UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño y Desarrollo de una Aplicación en Android capaz de detectar Fibrilación Atrial en un Electrocardiograma

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA TELEMÁTICA

Autor: Eva Sánchez-Guerrero Ayala

Director: Antonio Javier García Sánchez

Co-director: Joan García Haro

Cartagena, 13 de abril de 2021

Resumen

Este proyecto tiene como objetivo el desarrollo de una aplicación de salud en Android llamada MyHeartFitness. La aplicación tomará como entrada un electrocardiograma proporcionado por el usuario. A partir del electrocardiograma, MyHeartFitness aplicará un algoritmo para la detección de fibrilación atrial. La aplicación presentará una interfaz sencilla e intuitiva mediante la cual el usuario podrá subir el electrocardiograma y obtener el resultado del análisis en unos pocos minutos. Además, también será posible si el usuario lo desea, recoger datos personales como género, edad y otros parámetros biológicos que puedan servir para estudios futuros y mejora del algoritmo.

Palabras clave

Android, aplicación, algoritmo, fibrilación atrial, electrocardiograma.

Abstract

The aim of this Project is to develop a health application on Android called MyHeartFitness. The application will take an electrocardiogram provided by the user as input. From the electrocardiogram, MyHeartFitness will apply an algorithm for the detection of Atrial Fibrillation. The application will present a simple and intuitive interface through which the user can upload the electrocardiogram and obtain the result of the analysis in a few minutes. In addition, it will also be possible if the user so wishes, to collect personal data such as gender, age and other biological parameters that can be used for future studies and improvement of the algorithm.

Keywords

Android, application, algorithm, atrial fibrillation, electrocardiogram.

A mi familia, por todo el apoyo y consejos en los buenos y malos momentos a lo largo de mi carrera.

A mi pareja, por estar siempre ahí y creer en mí.

A mis tutores Antonio Javier y Joan por la gran disponibilidad y entendimiento que han demostrado a lo largo de este proyecto.

Gracias.

Índice

Índice de figuras

Índice de Anexos

1. Introducción

1.1. Motivación y Objetivos

A lo largo de mis estudios de Ingeniería Telemática he tenido la oportunidad de formarme en distintos ámbitos tecnológicos. De todos ellos siempre me ha resultado más interesante el desarrollo de software y la aplicación de este en el ámbito de la medicina.

Las enfermedades cardiovasculares (ECV) son la principal causa de muerte en todo el mundo[1]. En el año 2016 el número total de muertes por ECV fue de 17,9 millones, lo cual representa el 31% de todas las muertes a nivel mundial. Para tratar a personas con ECV es fundamental la detección precoz y el tratamiento temprano.

MyHeartFitness es una aplicación de salud para Android diseñada para ser intuitiva y accesible para el usuario y cuya finalidad es poder aplicar un algoritmo de predicción sobre los intervalos RR de un electrocardiograma obtenido mediante cualquier sensor biomédico de bajo costo. Su funcionamiento es muy simple y no requiere de registro por parte del usuario. De esta manera, si los usuarios disponen de un electrocardiograma en formato digital, podrán hacer uso de la aplicación para detectar si existen anomalías en este. MyHeartFitness puede ayudar a la detección de fibrilación atrial prematuramente si se detectan anomalías de las que paciente no es consciente todavía.

El electrocardiograma se dará como entrada a la aplicación ya sea en formato PDF o JPEG. Posteriormente, el algoritmo aplicará Análisis de Cuantificación de Recurrencia Simbólica o SRQA a este electrocardiograma para la detección de fibrilación atrial. El algoritmo será capaz de analizar anomalías en un electrocardiograma, en particular se centrará en la detección de la Fibrilación Atrial (AF). La aplicación presentará una interfaz sencilla e intuitiva mediante la cual el usuario podrá subir el electrocardiograma y obtener el resultado del análisis en unos pocos minutos.

Además, también será posible si el usuario lo desea, recoger datos personales como género, edad y otros parámetros biológicos que puedan servir para estudios futuros y mejora del algoritmo.

[1] (Enfermedades Cardiovasculares (ECVs), 2017)

1.2. Fases del proyecto

Para comenzar a desarrollar el proyecto se han seguido una serie de pasos antes de iniciar el diseño de la aplicación ya que nunca había trabajado en el desarrollo de aplicaciones Android ni con el algoritmo que implementa la aplicación.

El primer paso fue analizar y entender el algoritmo que va a ejecutar la aplicación. Este algoritmo fue desarrollado previamente en código MATLAB[2] por lo que el objetivo era comprender los pasos que realiza el algoritmo para después traducirlo a Java que es el lenguaje con el que finalmente se ha escrito la aplicación Android objeto de este proyecto. Para ello, me fue muy útil la lectura de los dos artículos escritos a cerca del algoritmo que ejecuta la aplicación[3] y el prototipo de esta[4].

Una vez comprendido el algoritmo, el siguiente paso fue la familiarización con la estructura de aplicaciones Android y su entorno de desarrollo. El software usado para la programación de la aplicación es Android Studio por lo que fue necesario comprender la estructura de los proyectos de aplicaciones Android y el contenido de los archivos principales. Para la familiarización con Android Studio y el desarrollo en Android ha sido muy útil la página oficial de desarrolladores Android[5], donde se puede encontrar una gran variedad de tutoriales y ejemplos para este tipo de aplicaciones.

Antes de comenzar a escribir el código de la aplicación se definieron los contenidos y visualización de la aplicación teniendo en mente que el objetivo es que la aplicación sea intuitiva para el usuario. Por ello, se decidió crear una aplicación simple con tres secciones principales.

Tras haber definido la estructura básica de la aplicación se llevó a cabo el desarrollo de la interfaz de esta. Esto incluía realizar el código de las tres vistas definidas previamente y el funcionamiento básico de la aplicación.

[2] (MyHeartFitnesApp, 2020)
[3] (Pérez-Valero, y otros, 2019)
[4] (Pérez-Valero, Garcia-Sanchez, Ruiz Marín, & Garcia-Haro, 2020)
[5] (Guía para Desarrolladores Android, s.f.)

Por último, en la fase de desarrollo se realizó la traducción del código del algoritmo escrito en MATLAB a código Java. Esta fase fue la más extensa dada la diferencia entre ambos lenguajes de programación. Algunas funciones usadas en el algoritmo original son nativas de MATLAB por lo que se tuvieron que implementar en Java o buscar librerías que proporcionaran estas funciones.

Finalmente, y una vez la aplicación ya estaba terminada se llevó a cabo la comprobación de resultados. Para comprobar los resultados obtenidos se ejecutó el algoritmo tanto en MATLAB como en Java con los mismos datos de entrada, verificando que ambos algoritmos ofrecían datos de salida bastante similares.

1.3. Estructura de la memoria

Esta memoria se ha estructurado en distintas secciones con el fin de facilitar el seguimiento de su lectura. La estructura del documento se indica a continuación:

- **Estado del arte**: este capítulo de la memoria pretende hacer una evaluación de las distintas soluciones similares a la propuesta en este trabajo que existen actualmente. Además, se introduce el sistema operativo Android y las tecnologías usadas para el desarrollo del trabajo.
- **Algoritmo ReAD-AF**: se explica brevemente el funcionamiento del algoritmo que ejecutará la aplicación junto con las características de este.
- **Funcionamiento de la aplicación**: en este punto se introduce el funcionamiento a grandes rasgos de la aplicación.
- **Implementación de la aplicación**: se presentan los pasos realizados para el desarrollo de la aplicación, así como su diseño y los distintos componentes de esta.
- **Resultados**: se introducen las fuentes de los datos usados para el algoritmo así como una comparativa entre el algoritmo original y el algoritmo traducido a Java.
- **Conclusiones**: se presentan las conclusiones del trabajo realizado, así como alguna línea futura para su mejora.

2. Estado del arte

2.1. Sistemas Operativos para móviles

Hoy en día existe una gran variedad de sistemas operativos para móviles, pero solo unos pocos son usados ampliamente a nivel mundial. En la actualidad los sistemas operativos más utilizados son Android e iOS, que combinados alcanzan prácticamente el 100% de la cuota de mercado global.

Los dispositivos Android actualmente representan el 71,8% de la cuota de mercado global [6], lo que hace que las perspectivas de desarrollo de aplicaciones Android sean extremadamente amplias. Si nos fijamos en la cuota de mercado en España [7], los dispositivos Android constituyen el 79.6% del total.

Por otro lado, los dispositivos iOS son cada vez más económicos y populares debido a su alto rendimiento y su continua innovación. En general, ambos sistemas operativos tienen una gran popularidad que no parece que vaya a decaer en los próximos años, por lo que las perspectivas de desarrollo de aplicaciones de Android siguen siendo amplias.

