” o “micción” para la fase de desarrollo, se iniciara la grabación del audio de la micción.

La aproximación al problema fue en primer lugar utilizar el reconocimiento nativo del sistema Android. Para esta iteración el objetivo fue crear una aplicación de prueba que reconociera voz constantemente y plasmara el texto detectado en pantalla, al tiempo que aumentara un contador de “coincidencias” cada vez que en el texto aparecía una de las palabras clave de prueba anteriormente mencionadas. El sentido de esa elección de palabras en particular fue simple: dos palabras que se dicen y pronuncian de igual manera en inglés y en español, y otra en español relevante a nuestra cuestión. De esta manera se podía comprobar fácilmente si se estaba reconociendo en inglés o español durante la fase de desarrollo. Una vez esta aplicación funcionara correctamente, sólo habría que portar este código a la aplicación Urosound.

Para implementar el reconocimiento de voz [43], se hizo uso de la clase SpeechRecognizer [44], que permite grabar audio durante un cierto tiempo no modificable por el programador y que devuelve un callback que puede recogerse mediante la interfaz RecognitionListener [45] [46], que implementa una serie

```
speechRecognizer = SpeechRecognizer.createSpeechRecognizer( context: this)
val speechRecognizerIntent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)
speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS, value: true);
speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_PREFER_OFFLINE, value: true);
speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, "en_US")
speechRecognizer!!.setRecognitionListener(object : android.speech.RecognitionListener {
    override fun onReadyForSpeech(bundle: Bundle) {
        Log.i( tag: "SpeechRecognizer", msg: "onReadyForSpeech")
    }
```

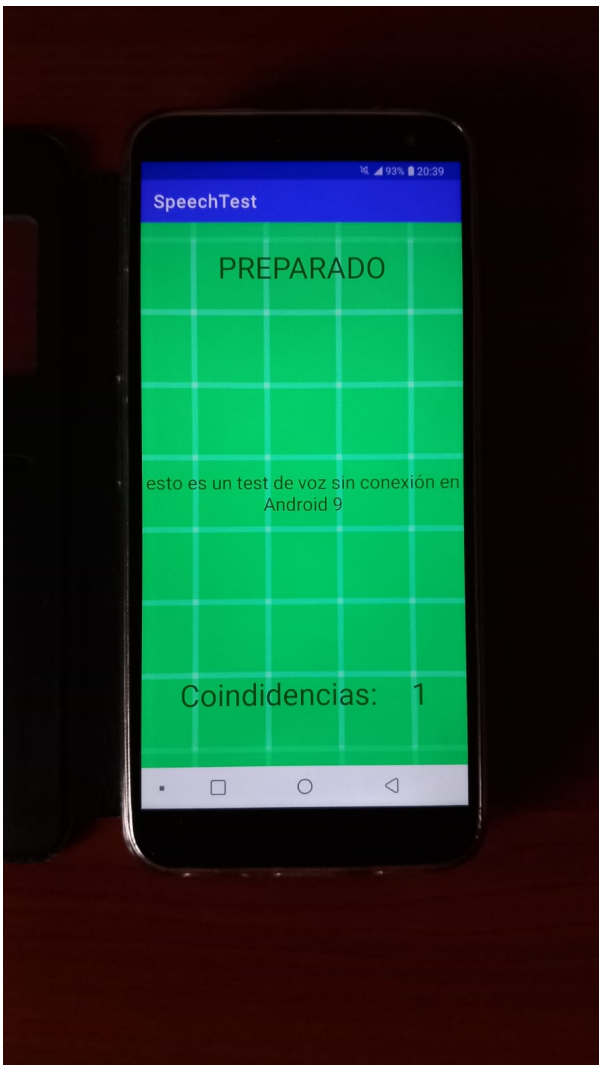
Figura 3.1: Parametrización del reconocimiento de voz en inglés

de métodos de eventos que se pueden dar en el proceso del reconocimiento de voz. Dichos eventos son onBeginningOfSpeech, onBufferReceived, onEndOfSegmentedSession, onEndOfSpeech, onError, onEvent, onLanguageDetection, onPartialResults, onReadyForSpeech, onResults, onRmsChanged y onSegmentResults. Los eventos relevantes para nuestra cuestión son onError, onResults y onPartialResults. La idea es que para conseguir que el reconocimiento de voz sea "constante" en el sentido de perpetuo hasta que se detecte una palabra clave, se ha decidido reiniciar el reconocedor cada vez que termine, es decir, en los eventos onError y onResults. Por otra parte tanto en onResults como en onPartialResults se obtiene el texto detectado y se comprueba si contiene al menos una palabra clave. De ser así, se cierra el reconocedor inmediatamente y se comienza la grabación. Esta implementación se llevó a cabo siguiendo unas cuantas guías online [47] [48] [49].

Una vez conseguido esto, también se deseaba que el reconocimiento fuera offline para no depender del acceso a la red y asegurar que iba a estar siempre disponible. Desgraciadamente, no se pudo implementar esta parte debido a problemas con el smartwatch. Al parecer, el reconocimiento de voz sólo funciona de forma online para nuestro planteamiento en particular, ya que cuando se parametriza la instancia de SpeechRecognizer para reconocimiento exclusivamente offline, siempre devuelve el evento onError con el código ERROR_NETWORK (0x00000002), creando un bucle infinito al reiniciarse constantemente el reconocedor y dar error de conexión a internet. Esto puede ser por múltiples motivos, según la implementación interna del reconocedor. Aparte de esto, el texto reconocido compartía hasta tres idiomas diferentes, japonés, chino y español, a pesar de parametrizarlo múltiples veces para "sólo español", "sólo inglés" o "sólo el idioma del sistema" (seleccionado en el smartwatch como español). En la figura 3.2a se puede ver texto en japonés.

Ante unos resultados tan desconcertantes y para tratar de solventar este problema, se portó el mismo código a Android base en lugar de Wear OS y se instaló en mi móvil personal para hacer pruebas. Aparentemente, ese mismo código, no sólo funcionaba exclusivamente en el idioma seleccionado, sino que además también de manera offline sin ningún tipo de problema, además de reducirse los tiempos de reconocimiento (ver figura 3.2b). Estos resultados declaran que el problema es del propio Oppo, o bien alguna configuración de usuario (aunque es improbable, dado que se ha revisado exhaustivamente) o alguna implementación de este sistema en Wear OS o en la capa del fabricante. Por otra parte, se descubrió que si el smartwatch estaba conectado por Bluetooth al móvil, el reconocimiento de voz se comportaba exactamente como debería (en español, con tiempos de espera razonables, etc), ya que éste se estaba computando automáticamente en el móvil y pasando por Bluetooth al smartwatch de

(a) Reconocimiento de la palabra “gracias” en japonés (arigato)



(b) Aplicación de prueba portada al móvil

Figura 3.2: Pruebas de la API de reconocimiento de voz nativa

manera espontánea.

También se trató de instalar los paquetes de reconocimiento de voz offline en el smartwatch sin éxito, ya que desde la propia Play Store sólo estaban disponibles los de TTS (Text-To-Speech), y no los de STT, además de no aparecer ninguna opción en los ajustes. Por último se intentó instalar estos paquetes dinámicamente desde el código pero tampoco fue posible. Finalmente, y ante la incertidumbre de estos errores, se decidió buscar otras alternativas [50] [51] [52] [53] [54] al reconocimiento nativo, como se explicará en la iteración 4. Se puede encontrar información adicional acerca de lo hablado en esta sección en la bibliografía [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75].

3.1.2. Permisos de *root* en el smartwatch

Por otra parte, en esta iteración también se intentó *rootear* el smartwatch para obtener permisos de superusuario en el dispositivo y poder desactivar aplicaciones y servicios innecesarios a la casuística de nuestro problema, para intentar aumentar la autonomía del dispositivo lo máximo posible. Esto en última instancia no se terminó llevando a cabo, ya que existía el riesgo de dejar el dispositivo inutilizable en caso de cometer algún error, y se decidió dejar para el final. Sin embargo, por problemas de tiempo no se realizó el proceso; no obstante sí que se hizo una investigación sobre la materia y a continuación se proporciona la información y enlaces más relevantes al respecto.

Rootear u obtener permisos de Root o Superusuario en Android es el proceso por el cual se aprovechan fallas de seguridad o exploits de un dispositivo Android para desbloquear las restricciones impuestas por los operadores de telefonía móvil y los fabricantes de hardware en algunos dispositivos y conseguir en última instancia privilegios de Root, es decir, acceso a la cuenta de usuario Unix que posee todos los derechos en todos los modos (monousuario o multiusuario). El proceso análogo para los dispositivos que corren iOS se llama *jailbreak*.

Las razones por las que se lleva a cabo este proceso pueden ser variadas, pero generalmente incluyen la instalación de ROMs o lanzadores personalizados que modifican el propio sistema operativo, la desinstalación de aplicaciones *bloatware* o capas de personalización impuestas por los fabricantes o incluso la desinstalación de las propias aplicaciones base del sistema operativo, el acceso a funciones protegidas, como acceso completo al sistema de archivos, capacidad para hacer copias de seguridad del sistema entero, sacar capturas de pantalla en estados no permitidos, o la instalación de aplicaciones que gestionen el hardware, como la frecuencia del procesador o monitores de batería a bajo nivel.

Cabe destacar que no todo son ventajas al rootear un dispositivo Android. Este proceso es en cierto modo una clase de modificación no oficial que anula la garantía del fabricante y por lo tanto, también pierde el soporte técnico en caso de necesitarlo en un futuro. Además, es posible que no sea posible instalar nuevas actualizaciones del sistema, y aumenta el riesgo de que se instale malware en el dispositivo así como aumenta el daño que éste pueda hacer, al disponer de permisos de superusuario. Del mismo modo, hay aplicaciones que dejan de funcionar si detectan que se dispone de permisos de superusuario, generalmente aplicaciones bancarias como Google Pay. Aparte de esto, también existe la posibilidad de bloquear o inutilizar el dispositivo si hubiera algún problema durante el proceso de rooteo, por lo que sigue siendo peligroso el hecho de intentarlo.

A continuación se describen los pasos que se deben seguir en líneas generales para rootear un disposivo Android:

1. **Desbloquear el gestor de arranque (Bootloader Unlock).** Es objetivo de esto es poder instalar software personalizado en el dispositivo. Este proceso es específico según el fabricante y el modelo particular. En el caso de nuestro reloj Oppo, hay poca documentación al respecto, siendo la mayor parte documentación no oficial proveniente de una serie de posts en el foro de hacking *XDA-Developers* con enlaces a tutoriales de YouTube. Estos posts, escritos por un usuario malasio con nickname *nelikp* que posee un Oppo OW19W8AP (versión asiática, AP significa Asia-

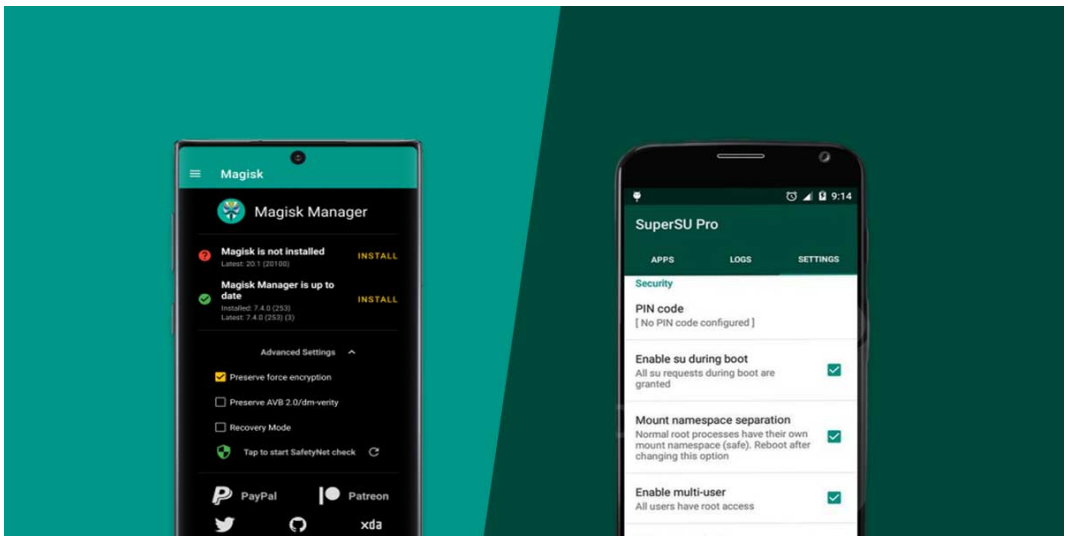


Figura 3.3: Interfaz de Magisk (izquierda) y SuperSU (derecha) en un móvil Android

Pacific), tienen baja calidad y explicaciones, pero es casi la única documentación disponible para este modelo concreto. El modelo que utiliza este usuario es el de 46mm, y a pesar de tener el codename “beluga” (el mismo que el nuestro), es posible que el sistema que emplea él no sea compatible con nuestro modelo OW19W6EU de 41mm. La documentación encontrada sobre esta cuestión está en la bibliografía [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96] .

2. **Instalar una imagen de modo de recuperación personalizada (Custom Recovery).** Tras desbloquear el gestor de arranque, el siguiente paso es instalar una imagen de recuperación personalizada. La más utilizada en general es TWRP (Team Win Recovery Project) [97] [98] [99], con la que se pueden hacer copias de seguridad del sistema, instalar ROMs personalizadas y acceder a archivos del sistema, entre otras funciones. En nuestro caso concreto, no hay ninguna imagen oficial para nuestro modelo en la web de TWRP, pero sí hay un par de ellas extraoficiales en los posts de *nelikp* y en los foros de *XDA-Developers*. Tanto su proveniencia y funcionalidad son dudosas, pero existe la posibilidad de que sean funcionales.

3. **Flashear una aplicación/sistema de obtención de superusuario.** Para este último paso existen dos alternativas famosas, SuperSU [100] y Magisk [101] [102]. Existen algunas diferencias entre estas dos aplicaciones, siendo SuperSU open source y Magisk privativo. Por otro lado, SuperSU se instala reemplazando la partición del sistema por completo, mientras que Magisk reemplaza sólamente la imagen de arranque con la suya propia, manteniendo intacto el resto de archivos del sistema. Por último, Magisk tiene más funcionalidades que SuperSU, además de tener una interfaz mejor cuidada. Por lo general esta elección se suele basar en preferencia personal. En la guía de *nelikp* se utiliza Magisk.

3.2. Segunda iteración

3.2.1. Detección de proximidad con baliza BLE

A lo largo de la segunda iteración se trató de implementar un sistema detección de proximidad del paciente al inodoro mediante el uso de una baliza (beacon) Bluetooth Low Energy [103] [104] (BLE). La cauística es la siguiente: la baliza se coloca en el inodoro que el paciente utiliza para el tratamiento dentro de su hogar, y emite constantemente paquetes de "advertising" en intervalos reducidos de tiempo menores a 1 segundo. Por otro lado, el smartwatch que lleva el paciente en la muñeca, está constantemente a la escucha de paquetes de advertising BLE. Una vez los paquetes sean recibidos en el smartwatch, se filtrarán según la dirección MAC de la baliza, el major, o el minor, o cualquier combinación de éstos según la preferencia que se desee. Al mismo tiempo, se calculará la distancia aproximada del smartwatch a la baliza en función de la potencia Bluetooth recibida de los paquetes de advertising. Si los paquetes recibidos son afines a los filtros seleccionados y la distancia aproximada es igual o menor a la asignada como parámetros de antemano, entonces se puede confirmar que el paciente está cerca del inodoro, y se puede activar el siguiente paso. Cabe destacar que este "siguiente paso" varía un poco a lo largo del proyecto como se explicará más adelante.

Para esta iteración, cuando se detecta la presencia cercana del paciente, se activa el mecanismo de reconocimiento de voz implementado en la iteración anterior. Es importante recordar que la proximidad del paciente al inodoro no significa que automáticamente vaya a orinar o realizar una toma flujométrica, sin embargo *puede* ser que sí. Se toma como suposición que a lo largo del día, sólamente habrá un pequeño porcentaje de tiempo en el cual el paciente se encuentre dentro del radio de la baliza, por lo que se optimiza la batería del reloj utilizando el escaneo BLE constante en lugar del reconocimiento de voz constante, que consume más recursos. De este modo se activa sólo el reconocimiento de voz únicamente cuando el paciente va al baño, utilizando la palabra clave como confirmación directa para comenzar la grabación de la micción. Esto también evita que una mala elección de palabra clave o una conversación personal active accidentalmente la grabación flujométrica en un lugar o situación que no sea el servicio de su hogar.

La aproximación al problema fue utilizar la API nativa de Bluetooth Low Energy de Android. El objetivo de esta iteración era crear una aplicación de prueba que escuchara constantemente todos los paquetes de advertising entrantes, mostrara por pantalla la potencia de la señal (RSSI, del inglés Received Signal Strength Indicator) y de la misma manera mostrara también la distancia aproximada en función de los factores de calibración (fig. 3.4). Una vez esta aplicación funcionara correctamente, sólo habría que portar este código a la aplicación Urosound.

Para comprender el funcionamiento a más bajo nivel, es importante remarcar que en primer lugar sólo estamos utilizando la funcionalidad de paquetes de advertising BLE. Éstos generalmente se utilizan para que los dispositivos BLE se "den a conocer" al resto de dispositivos de su alrededor y posteriormente conectarse mediante una conexión GATT (del inglés, Generic ATTribute Profile) y poder así intercambiar información. En nuestro caso particular resulta irrelevante esta funcionalidad, ya que sólo se desea saber la distancia entre la baliza y el smartwatch, que se calcula en base a la potencia de la

Figura 3.4: Aplicación de prueba detectando la baliza BLE próxima al smartwatch

señal y otros parámetros, de modo que no es necesario mantener ninguna conexión extra. La fórmula para calcular la distancia entre ambos dispositivos es la de la figura 3.5. Como se puede apreciar, toma 3 valores como parámetros:

$$\text{Distancia} = 10^{\left(\frac{\text{RSSI@1m} - \text{RSSI}}{10 \times N}\right)}$$

Figura 3.5: Fórmula de la distancia aproximada

- **RSSI:** Ésta es la potencia con la que llega la señal de la baliza al smartwatch, medida directamente en el receptor Bluetooth del mismo. Su medida son los dBm, o decibelios-milivatios.
- **RSSI@1m:** También llamado “potencia medida”, es un valor constante que se tiene que haber precalculado o precalibrado de antemano. Representa la potencia de la señal en dBm situados a 1 metro de distancia de la fuente emisora (la baliza). En ciertas balizas, este valor viene adjunto

en la información de los paquetes de advertising gracias a las pruebas de los fabricantes, sin embargo muchas veces este valor varía en función del entorno y es mejor medirlo personalmente.

- **Factor Medioambiental (N):** Abreviado "N" por sus siglas en inglés (eNvironmental factor), es otro valor constante sin unidades de medida que está comprendido entre 2 y 4. Representa la densidad del entorno, o dicho de otra manera, la dificultad de las ondas de Bluetooth para atravesar el medio en el que se encuentren, siendo 2 un entorno con la menor resistencia posible y 4 un entorno con la mayor. Este parámetro es difícil de calibrar dado que depende de muchos factores, como la geometría tridimensional del entorno, la presencia de metales que reflecten las ondas, la presencia de otras señales de Bluetooth o radio que puedan causar interferencias u otros factores complejos de medir. También hay que tener en cuenta que la presencia de objetos (incluyendo el cuerpo humano) influirá en la calidad de la señal. Para nuestro caso particular hemos elegido un valor axiomático de 2.4, pero este se puede modificar en función de las necesidades del usuario.

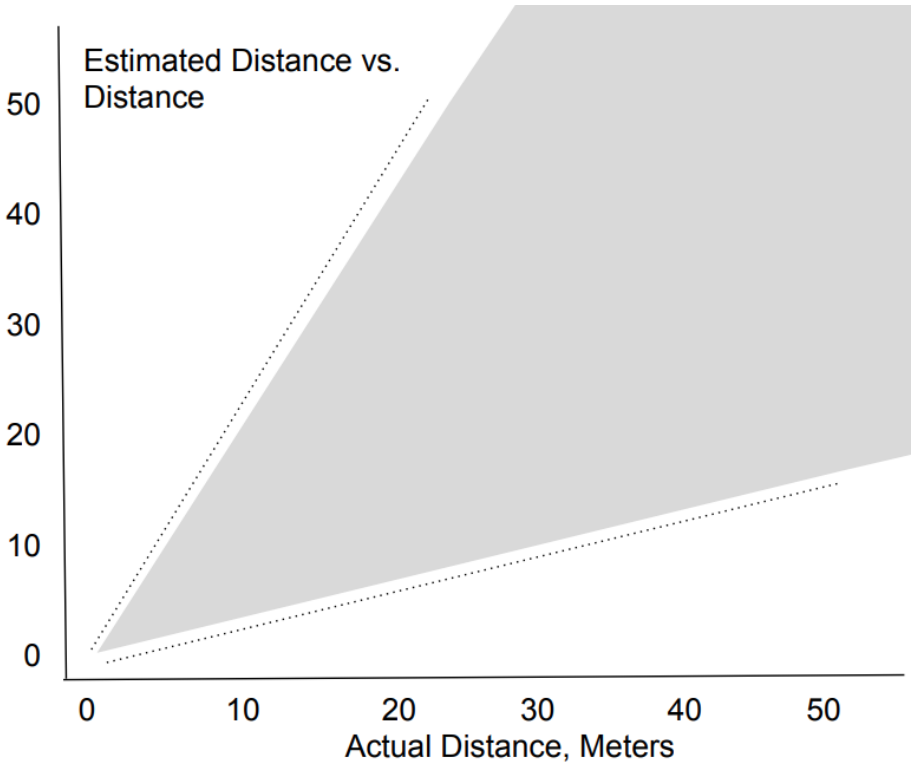


Figura 3.6: Distancia Estimada vs Distancia Real usando Bluetooth

Existen procedimientos [105] [106] para calibrar el RSSI a 1 metro para dispositivos móviles y beacons, pero a continuación se describe el método utilizado para este proyecto y su motivación. Para comprender el procedimiento de calibración, hay que tener en cuenta varias cuestiones. La primera es que la señal de Bluetooth sólo tiene una precisión aceptable desde los 1 ó 2 metros hasta los 10 metros. Existen estudios detallados sobre esta materia [107] [108] [109] [110] [111] [112]. En la figura 3.6 se puede ver una comparativa de la distancia estimada y la real utilizando ondas de Bluetooth. La segunda cuestión es que la forma y materiales del cuarto de baño de cada paciente será diferente. Por último, las personal que realizarán estas calibraciones concretas serán en primer lugar los doctores de urología o bien los propios pacientes, en caso de proceder así. En cualquier caso es inviable utilizar un

procedimiento de calibración exhaustivo, tanto por el factor humano como por las propias limitaciones de la tecnología utilizada. Es por esto que se ha decidido calibrar la potencia de la señal a 1 metro simplemente colocando la baliza BLE en la parte superior del inodoro personal y, llevando el reloj en la muñeca deseada, medir con algún instrumento de medida (en mi caso una cinta métrica) la distancia de un metro aproximado entre la baliza y el smartwatch, estando la persona con el smartwatch (yo mismo, en mi caso) de pie, en posición natural y con los brazos pegados a la cintura mirando hacia el inodoro. Cabe remarcar que no es el procedimiento más preciso posible, teniendo en cuenta que también la orientación tanto del smartwatch como de la baliza influye en la potencia de la señal, pero es un sistema relativamente sencillo para ejecutar por cualquier persona. Además de esto, por lo general la postura descrita se suele efectuar al ir a orinar, ya que normalmente se levanta la tapa del váter y en el caso de los varones, se acerca al menos una de las manos a los genitales para asistir en la micción. En cualquier caso el objetivo final es que se pueda asegurar que un paciente está próximo al inodoro.