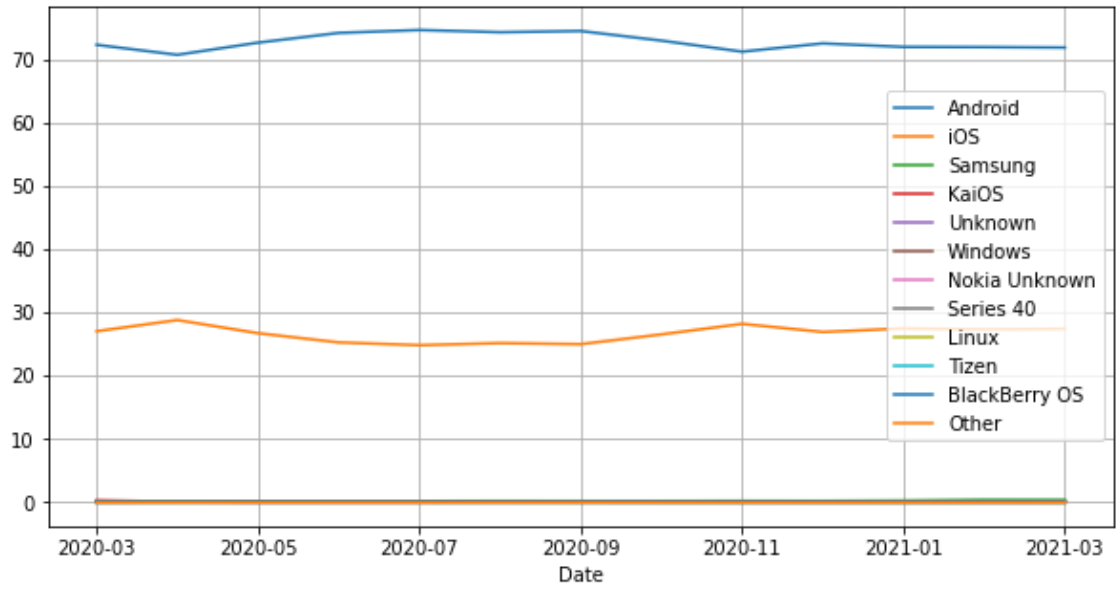

Figura 1. Cuota de Mercado Global de Sistemas Operativos Móviles.

[6] (Cuota de Mercado Global de Sistemas Operativos Móviles, 2021)
[7] (Cuota de Mercado en España de Sistemas Operativos Móviles, 2021)

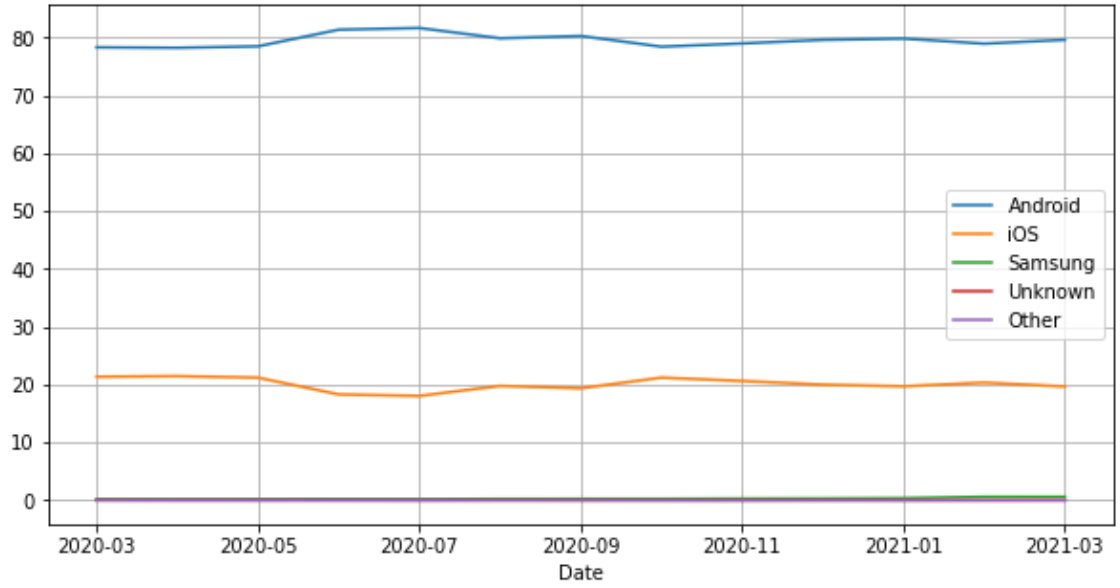

Figura 2. Cuota de Mercado en España de Sistemas Operativos Móviles.

Generalmente, Android tiene un entorno mucho menos restringido que iOS. En términos de distribución, las aplicaciones desarrolladas para Android se ejecutarán en prácticamente cualquier dispositivo Android y es poco probable que se encuentren problemas con la compatibilidad del hardware.

Asimismo, el proceso de desarrollo también es más flexible para Android que para iOS. Mientras que para el desarrollo en iOS necesitamos software que solo funciona en macOS, para el desarrollo en Android no se requiere ningún software en específico, dado que podemos desarrollar aplicaciones en Windows, macOS y Linux.

Se ha elegido Android para desarrollar la aplicación ya que es un sistema operativo de código abierto, lo que hace que esté disponible en dispositivos móviles de gama alta y de gama baja. Observando las estadísticas, está claro que el mercado está dominado por dispositivos Android, por lo que una aplicación en este sistema operativo llegará a un mayor número de usuarios potenciales.

2.2. Android

Android es un sistema operativo[8] que está basado en una versión modificada del kernel de Linux y otro software de código abierto. Principalmente está orientado a y diseñado para dispositivos móviles con pantalla táctil, como smartphones, tablets y smartwatches.

Figura 3. Logo de Android.

2.2.1. Arquitectura

La Arquitectura del sistema operativo Android consiste en cinco componentes principales que se detallan a continuación y se pueden ver en la Figura 4.

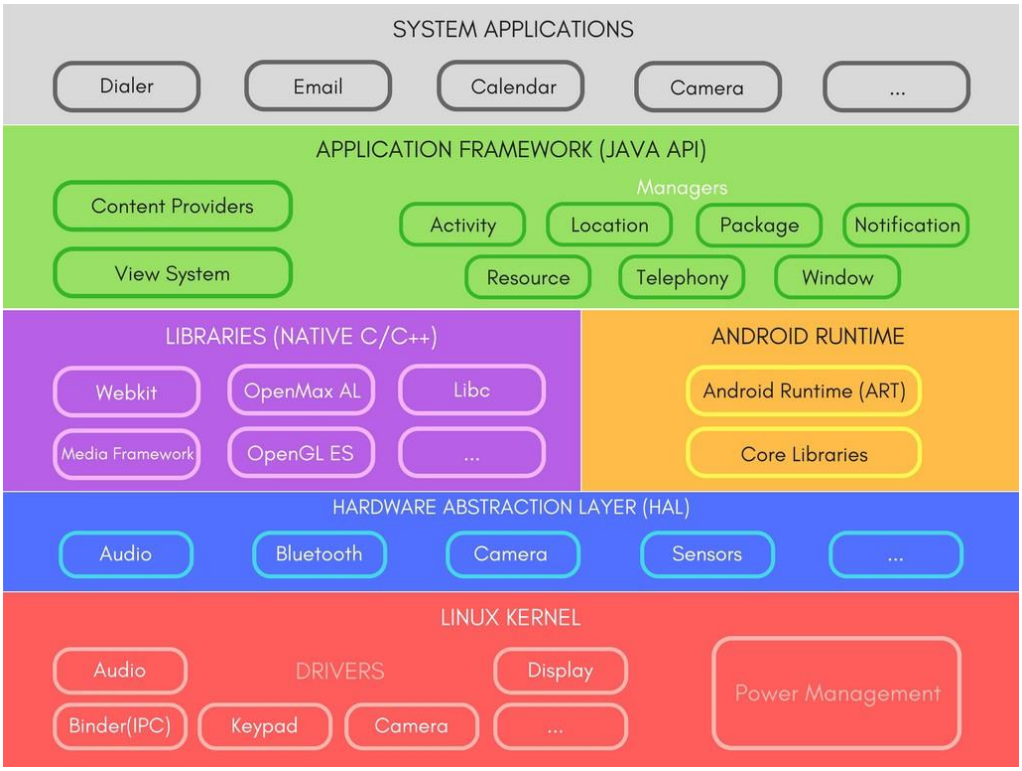

Figura 4. Arquitectura del Sistema Operativo Android.

[8] (Android, 2021)

- **Aplicaciones**

Son las aplicaciones que se ponen a disposición del usuario. Estas aplicaciones pueden venir preinstaladas en el dispositivo o ser instaladas por el propio usuario desde la tienda de aplicaciones. Algunas de las aplicaciones básicas son el calendario, mapas, navegadores, contactos, etc. Estas aplicaciones están escritas en el lenguaje de programación Java.

- **Framework de aplicaciones**

Los programadores de aplicaciones Android tienen acceso a las mismas API que usan las aplicaciones básicas que vienen ya instaladas por defecto. De esta manera se simplifica la reutilización de componentes y el uso de funciones de otras aplicaciones siempre y cuando las reglas de seguridad lo permitan.

- **Bibliotecas**

Android dispone también de una serie de bibliotecas escritas en C / C ++ que son usadas por varios componentes del sistema. Estas bibliotecas contienen funciones que pueden usar los desarrolladores a través del framework de aplicaciones de Android. Un ejemplo de biblioteca sería la de SQLite.

- **Runtime de Android**

Las aplicaciones Android corren su propio proceso cada una, con una instancia de la máquina virtual Dalvik. Esta máquina virtual está diseñada para que el dispositivo ejecute varias máquinas virtuales simultáneamente de manera eficaz. Con Davlik cada vez que se ejecuta una aplicación en el dispositivo móvil, solo parte del código que permite ejecutar la aplicación se compilará en ese momento. El resto del código en cambio se ejecutará posteriormente, cuando sea necesario. Esto hace que se ocupe menos espacio en la memoria.

A partir de la versión 5.0 de Android se usa el runtime ART. ART compila todo el código de lenguaje de alto nivel en código máquina cuando instalamos la aplicación en lugar de hacerlo

dinámicamente como en Davlik. Esto hace que no haya retardos al abrir la aplicación en nuestro dispositivo ya que ya está compilada al completo. Una de las ventajas de este runtime es que el rendimiento de la batería del dispositivo mejora considerablemente y se reduce el tiempo de inicio de las aplicaciones.

- **Núcleo Linux**

Como se ha indicado anteriormente, Android se basa en una versión modificada del kernel de Linux para proporcionar servicios básicos del sistema, como seguridad, administración de memoria, administración de procesos, pila de red y modelo de controlador. El kernel también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

2.2.2. Versiones

Desde su lanzamiento inicial en 2007 con la versión Android 1.0, Android ha tenido numerosas actualizaciones. Estas actualizaciones se realizan para eliminar bugs o agregar nuevas funcionalidades al sistema operativo. La última actualización es Android 11, lanzada en 2020.

En la Tabla 1 podemos ver el historial de versiones de Android junto con su nivel de API.

Nombre	Logo	Versión	Año	Nivel de API
(No name)	-	1.0-11	2008 - 2009	1 - 2
Cupcake		1.5	2009	3
Donut		1.6	2009	4
Eclair		2.0 - 2.1	2009	5 - 7
Froyo		2.2 - 2.2.3	2010	8
Gingerbread		2.3 - 2.3.7	2010	9 - 10
Honeycomb		3.0 - 3.2.6	2011	11 - 13
Ice Cream Sandwich		4.0 - 4.0.4	2011	14 - 15
Jelly Bean		4.1 - 4.3.1	2012	16 - 18
KitKat		4.4 - 4.4.4	2013	19 - 20
Lollipop		5.0 - 5.1.1	2014	21 - 22
Marshmallow		6.0 - 6.0.1	2015	23
Nougat		7.0 - 7.1.2	2016	24 - 25
Oreo		8.0 - 8.1	2017	26 - 27
Pie		9	2018	28
Android 10		10	2019	29
Android 11		11	2020	30
Android 12		12	TBA[9]	31

[9] TBA (to be announced), indica que a pesar de que algo está programado o previsto que suceda, un aspecto particular de esa cosa sigue siendo a convenir.

Tabla 1. Historial de versiones de Android.

En la Figura 5 se representa la cuota de mercado de las distintas versiones de Android en dispositivos móviles de todo el mundo entre el año 2018 y 2021[10]. Vemos que las versiones con mayor cuota de mercado en el año 2020 son Pie (lanzada en 2018) y 10 (lanzada en 2019) con un 32% y 23% de la cuota de mercado en 2020 respectivamente. Sin embargo, aunque estas versiones son las que mayor cuota tienen, hay otras versiones anteriores que siguen teniendo un alto porcentaje.

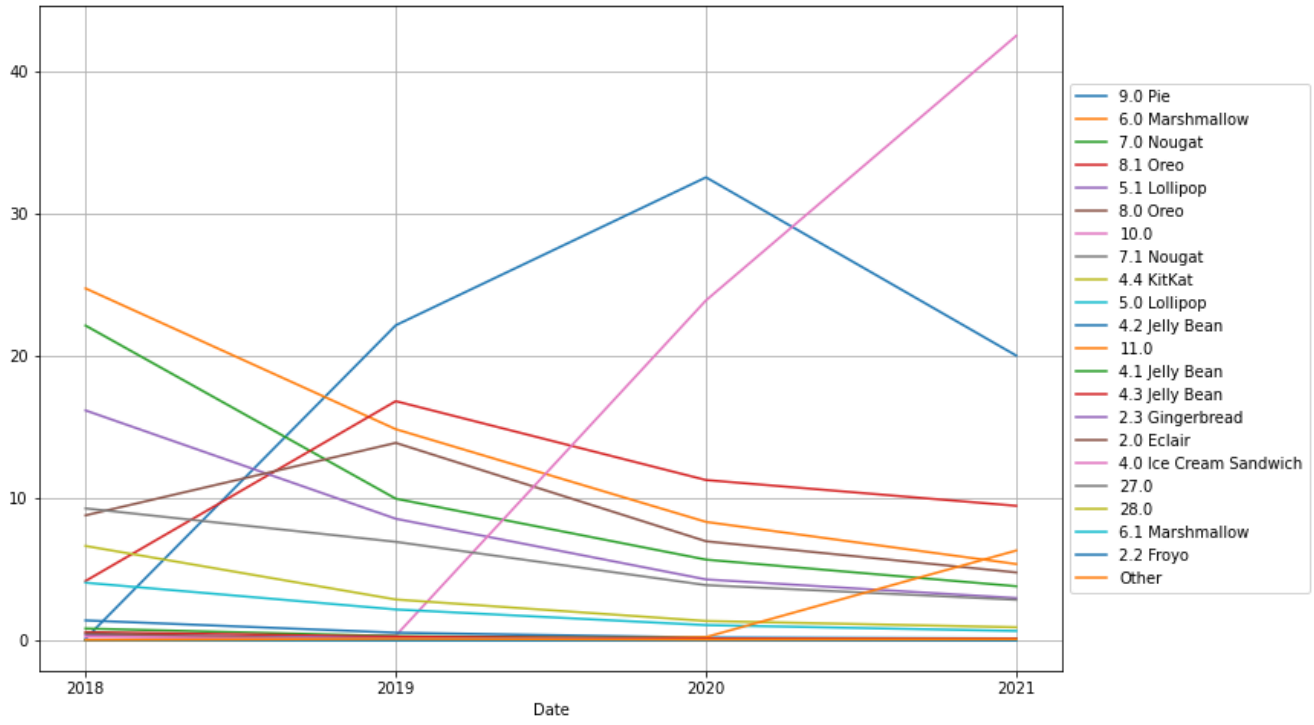

Figura 5. Cuota de Mercado Global de Versiones Android 2018-2021.

Si bien las últimas versiones de Android ofrecen unas API mejoradas para las aplicaciones, es importante desarrollar aplicaciones Android admitiendo versiones anteriores hasta que se actualicen más dispositivos.

[10] (Cuota de Mercado Global de Versiones Android, 2021)

2.3. Enfoques existentes

En la actualidad, existe una gran variedad de dispositivos portátiles que permiten estudiar el comportamiento del corazón de un paciente a través de exámenes cardíacos periódicos que se pueden realizar en cualquier momento y en cualquier lugar. Este tipo de monitorización permite al paciente obtener datos relacionados con su salud más frecuentemente sin tener que visitar un médico cada vez que quiera monitorizar esta información.

Por ello, este tipo de sensores portátiles se están volviendo cada vez más populares entre personas de distinta edad. Como consecuencia, algunas empresas líderes tecnológicas, como es el caso de Apple, están invirtiendo en desarrollar dispositivos que podemos llevar por ejemplo en la muñeca y nos proporcionan información sobre la salud del usuario.

2.3.1. Relojes inteligentes

Una de las formas más simples de monitorizar la salud de un individuo es una pulsera[11] que contiene distintos sensores que permiten hacer un seguimiento y adquirir datos referentes a la actividad física y a la frecuencia cardíaca del usuario, por mencionar sólo algunos de ellos.

Un ejemplo de esto es el reloj inteligente Fitbit, que además proporciona una aplicación para teléfonos móviles que ayuda al usuario realizar un seguimiento de la actividad, el ejercicio, el sueño y el peso durante todo el día.

[11] (Crecimiento de Sensores Portátiles en la Atención Médica, 2020)

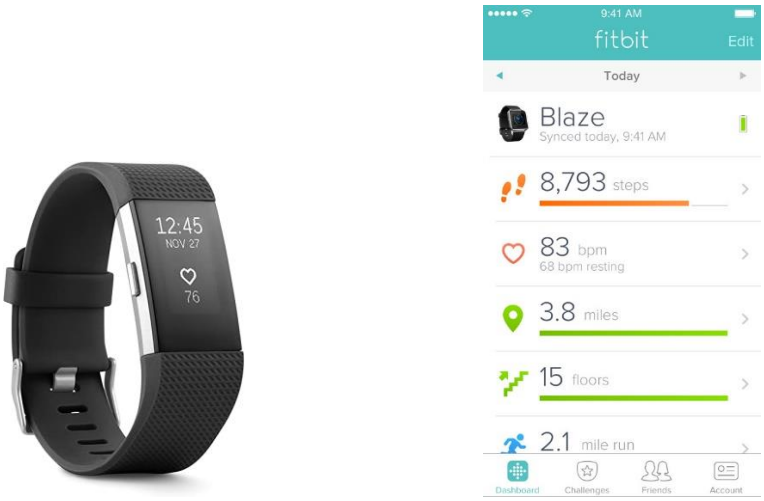

Figura 6. Reloj inteligente y aplicación de Fitbit.

Recientemente, Apple ha lanzado la aplicación Apple Heart Study[12] que usa datos capturados por el reloj inteligente Apple Watch para identificar ritmos cardiacos irregulares, incluidos los de enfermedades cardíacas potencialmente graves.

Estos dispositivos proporcionan una serie de funciones que nos dan información relevante sobre el usuario prácticamente en tiempo real. Sin embargo, la principal desventaja de estos métodos comúnmente es su baja precisión en la recopilación de datos de ECG. Sin embargo, el método desarrollado[13] no requiere la calibración de datos de ECG para proporcionar buenos resultados.

[12] (Apple Heart Study, 2021)
[13] (Pérez-Valero, Garcia-Sanchez, Ruiz Marín, & Garcia-Haro, 2020)

2.3.2. Algoritmos

Paralelamente al desarrollo de estos sensores portátiles para la monitorización de la salud, se han implementado varios algoritmos para comprender, detectar y clasificar diferentes enfermedades a partir de un electrocardiograma.

Algunos algoritmos relacionados para la detección de Fibrilación Atrial son algoritmos basados en análisis de ondas P[14] o algoritmos que requieren de una amplia cantidad de características obtenidas de un electrocardiograma en redes neuronales artificiales[15].