Una vez descrita la dinámica con respecto al tratamiento de la señal y el cálculo y calibración de la distancia, se puede hablar de los paquetes de advertising Bluetooth Low Energy. En la figura 3.7 se puede apreciar la estructura que tienen estos paquetes. Como se puede apreciar, existen 3 especificaciones diferentes para el campo de los datos: iBeacon, AltBeacon y Eddystone [113] [114] [115] [116] [117]. iBeacon es un formato desarrollado por Apple en 2013, y como tal es nativo para dispositivos iOS pero también compatible con Android. Eddystone fue desarrollado por Google en 2015 como alternativa a iBeacon, y es nativo tanto para Android como para iOS. AltBeacon por su parte, fue desarrollado en 2014 por Radius Networks como una opción open source que no favorezca a ningún vendedor, aunque resultó no ser tan utilizado. En nuestro caso particular, las dos balizas eran explícitamente compatibles con iBeacon así que se utilizó este formato para la implementación. Cabe destacar que la baliza de Global Tag también era compatible con Eddystone, pero este formato es más complejo de utilizar debido a que existen 4 subtipos de paquetes (UID, EID, URL, TLM). En cualquier caso, los requisitos pedían específicamente que la implementación hiciera uso de los campos Major y Minor, sólo presentes en los paquetes iBeacon. Dado por tanto el formato que tienen los paquetes iBeacon, es lógico explicar sus campos más relevantes para nuestra cuestión:

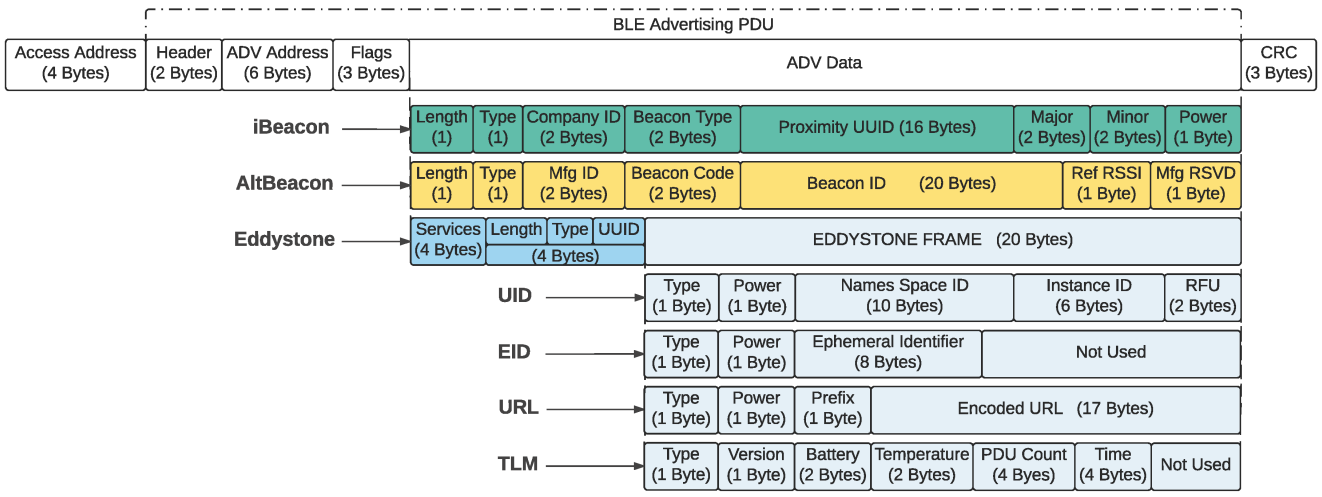


Figura 3.7: Formato de los paquetes de advertising BLE

- **Dirección MAC:** En la figura aparece bajo el nombre de “ADV Adress”, y hace referencia a la dirección física de la baliza BLE. Hay ciertas balizas que tienen la capacidad de cambiar esta dirección y hay otras que la cambian automáticamente cada X tiempo, por lo que según el modelo que se utilice, puede ser mala idea escoger este campo para filtrar los paquetes entrantes. Por fortuna, las dos balizas utilizadas tienen una dirección MAC fija. Es importante recordar que existen técnicas como el MAC Spoofing [118] [119] mediante las cuales un atacante malicioso podría cambiar la dirección MAC de algún dispositivo Bluetooth que éste poseyera y hacerse pasar por la baliza que tratamos de escanear. Afortunadamente, un ataque de este estilo para nuestro caso de uso particular es más que improbable y lo único que podría hacer es potencialmente engañar al smartwatch haciéndole pensar que se encuentra cerca del inodoro del paciente. Si bien este riesgo se puede ignorar, tampoco se puede hacer mucho al respecto, ya que el protocolo de advertising BLE carece de encriptación.
- **Company ID:** Es un campo constante que siempre tiene el mismo valor (0x4C00) independientemente del modelo y es el identificador de Apple [120].
- **UUID:** También llamado “Proximity UUID” (de sus siglas en inglés Universal Unique Identifier), es un número de 16 bytes (128 bits) utilizado generalmente para identificar unívocamente un elemento dentro de un contexto, permitiendo que coexista el mismo UUID en contextos diferentes. En el contexto de los BLEs, suele utilizarse para identificar un servicio concreto, por ejemplo un BLE que manda información sobre la temperatura de una zona concreta de un almacén. En nuestro caso particular no es necesario utilizar este valor.
- **Major:** Se trata de un número de 2 bytes, que puede por lo tanto tomar valores entre el -32768 y el 32767 o bien entre el 0 y el 65535, según la interpretación del bit de signo. Por defecto suele ser 0 y su objetivo es identificar un servicio, posición o baliza junto al otro valor Minor. Un ejemplo visual para entender esto es imaginar un centro comercial. El valor Major podría identificar la zona o sección en la que se encuentra una baliza BLE y el valor Minor podría identificar la tienda o zona dentro de la tienda en la que se encuentra dicha baliza. Esto podría ser tan simple como Major = 2, siendo 2 la zona de tiendas de ropa, y Minor = 4 identificando a una zapatería concreta. En nuestro proyecto no se sigue ninguna pauta con respecto a esto y siempre se utilizan los valores 0 y 0, pero la implementación permite parametrizarlos en función de las necesidades del mundo real.
- **Minor:** Tiene el mismo tamaño y características que el Major, y su funcionalidad está descrita en el punto anterior.
- **Tx Power:** Significa Potencia de Transmisión y es un valor constante precalculado por el fabricante que representa el RSSI a 1 metro de distancia como fue explicado anteriormente. Puede ser de utilidad para obtener la distancia aproximada si no se han aplicado calibraciones concretas o no se pueden hacer, pero suele tener un mayor error de precisión.

Con este contexto, se puede abordar el tema de la implementación en Wear OS. Para implementar el sistema de escaneo de balizas BLE se ha utilizado una guía [121] [122] muy completa al respecto. La idea principal es utilizar el servicio de escaneo BLE nativo de Android a través de la clase BluetoothLeScanner [123] y parametrizarlo con un objeto de tipo ScanFilter [124], donde se puede filtrar

```
private val bluetoothAdapter: BluetoothAdapter by lazy {
    val bluetoothManager = getSystemService(Context.BLUETOOTH_SERVICE) as BluetoothManager
    bluetoothManager.adapter  ^lazy
}

private val bleScanner by lazy {
    bluetoothAdapter.bluetoothLeScanner
}

private val scanSettings = ScanSettings.Builder()  ScanSettings.Builder
    .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)  ScanSettings.Builder!
    .setCallbackType(ScanSettings.CALLBACK_TYPE_ALL_MATCHES)
    .setMatchMode(ScanSettings.MATCH_MODE_STICKY)
    .build()

private val scanFilter = listOf(
    ScanFilter.Builder()  ScanFilter.Builder
        .setManufacturerData(
            manufacturerId: 0x004C,
            ByteBuffer.allocate( capacity: 23)
                .put(ByteArray( size: 18) { 0x00.toByte() })
                .put(ByteArray( size: 4) { 0x00.toByte() })
                .put(ByteArray( size: 1) { 0x00.toByte() })
                .array(),
            ByteBuffer.allocate( capacity: 23)
                .put(ByteArray( size: 18) { 0x00.toByte() })
                .put(ByteArray( size: 4) { 0xFF.toByte() })
                .put(ByteArray( size: 1) { 0x00.toByte() })
                .array()
        )  ScanFilter.Builder!
        .build()
```

Figura 3.8: Inicialización y parametrización del escáner BLE

por Major, Minor u otros valores. También se le da por parámetros un ScanCallback [125] para poder recibir el resultado de los escaneos. En la figura 3.8 se puede ver el fragmento de código donde se declara e inicializa el escáner. El objeto scanSettings se utiliza para asignar algunos ajustes para los escaneos. SCAN_MODE_LOW_LATENCY es el más rápido de los 3 modos posibles, pero también el que más consume. En instancias más avanzadas de la aplicación se decidió cambiar el modo a SCAN_MODE_BALANCED, para llegar a un compromiso entre el consumo de batería y la frecuencia de los escaneos. El modo que menos consume, SCAN_MODE_LOW_POWER, resultó no ser lo suficientemente frecuente para nuestro caso de uso, llegando a tardar alrededor de 10 segundos entre escaneos. CALLBACK_TYPE_ALL_MATCHES se utiliza para recoger todos los paquetes entrantes, en lugar de sólo el primero de cada dispositivo u otras opciones disponibles. Por último, MATCH_MODE_STICKY descarta paquetes cuya señal es muy tenue, ya que son irrelevantes para nuestro caso de uso. En la figura 3.8 también se puede ver que se aplica una máscara en los bytes del Major y el Minor, para

denotar que se desea filtrar por esos campos, a pesar de que los valores insertados en el filtro son 0 y 0. Para que los escaneos sean “constantes”, se utiliza un Handler que actúa sobre un Runnable. Este Runnable programa otra llamada recursiva a la misma función al cabo de 7 segundos, para el escaneo anterior en caso de seguir activo y comienza uno nuevo. La razón detrás de los 7 segundos es que Android descarta un escaneo nuevo si se han iniciado ya otros 5 escaneos en los últimos 30 segundos [126] [127]. Según esta restricción, 7 segundos deja 1 segundo de margen para que no se cancele ningún escaneo mientras que al mismo tiempo se asegura de que reinician los escaneos con la mayor frecuencia posible. Esto sólo sirve para cerciorarse de que siempre está activo, pero no influye en la frecuencia de los escaneos según el modo elegido, como se ha descrito antes. En el ScanCallback, la lógica que se ejecuta al recibir los resultados se encuentra en el método onScanResult. Allí se calcula la distancia aproximada, y si es menor que el umbral deseado (2 metros para las pruebas) entonces la pantalla del smartwatch se enciende si ésta estaba apagada, y se iniciaría el reconocimiento de voz.

3.2.2. Gestión programática de la pantalla y la CPU

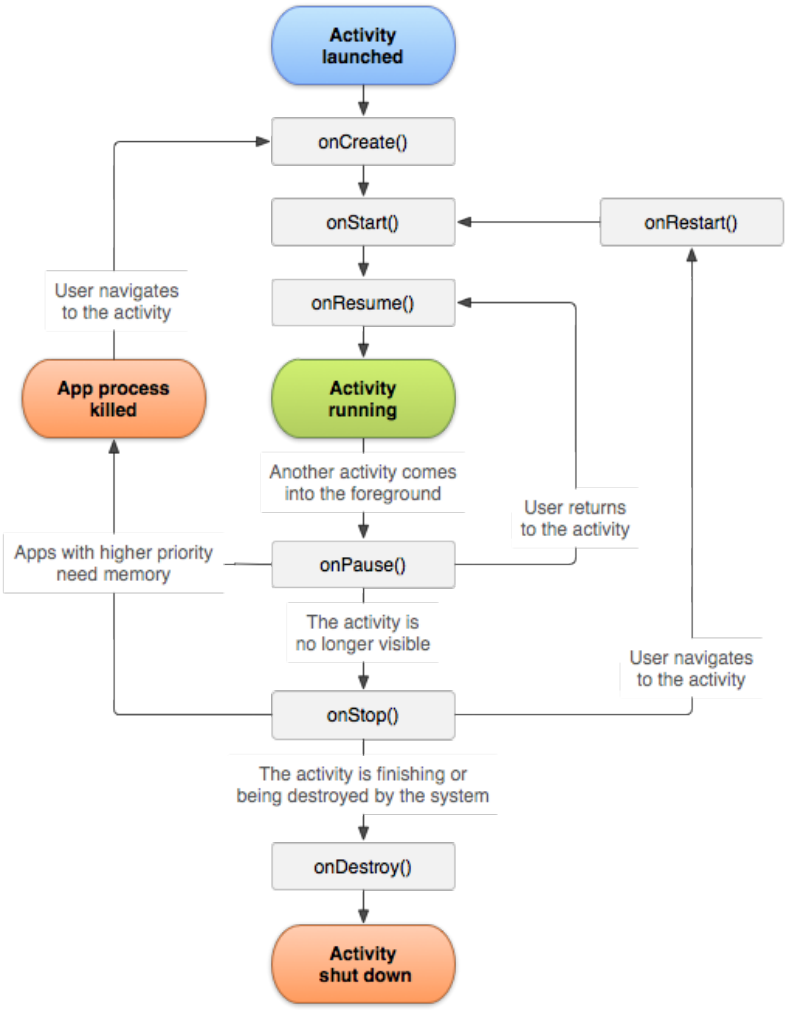


Figura 3.9: Ciclo de vida de las actividades en Android

Antes de explicar cómo se enciende la pantalla, es conveniente explicar cómo se mantienen los esca-

neos en funcionamiento con la pantalla apagada. Para esto, es relevante comprender el funcionamiento del ciclo de vida de las actividades en Android [128] (fig. 3.9). Cuando una actividad se encuentra en primer plano y el usuario apaga la pantalla del dispositivo, dicha actividad ejecuta el método onPause, seguido del método onStop ya que la actividad deja de ser visible. Cuando se vuelve a encender la pantalla, si la aplicación sigue viva, se ejecutan los métodos onRestart, onStart y onResume. Por otra parte, si una actividad nueva entra en primer plano mientras estaba la anterior, sólo se ejecuta onPause y no onStop. Esto se debe a que onStop se ejecuta únicamente cuando una actividad deja de ser visible, por ejemplo cuando se apaga la pantalla, cuando el usuario da marcha atrás o cuando se pulsa el botón Home. La parte clave es que una vez se apaga la pantalla, Android tratará de entrar en "modo de suspensión" tan pronto como le sea posible para ahorrar energía, apagando la CPU o activando el modo Doze [129] [130]. En un contexto normal, esto sería beneficioso para la autonomía de la batería, pero en nuestro caso es perjudicial, ya que deseamos que se realicen escaneos BLE constantemente a lo largo de todo el día.

La solución a esta problemática es adquirir un PartialWakeLock [131] en el método onPause y liberarlo en el método onResume. Esto permite que la CPU se mantenga activa mientras la pantalla permanece apagada. A pesar de esto, es posible que Android mate el proceso espontáneamente y sin avisar, ni siquiera ejecutando el método onDestroy [132]. La aproximación a esta cuestión se aborda en la iteración 4. Por otra parte, de igual manera la pantalla puede estar configurada desde los ajustes del sistema para que se suspenda una vez pasado un tiempo determinado. En el caso de operaciones críticas, como la grabación de los audios de micción, se ha hecho uso de banderas de ventana [133] para mantener la pantalla activa permanentemente, y liberándolas una vez termina la sección crítica. De esta forma se puede prevenir la problemática de la suspensión de la pantalla y por tanto de la CPU completamente, al tiempo que la interfaz puede proporcionar feedback relevante al usuario. Cabe remarcar que el efecto de las banderas de ventana puede ser negado directamente por el usuario si éste decide apagar la pantalla manualmente (cosa que no debería hacerse mientras se graba la micción, por lo general), pero en tal caso se seguiría accionando el PartialWakeLock comentado anteriormente.

Sólo falta concluir cómo se enciende la pantalla de forma programática una vez un escaneo detecta que el paciente está cerca de la baliza. Esta cuestión es algo más complicada de hacer de forma limpia, ya que la API de Android tiene algunos métodos deprecados. La idea utilizada originalmente (fig. 3.10) fue hacer uso del método requestDismissKeyguard [134] de la clase KeyguardManager para desbloquear automáticamente la pantalla de bloqueo. Esto sólo funciona si no hay ningún tipo de seguridad aplicada a la pantalla de bloqueo, como un pin o un patrón de desbloqueo. Además de esto, se hace un intent a una actividad invisible [135] (fig. 3.11), que aplica ciertas banderas de ventana e inmediatamente después ejecuta finish(), volviendo a la actividad original. Como la actividad no tiene componentes en su layout, es invisible, y no se nota de cara al usuario. Posteriormente este método se simplificó aplicando las banderas de ventana en la propia actividad (fig. 3.12), eliminando la necesidad de crear una actividad nueva.
Hay información adicional en la bibliografía [136] [137] [118] [119] [120] [138] [139] [140] [141] [142] [143] [144] [145].

```
val keyguardManager = getSystemService(KEYGUARD_SERVICE) as KeyguardManager
keyguardManager.requestDismissKeyguard( activity: this,  callback: null)
val intent = Intent( packageContext: this, WakeDevice::class.java)
startActivity(intent)
```

Figura 3.10: Método original para encender la pantalla

```
class WakeDevice : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_wake_device)
        setShowWhenLocked(true)
        setTurnScreenOn(true)
        window.addFlags( flags: WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON or
                WindowManager.LayoutParams.FLAG_ALLOW_LOCK_WHILE_SCREEN_ON)
        finish()
    }
}
```

Figura 3.11: WakeDevice es una actividad sin componentes en el layout, por lo tanto invisible

```
setShowWhenLocked(true)
setTurnScreenOn(true)
window.addFlags( flags: WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON or
        WindowManager.LayoutParams.FLAG_ALLOW_LOCK_WHILE_SCREEN_ON)
val keyguardManager = getSystemService(KEYGUARD_SERVICE) as KeyguardManager
keyguardManager.requestDismissKeyguard( activity: this,  callback: null)
```

Figura 3.12: Método simplificado para encender la pantalla aplicando banderas de ventana

3.3. Tercera iteración

3.3.1. Implementación del algoritmo de IA de detección de micción

A lo largo de esta iteración se trató de implementar múltiples algoritmos de detección de micción. El objetivo de estos algoritmos es clasificar un audio de breve duración como micción o no micción. El valor que aporta un algoritmo de este estilo es poder confirmar en tiempo real que el paciente está orinando sin mayor interacción del usuario. La idea es que el algoritmo se ejecute en bucle tras detectar la proximidad del paciente a la baliza colocada en el inodoro, sustituyendo al sistema de activación por palabra clave mediante reconocimiento de voz, automatizando aún más el proceso para que sea más transparente a ojos del usuario.

Cabe destacar que estos algoritmos han sido desarrollados completamente por D.ª Laura Arjona,

profesora en la Universidad de Deusto (Bilbao) y no se ha tenido parte en el desarrollo ni acceso al código fuente de los mismos. Desgraciadamente, por problemas logísticos, de disponibilidad y de tiempo, la comunicación con ella ha sido pobre y ha llevado a malentendidos con respecto al código, produciendo algunos retrasos. Sin embargo, más adelante se detalla la información relevante obtenida del trabajo con dichos algoritmos.

Originalmente estos algoritmos (descritos en el paper [146]) fueron diseñados para su ejecución en una Raspberry Pi a través de una serie de scripts en Python, que actúan sobre el algoritmo en cuestión y parametrizan la entrada y la salida. El primer planteamiento fue ejecutar directamente los scripts de Python mediante una librería para Android llamada Chaquopy [147], que permite la ejecución de un intérprete de Python de forma relativamente nativa en Android. El procedimiento para implementar dicha librería [148] es sencillo y está descrito en su página web. Se debe importar y configurar la librería a través del gestor de dependencias Gradle [149], pero la diferencia es que también se configura desde Gradle parámetros como la ruta a la instalación de Python en nuestro sistema, la versión de Python, las arquitecturas objetivo de compilación, los directorios que contienen el código fuente en Python o los módulos de Python que se desean importar por medio del comando "pip" [150] o como paquetes Wheel [151]. Es muy importante recordar que Chaquopy no tiene soporte para todos los módulos de Python que existen [152]. Según su página web:

In our most recent tests, Chaquopy could install over 90 % of the top 1000 packages on PyPI.

En nuestro caso, los scripts de Python provienen del repositorio de la aplicación AutoFlow [153], y contienen 3 algoritmos diferentes, uno de Machine Learning (ML) utilizando Joblib [154] para su ejecución y dos de Deep Learning, uno con Keras [155] y otro con Tensorflow Lite [156]. Los scripts importan ciertos módulos [157] [158], y según la lista de módulos soportados de Chaquopy [159], "tensorflow_hub", "bleak" y "protobuf" no se soportan para los algoritmos de Deep Learning, y "librosa" y "bleak" tampoco para el de Machine Learning. Como es lógico, tras la implementación ninguno de los dos pudo ser compilado y ejecutado correctamente. Sim embargo, esto sólo significó que los scripts de Python que envuelven a los algoritmos de IA no podrían ejecutarse, por lo que se intentó eliminar todas las dependencias incompatibles y cargar y ejecutar los modelos, con el objetivo de comprobar si al menos éstos se podían ejecutar aquella parametrización. El resultado fue que en efecto se podían ejecutar de principio a fin, pero la inicialización de los modelos tardaba aproximadamente unos 30 segundos y la inferencia (ejecución) de los mismos otros 30 o más segundos. Estos resultados son en cierto modo esperables, ya que se está ejecutando un modelo de IA sobre un algoritmo de Python que a su vez se ejecuta sobre una instancia virtual del intérprete de Python, que a su vez se ejecuta sobre la máquina virtual de Java de Android. Además de esto, el poder de procesamiento de un smartwatch es más bajo que el de una Raspberry Pi.

Tras discutir esta cuestión, se tomó la decisión de utilizar el algoritmo de Tensorflow Lite de forma nativa en Android, ya que existe soporte para este tipo de modelos, aunque no para Tensorflow. Tensorflow Lite al fin y al cabo fue diseñado para dispositivos móviles o de bajo poder de procesamiento como las Raspberrys. Otro punto a favor de esta decisión fue que la parametrización de la entrada no requería de módulos adicionales, no soportados por Chaquopy. Sin embargo, dicha parametrización requería del uso de funciones del paquete de Tensorflow de Python que no se encontraban en la librería

de Tensorflow Lite de Android. Esto fue un problema, ya que para inicializar la entrada del algoritmo se empleaban objetos de tipo Tensor en Python, y si bien la traducción entre tipos primitivos de Python y Java es relativamente viable, en el caso del objeto Tensor de Python fue imposible. Ante esta nueva problemática, la solución adoptada fue tratar de traducir la parametrización del script de Python a Java nativo, liberando como efecto secundario, la necesidad de ejecutar nada en Python y por tanto, borrando Chaquopy del proyecto. Sin embargo, mis escasos conocimientos sobre el ámbito de la inteligencia artificial, el hecho de no saber el funcionamiento interno de éste ni de los otros algoritmos ni tener el código fuente, y la indisponibilidad de D.ª Laura Arjona hicieron muy dificultosa esta traducción. Las directrices de las que disponía sobre el algoritmo de Tensorflow Lite eran que éste tomaba *la señal de audio tal cual* y devolvía la String "void" o "no_void" en función de si la entrada de audio se identificaba como micción o no micción tras la inferencia del algoritmo. Además de esto, según este paper [146], se debían introducir los últimos 5 segundos grabados para ejecutar la inferencia cada segundo. Sin embargo, la versión más actualizada requiere de 10 segundos. La señal del audio debe ser sin formato (PCM), el número de muestras de sonido cada segundo debe ser 16000, los bits por cada muestra deben ser 16 (es decir, 2 bytes, 1 short) y debe ser grabado en MONO, no en STEREO. En el caso de la Raspberry Pi, las grabaciones se hacen en WAV con un script y luego en el script de la inferencia se extrae la señal del sonido y se hacen algunas transformaciones antes de introducirlo en el algoritmo de Tensorflow Lite. Cabe destacar que PCM [160] es el equivalente a "audio sin formato" y WAV [161] no es más que PCM con una cabecera que contiene la información sobre la señal de audio. La toma de audio PCM se abordará con más detalle en la Iteración 4.

Hasta este momento, el algoritmo de Tensorflow Lite se ejecutaba de forma nativa, pero al no introducirle una entrada correcta, tampoco devolvía una salida correcta, siendo ésta siempre "no_void" independientemente de la entrada. A día 7 de julio (!!) finalmente descubrí la manera correcta de parametrizar el algoritmo. En primer lugar, recordemos que la entrada original de la que disponemos es un array de 160000 shorts. Esto se debe a que son:

$$1 \text{ canal (MONO)} \times 10 \text{ segundos} \times 16000 \text{ muestras por segundo} = 160000 \text{ muestras de audio totales}$$

Son de tipo short por que como se ha explicado anteriormente, son 16 bits por cada muestra, 2 bytes. Este array de shorts debe convertirse en un array de floats, que ocupan 4 bytes, y posteriormente ser normalizados entre -1 y 1. La conversión de short a float es inmediata, siendo por ejemplo el valor short 123 equivalente de forma aproximada [162] a 123.0. para normalizarlo entre -1 y 1 simplemente se divide el array de floats entre 32768.0, ya que sus valores están en el rango -32768 a 32767, puesto que han sido convertidos directamente desde el tipo short. Además de esto, resulta que no es necesario utilizar objetos de tipo Tensor [163], por lo que la entrada es sólo el array de floats normalizado (ver figura [meter foto del código de la inferencia]). Tras unas cuantas pruebas, parece que se comporta tal y como se espera, aunque tal y como advirtió D.ª Laura Arjona, este algoritmo a menudo clasifica el silencio como "void", siendo esto un comportamiento no deseado, menos aún para un sistema de detección constante de micción como el que se trata de implementar. Para solventar este problema se ha decidido implementar un cálculo RMS [164] sobre las muestras de audio, para obtener el volúmen promedio en decibelios (dB) de dicho fragmento de audio, y descartar los resultados si los decibelios no son mayores de un cierto umbral. Se ha decidido utilizar 40 dB como umbral, pero este valor debería ajustarse y hacer pruebas antes de que la aplicación pase a producción. Por otra parte, el tiempo que tarda la

ejecución del algoritmo varía entre los 3-4 segundos, por lo que ejecutarlo cada segundo es inviable, ya que la inferencia dejaría de ser en tiempo real, alejándose más del presente cuanto más tiempo pasara. Incluso aunque se implementara como una inferencia retardada, los datos no procesados se seguirían almacenando en memoria RAM y ésta es finita. Lo mismo sucede con el almacenamiento interno del smartwatch. Esto podría llegar a ser una opción viable si se toma como suposición que el paciente no va a estar mucho tiempo *dentro del rango de acción de la baliza BLE*. Incluso entonces, se corre el riesgo de que la aplicación se cuelgue o se sobrecaliente el dispositivo. La otra opción sería conseguir un algoritmo que tardara menos o un smartwatch más potente. En cualquier caso eso forma parte del trabajo y la valoración de Deustotech. Por el momento, la implementación utilizada es que el algoritmo se ejecuta en bucle una vez cada 10 segundos, aunque existiría la posibilidad de ejecutarlo cada 5 segundos con los últimos 10 segundos grabados. 10 segundos son una latencia algo considerable, pero sabiendo que las micciones humanas duran un máximo de 90 segundos y que el algoritmo fue originalmente diseñado para inferir cada 5 segundos antes de pasar a ser 10, creo que el compromiso implementado es bastante respetable.

3.4. Cuarta iteración

El objetivo de esta última iteración fue integrar el código implementado en las otras iteraciones en la app Urosound.

3.4.1. Implementación de la API Vosk para reconocimiento de voz offline

Como ya se ha explicado en la primera iteración, la fiabilidad de la implementación del sistema de reconocimiento de voz nativo es baja debido al comportamiento incoherente en el smartwatch. Ante esto se buscó una alternativa y se encontró la API Vosk [165] [166] de reconocimiento de voz offline. Ésta tiene implementaciones para múltiples sistemas y lenguajes, incluyendo versiones lightweight para Raspberry Pi, Android e iOS. La instalación es muy sencilla y está explicada en su web [167] [168], donde hay ejemplos ya implementados.

Hay un par de diferencias remarcables con respecto a la API nativa. En primer lugar los modelos de lenguaje hay que descargarlos de la página oficial [169], y según el lenguaje hay más o menos modelos disponibles. En el caso del español y el inglés hay modelos portables de unos 50 Mb de tamaño y aparte otros modelos más grandes para procesamiento en un servidor, que pueden pesar varios Gb. En la aplicación se han integrado los dos modelos portables de español e inglés, aumentando en unos 100 Mb el tamaño de la aplicación. La otra diferencia es que estos modelos, al ser más pequeños, tienen menor precisión [170] y vocabulario que los de reconocimiento en servidor, y posiblemente también que los de la API nativa. Sin embargo, con una elección inteligente de palabra clave, esto no debería ser un problema.

La dinámica de la implementación es muy parecida a la nativa: en primer lugar se debe cargar el modelo de lenguaje deseado, se utiliza un objeto SpeechService que toma como parámetros una ins-

tancia de la clase Recognizer y recoge los resultados del habla en un RecognitionListener que contiene los eventos onPartialResult, onResult, onFinalResult, onError y onTimeout. De la misma manera, para que sea reconocimiento perpetuo, se reinicia cada vez que termina, y se pasa a la siguiente fase del proceso si detecta la palabra clave elegida.

3.4.2. Implementación del servicio de Bluetooth Low Energy en primer plano

El sistema de escaneos BLE es la única sección de código pensada para ejecutarse a lo largo de todo el día, aprovechando su coste energético inferior. Como tal, una buena práctica es implementar este sistema por medio de un servicio en primer plano [171] [172], ya que es la forma más robusta de asegurarse de que el proceso no se muere, es expulsado de la memoria o se suspende por políticas de ahorro de energía del sistema o por el modo Doze, a diferencia de las actividades o de los servicios en segundo plano [144], además de gastar menos recursos teóricos al no tener un layout visible.

La implementación de un servicio en primer plano [173] [174] [175] [176] se ha llevado a cabo con 4 clases adicionales:

- **BLEService:** Esta clase implementa el servicio como tal y contiene la lógica de escaneos BLE. Todos los servicios en primer plano requieren de una notificación persistente que no se puede descartar, y si se pulsa, en nuestro caso abre la actividad MainActivity. Además, si detecta que el paciente está dentro del rango de la baliza, se hace un Intent a MainActivity, ya que no se puede asegurar que esta actividad siga viva, y también se envía una señal mediante un BroadcastReceiver [177]. Ésta clase permite enviar datos entre actividades o entre aplicaciones. Es importante tanto hacer el Intent como enviar la señal, ya que no hay una manera fiable y no deprecada de asegurarse de que una actividad esté viva y además si no está viva, tampoco recibiría la señal del BroadcastReceiver. Si se hace un Intent a una actividad viva, en lugar de volver a ejecutarse el método onCreate, se ejecuta el método onNewIntent, pero sólo si se configura la etiqueta launchMode del manifiesto de la aplicación en el modo singleTask [178], el cual sólo permite una instancia de la actividad al mismo tiempo. Por otro lado, se obtiene un PartialWakeLock de forma permanente, hasta que se cierre el servicio. Recordemos que antes se obtenía y liberaba el PartialWakeLock en los métodos onPause y onResume respectivamente, pero esos sólo existen en las actividades y no en los servicios. Por otra parte, ya no se utiliza un Handler y un Runnable para ejecutar los escaneos, sino una Corrutina de Kotlin [179] en bucle, que es más recomendable. También es remarcable destacar la implementación de un BroadcastReceiver que monitoriza la señal BluetoothAdapter.ACTION_STATE_CHANGED, permitiendo saber si el Bluetooth del dispositivo se ha apagado o se ha suspendido. Esto ha sido muy útil en la fase de desarrollo, ya que ha permitido descubrir que el Bluetooth en el smartwatch Oppo en efecto se suspende aproximadamente a la media hora de no tenerlo puesto en la muñeca, haciendo que los escaneos dejen de funcionar. No se puede “despertar” al servicio de Bluetooth del dispositivo de forma programática, por lo que la única solución al respecto es llevarlo puesto siempre (lo cual es el comportamiento esperado de los pacientes de todas las maneras) o bien activar la pantalla cuando se detecte que el Bluetooth se suspende, en un intento de que se reactive él solo. Este

comportamiento se ha implementado como una opción en el panel de control del que se hablará más adelante, pero se espera que no sea necesario su uso. Esta suspensión automática del servicio de Bluetooth del dispositivo es así para este modelo de smartwatch concreto pero es posible que sucedan cosas parecidas en otros modelos. Por último, es relevante comentar el uso del valor de retorno START_STICKY [180] en el método onStartCommand, ya que éste permite que el servicio se vuelva a lanzar automáticamente en el remoto caso de que Android necesitara cerrarlo por problemas de memoria.

- **Actions:** Es una clase enumerada con dos valores, START y STOP. Estos se utilizan desde la MainActivity como parámetros al encender el servicio y apagarlo.

- **ServiceTracker** Esta clase controla el estado del servicio registrando si éste está vivo o no. Dentro de la clase hay un getter y setter para una propiedad que almacena este valor en una preferencia de la aplicación las cuales son persistentes en el almacenamiento interno. El valor de esta propiedad está definido en una enum dentro de la misma clase que tiene como valores posibles STARTED y STOPPED. Esta clase es necesaria porque como se ha comentado anteriormente, no se puede saber acerca del estado de vida de las actividades o servicios. En este caso, el tipo de servicio en primer plano que hemos implementado no puede morir espontáneamente, por lo que se puede guardar su estado en las preferencias. Esto no se cumple para las actividades ni los servicios en segundo plano, como hemos dicho anteriormente, por lo que no se podría implementar.

- **StartReceiver** Esta clase es la guinda del pastel para el objetivo de tener un servicio inmortal. Aquí se implementa un BroadcastReceiver que está a la escucha del evento global del sistema Intent.ACTION_BOOT_COMPLETED [181]. Este evento se genera cada vez que el dispositivo se enciende, y permite ejecutar código a las aplicaciones que recogen este evento, declarándolo en el manifiesto de la app y agregando el permiso correspondiente. De esta manera, cada vez que el dispositivo se reinicie, podemos reiniciar con él también el servicio BLE, cerciorándonos de que no se desactiva por error. Evidentemente, dentro de la app se ha añadido un botón que permite cerrar tanto la app como el servicio, como se verá más adelante.

3.4.3. Gestión de audio para el algoritmo de detección de micción

Como se ha explicado anteriormente, las grabaciones de Urosound manuales son en formato WAV, sin embargo, el algoritmo de Tensorflow requiere la señal de audio en formato PCM (plano). Para resolver esta cuestión se consideró ineficiente realizar todas las grabaciones en WAV, luego extraer la señal de audio y luego procesarla, ya que una vez se detectara que un fragmento de 10 segundos de audio fuera micción, habría que grabar un segundo audio hasta un máximo de 80 segundos, para cumplir con el requisito de grabar hasta 90 segundos. Esto significaría que también habría que extraer la señal de audio del segundo WAV, concatenar ambos audios y añadirles una cabecera. En su lugar, se ha empleado una versión modificada de la clase SoundRecorder [182], de una aplicación con comportamiento parecido llamada SoundWatch [183]. La versión de nuestro proyecto de esta clase utiliza una subclase RecordAudioAsyncTask que extiende a AsyncTask [184]. AsyncTask es una clase que permite

realizar trabajo en un hilo asíncrono en Android. Dentro de dicha clase se ha implementado un objeto AudioRecord [185], que graba sonido del micrófono a bajo nivel de código. En la clase RecordAudioAsyncTask se implementan los métodos de la clase abstracta AsyncTask, siendo éstos onPreExecute, doInBackground, onPostExecute y onCancelled. El método onPreExecute se ejecuta antes que el resto y sirve para inicializar los componentes. En doInBackground se hace el trabajo en cuestión, en nuestro caso recoge la señal de sonido grabada del AudioRecord en un búfer en bucle, al tiempo que la va almacenando en un fichero temporal por medio de un BufferedOutputStream. Dicha señal se ha parametrizado según los requisitos del algoritmo de IA, según se ha explicado anteriormente. Cuando se han almacenado 160000 shorts en el búfer, se ejecuta la inferencia, y si ésta devuelve "void", entonces se finaliza el proceso grabando el resto del audio, escribiendo la cabecera WAV en el archivo PCM, y mandando una señal a la MainActivity a través de un BroadcastReceiver para que actualice la interfaz. Finalmente, se ejcutaría el método onPostExecute para liberar los objetos de la memoria. Si por el contrario se cancelara la inferencia (por ejemplo si el paciente se aleja un tiempo de la baliza) o la grabación (si el paciente ha terminado la micción y quiere detenerla manualmente), se ejecutaría el método onCancelled. Para una mejor gestión de estos estados, se ha creado una enumeración local llamada State con los valores IDLE, INFERRING y RECORDING.

3.4.4. Otras mejoras para Urosound

Urosound, tal y como comentó D.ª Laura Arjona en una de las reuniones, fue una aplicación desarrollada rápidamente para obtener resultados lo antes posible. Como tal, no se ha modificado desde entonces, y he tomado esta oportunidad para desarrollar algunas mejoras para la app de forma voluntaria.

3.4.4.1. Refactorización del código existente

El código original de Urosound ha sido limpiado y refactorizado, eliminando código repetido, código innecesario y simplificando las llamadas, para estar conforme con los principios SOLID [186], KISS y DRY [187]. Además, se han mejorado los detalles de la interfaz, utilizando ConstraintLayout [188] en lugar de LinearLayout para las mismas. En el apéndice A se puede ver una comparativa del código antiguo y el nuevo. Las interfaces se pueden ver en la sección 4.5.

3.4.4.2. Panel de control mejorado

Originalmente, la aplicación constaba de dos actividades, MainActivity, que es la interfaz para los pacientes, donde pueden grabar los audios de las micciones, y SettingsActivity, que es un panel de control oculto, que permite al urólogo Asignar el código del paciente, enviar los audios al servidor y hacer una copia de seguridad local de los audios, moviéndolos al almacenamiento externo de la app en lugar del interno, inaccesible desde el explorador de archivos de Android. La manera de acceder al panel de control es pulsar 7 veces seguidas en un intervalo corto de tiempo en el texto inferior de la

MainActivity, donde pone “Urosound”. Este sistema no es infalible, pero se ha mantenido así como una decisión de diseño.

Debido al aumento de elementos configurables en esta versión de la aplicación, se ha creado un panel de control mejorado. El diseño visual es un título arriba del todo donde pone Panel de Control, y un ScrollView que actúa como lista scrolleable de las opciones. Estas opciones son representadas como botones con un icono representativo colocado a la izquierda del botón, y un texto indicando la funcionalidad de la opción. Cabe destacar que por cuestiones de diseño, al entrar en el panel de control se cierra el servicio BLE, y sólo es posible entrar en el menú si no se está grabando micción, ejecutando el algoritmo de IA o ejecutando el reconocimiento de voz, como se explica en el capítulo 4. Las opciones del Panel de Control son las siguientes:

- **ID del Paciente:** Esta opción permite registrar el código del paciente, requisito previo para la opción de Enviar Audios al Servidor. Al pulsarla, Aparece un AlertDialog [189] [190] personalizado con el nombre de la opción, un breve texto y un campo de texto para introducir el código del paciente. También dispone de dos botones, uno con la etiqueta Cancelar que descarta los cambios y otro con la etiqueta Confirmar, que guarda los cambios y muestra un breve Toast por pantalla, dando confirmación visual al usuario del guardado exitoso. Se ha elegido un AlertDialog dinámico en lugar de una nueva actividad para simplificar el código y optimizar el uso de recursos de la app. Cabe destacar que deslizar la pantalla hacia la derecha (gesto equivalente en el Oppo a volver hacia atrás), también se descartan los cambios como si se hubiera pulsado Cancelar. La manera de guardar el código del paciente, al igual que el resto de parámetros, es hacer uso de las SharedPreferences [191] de la app, que están diseñadas para almacenar valores primitivos y Strings en el almacenamiento persistente en un archivo XML.
- **Reconocimiento de Voz:** Esta opción se comporta de forma parecida a la anterior, diferenciándose en que ahora en lugar de un campo de texto en el AlertDialog, hay una lista con 3 RadioButton, que permiten seleccionar una única opción entre “[Deshabilitado]”, “Android (Online)” y “Vosk (Offline)”, para elegir el sistema de reconocimiento de voz deseado. Recordemos que si la opción elegida es “[Deshabilitado]”, se utilizará el sistema de detección de micción en tiempo real una vez que el paciente esté cerca de la baliza BLE, en lugar de activarse el reconocimiento de voz.
- **Palabra para Reconocimiento de Voz:** Esta opción tiene un campo de texto igual que la primera, y debe contener la palabra clave deseada para el mecanismo de reconocimiento elegido, siempre y cuando se haya elegido uno. Si este campo está vacío y el reconocimiento de voz está seleccionado, el reconocedor no responderá a ninguna palabra.
- **Filtrar por dirección MAC:** Esta opción incluye un campo de texto en el que se puede escribir la dirección MAC de la baliza en cuestión para aplicar a los filtros. Si el campo está vacío, no se filtrará por este campo, y si la dirección MAC está mal formateada, el filtro no funcionará y no se recibirán resultados.
- **Filtrar por Major:** Funciona de igual manera que el filtro por MAC, sólo que es un campo exclusivamente numérico, entre el -32768 y el 32767.
- **Filtrar por Minor:** Funcionamiento idéntico al filtro por Major.

- **Distancia de activación BLE:** En este caso, no hay un campo de texto en el AlertDialog, sino un SeekBar deslizable qe permite elegir un valor entre 0.5 y 5 metros para elegir la distancia a la que se considerará “cerca” del BLE y por tanto se activará el subsiguiente código correspondiente. La distancia por defecto actual es 2 metros.
- **Calibrar RSSI a 1 metro:** Esta opción no abre un AlertDialog sino una actividad llamada RSSICalibrationActivity. Esta actividad despliega la misma lógica del servicio BLE para hacer depuración. Así mismo, esta actividad presenta al usuario 3 valores en tiempo real para asistir en la calibración manual del RSSI de la baliza. Los 3 valores son el RSSI detectado proveniente de la baliza, el RSSI promedio de las últimas 10 medidas de RSSI tomadas, y el cálculo de la distancia aproximada en base a los parámetros N (Factor Medioambiental), el RSSI a 1 metro seleccionado y el RSSI obtenido en tiempo real en ese instante. Además, la interfaz cuenta con un SeekBar deslizable con el que se puede ajustar el RSSI a 1 metro, permitiendo al usuario calibrar manualmente dicho valor para el entorno concreto en el que se encuentre. Evidentemente, para realizar esta medición, el usuario debe colocar el smartwatch aproximadamente a 1 metro de distancia de la baliza.
- **Factor Medioambiental:** Esta opción es similar a la de Distancia de activación BLE en tanto en cuanto dispone de un SeekBar deslizable en el AlertDialog que permite ajustar el valor. Recordemos que este valor se comprende entre 2 y 4. El valor por defecto es 2.4, por recomendación de D.º Alfonso Bahillo.
- **Algoritmo de Reconocimiento de Micción:** Esta opción funciona como la de Reconocimiento de Voz, ya que presenta una lista de 2 RadioButton para elegir el algoritmo deseado. Actualmente, las dos opciones son TFLite (el modelo implementado) y Yamnet [192], un modelo desarrollado por Google que se pensó implementar en un momento del proyecto pero no se ha llegado a implementar. La opción de Yamnet, por tanto, no hace nada. La idea original fue implementar Yamnet debido a la problemática con la parametrización del algoritmo de TFLite, pero finalmente no fue necesario. Yamnet clasifica el audio en 521 [193] etiquetas diferentes, y la idea hubiera sido considerar micción las etiquetas “Water”, “Drip”, “Liquid” o “Splatter”, pero esto podría generar falsos positivos si el paciente se lavara las manos o se duchara, por lo que era una alternativa inferior.
- **Activar pantalla si el bluetooth se suspende:** Esta opción ha sido comentada anteriormente y permite desbloquear el dispositivo programáticamente si el Bluetooth se suspende en el dispositivo, en un intento de reactivarlo. El AlertDialog contiene un ToggleButton con el que se activa o desactiva la opción, que por defecto está desactivada.
- **Enviar Audios al Servidor:** Este botón ejecuta la funcionalidad original de enviar los audios del almacenamineto interno al servidor.
- **Crear Copia de Seguridad Local:** Este botón ejecuta la funcionalidad original de mover los audios del almacenamiento interno al externo para poder acceder a ello de forma manual a través del explorador de archivos de Android.
- **Habilitar/Deshabilitar Logs:** Esta opción también usa un ToggleButton en el AlertDialog y permite activar el registro de eventos relevantes para la depuración de la app en un archivo almacenado

en el almacenamiento interno. Estos logs han sido colocados manualmente y registran la fecha y hora del evento, el nivel de batería en ese momento, y el nombre del evento concreto. Un par de ejemplos son el inicio de un escaneo o la detección de un paciente, aunque hay más. Por defecto desactivado.