Sin embargo, estos algoritmos también presentan una serie de limitaciones como la alta dependencia de la robustez de los datos de entrenamiento y el hecho de que se basan en una gran cantidad de características, que suelen ser difíciles de calcular computacionalmente, lo cual no resulta ventajoso para su implementación mediante una aplicación móvil.

Con el fin de superar las limitaciones indicadas anteriormente, se han desarrollado varios algoritmos que detectan automáticamente la FA basándose únicamente en el intervalo RR, que es el tiempo entre dos latidos cardíacos consecutivos.

Este trabajo tiene como objetivo desarrollar aplicaciones y métodos similares para poder aplicar algoritmos predictivos en intervalos RR obtenidos de cualquier sensor biomédico de bajo costo.

Para el desarrollo de la aplicación se hace uso del algoritmo de predicción[16] propuesto, para detectar la fibrilación auricular, que se basa en un análisis cuantitativo de la recurrencia simbólica.

[14] (Censi, y otros, 2016)
[15] (Bollepalli, Challa, Jana, & Patidar, 2017)
[16] (Pérez-Valero, y otros, 2019)

2.4. Entorno de desarrollo

2.4.1. Software

- Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para aplicaciones Android. Antes de Android Studio, Eclipse era el IDE oficial, pero a partir del año 2013 fue remplazado. La primera versión estable del IDE fue lanzada en diciembre de 2014.

Este IDE está basado en el software IntelliJ IDEA de JetBrains y se publica de forma gratuita bajo la licencia Apache 2.0.

Figura 7. Logo de Android Studio

Además del potente editor de código y las herramientas de desarrollo de IntelliJ, Android Studio también proporciona más funciones para mejorar la eficiencia del trabajo al desarrollar aplicaciones de Android, como:

- Sistema de construcción flexible basado en Gradle.
- Emulador rápido y con muchas funciones.
- Entorno unificado dónde se puede desarrollar para todos los dispositivos Android.
- Integración con sistemas de control de versiones (VCS), como pueden ser Git o GitHub.
- Plantillas de código para compilar funciones de aplicaciones de uso frecuente e importar códigos de muestra.
- Varios marcos y herramientas de prueba.
- Previsualización de layouts en distintos dispositivos Android.

Android Studio está disponible para entornos con sistema operativo Windows 2003, Vista, 7, 8 y 10 (plataformas de 32 y 64 bits), GNU / Linux, Linux con GNOME o KDE y un mínimo de 2 GB de RAM, así como para macOS (a partir de 10.8.5).

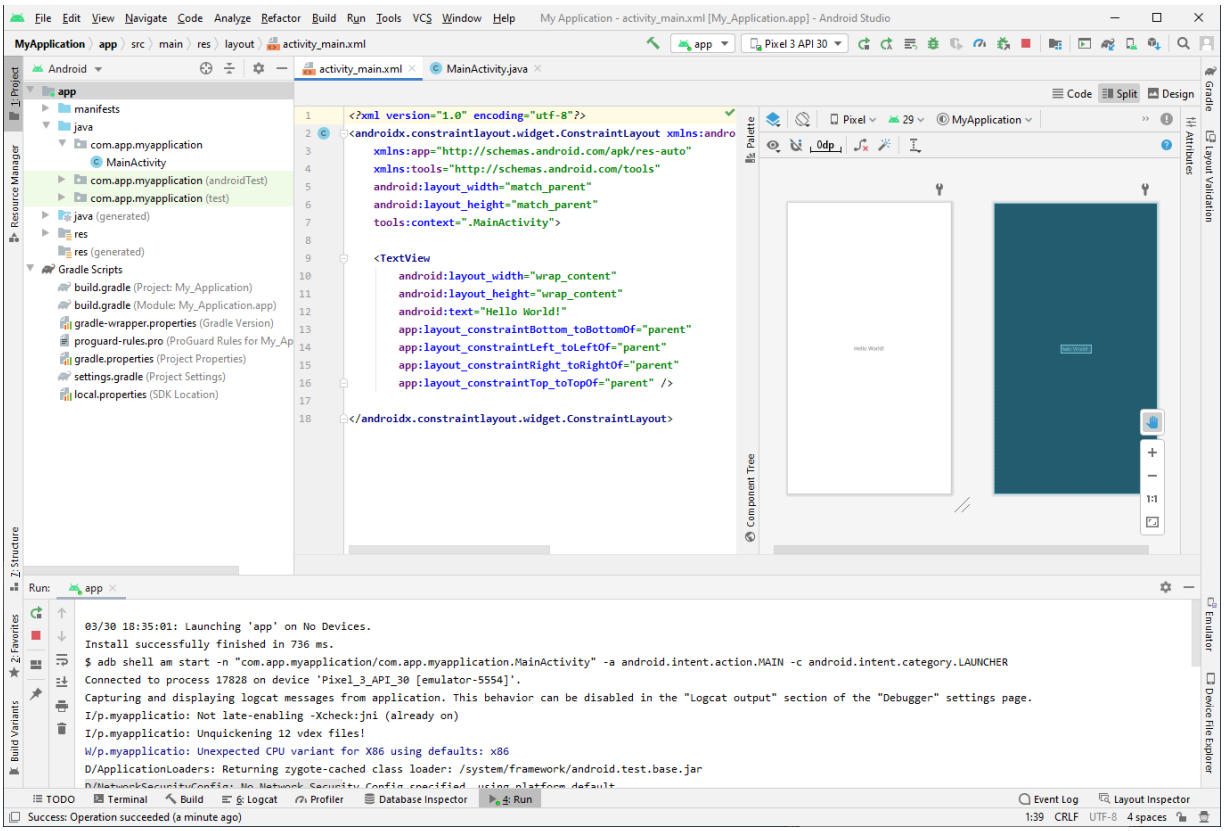

Figura 8. Vista general del IDE Android Studio.

Por todo lo indicado anteriormente, Android Studio ha sido el IDE elegido para el desarrollo de la aplicación. En caso de no haber trabajado con el IDE previamente, como es mi caso, al principio puede resultar complicado entender la estructura de archivos y las funciones de cada uno. Sin embargo, tras pasar varios días utilizando este entorno el desarrollo se hace mucho más fácil y rápido de lo que podría ocurrir con otros entornos.

La página oficial para desarrolladores de Android[17] ha resultado de gran ayuda para el desarrollo de la aplicación ya que contiene una gran cantidad de tutoriales y ejemplos muy útiles para alguien que está iniciándose en el desarrollo de aplicaciones Android.

[17] (Guía para Desarrolladores Android, s.f.)

- **MATLAB**

MATLAB (MATrix LABoratory) es un sistema de cálculo numérico que proporciona un IDE con su propio lenguaje de programación (lenguaje M). Este IDE Se puede utilizar en plataformas Unix, Windows, macOS y GNU / Linux.

Figura 9. Logo de MATLAB.

Las funciones básicas de MATLAB incluyen el procesamiento de matrices, representación de datos y funciones, implementación de algoritmos, creación de interfaz de usuario (GUI) y programas en otros lenguajes y comunicación con otros dispositivos hardware.

Es un software muy utilizado en universidades y es el lenguaje de programación con el que se desarrolló el código original para el algoritmo ReAD-AF[18]. Durante el desarrollo del proyecto se ha hecho uso de MATLAB para estudiar el funcionamiento del algoritmo. Además, dado que el objetivo es la traducción de este algoritmo a Java, se ha usado para hacer pruebas y comparaciones con el código de la aplicación Android.

[18] (MyHeartFitnesApp, 2020)

- **Git**

Git es un sistema de control de versiones distribuido de código abierto y gratuito, diseñado para procesar de forma rápida y eficiente todo, desde proyectos pequeños hasta proyectos considerablemente grandes.

El control de versiones o VCS es un sistema que registra los cambios realizados en un archivo o grupo de archivos durante un período de tiempo para que se pueda recuperar una versión específica más adelante.

Figura 10. Logo de Git.

En el desarrollo de aplicaciones, el control y seguimiento de versiones es fundamental porque nos permite regresar a versiones anteriores de archivos, regresar a versiones anteriores de todo el proyecto, comparar contenido que ha cambiado con el tiempo, etc. Generalmente, usar VCS también significa que, si estropeamos o perdemos archivos, podemos restaurarlos fácilmente.

En el control de versiones distribuido los usuarios descargan el repositorio entero (todas las versiones anteriores) además de la última versión publicada. De esta forma, si el servidor deja de funcionar y estos sistemas están colaborando a través del servidor, cualquier repositorio disponible en el cliente se puede copiar al servidor para su restauración. Cada clon es en realidad una copia completa de todos los datos.

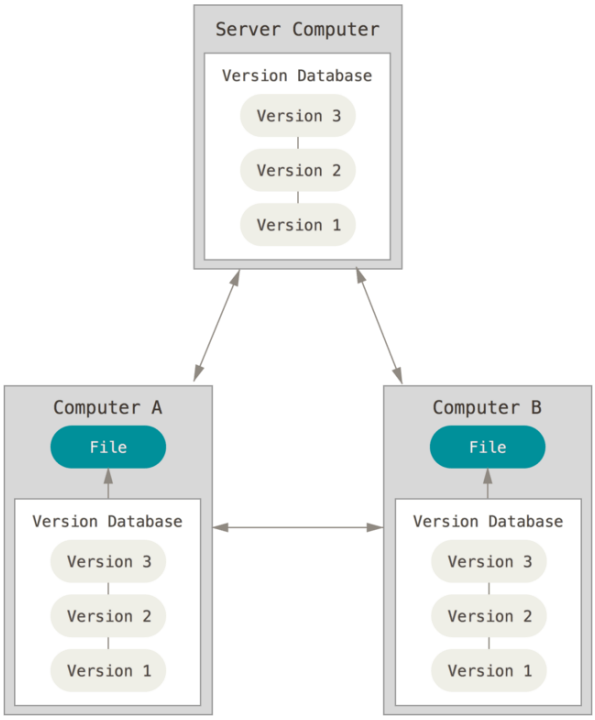

Figura 11. Estructura del Control de Versiones Distribuido.