- **Idioma:** Si bien es cierto que la aplicación ha sido traducida en español y en inglés como se explica en la siguiente sección, cambiar de idioma en la aplicación es algo complicado [194], por lo que esta opción es un campo de texto que indica que el idioma de la app va en función del idioma del dispositivo, y si no coincidiera, se emplearía el inglés. Sería posible para trabajo futuro intentar que esta opción cambiara el idioma de la aplicación, pero cabe recordar que ni siquiera la traducción de la aplicación se encuetra entre los objetivos del TFG.
- **Ayuda:** Es otro campo de texto que actúa como manual de usuario local, explicando información relevante para el uso de la app.
- **Restaurar Valores por Defecto:** Es otro campo de texto, que explica cómo restaurar los valores de fábrica. Recomendable al cambiar de un paciente a otro.
- **Acerca De:** Información sobre la versión de la aplicación y los créditos.

3.4.4.3. Localización de la app en Español e Inglés

La aplicación original hacía uso de los recursos de Strings [195] para almacenar el texto de las interfaces. Se ha estandarizado el formato y simplificado la estructura, así como añadido campos nuevos tanto para texto que no había sido estandarizado como para el texto de las nuevas iteraciones. Además, se ha traducido este mismo texto al inglés, para permitir su posible uso internacional.

Referencias extras de todas las iteraciones: [196] [197] [198] [199] [200] [201] [202] [203] [204] [205] [206] [207] [153] [208] [209] [210] [211] [212] [213] [214] [215] [216] [217] [218] [219]

Capítulo 4

Estado final de la aplicación

En este capítulo se detalla el estado final de la aplicación por medio de diagramas, se explican los patrones de diseño utilizados e información relevante sobre la aplicación.

4.1. Análisis

En la fase de análisis se recopilan y determinan los requisitos del cliente. El objetivo es establecer una comprensión clara de los objetivos del proyecto, los usuarios finales, los casos de uso, los requisitos técnicos y los límites y restricciones del sistema. Los requisitos funcionales representan las funcionalidades que se quieren implementar, mientras que los no funcionales aplican restricciones concretas sobre los funcionales. Los requisitos funcionales de información por su parte, detallan la información y los datos que se van a tratar. Tras las reuniones iniciales con los tutores, se obtuvo una tabla (figura 4.1) de requisitos funcionales, no funcionales y de información originales, aunque estos cambiaron ligeramente a lo largo del proyecto.

Requisitos Funcionales	
RF01	La aplicación permitirá el uso de comandos de voz para grabar y parar la grabación (1er bloque)
RF02	La aplicación permitirá la activación automática de la grabación por proximidad (2do bloque)
RF03	La aplicación permitirá la ejecución del algoritmo de flujometría en local (3er bloque)
RF04	La aplicación permitirá el envío de los datos recogidos a un servidor externo
RF05	La aplicación permitirá usar un “modo manual” para empezar y finalizar las grabaciones (Originalmente funcional)
RF06	La aplicación permitirá eliminar audios erróneos (Originalmente funcional)
Requisitos No Funcionales	
RNF01	La aplicación debe permitir que la batería del smartwatch Oppo dure al menos 3 días seguidos sin agotarse
RNF02	Una vez se active la grabación automática, ésta deberá grabar entre 45 y 90 segundos
Requisitos Funcionales de Información	
RFI01	La aplicación guardará los audios en formato WAV para su análisis en el servidor

Tabla 4.1: Tabla de requisitos iniciales

4.1.1. Casos de uso

El diagrama de casos de uso de una aplicación permite visualizar de forma simplificada y a alto nivel las funcionalidades y el comportamiento del sistema de cara a los usuarios finales. En la figura 4.1 se puede ver el diagrama de casos de uso de la aplicación Urosound.

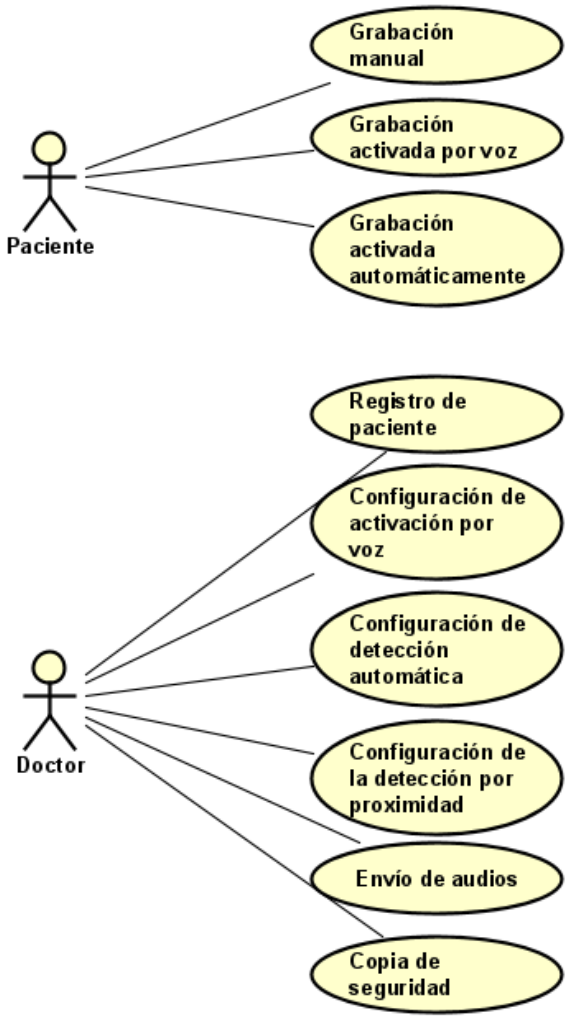


Figura 4.1: Diagrama de casos de uso de la aplicación

4.1.2. Modelo de dominio

El modelo de dominio en un proyecto de ingeniería de software permite la representación conceptual de las clases y sus relaciones. Proporciona una comprensión de los conceptos clave y las interacciones dentro del dominio de la aplicación o sistema en cuestión. En la figura 4.2 se presenta el modelo de dominio de nuestra aplicación.

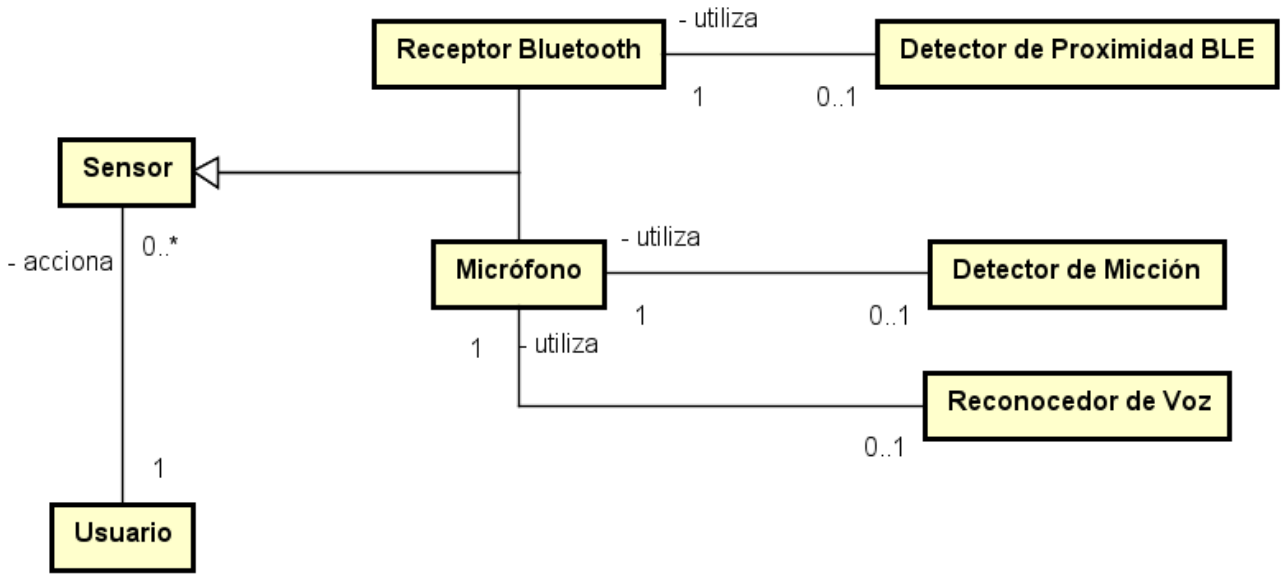


Figura 4.2: Modelo de dominio de la aplicación

4.1.3. Historias de usuario

Una manera de representar de manera simplificada los requisitos de la aplicación son las historias de usuario. Éstas son redactadas en lenguaje coloquial, por lo que permiten transmitir las necesidades, características o funcionalidad que se desea implementar desde el punto de vista del usuario final. En las tablas a continuación se muestran las historias de usuario más importantes implementadas o reimplementadas en esta versión de la aplicación Urosound. Éstas historias de usuario se componen por seis campos, el ID de la historia, el nombre de la historia, la prioridad de la misma, el riesgo de fracaso en la implementación, la descripción desde el punto de vista del usuario y una serie de reglas que permiten validar si la historia de usuario ha sido completada con éxito.

ID	HU01
Nombre	Grabación manual
Prioridad	Alta
Riesgo	Bajo
Descripción	Como paciente quiero poder grabar manualmente mis micciones
Validación	- Quiero comenzar a grabar las micciones manualmente - Quiero poder parar las grabaciones si termino antes de 90 segundos - Quiero poder elegir guardar o descartar el audio al acabar la grabación

Tabla 4.2: Historia de usuario HU01

ID	HU02
Nombre	Grabación activada por voz
Prioridad	Alta
Riesgo	Medio
Descripción	Como paciente quiero poder iniciar las grabaciones de micción al decir una palabra clave
Validación	- Quiero que se active el reconocimiento de voz una vez me encuentre cerca de la baliza BLE - Quiero que se active la grabación una vez diga la palabra clave - Quiero poder elegir guardar o descartar el audio al acabar la grabación

Tabla 4.3: Historia de usuario HU02

ID	HU03
Nombre	Grabación activada automáticamente
Prioridad	Alta
Riesgo	Alto
Descripción	Como paciente quiero que la grabación se active automáticamente cuando el algoritmo de IA detecte mi micción
Validación	- Quiero que se active el la detección de micción una vez me encuentre cerca de la baliza BLE - No debo poder parar la grabación hasta que termine una vez comience automáticamente - Quiero poder elegir guardar o descartar el audio al acabar la grabación

Tabla 4.4: Historia de usuario HU03

ID	HU04
Nombre	Registro de paciente
Prioridad	Alta
Riesgo	Bajo
Descripción	Como doctor quiero poder registrar al paciente con un código
Validación	- Quiero poder almacenar el código del paciente en la aplicación - Quiero poder registrar al paciente en el servidor remoto

Tabla 4.5: Historia de usuario HU04

ID	HU05
Nombre	Configuración de activación por voz
Prioridad	Alta
Riesgo	Bajo
Descripción	Como doctor quiero poder configurar el reconocimiento de voz
Validación	- Quiero poder elegir el sistema de reconocimiento de voz - Quiero poder elegir la palabra clave

Tabla 4.6: Historia de usuario HU05

ID	HU06
Nombre	Configuración de detección automática
Prioridad	Alta
Riesgo	Bajo
Descripción	Como doctor quiero poder configurar la detección de micción en tiempo real
Validación	- Quiero poder elegir el sistema de detección de micción

Tabla 4.7: Historia de usuario HU06

ID	HU07
Nombre	Configuración de la detección por proximidad
Prioridad	Alta
Riesgo	Bajo
Descripción	Como doctor quiero poder configurar la detección de proximidad por BLE
Validación	- Quiero poder elegir filtrar por dirección MAC - Quiero poder elegir filtrar por Major - Quiero poder elegir filtrar por Minor - Quiero poder elegir la distancia de activación - Quiero poder configurar el factor medioambiental - Quiero poder calibrar el RSSI a 1 metro

Tabla 4.8: Historia de usuario HU07

ID	HU08
Nombre	Envío de audios
Prioridad	Alta
Riesgo	Bajo
Descripción	Como doctor quiero poder enviar los audios del paciente al servidor
Validación	- Quiero poder enviar los audios grabados por el paciente al servidor externo

Tabla 4.9: Historia de usuario HU08

ID	HU09
Nombre	Copia de seguridad
Prioridad	Media
Riesgo	Bajo
Descripción	Como doctor quiero poder hacer una copia de seguridad de los audios del paciente
Validación	- Quiero poder enviar los audios grabados por el paciente al almacenamiento externo del smartwatch

Tabla 4.10: Historia de usuario HU09

4.1.4. Diagramas de actividad

Una manera de representar el flujo de ejecución de un sistema o aplicación junto con las intervenciones del usuario a alto nivel son los diagramas de actividad. A continuación se muestran los diagramas de actividad correspondientes a cada una de las historias de usuario mostradas antes, junto con una breve descripción de las mismas.

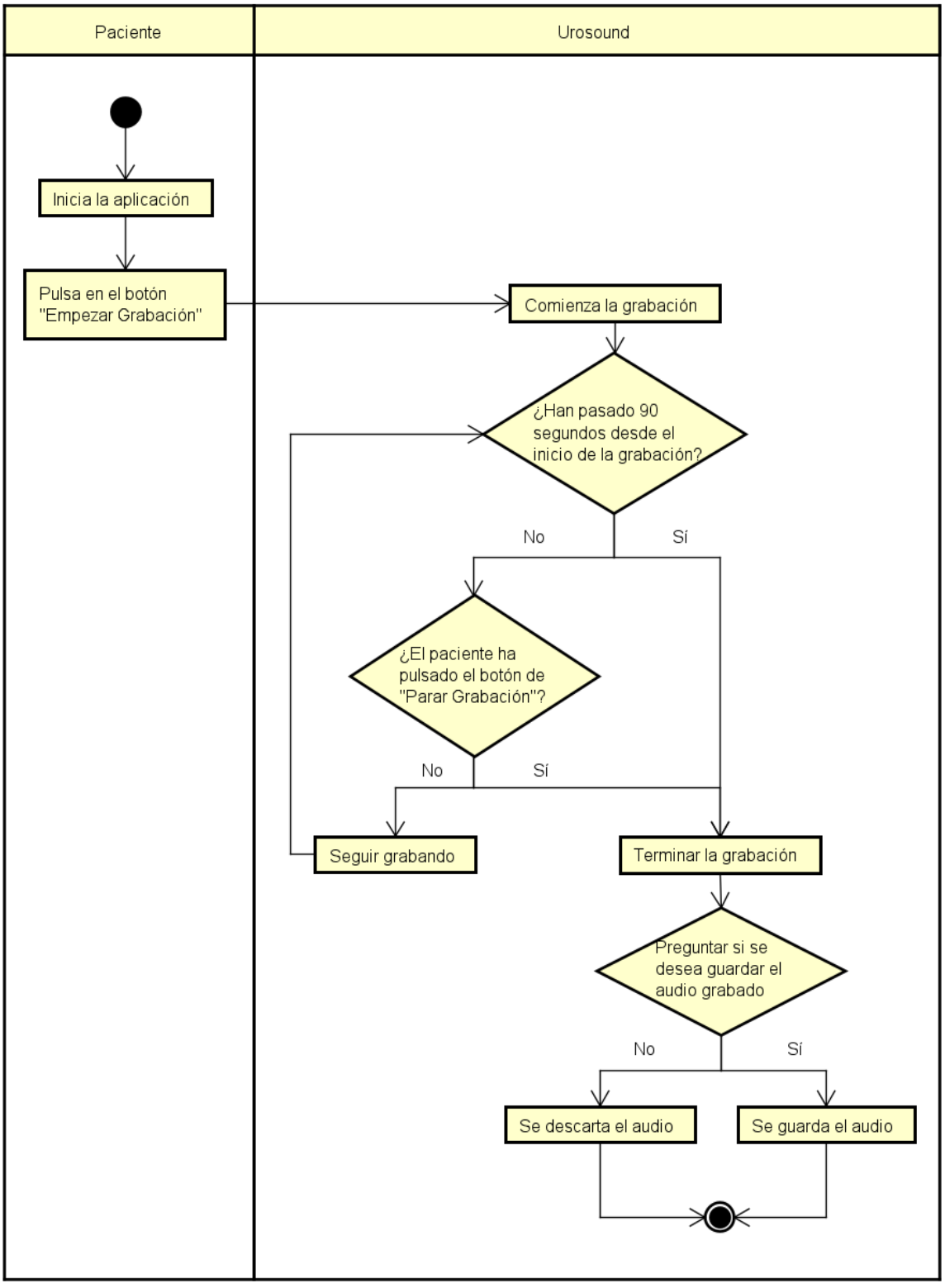


Figura 4.3: Diagrama de actividad de la historia de usuario HU01

En la historia de usuario **Grabación Manual** (fig. 4.3), el paciente pulsa el botón de “Empezar Grabación”. Después, puede volver a pulsar el botón, que ahora tiene la etiqueta “Parar Grabación”, dentro de una ventana de 90 segundos como máximo, o se parará la grabación automáticamente. Al terminar la grabación, se le presentará al paciente un diálogo para guardar o descartar el audio grabado.

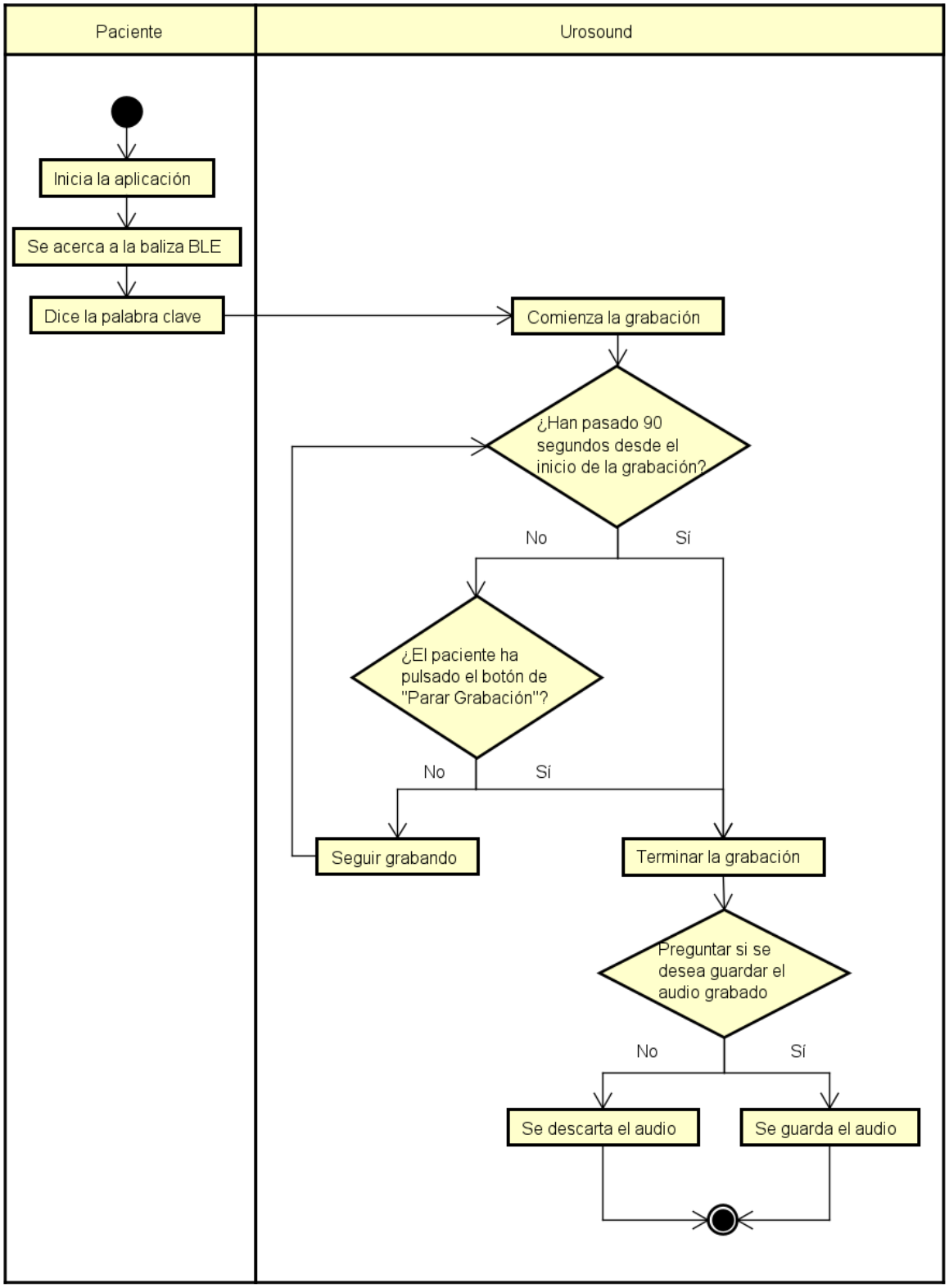


Figura 4.4: Diagrama de actividad de la historia de usuario HU02

En la historia de usuario **Grabación activada por voz** (fig. 4.4), el flujo es igual al de la historia de usuario HU01, pero esta vez el paciente comienza acercándose dentro del rango de la baliza BLE y después dice la palabra clave elegida para empezar la grabación.

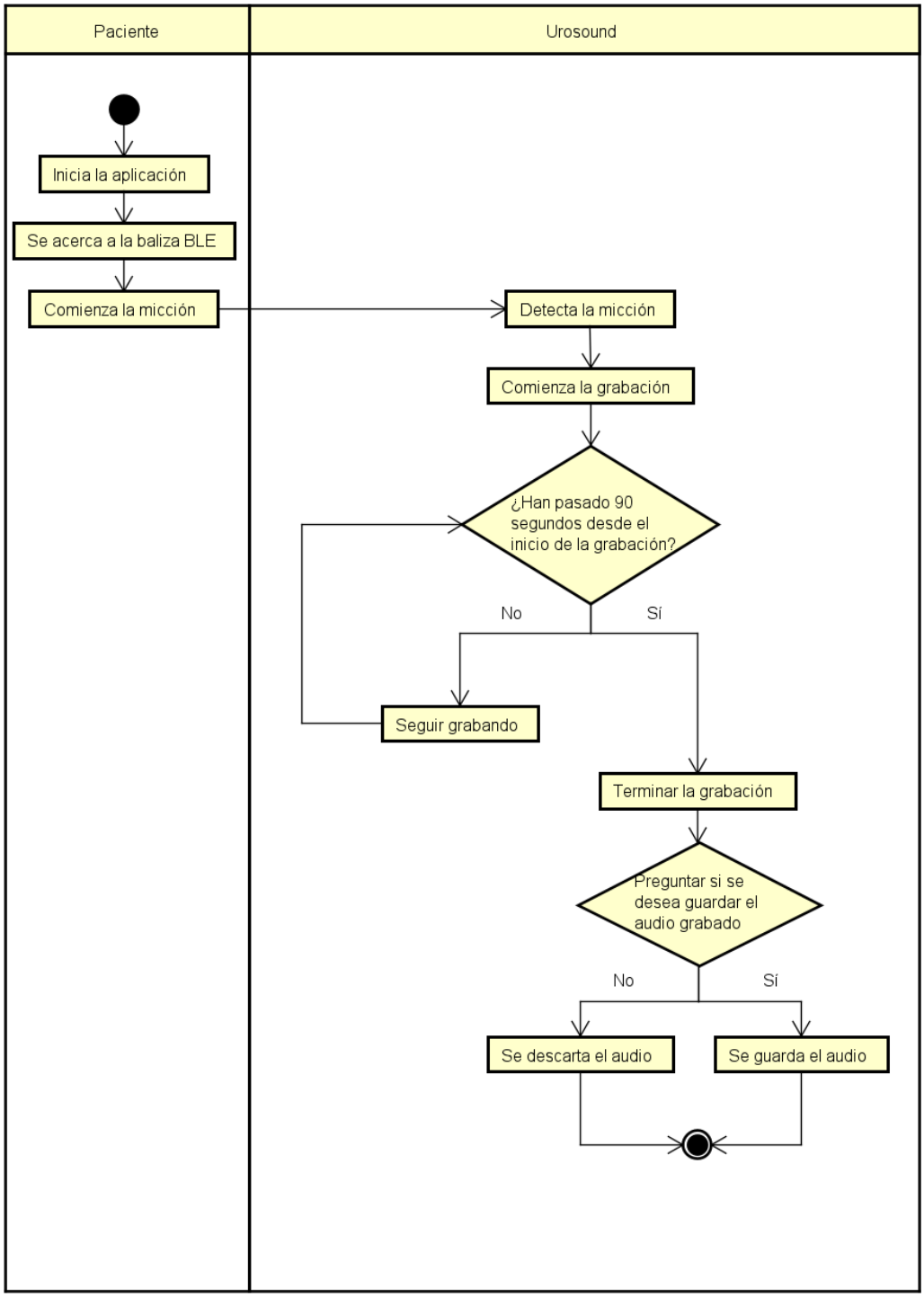


Figura 4.5: Diagrama de actividad de la historia de usuario HU03

En la historia de usuario **Grabación activada automáticamente** (fig. 4.5), el flujo es igual al de la historia de usuario HU02, sólo que en lugar de decir una palabra clave, simplemente comienza la micción y se detecta automáticamente, empezando también la grabación.

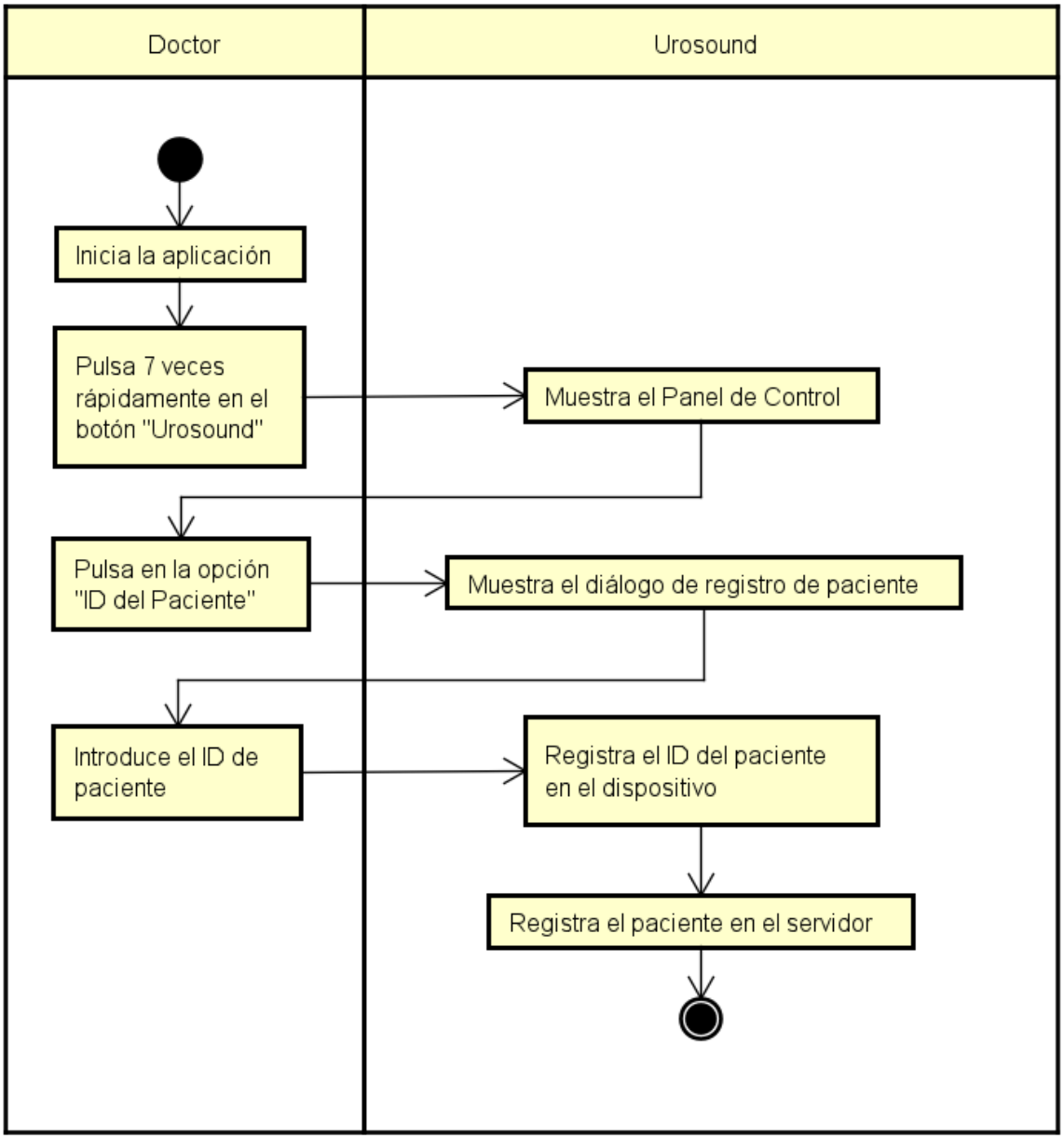


Figura 4.6: Diagrama de actividad de la historia de usuario HU04

En la historia de usuario **Registro de paciente** (fig. 4.6), el doctor accede al panel de control oculto pulsando 7 veces rápidamente en la etiqueta de “Urosound”. Después navega hasta la opción de “ID del paciente” e introduce el ID correspondiente.

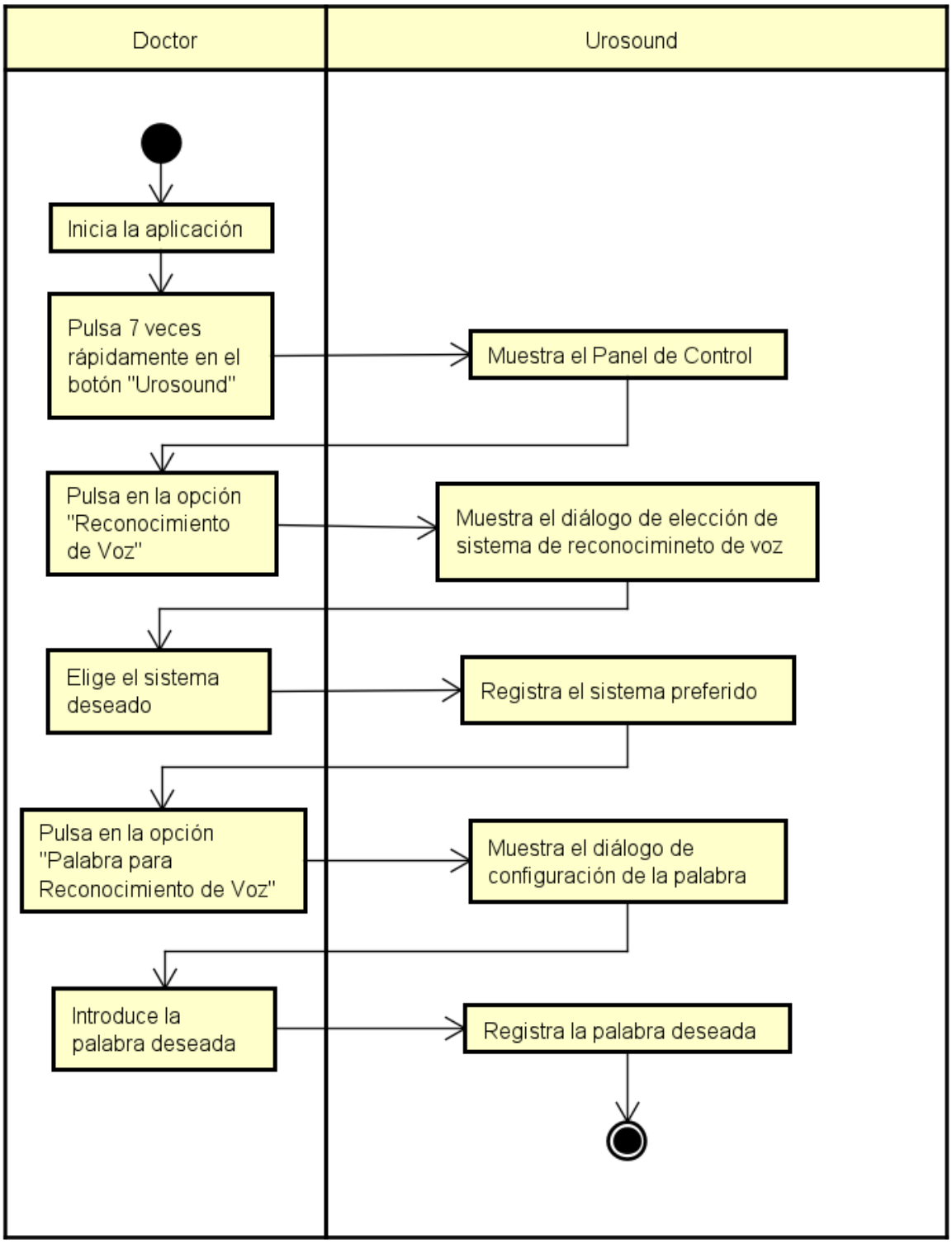


Figura 4.7: Diagrama de actividad de la historia de usuario HU05

En la historia de usuario **Configuración de activación por voz** (fig. 4.7), el doctor accede al panel de control y navega hasta las opciones de “Reconocimiento de Voz” y “Palabra para Reconocimiento de Voz” para elegir las preferencias deseadas.

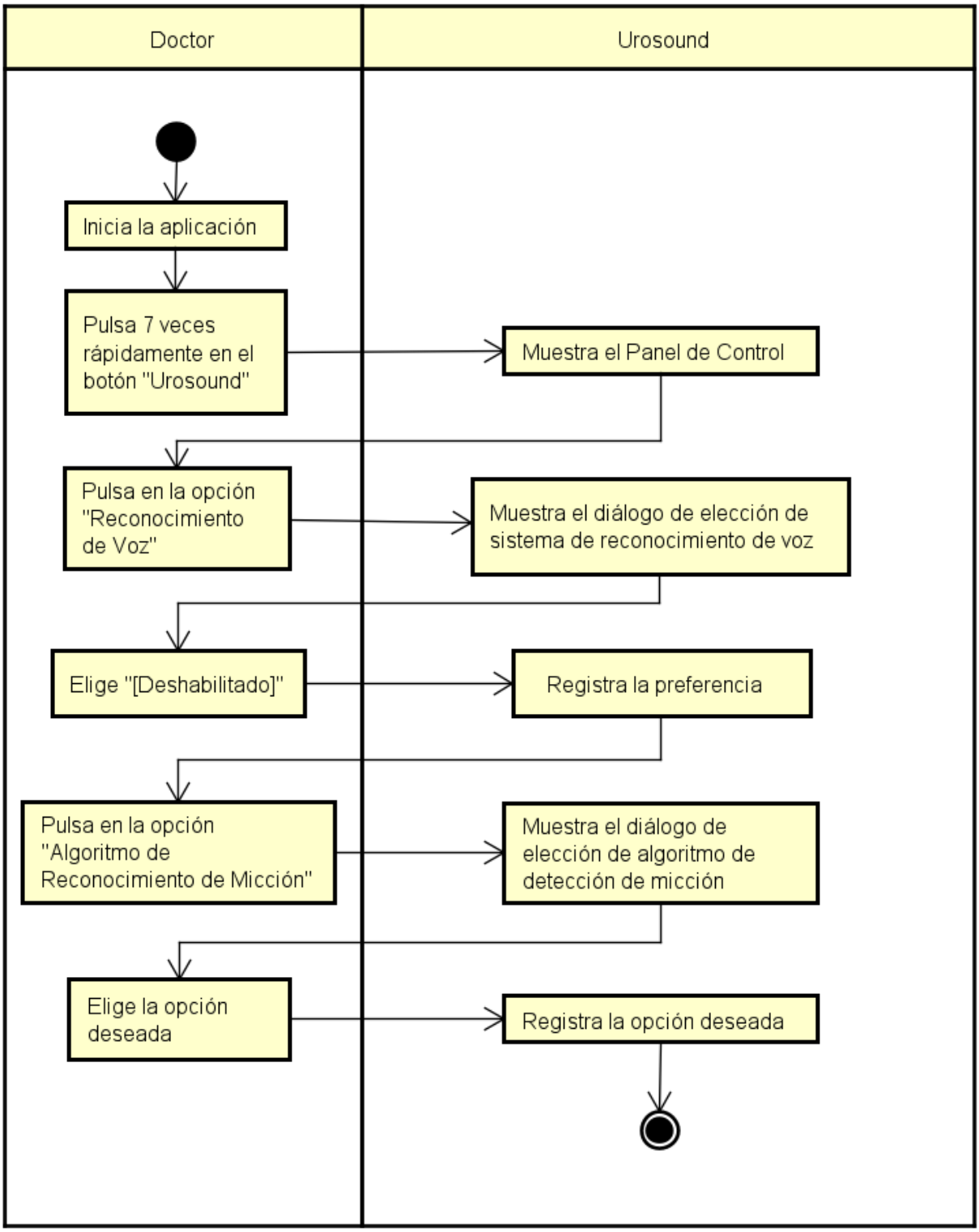


Figura 4.8: Diagrama de actividad de la historia de usuario HU06

En la historia de usuario **Configuración de detección automática** (fig. 4.8), el doctor de nuevo accede al panel de control y navega hasta las opciones de “Reconocimiento de Voz” y “Algoritmo de Reconocimiento de Micción” para elegir las preferencias deseadas. En el caso de la primera opción, se debe elegir “[Deshabilitado]”.

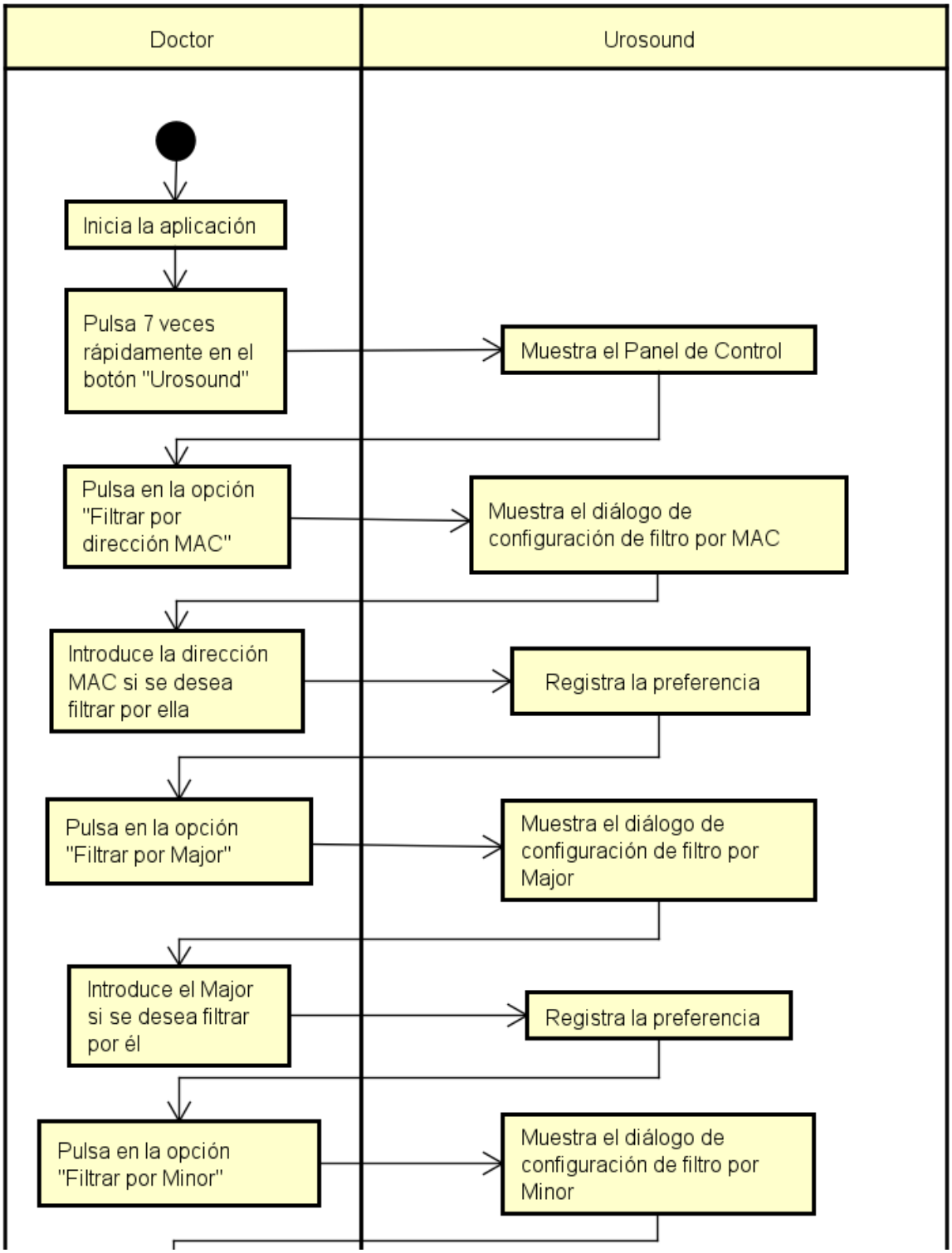


Figura 4.9: Diagrama de actividad de la historia de usuario HU07 (parte 1)

En la historia de usuario **Configuración de la detección por proximidad** (figs. 4.9 y 4.10), el doctor entra en el panel de control y configura las opciones “Filtrar por dirección MAC”, “Filtrar por Major”, “Filtrar por Minor”, “Distancia de activación BLE”, “Factor Medioambiental” y “Calibrar RSSI a 1 metro”. Es recomendado configurar el factor medioambiental antes del RSSI a 1 metro.