Además de ser muy útil, Git es fácil de aprender y ocupa poco espacio. Por todas estas funcionalidades, se ha usado durante el desarrollo de la aplicación para mantener un historial de versiones de la aplicación.

- **GitHub**

GitHub es una plataforma basada en la nube creada para alojar proyectos utilizando el sistema de control de versiones Git. Actualmente, es una de las principales plataformas para crear proyectos abiertos para aplicaciones.

Figura 12. Logo de GitHub

El código de los proyectos abiertos alojados en GitHub puede ser descargado y revisado por cualquier usuario, lo que ayuda a mejorar el producto y crear ramificaciones a partir de él. También existe la opción de crear repositorios privados de manera que solo personas autorizadas puedan acceder y contribuir a él.

Además, la interfaz de usuario de GitHub es más fácil de usar que la de Git, lo que la hace accesible para aquellas personas con pocos o ningún conocimiento técnico. Esto significa que se puede incluir a más miembros del equipo en el progreso y la gestión de un proyecto, haciendo que el proceso de desarrollo sea más fluido.

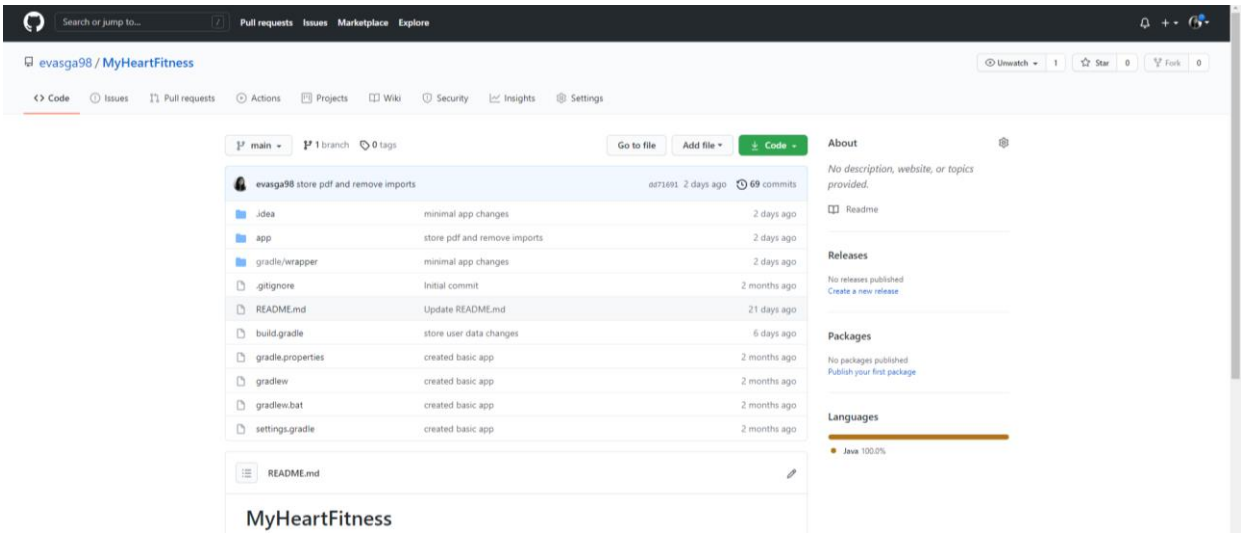

Figura 13. Repositorio de la aplicación MyHeartFitness en GitHub.

GitHub ha sido la plataforma utilizada durante el desarrollo de la aplicación MyHeartFitness y se encuentra en un repositorio público[19].

[19] (evasga98/MyHeartFitness, 2021)

2.4.2. Hardware

Para realizar el desarrollo de aplicaciones Android en Android Studio lo único que necesitamos es tener un equipo con alguna de las siguientes características:

- Windows 2003, Vista, 7, 8 y 10 en plataformas de 32 o 64 bits.
- GNU / Linux, Linux con GNOME o KDE y un mínimo de 2 GB de RAM.
- macOS (a partir de 10.8.5).

El desarrollo de la aplicación se ha realizado en dos equipos con dos sistemas operativos distintos:

- Windows 10, plataforma de 64 bits y 8GB de RAM.
- Ubuntu 20.04, plataforma de 64 bits y 16 GB de RAM.

Desde ambos equipos ha sido posible trabajar con el IDE sin ningún problema y la integración con Git ha ayudado mucho a poder realizar el control de versiones desde los dos equipos, así como a trabajar siempre desde la última versión publicada.

Por otro lado, las pruebas de la aplicación no se han realizado en ningún dispositivo Android físico ya que no se dispone de uno. Para ello se ha usado el emulador que proporciona Android Studio donde podemos probar la aplicación en distintos dispositivos virtuales con distintas versiones Android.

3. Algoritmo ReAD-AF

ReAD-AF (Recurrence Analysis to Detect Atrial Fibrilation)[20] es un algoritmo novedoso basado en un modelo logístico que aplica Análisis de Cuantificación Repetida Simbólica (SQRA) para la detección de fibrilación auricular[21]. El uso de SQRA nos permite analizar cambios dinámicos en una serie temporal. El algoritmo tiene como objetivo determinar si un individuo se encuentra en un estado normal (ritmo Sinusal Normal NS) o tiene fibrilación atrial (AF). Esta solución se basa únicamente en el análisis de los intervalos RR de la señal de un electrocardiograma, por lo que el coste de preprocesamiento de los datos es mínimo.

Por lo tanto, ReAD-AF es un algoritmo de regresión logística que usa SRQA en una serie temporal de intervalos RR. Los intervalos RR son mediciones de tiempo entre latidos del corazón consecutivos.

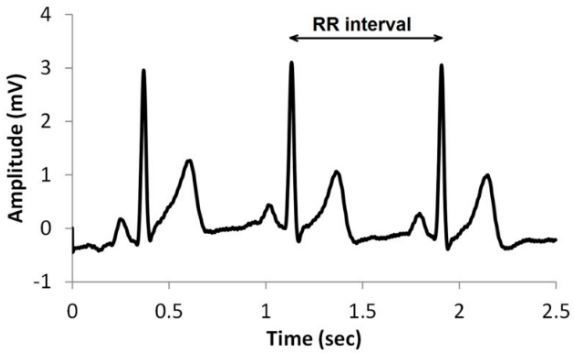

Figura 14. Intervalo RR en un Electrocardiograma.

El algoritmo ReAD-AF es robusto y capaz de detectar personas con AF con gran precisión. Además, el algoritmo trabaja con un único dato de entrada, esto significa una reducción en el coste computacional del modelo y, por lo tanto, facilita la portabilidad a plataformas móviles como son los teléfonos móviles.

Este algoritmo está escrito en MATLAB originalmente[22] y en el Anexo A se encuentra su pseudocódigo.

[20] (Pérez-Valero, Garcia-Sanchez, Ruiz Marín, & Garcia-Haro, 2020)
[21] (Pérez-Valero, y otros, 2019)
[22] (MyHeartFitnesApp, 2020)

4. Funcionamiento de la aplicación

La estructura del funcionamiento de la aplicación es muy sencilla con el fin de facilitar su uso al usuario. Los usuarios pueden hacer uso de la aplicación sin necesidad de registrarse ni proporcionar datos personales. A su vez, la aplicación se comunicará con una base de datos para almacenar los datos que el usuario obtiene como resultado del algoritmo.

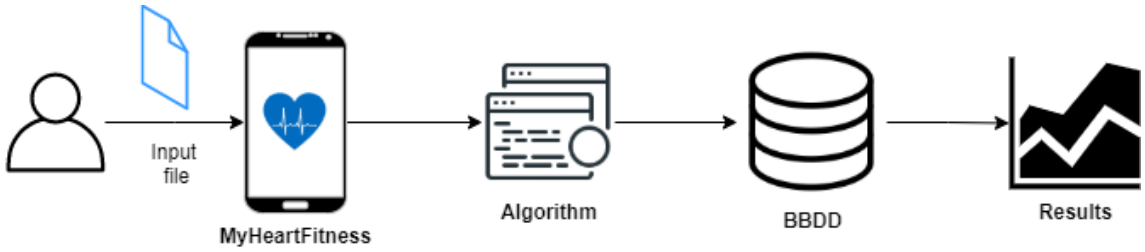

Figura 15. MyHeartFitness. Estructura general.

En la Figura 15 se representa el funcionamiento de la aplicación de manera general. El usuario que está haciendo uso de la aplicación MyHeartFitness introduce un archivo desde el almacenamiento del dispositivo que tendrá formato PDF o JPEG. La aplicación entonces ejecutará el algoritmo ReAD-AF que cuando termine de ejecutarse, almacenará los resultados obtenidos en una base de datos. Por último, estos resultados obtenidos podrán ser consultados por el usuario en la aplicación. Los resultados se mostrarán en la vista principal de la aplicación en forma de gráfica.