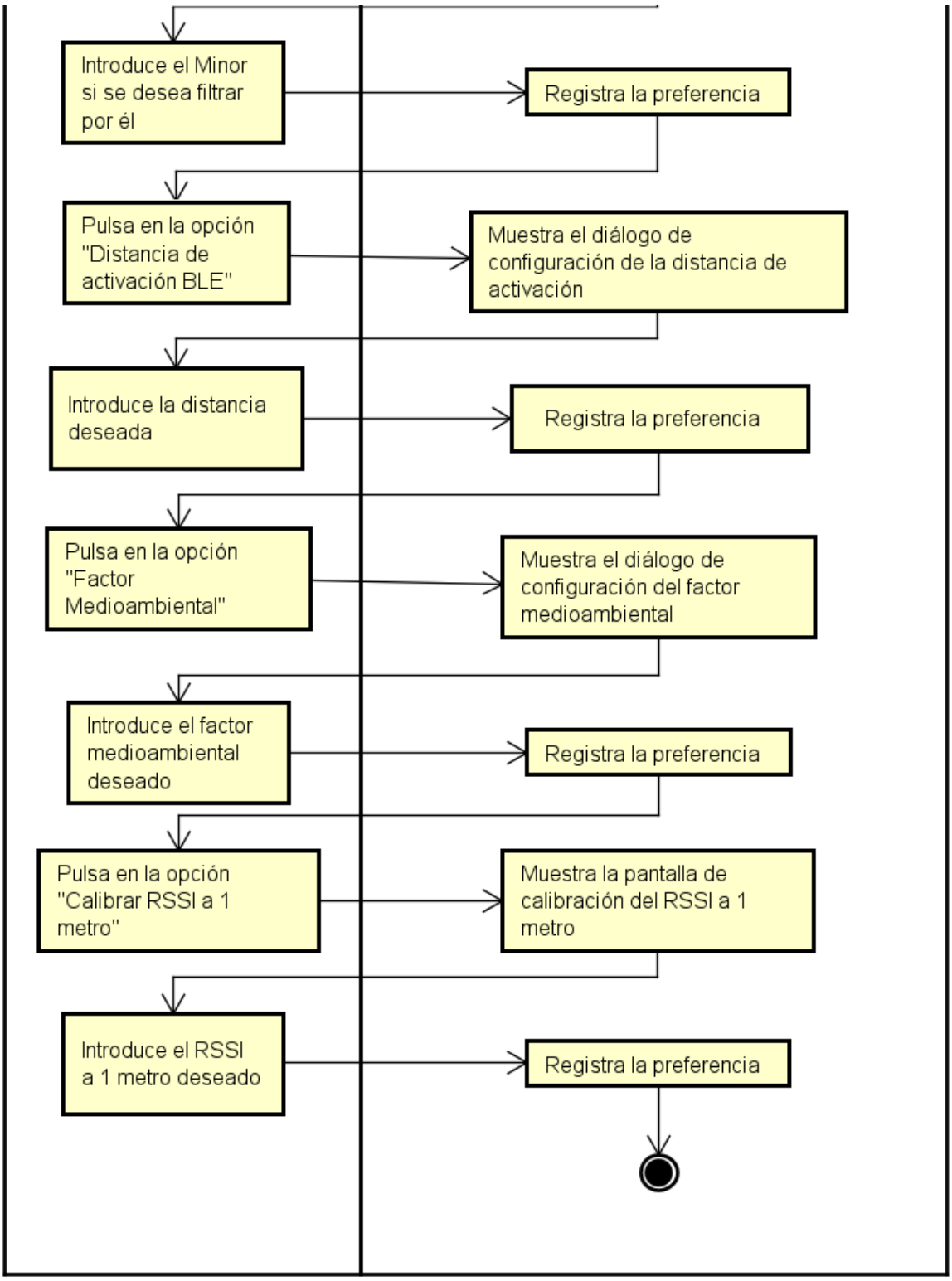


Figura 4.10: Diagrama de actividad de la historia de usuario HU07 (parte 2)

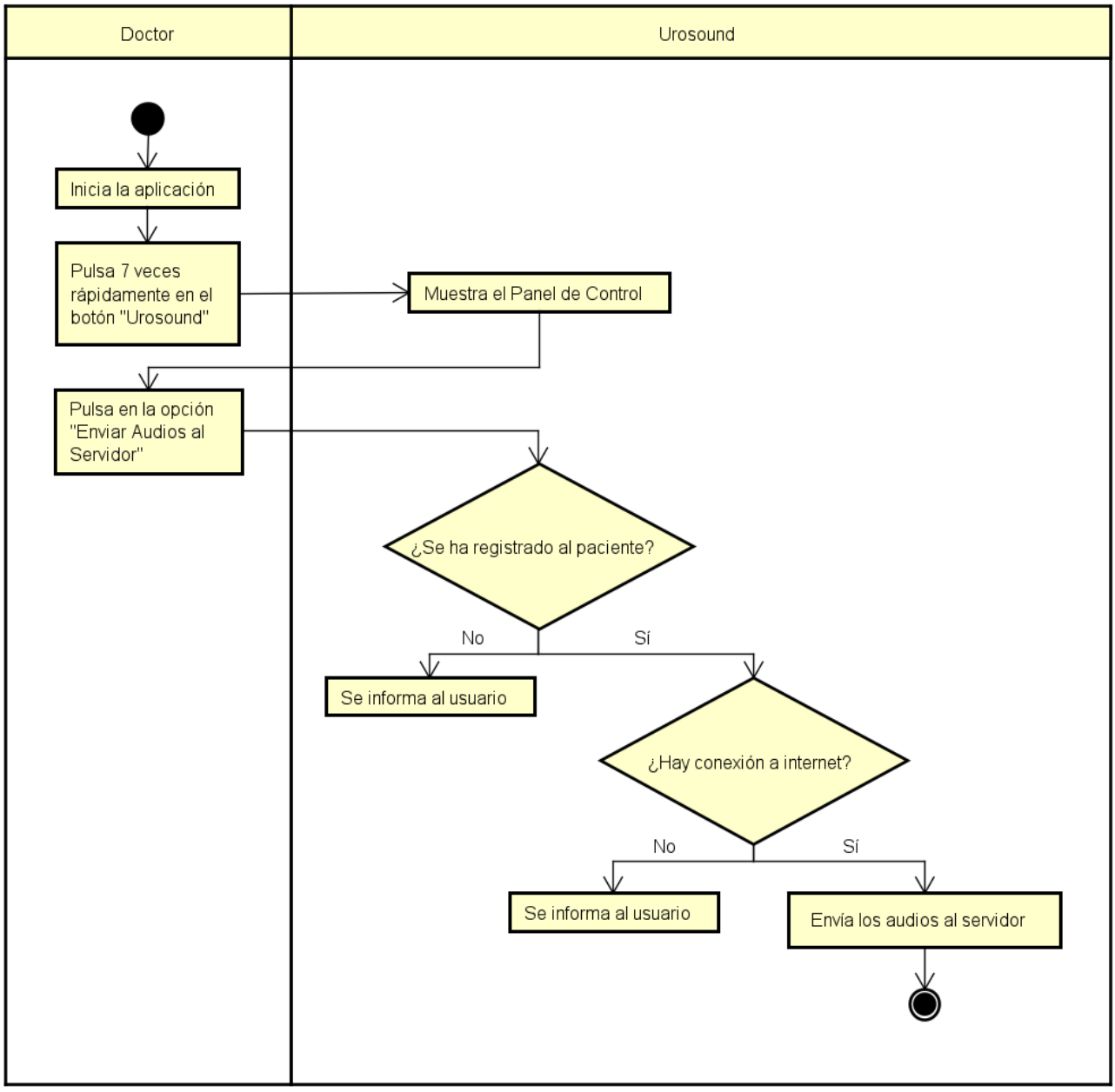


Figura 4.11: Diagrama de actividad de la historia de usuario HU08

En la historia de usuario **Envío de audios** (fig. 4.11), el doctor una vez más entra en el panel de control y pulsa en la opción “Enviar Audios al Servidor”, suponiendo que el paciente ya ha sido registrado y que hay conexión a internet. En caso contrario aparecerá un mensaje informando al usuario.

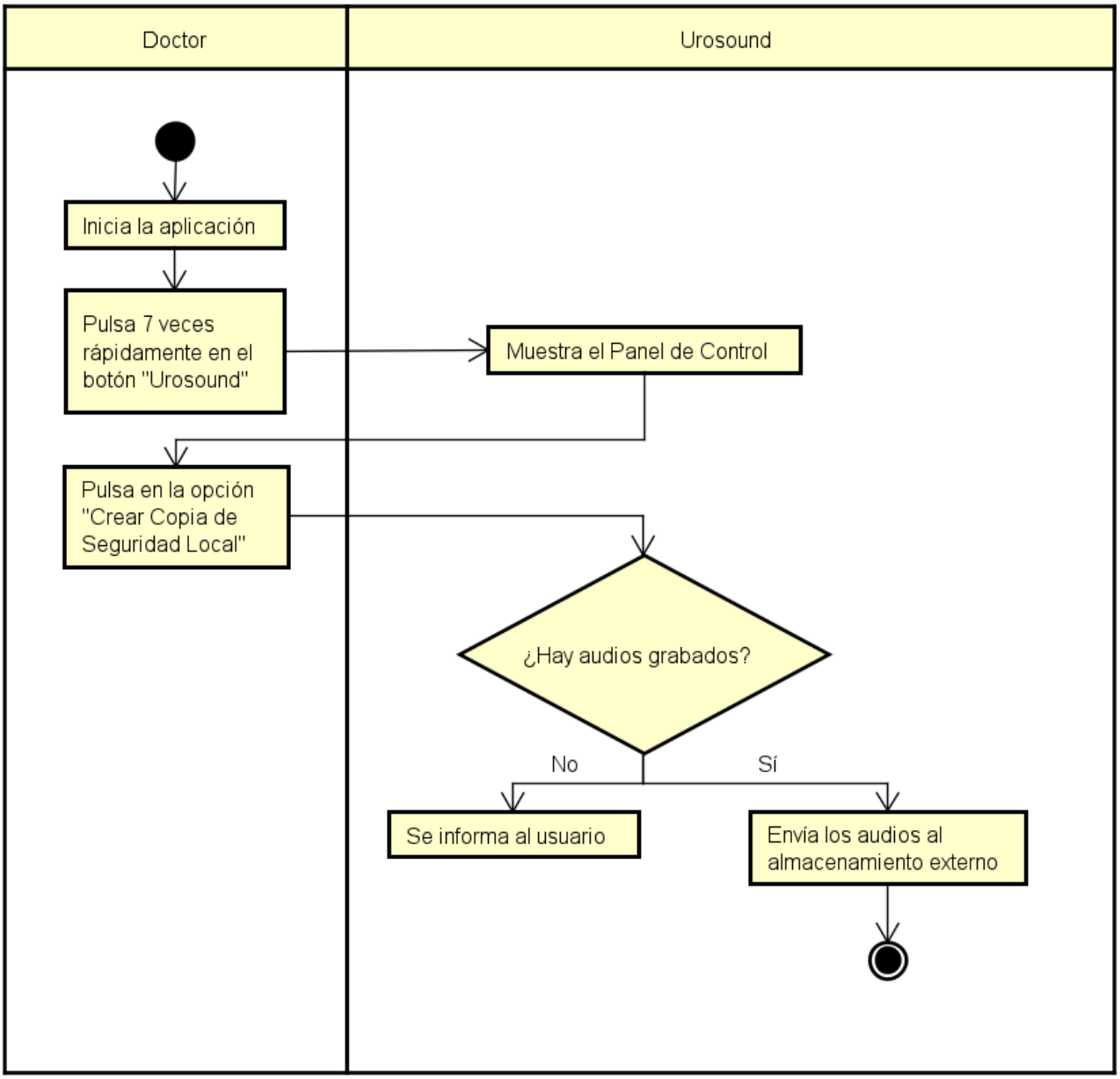


Figura 4.12: Diagrama de actividad de la historia de usuario HU09

En la historia de usuario **Copia de seguridad** (fig. 4.12), el doctor accede al panel de control y pulsa en la opción “Crear Copia de Seguridad Local”, suponiendo que hay al menos un audio grabado, ya que en caso contrario aparecerá un mensaje informando al usuario.

4.2. Diseño

4.2.1. Diagrama de paquetes

Los diagramas de paquetes permiten ver las dependencias entre los paquetes que componen la aplicación. En la figura 4.13 se presenta el diagrama de paquetes de nuestra aplicación.

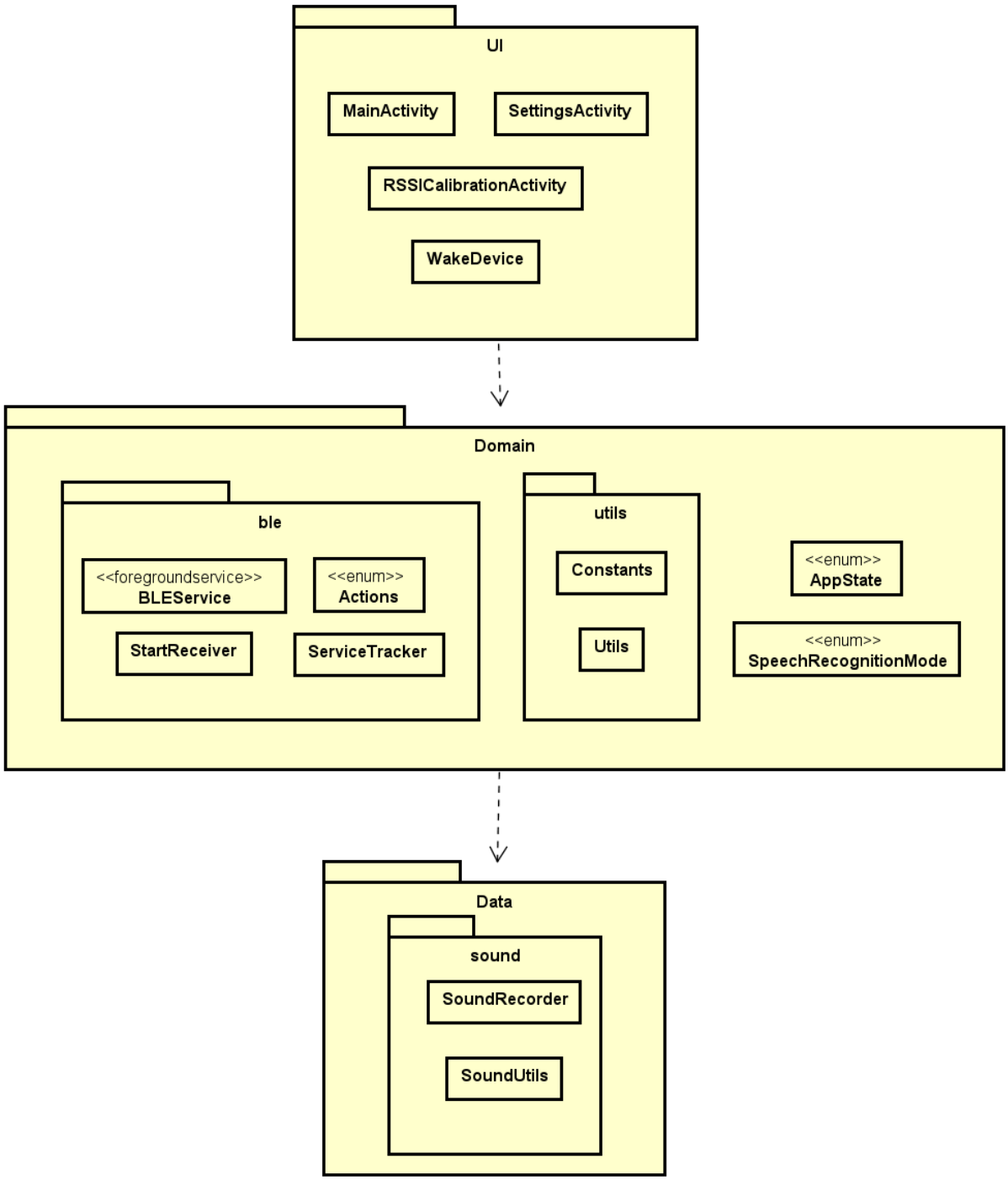


Figura 4.13: Diagrama de paquetes de la aplicación

4.2.2. Diagrama de despliegue

El diagrama de despliegue se utiliza para representar la distribución física de los dispositivos hardware y sus entornos de ejecución en un entorno real. El diagrama de despliegue de Urosound se puede ver en la figura 4.14.

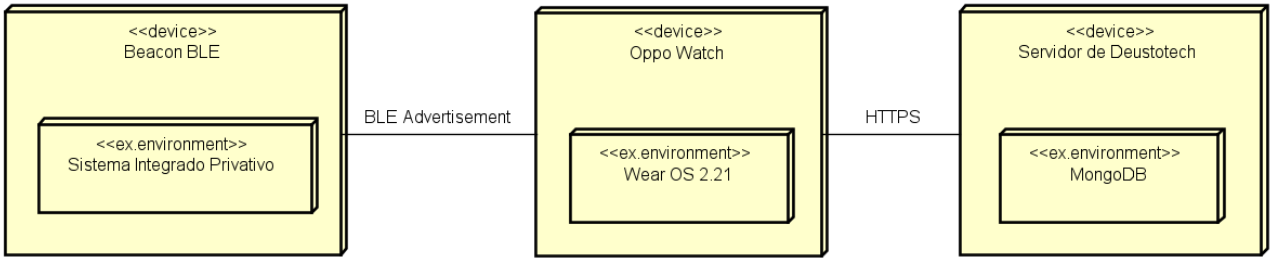


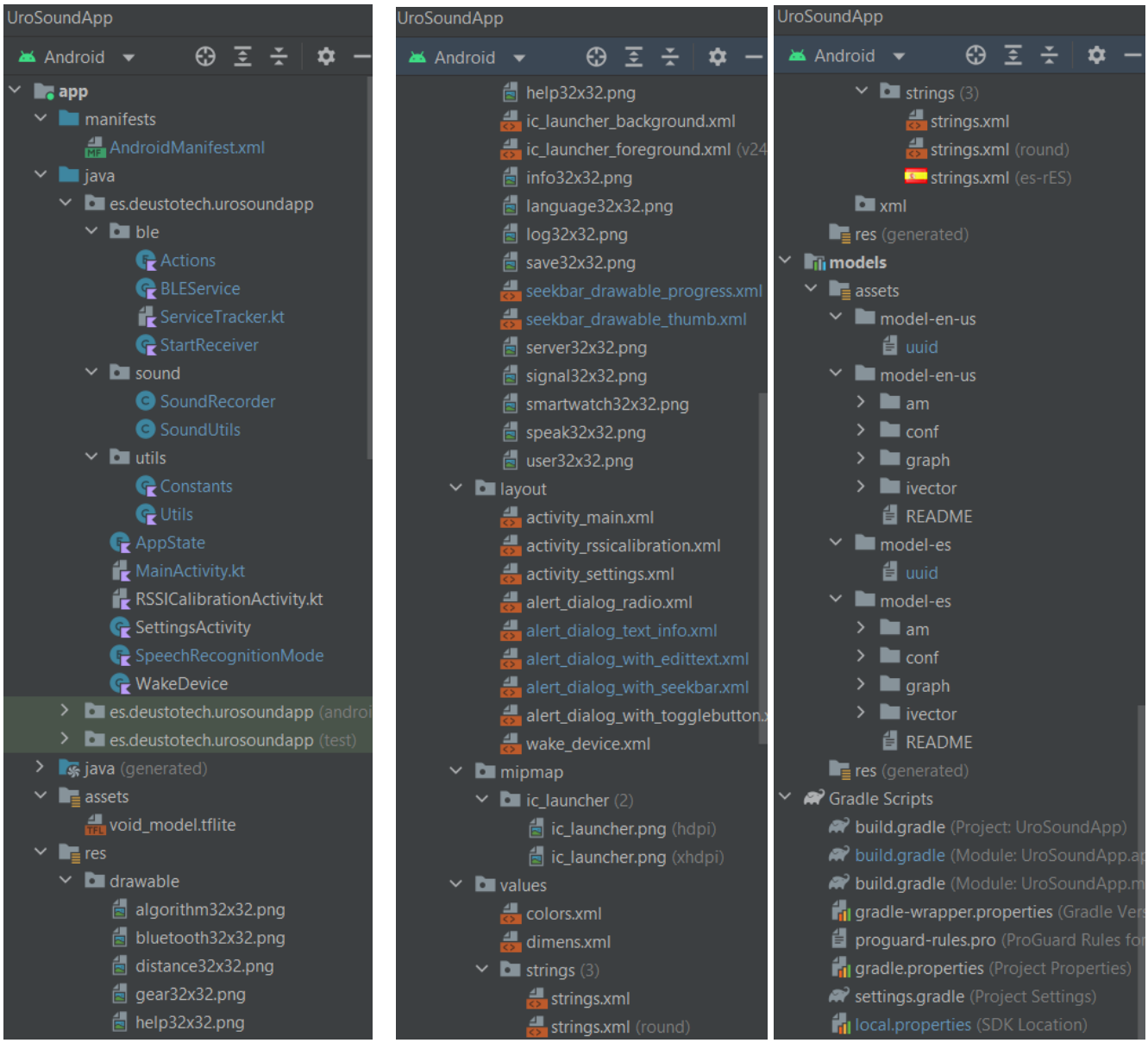
Figura 4.14: Diagrama de despliegue de Urosound

4.3. Estructura del proyecto

La estructura del proyecto es la que está por defecto en Android Studio, utilizando Gradle como gestor de dependencias y configuración del proyecto. En la figura 4.15 se pueden ver los archivos del proyecto. A continuación se describen las partes más importantes de la estructura:

- **Manifiesto de Android:** Es un archivo XML donde se almacena información general de la aplicación, importante para el sistema operativo. Aquí se encuentran los permisos de la aplicación, la arquitectura, la definición de las actividades y servicios y sus características y otros metadatos importantes.
- **Clases fuente:** En el directorio `/app/src/main/java/es/deustotech/urosoundapp/` se encuentra el código fuente del proyecto, escrito mayoritariamente en Kotlin, aunque hay un par de clases en Java.
- **Directorio assets:** En este directorio se pueden almacenar otros archivos o binarios que asistan al código del proyecto. En este caso, ahí se encuentra el modelo de Tensorflow Lite.
- **Directorio de recursos:** Aquí se encuentran los recursos visuales y archivos de localización. En la carpeta “drawable” están los iconos utilizados en el menú de opciones y módulos visuales, como el diseño de la barra SeekBar deslizable. En la carpeta layout se encuentran los diseños de las interfaces de las actividades y de los AlertDialogs implementados. En strings se encuentran los ficheros de strings localizadas para español e inglés.
- **Subproyecto models:** Este subproyecto de Gradle contiene los modelos de lenguaje de Vosk en español y en inglés, así como la configuración de los mismos.

- **Configuración de Gradle:** Los tres ficheros más relevantes de la configuración son los tres primeros que se pueden ver en la figura 4.15c, bajo el nombre de “build.gradle”. El primero contiene la configuración y dependencias a nivel del proyecto entero, el segundo de la app Urosound, y el tercero del subproyecto de los modelos de Vosk.



(a) Parte 1 (b) Parte 2 (c) Parte 3

Figura 4.15: Estructura de archivos del proyecto

4.4. Patrones de diseño

Los patrones de diseño son una parte fundamental de todo sistema software, ya que son soluciones reutilizables para problemas comunes que surgen de forma recurrente durante el desarrollo. En las siguientes subsecciones se explican brevemente los patrones empleados más relevantes.

4.4.1. Patrón Singleton

El patrón Singleton garantiza que una clase sólo tenga una instancia en todo el ciclo de vida de la aplicación y proporciona un punto de acceso global a esa instancia. Es útil cuando se necesita una única instancia compartida de una clase, como una clase de configuración o una conexión a una base de datos. En nuestro caso, se ha utilizado el patrón Singleton en la clase SharedPreferences, que permite la administración de las preferecians persistentes de la aplicación. Es importante que sólo exista una instancia de esta clase para garantizar la integridad de los datos, de ahí el uso de este patrón.

4.4.2. Patrón Observador

El patrón Observador está muy presente en las aplicaciones que contienen orientación a eventos, como cualquier aplicación Android. El objetivo de este patrón es la implementación de métodos que se ejecutan cuando reciben una señal o evento concreta, a la que están a la escucha. Un ejemplo es cualquier elemento interactuable de la interfaz, como un botón, que ejecuta el método onClick correspondiente al ser pulsado. Otro ejemplo son los métodos del ciclo de vida de Android, que se ejecutan cuando se reciben eventos concretos como el cambio de actividades o el apagado de la pantalla. De la misma manera, los métodos de cualquier interfaz Callback se ejecutan cuando sucede su evento correspondiente, como los callbacks de los reconocedores de voz o el de el escáner BLE. También se aplica en los objetos de tipo CountDownTimer y en el AsyncTask utilizado para la grabación asíncrona de audio. Por último, podemos encontrar este patrón en los BroadcastReceiver, que esperan una señal concreta para ejecutar su código, por ejemplo, el de la clase StartReceiver, que espera al evento Intent.ACTION_BOOT_COMPLETED generado al iniciar el dispositivo para ejecutar el reinicio del servicio BLE.