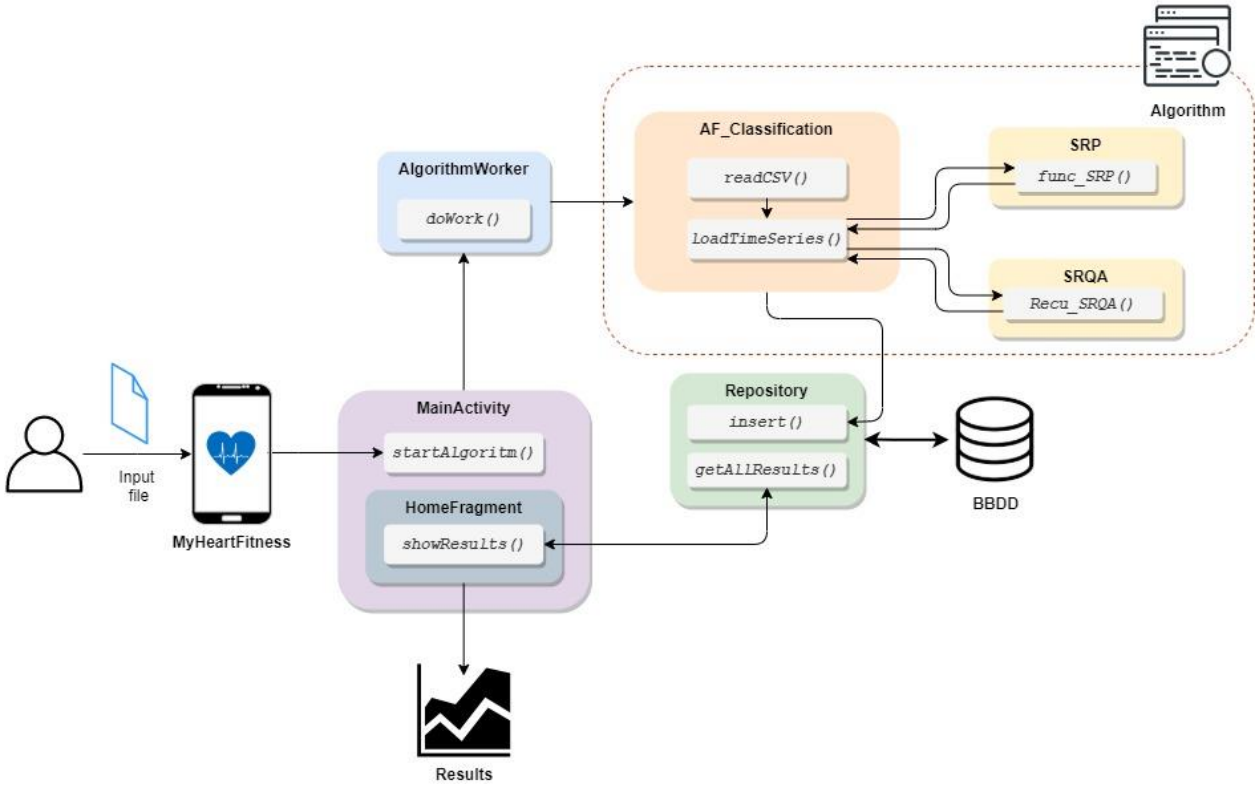

Figura 16. MyHeartFitness. Estructura detallada.

Entrando en más detalle, la Figura 16 muestra un esquema donde se especifica la operativa de la aplicación para ejecutar el algoritmo ReAD-AF y mostrar los resultados al usuario.

Cuando el usuario requiere la ejecución del algoritmo, desde el módulo MainActivity se inicia el algoritmo en segundo plano mediante una instancia de la clase AlgorithmWorker.

La clase AlgorithmWorker, comienza la ejecución del algoritmo en segundo plano llamando al método `readCSV()` del módulo AF_Classification, que implementa el algoritmo ReAD-AF. Con esta función, se toman los datos de un fichero llamado data.csv donde se encuentran almacenados los intervalos RR que se van a procesar por el algoritmo. Una vez se han leído los datos del archivo `CSV`, se comienza el procesado de estos mediante la función `loadTimeSeries()` cuyo pseudocódigo se encuentra en el Anexo A. Esta función nos dará el resultado de clasificación del usuario para cada uno de los intervalos leídos del archivo de entrada.

Al finalizar, el módulo AF_Classification solicitará guardar los resultados en una BBDD llamada `results_database` mediante el método `insert()` del módulo Repository. Para

mostrar los datos, en la clase HomeFragment se llama al método `showResults()` que consulta la base de datos y lee los datos obtenidos del algoritmo. Tras leer estos datos de la BBDD, se representan en la aplicación para que el usuario pueda interpretarlos mediante una gráfica.

En el siguiente punto de la memoria, se realiza una explicación de las diferentes clases implementadas (Figura 16) para el funcionamiento de la aplicación.

5. Implementación de la aplicación

En este punto se va a explicar en detalle los aspectos a destacar en la implementación de la aplicación MyHeartFitness, los elementos que la componen y sus funciones.

5.1. Creación del proyecto

Para la implementación de la aplicación lo primero es crear un nuevo proyecto en Android Studio. El desarrollo de la aplicación en cuestión, parte de la última versión del software Android Studio y se inicia con una plantilla que se nos ofrece por defecto.

Una vez elegimos la opción de crear un nuevo proyecto, Android Studio nos permite crearlo a partir de una plantilla. En este caso se ha decidido partir de la plantilla "Bottom Navigation Activity" ya que crea una actividad con tres vistas por defecto.

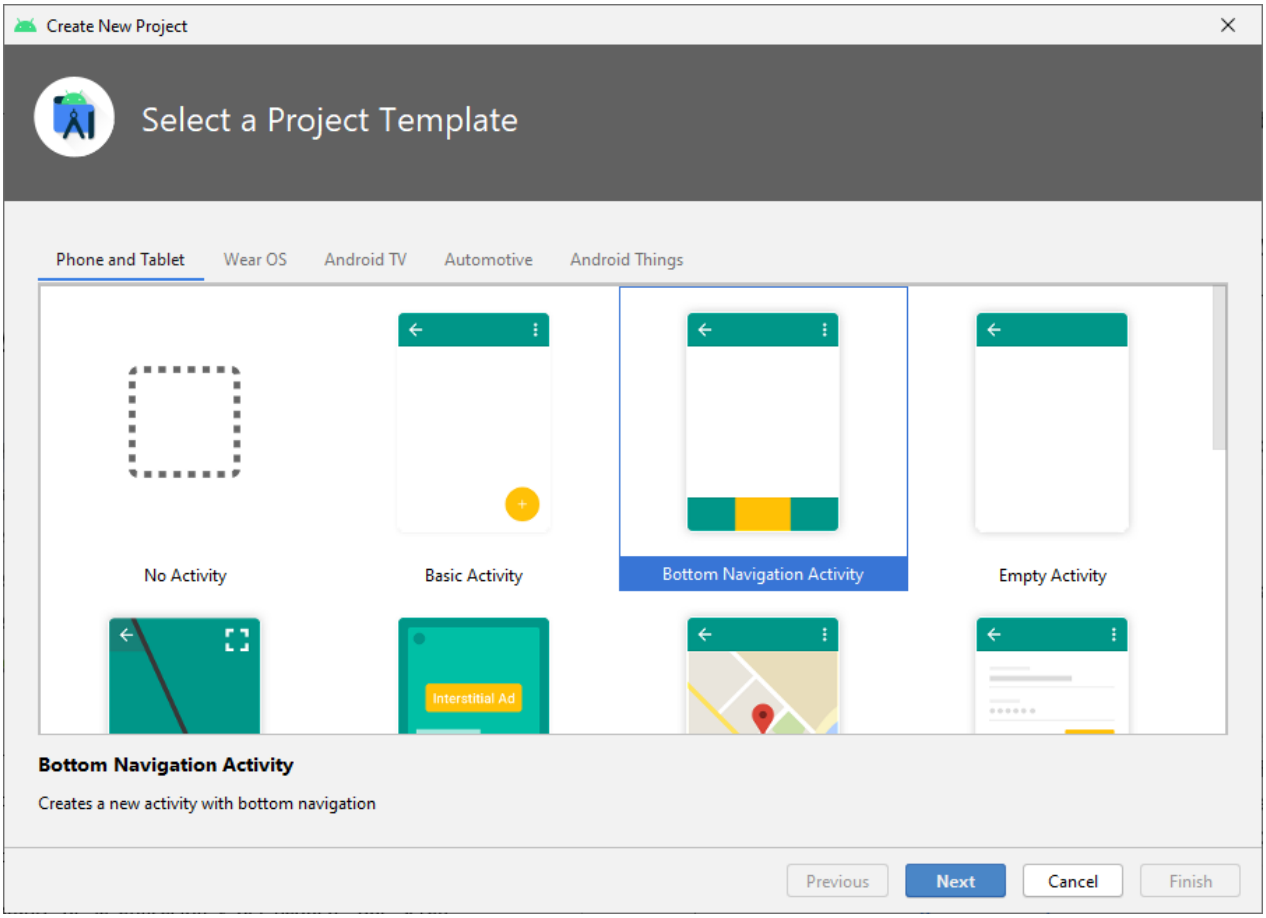

Figura 17. Creación de un Proyecto en Android Studio.

Lo siguiente es realizar la configuración del proyecto. Para ello introducimos el nombre de la aplicación y del paquete, que serán MyHeartFitness y com.app.myheartfitness respectivamente. En este momento es donde tenemos que decidir a qué dispositivos Android estará orientada la aplicación. Como queremos llegar al mayor número de dispositivos posible, seleccionamos el SDK mínimo API 16 ya que Android Studio nos indica que las aplicaciones desarrolladas con este API funcionarán en aproximadamente un 99,8% de dispositivos Android.

- Nombre: MyHeartFitness
- Nombre del paquete: com.app.myheartfitness
- Lenguaje: Java
- SDK Mínimo: API 16

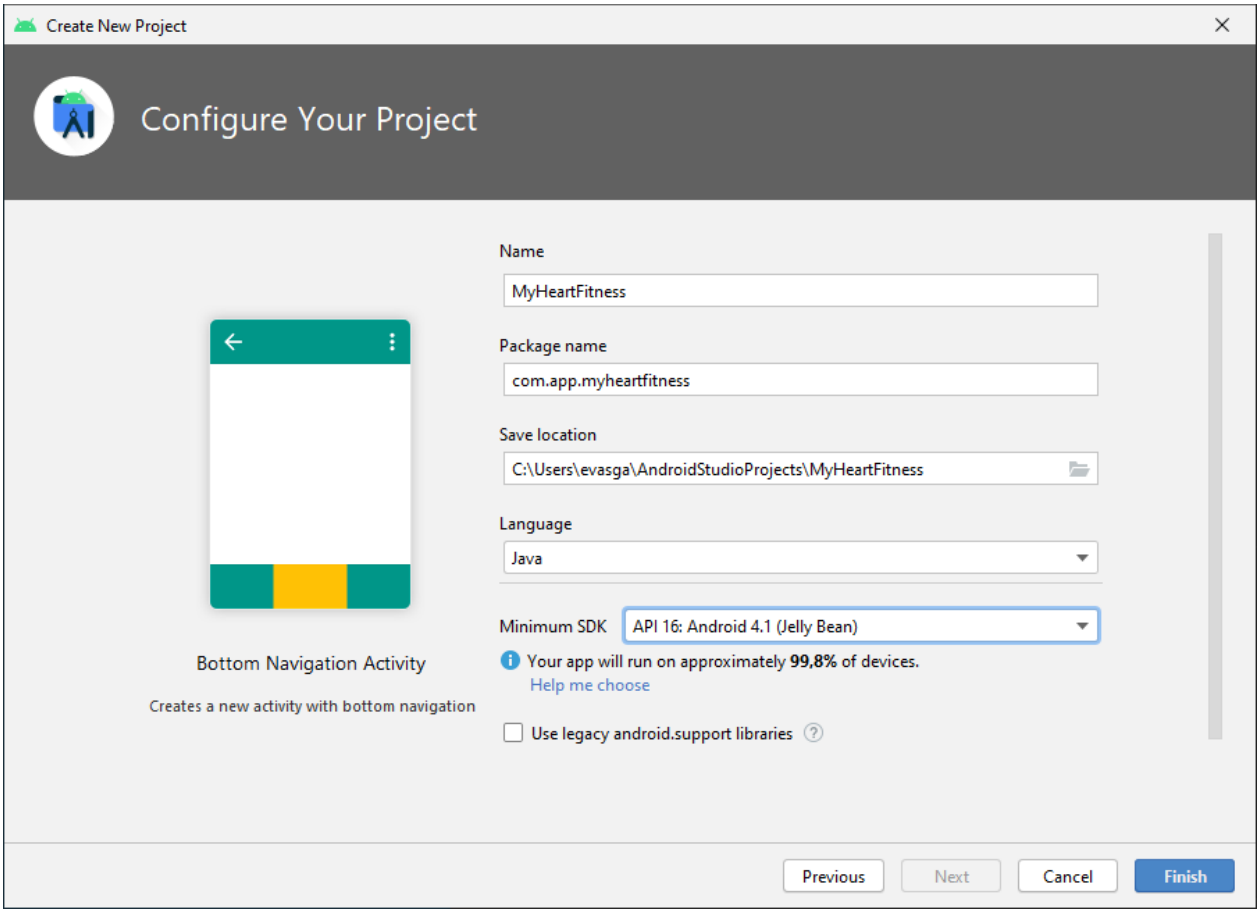

Figura 18. Configuración de un Proyecto en Android Studio.

Como se ha indicado previamente, el desarrollo de aplicaciones en Android Studio puede realizarse en dos lenguajes de programación principales: Java y Kotlin. Se ha decidido implementar la aplicación en Java dado mi conocimiento y experiencia previa con este lenguaje de programación.

5.2. Estructura del proyecto

Una vez hemos terminado de configurar el proyecto inicial de Android Studio tenemos la siguiente estructura de archivos y directorios:

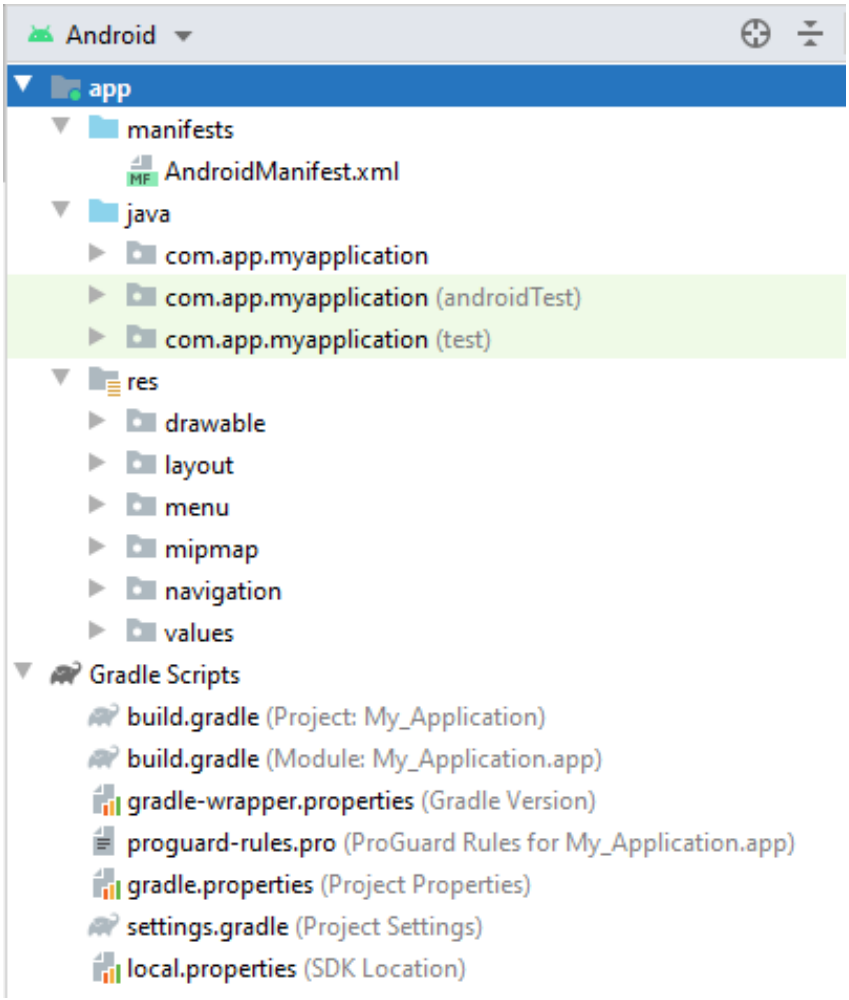

Figura 19. MyHeartFitness. Estructura del Proyecto en Android Studio.

5.2.1. Manifests

La carpeta Manifests contiene el archivo AndroidManifest.xml que se utiliza para crear aplicaciones de Android.

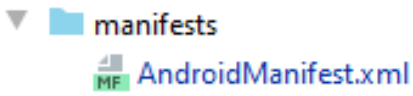

Figura 20. MyHeartFitness. Directorio Manifests.

Este archivo contiene información sobre nuestra aplicación, como su nombre, la versión de Android, los metadatos, la actividad con la que se inicia la aplicación y otros componentes de la aplicación. Actúa como intermediario entre el sistema operativo Android y nuestra aplicación.

Además de todo esto, en el archivo AndroidManifests también se indican los permisos que requiere la aplicación. En nuestro caso estos permisos son los siguientes:

1. Acceso para leer y escribir en la memoria del dispositivo con el objeto de poder leer los archivos PDF y JPEG que introduce el usuario y guardarlos para procesarlos posteriormente.
2. Acceso para usar los sensores biométricos del dispositivo en caso de que el usuario quiera autenticación para entrar en la aplicación. Este permiso no es obligatorio, pero si el usuario solicita autenticación tendrá que ser activado.

Otra de las características que tiene este archivo es que nos permite definir cuál es la actividad con la que se inicia la aplicación, en nuestro caso esta actividad es AuthenticateActivity.

En el Anexo B se encuentra el contenido (código) del archivo AndroidManifests de la aplicación MyHeartFitness.

5.2.2. Java

La carpeta Java contiene el paquete `com.myheartfitness.app` donde se encuentran otros paquetes que alojan todos los archivos de código fuente de Java que creamos durante el desarrollo de la aplicación, incluidos otros archivos de prueba.

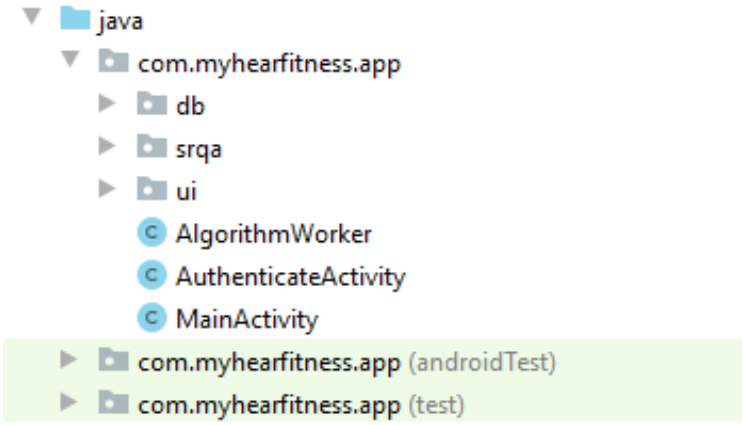

Figura 21. MyHeartFitness. Directorio Java.