4.4.3. Patrón Builder

El patrón Builder se utiliza para crear nuevos objetos de una manera flexible y personalizable. Este patrón se utiliza para crear y personalizar la apariencia y comportamiento de los AlertDialog en el menú de opciones. También se utiliza para crear los filtros del escáner BLE, permitiendo colocar los bytes a bajo nivel de forma flexible.

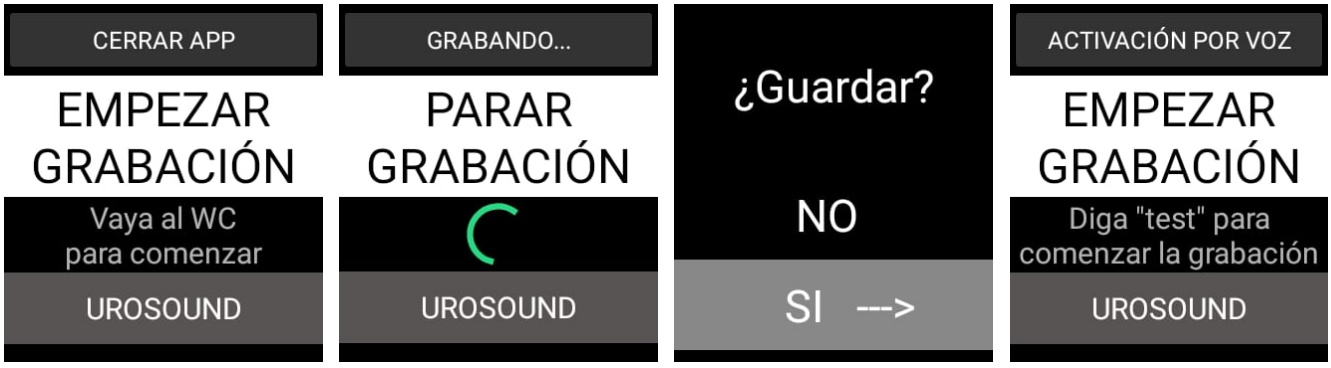
4.4.4. Patrón Adapter

El patrón Adapter se utiliza para convertir la interfaz de una clase en otra interfaz diferente para poder interactuar con otra clase. Esto permite que clases incompatibles trabajen juntas sin modificar su código fuente. Un ejemplo de este patrón en el proyecto es la clase BluetoothAdapter, que permite la interacción externa con la clase BluetoothManager, que de otra forma no se podría.

4.5. Interfaces

En esta sección se presentan capturas de pantalla de las interfaces de la aplicación, para obtener una mejor idea de lo que verán los usuarios, así como para comparar con las interfaces originales de Urosound 1.5. Las figuras se describen a continuación:

- **Figura 4.16a:** Vista al entrar en la aplicación.
- **Figura 4.16b:** Vista tras pulsar el botón de “Empezar Grabación”
- **Figura 4.16c:** Vista tras terminar la grabación
- **Figura 4.16d:** Vista al estar dentro del rango de la baliza BLE con el reconocimiento de voz seleccionado
- **Figura 4.17a:** Vista al estar dentro del rango de la baliza BLE con la detección de micción automática activada
- **Figura 4.17b:** Vista mientras se graba tras haberse detectado la micción automáticamente. Recordemos que en este caso no se puede parar la grabación.
- **Figuras 4.17c a la 4.18d:** Vista del panel de control. Se accede tras pulsar 7 veces rápidamente en la etiqueta “UROSOUND” del menú principal.
- **Figuras 4.19a a la 4.22d:** Vistas de los AlertDialog que aparecen al pulsar cada una de las opciones. Cabe destacar que para las opciones de legado “Enviar audios al servidor” y “Crear copia de seguridad local” (ver figura 4.18b) no hay diálogo de confirmación, aunque aparecen Toasts con retroalimentación para el usuario al ser pulsadas.



(a) (b) (c) (d)

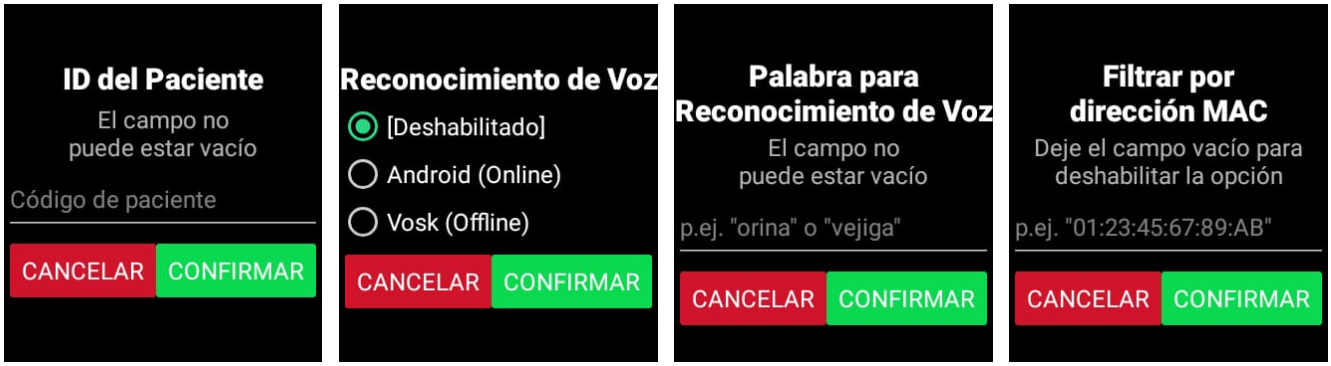
Figura 4.16: Interfaces de la app (Parte 1)

(a) (b) (c) (d)

Figura 4.17: Interfaces de la app (Parte 2)

(a) (b) (c) (d)

Figura 4.18: Interfaces de la app (Parte 3)



(a) (b) (c) (d)

Figura 4.19: Interfaces de la app (Parte 4)

(a) (b) (c) (d)

Figura 4.20: Interfaces de la app (Parte 5)

(a) (b) (c) (d)

Figura 4.21: Interfaces de la app (Parte 6)

(a) (b) (c) (d)

Figura 4.22: Interfaces de la app (Parte 7)

4.6. Pruebas

Originalmente no se planteó una necesidad específica de una batería de pruebas para el proyecto, dada la simplicidad conceptual de la funcionalidad y la logística de la implementación, además de las restricciones temporales del proyecto. Sin embargo, se ha probado la aplicación con el uso de las interfaces, los flujos descritos en los diagramas de actividad y las versiones parciales de cada iteración.

Capítulo 5

Conclusiones

En este último capítulo se hace un resumen del trabajo realizado y del aprendizaje percibido, así como una valoración personal.

Este trabajo se supone que es uno de los primeros pasos en una serie de trabajos planificados para los siguientes años, con el objetivo de impulsar la tecnología de la uroflujometría sonora en el mundo real, así como de facilitar el tratamiento para que pueda ser efectuado cómodamente en el hogar de los pacientes. De la misma manera, también se pretende hacer que el tratamiento sea lo más transparente posible a los usuarios, especialmente los niños y ancianos, y aquellos colectivos que no están muy familiarizados con las nuevas tecnologías.

Personalmente, creo que es muy poco probable que hubiera conseguido una oferta de TFG mejor que ésta. He aprendido a programar en Kotlin, he aprendido mucho sobre el sistema operativo Android y su versión Wear OS para relojes inteligentes, así como el funcionamiento e implementación de sistemas de reconocimiento de voz en tiempo real, de gestión de información por Bluetooth Low Energy, el funcionamiento de bloqueos de activación (WakeLocks) y automatización, gestión de servicios en Android, gestión de la señal de audio a muy bajo nivel, ejecución de código Python en Android y manejo de modelos de inteligencia artificial. En mi opinión son temas muy interesantes y muy diversos. Además, el hecho de trabajar en una aplicación del mundo real que se va a utilizar en el futuro representa mucho valor, sin olvidar que se trata del ámbito de la medicina, lo cual es mucho mejor. Además de esto, mis tutores D.º Mario Corrales y D.º Alfonso Bahillo han hecho un gran trabajo guiándome en los obstáculos encontrados, siendo flexibles y en general tutorizando el trabajo desarrollado.

Finalmente, y a pesar de la falta lógica de tiempo por ser un proyecto de tal envergadura, puedo decir que se han cumplido todos los objetivos iniciales dentro de las limitaciones encontradas, además de las mejoras adicionales implementadas en Urosound, por lo que puedo decir que el proyecto ha sido un éxito.

5.1. Trabajo futuro

En esta sección se hace un resumen del posible trabajo futuro pendiente por hacer:

- En primer lugar, optimizar o mejorar el modelo de inteligencia artificial de detección de micción en tiempo real, ya que muchas veces detecta falsos positivos y puede llegar a ser inviable en función del comportamiento de los pacientes y su ambiente sonoro concreto. De igual manera, sería conveniente que el algoritmo se ejecutara cada segundo o con la frecuencia adecuada para que se reduzca la posibilidad de fallar una detección de micción.
- En segundo lugar, se podría explorar la optimización de la autonomía del dispositivo haciendo uso de la investigación sobre obtención de root realizada en este TFG.
- En tercer lugar, se podrían pulir ciertos detalles menores del funcionamiento de la aplicación para que fuera más robusta o eficiente, incluyendo el código no modificado correspondiente al envío de los audios al servidor o las opciones no completas, como la restauración de los valores por defecto o el cambio de idioma de forma dinámica.
- Otras posibles mejoras incluyen una investigación de la mejor manera de implementar la API nativa de reconocimiento de voz en el Oppo (aunque no recomiendo intentarlo), el uso de otros modelos como Yamnet para la detección de micción a pesar de sus restricciones para nuestrro caso de uso particular (ya que podría dar falsos positivos con otros sonidos de agua en el baño) o posibles modificaciones a la interfaz o al flujo de ejecución en función de la retroalimentación recibida por parte de los usuarios.

Apéndices

Apéndice A

Funcionalidad implementada

El código de la aplicación se encuentra en el repositorio de GitLab de la Escuela de Ingeniería Informática de la UVa. El enlace es el siguiente:
`https://gitlab.inf.uva.es/marordo/urosound-automatization`

La versión original de Urosound se ha subido al repositorio:
`https://gitlab.inf.uva.es/marordo/urosound-original`
Se puede utilizar para ver la comparativa entre el código antiguo y el implementado en este proyecto.

Apéndice B

Manual de despliegue

En este apéndice se explica el procedimiento para desplegar la aplicación para su desarrollo o instalación. El código de la aplicación se encuentra en el repositorio de GitLab de la Escuela de Ingeniería Informática de la UVa. El enlace es el siguiente:
`https://gitlab.inf.uva.es/marordo/urosound-automatization`

Los pasos para desplegar la aplicación son los siguientes:

1. Descargar e instalar Android Studio, y configurarlo de acuerdo a las preferencias personales correspondientes si se desea.
2. Descargar el proyecto del repositorio y descomprimirlo, recomendable en formato zip, aunque también se puede clonar si se desea.
3. Abrir el proyecto en Android Studio.
4. Hacer click en la opción “Build Project” para compilar el proyecto.
5. Instalar la aplicación en el smartwatch correspondiente. El procedimiento es ir al menú de configuración en el smartwatch, luego a Sistema, luego a Información, presionar 7 veces el número de versión de compilación, volver al menú de configuración, ir al nuevo menú Opciones para desarrolladores y habilitar la opción Depuración ADB. De este modo, se puede conectar el smartwatch al pc a través de la plataforma de carga por USB e instalar la aplicación desde el botón correspondiente en Android Studio.
6. Configurar la Baliza BLE. Este proceso varía en función de la marca y el modelo concreto, por lo que habrá que mirar la documentación correspondiente.

Apéndice C

Manual de uso

En este apéndice se explica de manera breve el manual de uso de la aplicación. Para comenzar, se presuponen completados los pasos del apéndice anterior B. Para el correcto funcionamiento de la aplicación, tanto el Bluetooth como la Ubicación del smartwatch deben estar encendidas. La Ubicación no se utiliza para el GPS, pero es requerida por el sistema BLE. Opcionalmente, se pueden configurar el idioma del reconocimiento de voz del sistema desde el menú de Ajustes. También se desaconseja activar opciones de ahorro de batería, ya que podrían desactivar de forma espontánea el mecanismo de detección BLE, sin embargo esto depende del modelo de smartwatch concreto, por lo que se recomienda experimentar estas opciones. Evidentemente, siempre será mejor aumentar la autonomía de la batería, por lo que se pueden activar si dichas opciones no influyen negativamente en el funcionamiento de la aplicación. De la misma manera es buena idea desactivar otras aplicaciones o servicios del sistema que sean innecesarios para la ejecución de la aplicación.

Al abrir la aplicación se abre el menú principal (todas las interfaces se pueden ver en la sección 4.5). La primera vez que se abra la aplicación se solicitarán una serie de permisos para su correcto funcionamiento; hay que dárselos. Cuando se abre la aplicación también se abre un servicio en primer plano. Éste aparecerá como una notificación persistente en la barra de notificaciones del smartwatch, y su propósito es realizar los escaneos de BLE constantes. Se mantendrá ahí tras cerrar la app e incluso después de reiniciar el dispositivo, por lo que para cerrar tanto la app como el servicio, se debe hacer uso del botón superior del menú principal “Cerrar app”. El paciente dispone siempre de un modo manual para iniciar las grabaciones, pulsando en el botón grande blanco que dice “Empezar Grabación” o bien pulsar el botón lateral físico del smartwatch. Por otro lado, existen 2 maneras automáticas de iniciar la grabación: reconocimiento de voz con palabra clave o detección automática. Ambas se configuran desde el panel de control oculto, exclusivo para el doctor. En el caso del reconocimiento de voz, la grabación se activará automáticamente si el paciente dice una palabra clave configurable una vez se encuentra dentro del rango de la baliza BLE. En el caso de la detección automática, ésta empezará automáticamente una vez se detecte por audio que el paciente ha empezado a orinar, una vez éste esté dentro del rango de la baliza.

Para acceder al panel de control oculto, hay que pulsar rápidamente 7 veces en el botón inferior del menú principal, donde pone “Urosound”. Cuidado con pulsar muchas veces o muy rápido, porque

es posible pulsar accidentalmente alguna opción del panel de control. El panel puede tardar un par de segundos en abrirse. Esto se debe a que cuando se accede al panel, se desactiva automáticamente el servicio BLE para poder ser configurado.

Como doctor, lo primero recomendable sería introducir el ID del paciente. Después se puede configurar el reconocimiento de voz, la detección automática, los filtros y calibración del sistema BLE de detección de proximidad. Es importante recordar que la baliza BLE debería colocarse encendida en el inodoro del paciente para que el proceso funcione. Desde el panel de control también se pueden enviar los audios del paciente al servidor una vez el tratamiento ha terminado o enviarlos al almacenamiento externo del smartwatch en caso de que se deseen extraer manualmente con la opción “Crear Copia de Seguridad Local”. el resto de opciones proporcionan información adicional.

Bibliografía

[1] “Uroflowmetry image,” visitado: 2023-07-11. [Online]. Available: https://patients.uroweb.org/tests/uroflowmetry/

[2] “Menhealth,” visitado: 2023-07-04. [Online]. Available: https://play.google.com/store/apps/details?id=com.menhealth

[3] “Minzeflow,” visitado: 2023-07-04. [Online]. Available: https://play.google.com/store/apps/details?id=com.minze.flow

[4] “Telesonouroflow bladder diary,” visitado: 2023-07-04. [Online]. Available: https://play.google.com/store/apps/details?id=com.tradersmicro.TeleSonoUroflow

[5] L. Arjona, L. E. Díez, A. Bahillo, and A. Arruza-Echevarría, *UroSound: A Smartwatch-based Platform to Perform Non-Intrusive Sound-based Uroflowmetry*. Universidad de Deusto, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9670631

[6] “Android web oficial,” visitado: 2023-07-04. [Online]. Available: https://www.android.com/intl/es_es/

[7] “Android wikipedia,” visitado: 2023-07-04. [Online]. Available: https://es.wikipedia.org/wiki/Android

[8] “Ciclo de vida de las actividades android,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419

[9] “Android system architecture image,” visitado: 2023-07-11. [Online]. Available: https://es.m.wikipedia.org/wiki/Archivo:Android-System-Architecture.svg

[10] “Wear os web oficial,” visitado: 2023-07-04. [Online]. Available: https://wearos.google.com/intl/es_es/

[11] “Wear os wikipedia,” visitado: 2023-07-04. [Online]. Available: https://es.wikipedia.org/wiki/Wear_OS

[12] “Java web oficial,” visitado: 2023-07-04. [Online]. Available: https://www.java.com/es/

[13] “Java wikipedia,” visitado: 2023-07-04. [Online]. Available: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[14] “Kotlin web oficial,” visitado: 2023-07-04. [Online]. Available: https://kotlinlang.org/

[15] “Kotlin wikipedia,” visitado: 2023-07-04. [Online]. Available: https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n)

[16] “Iterative methodology image,” visitado: 2023-07-11. [Online]. Available: https://medium.com/sue%C3%B1os-graficos/dise%C3%B1o-iterativo-la-metodolog%C3%ADa-que-perfeccionar%C3%A1-tus-proyectos-21034b0d277e

[17] “Lenovo ideapad 330,” visitado: 2023-07-04. [Online]. Available: https://www.lenovo.com/es/es/p/laptops/ideapad/ideapad-300/ideapad-330-(15_intel)/88ip3000996

[18] “Lg k40,” visitado: 2023-07-04. [Online]. Available: https://www.lg.com/pe/celulares/lg-LMX420HM

[19] “Android api levels,” visitado: 2023-07-04. [Online]. Available: https://apilevels.com/

[20] “Oppo watch españa especificaciones,” visitado: 2023-07-04. [Online]. Available: https://www.oppo.com/es/accessories/watch/specs/

[21] “Global tag srl beacon,” visitado: 2023-07-04. [Online]. Available: https://www.global-tag.com/es/portfolio/round-beacony-tag-ble-beacon/

[22] “Minew tech beacon,” visitado: 2023-07-04. [Online]. Available: https://www.minew.com/product/i6-disposable-tag/

[23] “Wear os development,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/wear?hl=es-419

[24] “Introducción a wear os,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/training/wearables?hl=es-419

[25] “Cómo depurar una app de wear os,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/training/wearables/apps/debugging?hl=es-419

[26] “Cómo configurar un smartwatch,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/training/wearables/apps/creating?hl=es-419#set-up-watch

[27] “Cómo ejecutar apps en android emulator,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/studio/run/emulator?hl=es-419

[28] “Cómo actualizar a la versión más reciente de wear os,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/training/wearables/upgrade?hl=es-419

[29] “Oppo watch user manual,” visitado: 2023-07-04. [Online]. Available: https://m.media-amazon.com/images/I/B1pZ7+wHegS.pdf

[30] “Lightblue® — bluetooth le,” visitado: 2023-07-04. [Online]. Available: https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer

[31] “nrf connect for mobile,” visitado: 2023-07-04. [Online]. Available: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

[32] “Android studio web oficial,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/studio

[33] “Firefox web oficial,” visitado: 2023-07-04. [Online]. Available: https://www.mozilla.org/es-ES/firefox/new/

[34] “Notepad++ web oficial,” visitado: 2023-07-04. [Online]. Available: https://notepad-plus-plus.org/

[35] “Sourcetree web oficial,” visitado: 2023-07-04. [Online]. Available: https://www.sourcetreeapp.com/

[36] “Gitlab inf uva,” visitado: 2023-07-04. [Online]. Available: https://gitlab.inf.uva.es/

[37] “Teams web oficial,” visitado: 2023-07-04. [Online]. Available: https://www.microsoft.com/es-es/microsoft-teams/download-app

[38] “Overleaf web oficial,” visitado: 2023-07-04. [Online]. Available: https://es.overleaf.com/

[39] “Astah web oficial,” visitado: 2023-07-04. [Online]. Available: https://astah.net/

[40] “Sueldo desarrollador android junior,” visitado: 2023-07-04. [Online]. Available: https://www.glassdoor.es/Sueldos/junior-software-developer-sueldo-SRCH_KO0,25.htm

[41] “Sueldo desarrollador android,” visitado: 2023-07-04. [Online]. Available: https://www.glassdoor.es/Sueldos/desarrollador-android-sueldos-SRCH_KO0,21.htm

[42] “Astah pricing,” visitado: 2023-07-04. [Online]. Available: https://astah.net/pricing/individual/

[43] “Entrada de voz de formato libre,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/training/wearables/user-input/voice?hl=es-419#kotlin

[44] “Speechrecognizer docs,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/reference/android/speech/SpeechRecognizer

[45] “Recognitionlistener docs,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/reference/android/speech/RecognitionListener

[46] “Recognitionlistener source code,” visitado: 2023-07-04. [Online]. Available: https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/java/android/speech/RecognitionListener.java;l=33;drc=140bfb0cf46a606dcf5268882e12172ef2091553

[47] “Offline speech to text without any popup dialog in android,” visitado: 2023-07-04. [Online]. Available: https://www.geeksforgeeks.org/offline-speech-to-text-without-any-popup-dialog-in-android/

[48] “Android speech to text tutorial,” visitado: 2023-07-04. [Online]. Available: https://medium.com/voice-tech-podcast/android-speech-to-text-tutorial-8f6fa71606ac

[49] “Implement continuous speech recognition on android,” visitado: 2023-07-04. [Online]. Available: https://betterprogramming.pub/implement-continuous-speech-recognition-on-android-1dd2f4b562fd

[50] “Kontinuousspeechrecognizer github,” visitado: 2023-07-04. [Online]. Available: https://github.com/StephenVinouze/KontinuousSpeechRecognizer

[51] “Pocketsphinx on android,” visitado: 2023-07-04. [Online]. Available: https://cmusphinx.github.io/wiki/tutorialandroid/

[52] “Pocketsphinx github,” visitado: 2023-07-04. [Online]. Available: https://github.com/cmusphinx/pocketsphinx

[53] “Vosk android demo github,” visitado: 2023-07-04. [Online]. Available: https://github.com/alphacep/vosk-android-demo/

[54] “Vosk official website,” visitado: 2023-07-04. [Online]. Available: https://alphacephei.com/vosk/android

[55] “How to set the language in speech recognition on android?” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/10538791/how-to-set-the-language-in-speech-recognition-on-android/10548680#10548680

[56] “Auto download offline speech recognition language on android,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/42033223/auto-download-offline-speech-recognition-language-on-android

[57] “Recognizerintent additional parameters request,” visitado: 2023-07-04. [Online]. Available: https://issuetracker.google.com/issues/36977959

[58] “Speech recognition - get all languages supported - android,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/10610826/speech-recognition-get-all-languages-supported-android

[59] “Idiomas compatibles con speech-to-text,” visitado: 2023-07-04. [Online]. Available: https://cloud.google.com/speech-to-text/docs/speech-to-text-supported-languages?hl=es-419

[60] “Android system settings for speech and voice recognition,” visitado: 2023-07-04. [Online]. Available: https://speaking.email/FAQ/87/android-system-settings-for-speech-and-voice-recognition

[61] “How to set the language in speech recognition on android?” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/10538791/how-to-set-the-language-in-speech-recognition-on-android

[62] “android speech recognition api.system always recognizes default language,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/25503059/android-speech-recognition-api-system-always-recognizes-default-language

[63] “Speechrecognizer with google search version 3.6.14.1337016 can’t recognize other voice language except default,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/25417439/speechrecognizer-with-google-search-version-3-6-14-1337016-cant-recognize-other

[64] “Automatically download android tts engine,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/16835847/automatically-download-android-tts-engine

[65] “How to open download languages of google settings from another app android programmatically,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/52816769/how-to-open-download-languages-of-google-settings-from-another-app-android-progr

[66] “Offline speech recognition in android (jellybean),” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/17616994/offline-speech-recognition-in-android-jellybean/17674655#17674655

[67] “Recognizerintent, extra_prefer_offline,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/reference/android/speech/RecognizerIntent.html#EXTRA_PREFER_OFFLINE

[68] “Detect installed languages for offline recognition,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/17803487/detect-installed-languages-for-offline-recognition

[69] “Speechrecognizer, checkrecognitionsupport,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/reference/android/speech/SpeechRecognizer#checkRecognitionSupport(android.content.Intent,%20java.util.concurrent.Executor,%20android.speech.RecognitionSupportCallback)

[70] “Speechrecognizer, triggermodeldownload,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/reference/android/speech/SpeechRecognizer#triggerModelDownload(android.content.Intent)

[71] “Servicios de voz de google,” visitado: 2023-07-10. [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.tts

[72] “Issue speechrecognizer.error_no_match,” visitado: 2023-07-10. [Online]. Available: https://issuetracker.google.com/issues/37053152

[73] “Android speechrecognizer fails on second listen,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/31274294/android-speechrecognizer-fails-on-second-listen

[74] “Speechrecognizer throws onerror on the first listening,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/31071650/speechrecognizer-throws-onerror-on-the-first-listening

[75] “Android speechrecognizer when do i get error_client when starting the voice recognizer?” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/24995565/android-speechrecognizer-when-do-i-get-error-client-when-starting-the-voice-reco

[76] “Wear os development and hacking,” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/c/wear-os-development-and-hacking.2983/

[77] “Wear os software and hacking general,” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/f/wear-os-software-and-hacking-general.2986/

[78] “[rom] stock system dump imgs,” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/t/rom-stock-sytem-dump-imgs-img.4471105/

[79] “Oppo watch kernel sources released,” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/t/oppo-watch-kernel-sources-released.4140567/

[80] “[dev] [twrp] oppo watch 46mm [beluga],” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/t/dev-twrp-oppo-watch-46mm-beluga.4471083/

[81] “Oppo watch international version fastboot boot don’t work! (unlocked),” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/t/oppo-watch-international-version-fastboot-boot-dont-work-unlocked.4403469/

[82] “Twrp oppo,” visitado: 2023-07-04. [Online]. Available: https://twrp.me/Devices/Oppo/

[83] “Asteroidos beluga,” visitado: 2023-07-04. [Online]. Available: https://asteroidos.org/watches/beluga/

[84] “Hacks, tricks, news and updates for wear os smartwatches (ticwatch, fossil, missfit, skagen, xiaomi, oppo, suunto...),” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/t/hacks-tricks-news-and-updates-for-wear-os-smartwatches-ticwatch-fossil-missfit-skagen-xiaomi-oppo-suunt
4233055/

[85] “[recovery][unofficial] twrp 3.7.0_9-0 for oppo watch,” visitado: 2023-07-04. [Online]. Available: https://forum.xda-developers.com/t/recovery-unofficial-twrp-3-7-0_9-0-for-oppo-watch.4548153/

[86] “Reset your watch to factory settings,” visitado: 2023-07-04. [Online]. Available: https://support.google.com/wearos/answer/6056905?hl=en

[87] “Herramientas de la plataforma del sdk (adb, fastboot),” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/studio/releases/platform-tools?hl=es-419

[88] “Wikipedia bootloader unlocking,” visitado: 2023-07-04. [Online]. Available: https://en.wikipedia.org/wiki/Bootloader_unlocking

[89] “Wikipedia rooting (android),” visitado: 2023-07-04. [Online]. Available: https://en.wikipedia.org/wiki/Rooting_(Android)

[90] “Android boot flow,” visitado: 2023-07-04. [Online]. Available: https://source.android.com/docs/security/features/verifiedboot/boot-flow?hl=en

[91] “Android boot sequence,” visitado: 2023-07-04. [Online]. Available: https://www.researchgate.net/figure/Android-boot-sequence_fig4_292906634

[92] Y. an Tan, Y. Xue, C. Liang, J. Zheng, Q. Zhang, J. Zheng, and Y. Li, *A root privilege management scheme with revocable authorization for Android devices*. Elsevier, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1084804518300250

[93] “Oppo watch ow19w6ap stock firmware rom (flash file),” visitado: 2023-07-10. [Online]. Available: https://oppostockrom.com/oppo-watch-ow19w6ap

[94] “Oppo watch | easy firmware,” visitado: 2023-07-10. [Online]. Available: https://easy-firmware.com/index.php?a=downloads&b=folder&id=65251

[95] “Oppo watch ow19w6ap flash file firmware (stock rom),” visitado: 2023-07-10. [Online]. Available: https://romprovider.com/oppo-watch-ow19w6ap-firmware-support/

[96] “Ow19w6 factory firmware,” visitado: 2023-07-10. [Online]. Available: https://romdevelopers.com/index.php?a=downloads&b=tags&tag=OW19W6+Factory+Firmware

[97] “Wikipedia twrp,” visitado: 2023-07-04. [Online]. Available: https://en.wikipedia.org/wiki/TWRP_(software)

[98] “About twrp,” visitado: 2023-07-04. [Online]. Available: https://twrp.me/about/

[99] “Twrp github,” visitado: 2023-07-04. [Online]. Available: https://github.com/Teamwin/android_bootable_recovery/

[100] “Wikipedia supersu,” visitado: 2023-07-04. [Online]. Available: https://en.wikipedia.org/wiki/SuperSU

[101] “Magisk github,” visitado: 2023-07-04. [Online]. Available: https://github.com/topjohnwu/Magisk

[102] “How to install magisk on your android phone,” visitado: 2023-07-04. [Online]. Available: https://www.xda-developers.com/how-to-install-magisk/

[103] “Introducción general a bluetooth,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419

[104] “Bluetooth low energy,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth/ble-overview

[105] “Procedimiento de calibración de notificaciones rss de exposición ble,” visitado: 2023-07-04. [Online]. Available: https://developers.google.com/android/exposure-notifications/ble-attenuation-procedure?hl=es-419

[106] “Cálculo de la calibración de las notificaciones de exposición ble,” visitado: 2023-07-04. [Online]. Available: https://developers.google.com/android/exposure-notifications/ble-attenuation-computation?hl=es-419

[107] D. L. Hernández-Rojas, T. M. Fernández-Caramés, and P. F.-L. C. J. Escudero, *Design and Practical Evaluation of a Family of Lightweight Protocols for Heterogeneous Sensing through BLE Beacons in IoT Telemetry Applications*. Department Computer Science, Academic Unit of Civil Engineering, Universidad Técnica de Machala, Machala 070150, Ecuador AND Department Computer Engineering, Faculty of Computer Science, Universidade da Coruña, 15071 A Coruña, Spain, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/18/1/57

[108] L. Liu, B. Li, L. Yang, and T. Liu, *Real-Time Indoor Positioning Approach Using iBeacons and Smartphone Sensors*. College of Surveying and GeoInformatics, Tongji University, Shanghai 200092, China, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/6/2003

[109] “The challenge of bluetooth distance estimation,” visitado: 2023-07-04. [Online]. Available: http://www.davidgyoungtech.com/2020/05/15/how-far-can-you-go

[110] “Understanding ibeacon distancing,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/20416218/understanding-ibeacon-distancing

[111] “Micro-location part 2: Ble and rssi,” visitado: 2023-07-04. [Online]. Available: https://abaltatech.com/blog/2021/01/microlocation2/

[112] “Analyzing bluetooth advertising with ubertooth,” visitado: 2023-07-04. [Online]. Available: http://j2abro.blogspot.com/2014/06/analyzing-bluetooth-advertising-with.html

[113] “The beacon war – know what is a ibeacon, a eddystone and a altbeacon,” visitado: 2023-07-04. [Online]. Available: https://www.k2bindia.com/ibeacon-vs-eddystone-vs-altbeacon/

[114] “Android beacon library,” visitado: 2023-07-04. [Online]. Available: https://altbeacon.github.io/android-beacon-library/

[115] “What is the ibeacon bluetooth profile,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/18906988/what-is-the-ibeacon-bluetooth-profile

[116] “Bluetooth core specification 5.4,” visitado: 2023-07-04. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-4/

[117] “Eddystone specification,” visitado: 2023-07-11. [Online]. Available: https://github.com/google/eddystone

[118] “Mac spoofing wikipedia,” visitado: 2023-07-04. [Online]. Available: https://es.wikipedia.org/wiki/MAC_spoofing

[119] “Ibeacon spoofing wikipedia,” visitado: 2023-07-04. [Online]. Available: https://en.wikipedia.org/wiki/IBeacon#Spoofing

[120] “Ibeacon packet structure wikipedia,” visitado: 2023-07-04. [Online]. Available: https://en.wikipedia.org/wiki/IBeacon#BLE_Advertisement_Packet_Structure_Byte_Map

[121] “The ultimate guide to android bluetooth low energy,” visitado: 2023-07-04. [Online]. Available: https://punchthrough.com/android-ble-guide/

[122] “Ble starter android github,” visitado: 2023-07-04. [Online]. Available: https://github.com/PunchThrough/ble-starter-android

[123] “Bluetoothlescanner,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/BluetoothLeScanner

[124] “Scanfilter,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/ScanFilter

[125] “Scancallback,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/ScanCallback

[126] “Commit restrictivo 1,” visitado: 2023-07-11. [Online]. Available: https://android-review.googlesource.com/c/platform/packages/apps/Bluetooth/+/215844/15/src/com/android/bluetooth/gatt/AppScanStats.java

[127] “Commit restrictivo 2,” visitado: 2023-07-11. [Online]. Available: https://android-review.googlesource.com/c/platform/packages/apps/Bluetooth/+/215844/15/src/com/android/bluetooth/gatt/ScanManager.java

[128] “Cómo interpretar el ciclo de vida de una actividad,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419

[129] “Cómo mantener activo el dispositivo,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/training/scheduling/wakelock?hl=es-419

[130] “Modo doze,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/training/monitoring-device-state/doze-standby?hl=es-419

[131] “Partialwakelock,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/topic/performance/vitals/wakelock?hl=es-419

[132] “Estado de actividad y expulsión de memoria,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419#asem

[133] “Windowmanager.layoutparams,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/reference/android/view/WindowManager.LayoutParams

[134] “Keyguardmanager, requestdismisskeyguard,” visitado: 2023-07-04. [Online]. Available: https://developer.android.com/reference/android/app/KeyguardManager#requestDismissKeyguard(android.app.Activity,%20android.app.KeyguardManager.KeyguardDismissCallback)

[135] “what is the proper, non-deprecated way to wake up the device?” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/30369476/what-is-the-proper-non-deprecated-way-to-wake-up-the-device

[136] “How to calculate distance from the rssi value of the ble beacon,” visitado: 2023-07-04. [Online]. Available: https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/

[137] “Beacony parameter 3.9.4,” visitado: 2023-07-04. [Online]. Available: https://www.global-tag.com/wp-content/uploads/2021/03/Beacony-Parameter-3.9.4.pdf

[138] “nv-bluetooth github,” visitado: 2023-07-04. [Online]. Available: https://github.com/TakahikoKawasaki/nv-bluetooth

[139] “Wake android device up,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/3621599/wake-android-device-up

[140] “Android - wake up and unlock device,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/14741612/android-wake-up-and-unlock-device

[141] “Android: Wake up screen from a service,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/32591360/android-wake-up-screen-from-a-service

[142] “Keep a service running even when phone is asleep?” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/8713361/keep-a-service-running-even-when-phone-is-asleep

[143] “Android sleep/standby mode,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/5120185/android-sleep-standby-mode

[144] “Restricciones para iniciar actividades en segundo plano,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/components/activities/background-starts?hl=es-419

[145] “Partial wake lock disabled by doze in android wear,” visitado: 2023-07-04. [Online]. Available: https://issuetracker.google.com/issues/228086086

[146] L. Arjona, G. Narayanswamy, S. Hernandez, A. Bahillo, and S. Patel, *Automatic Collection of Acoustic Voiding Events at Home with Machine Learning at the Edge*, 2022.

[147] “Chaquopy,” visitado: 2023-07-10. [Online]. Available: https://chaquo.com/chaquopy/doc/current/index.html

[148] “Using python code in android studio with chaquopy,” visitado: 2023-07-10. [Online]. Available: https://medium.com/geekculture/using-python-code-in-android-studio-with-chaquopy-2d4dc3469d4d

[149] “Gradle site,” visitado: 2023-07-11. [Online]. Available: https://gradle.org/

[150] “Pip wiki,” visitado: 2023-07-11. [Online]. Available: https://es.wikipedia.org/wiki/Pip_(administrador_de_paquetes)

[151] “Wheel wiki,” visitado: 2023-07-11. [Online]. Available: https://en.wikipedia.org/wiki/Setuptools#Package_format

[152] “Chaquopy requirements,” visitado: 2023-07-11. [Online]. Available: https://chaquo.com/chaquopy/doc/current/android.html#android-requirements

[153] “Github - deustotech/autoflow,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/tree/main

[154] “Joblib site,” visitado: 2023-07-11. [Online]. Available: https://joblib.readthedocs.io/en/stable/

[155] “Keras site,” visitado: 2023-07-11. [Online]. Available: https://keras.io/

[156] “Tensorflow site,” visitado: 2023-07-11. [Online]. Available: https://www.tensorflow.org/?hl=es-419

[157] “Autoflow/rpi/requirements_dl.txt,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/requirements_dl.txt

[158] “Autoflow/rpi/requirements_ml.txt,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/requirements_ml.txt

[159] “Chaquopy repo modules,” visitado: 2023-07-11. [Online]. Available: https://chaquo.com/pypi-7.0/

[160] “Pcm wiki,” visitado: 2023-07-11. [Online]. Available: https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_impulsos_codificados

[161] “Wav wiki,” visitado: 2023-07-11. [Online]. Available: https://es.wikipedia.org/wiki/Waveform_Audio_Format

[162] “Float precision,” visitado: 2023-07-11. [Online]. Available: https://en.wikipedia.org/wiki/Floating-point_arithmetic

[163] “Tensor reference,” visitado: 2023-07-11. [Online]. Available: https://www.tensorflow.org/api_docs/java/org/tensorflow/Tensor?hl=es-419

[164] “Valor eficaz,” visitado: 2023-07-10. [Online]. Available: https://es.wikipedia.org/wiki/Valor_eficaz

[165] “Vosk offline speech recognition api,” visitado: 2023-07-10. [Online]. Available: https://alphacephei.com/vosk/

[166] “Vosk speech recognition toolkit,” visitado: 2023-07-10. [Online]. Available: https://github.com/alphacep/vosk-api/tree/master

[167] “Vosk installation,” visitado: 2023-07-10. [Online]. Available: https://alphacephei.com/vosk/install

[168] “Automatic speech recognition with vosk,” visitado: 2023-07-20. [Online]. Available: https://medium.com/@johnidouglasmarangon/automatic-speech-recognition-with-vosk-828569219f2b

[169] “Vosk models,” visitado: 2023-07-20. [Online]. Available: https://alphacephei.com/vosk/models

[170] “Debugging accuracy problems in vosk,” visitado: 2023-07-10. [Online]. Available: https://alphacephei.com/vosk/accuracy

[171] “Services in android with example,” visitado: 2023-07-10. [Online]. Available: https://www.geeksforgeeks.org/services-in-android-with-example/

[172] “Foreground services,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/components/foreground-services

[173] “Building an android service that never stops running,” visitado: 2023-07-10. [Online]. Available: https://medium.com/koahealth/building-an-android-service-that-never-stops-running-5868f304724b

[174] “Foreground service dont run constantly,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/51823543/foreground-service-dont-run-constantly

[175] “Best way to run a never ending service in android,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/34216348/best-way-to-run-a-never-ending-service-in-android

[176] “How to create an always running service in android,” visitado: 2023-07-10. [Online]. Available: https://gist.github.com/varunon9/f2beec0a743c96708eb0ef971a9ff9cd

[177] “Broadcastreceiver reference,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/reference/android/content/BroadcastReceiver

[178] “Setting launchmode="singletask"vs setting activity launchmode="singletop",” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/25773928/setting-launchmode-singletask-vs-setting-activity-launchmode-singletop

[179] “Corrutinas de kotlin en android,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/kotlin/coroutines?hl=es-419

[180] “Service,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/app/Service

[181] “Manifest.permission, receive_boot_completed,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission#RECEIVE_BOOT_COMPLETED

[182] “Soundwatch/soundwatch/wearable/src/main/java/com/wearable/sound/utils/soundrecorder.java,” visitado: 2023-07-10. [Online]. Available: https://github.com/AccessibilityLab/SoundWatch/blob/master/SoundWatch/Wearable/src/main/java/com/wearable/sound/utils/SoundRecorder.java

[183] “Soundwatch,” visitado: 2023-07-10. [Online]. Available: https://github.com/AccessibilityLab/SoundWatch

[184] “Asynctask reference,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/reference/android/os/AsyncTask

[185] “Audiorecord,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/media/AudioRecord

[186] “Broadcastreceiver reference,” visitado: 2023-07-11. [Online]. Available: https://es.wikipedia.org/wiki/SOLID

[187] “Broadcastreceiver reference,” visitado: 2023-07-11. [Online]. Available: https://en.wikipedia.org/wiki/KISS_principle

[188] “Cómo crear una iu responsiva con constraintlayout,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/training/constraint-layout?hl=es-419#groovy

[189] “Alertdialog,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/app/AlertDialog

[190] “Alertdialog.builder,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/app/AlertDialog.Builder

[191] “Sharedpreferences reference,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/reference/android/content/SharedPreferences

[192] “Google yamnet classification model,” visitado: 2023-07-10. [Online]. Available: https://tfhub.dev/google/lite-model/yamnet/classification/tflite/1

[193] “models/research/audioset/yamnet/yamnet_class_map.csv,” visitado: 2023-07-10. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/audioset/yamnet/yamnet_class_map.csv

[194] “Change app language programmatically in android,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/2900023/change-app-language-programmatically-in-android

[195] “string resources reference,” visitado: 2023-07-11. [Online]. Available: https://developer.android.com/guide/topics/resources/string-resource?hl=es-419

[196] “Simple alarm app for android 10. how to turn on screen by service,” visitado: 2023-07-04. [Online]. Available: https://stackoverflow.com/questions/65598590/simple-alarm-app-for-android-10-how-to-turn-on-screen-by-service

[197] “Intentservice,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/app/IntentService

[198] “Support for long-running workers,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/background/persistent/how-to/long-running

[199] “Oppo - don’t kill my app!” visitado: 2023-07-20. [Online]. Available: https://dontkillmyapp.com/oppo

[200] “Android wave recorder: A powerful and efficient library to record wave form audio files (wav) in android,” visitado: 2023-07-20. [Online]. Available: https://github.com/squti/Android-Wave-Recorder

[201] “Detect enable/disable bluetooth,” visitado: 2023-07-20. [Online]. Available: https://stackoverflow.com/questions/47280616/detect-enable-disable-bluetooth

[202] “How to enable/disable bluetooth programmatically in android,” visitado: 2023-07-20. [Online]. Available: https://stackoverflow.com/questions/3806536/how-to-enable-disable-bluetooth-programmatically-in-android

[203] “Android kotlin : Getting location when app is in doze mode,” visitado: 2023-07-20. [Online]. Available: https://stackoverflow.com/questions/65041152/android-kotlin-getting-location-when-app-is-in-doze-mode

[204] “Powermanager, location_mode_no_change,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/reference/android/os/PowerManager#LOCATION_MODE_NO_CHANGE

[205] “Faq - how do i... read files in python,” visitado: 2023-07-10. [Online]. Available: https://chaquo.com/chaquopy/doc/current/faq.html#faq-read

[206] “Faq - write files in python,” visitado: 2023-07-10. [Online]. Available: https://chaquo.com/chaquopy/doc/current/faq.html#faq-write

[207] “onnewintent() lifecycle and registered listeners,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/8619883/onnewintent-lifecycle-and-registered-listeners

[208] “Autoflow/rpi/tflite_model/void_model.tflite,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/tflite_model/void_model.tflite

[209] “Autoflow/rpi/runinference.py,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/runInference.py

[210] “Autoflow/rpi/runinference_ml.py,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/runInference_ml.py

[211] “Autoflow/rpi/record.py,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/record.py

[212] “Autoflow/rpi/record_ml.py,” visitado: 2023-07-10. [Online]. Available: https://github.com/DeustoTech/AutoFlow/blob/main/RPI/record_ml.py

[213] “Compatibilidad con versiones de tensorflow,” visitado: 2023-07-10. [Online]. Available: https://www.tensorflow.org/guide/versions?hl=es-419

[214] “Cómo crear un fragmento de diálogo,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/topics/ui/dialogs?hl=es-419#kotlin

[215] “Android alert dialog using kotlin,” visitado: 2023-07-10. [Online]. Available: https://www.digitalocean.com/community/tutorials/android-alert-dialog-using-kotlin

[216] “Cómo acceder a archivos específicos de la app,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/training/data-storage/app-specific?hl=es-419

[217] “Android restore application’s default preferences,” visitado: 2023-07-10. [Online]. Available: https://stackoverflow.com/questions/46655743/android-restore-applications-default-preferences

[218] “Android manifest activity element,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/topics/manifest/activity-element?hl=es-419#lmode

[219] “Descripción general de los servicios,” visitado: 2023-07-10. [Online]. Available: https://developer.android.com/guide/components/services?hl=es-419