En el directorio `db` se encuentran los archivos de código relacionados con la lectura y escritura en la base de datos que usa la aplicación. En el directorio `sqra` se encuentran las clases que contienen el código del algoritmo. Por último, en el directorio `ui` se ubican los archivos encargados de controlar la interfaz gráfica y proporcionar los datos para la visualización de la aplicación.

A parte de los directorios, tenemos otros tres archivos principales:

- **AlgorithmWorker**: es una clase que nos permite ejecutar el algoritmo de la aplicación en segundo plano.
- **AuthenticateActivity**: es una clase que inicia una actividad para solicitar autenticación al usuario antes de permitirle entrar en la aplicación.
- **MainActivity** es una clase que define la actividad principal de la aplicación. Al ejecutar la aplicación se inicia una instancia de esta Actividad y se carga el diseño de la aplicación.

5.2.3. Resources

Es la carpeta de recursos y es una carpeta esencial porque contiene todas las fuentes que no son de código, como imágenes, diseños XML y cadenas de interfaz de usuario para nuestra aplicación de Android.

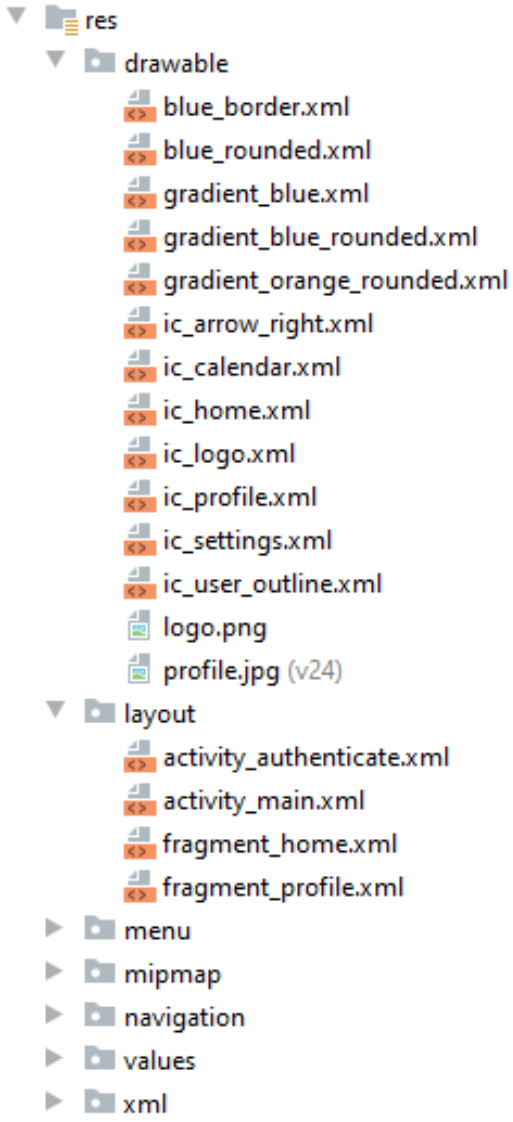

Figura 22. MyHeartFitness. Directorio Resources.

A continuación, se describen sus principales elementos:

- `/drawable` contiene diferentes tipos de imágenes que se utilizan para la aplicación. Aquí es donde se agregan los iconos, fondos e imágenes que usará nuestra aplicación.
- `/layout` contiene todos los archivos de diseño XML que usamos para definir la interfaz de usuario de la aplicación. Estos archivos suelen ir asociados a una Actividad ya que definen el diseño y estructura de la interfaz de usuario de estas.
- `/menu` contiene archivos XML que definen los menús de la aplicación (por ejemplo, menú de opciones, menú contextual o submenú)
- `/mipmap` contiene el archivo `launcher.xml`, que se utiliza para definir el icono de la aplicación. Contiene tipos de iconos con diferentes densidades, según el tamaño del dispositivo, como hdpi, mdpi, xhdpi.
- `/value` contiene archivos XML con valores simples, como strings, valores enteros y colores; entre los archivos más importantes destaca el archivo strings.xml que contiene recursos.

5.2.4. Gradle

El sistema de compilación de Android compila recursos y código fuente de la aplicación, y los empaqueta en APK que podemos probar, implementar, firmar y distribuir. Android Studio usa Gradle, que se refiere a un sistema de compilación automatizado, que contiene muchos archivos que se utilizan para definir configuraciones de compilación y que se pueden aplicar a todos los módulos de la aplicación.

En `build.gradle` (proyecto), hay scripts de compilación, y en `build.gradle` (módulo), se utilizan complementos e implementaciones para generar configuraciones que se pueden aplicar a todos los módulos de nuestra aplicación.

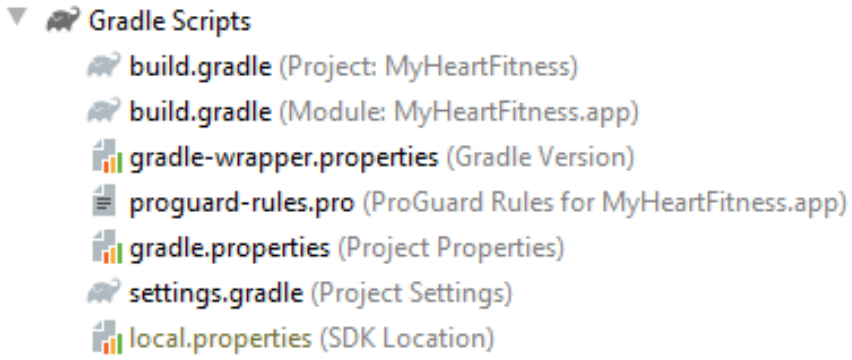

Figura 23. MyHeartFitness. Directorio Gradle.

5.3. Desarrollo de la aplicación

El desarrollo de la aplicación se ha realizado siguiendo varios pasos que se van a ir detallando a continuación.

5.3.1. Diseño

Una vez hemos creado, configurado y entendido la estructura del proyecto de Android Studio podemos comenzar a darle forma a la estructura de la aplicación.

La aplicación va a tener tres vistas que se muestran en la Figura 24 y desde cada una de ellas el usuario podrá realizar una serie de acciones diferentes.

La aplicación tendrá la siguiente estructura:

- **Home**: esta es la vista principal de la aplicación y la que se mostrará por defecto cuando abramos la aplicación. En ella podremos realizar distintas acciones como la subida de un archivo o la visualización de los datos obtenidos ejecutando el algoritmo.
- **Profile**: en esta vista se mostrarán los datos del usuario si este ha introducido algunos.
- **Settings**: esta vista nos permite introducir información del usuario como su nombre, fecha de nacimiento y foto de perfil. Además, en esta sección es donde el usuario podrá seleccionar si desea que la aplicación solicite autenticación antes de entrar o no.

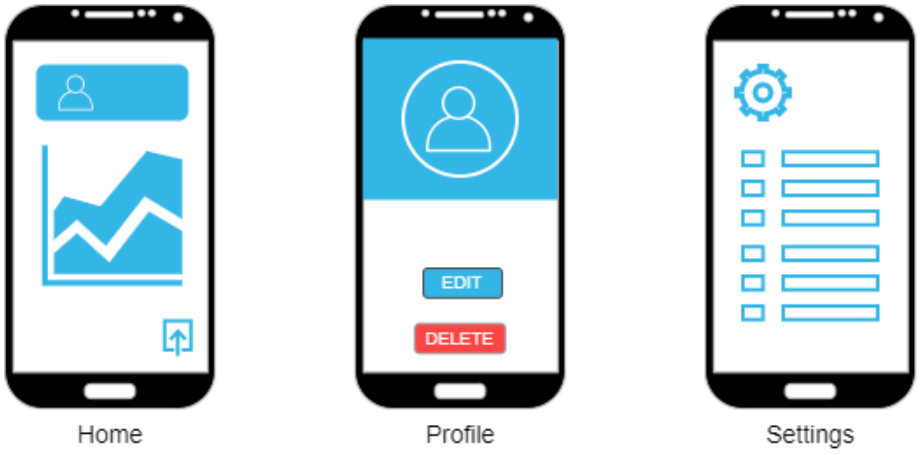

Figura 24. MyHeartFitness. Esquema de vistas.

Para implementar las diferentes vistas en la interfaz de usuario, usamos la clase Activity. Toda aplicación Android debe de tener al menos una clase Activity, pero puede tener más. En una aplicación Android, primero se muestra la actividad principal y a partir de esta se abren otras actividades en caso de que las haya. En MyHeartFitness tenemos dos clases Activity (AuthenticateActivity y MainActivity) cuya funcionalidad se indica a continuación.

- **AuthenticateActivity**

La actividad AuthenticateActivity se ha creado con el fin de dar la opción al usuario de proteger la información de MyHeartFitness solicitando autenticación. Un método para implementar esta funcionalidad es solicitar autenticación biométrica, por ejemplo, con reconocimiento facial o de huella digital.

Esta actividad se encarga de comprobar si el usuario ha decidido usar autenticación o no antes de dar acceso al contenido de la aplicación. Por defecto, la aplicación no va a solicitar autenticación por lo que esta actividad pasará directamente a MainActivity y entrará directamente a la aplicación. Sin embargo, si el usuario ha activado la autenticación en la configuración de la aplicación, entonces se requerirá que se identifique mediante huella digital si hay alguna configurada o mediante el PIN de desbloqueo del teléfono. Hasta que la autenticación no sea correcta el usuario no podrá acceder a la aplicación. El funcionamiento de esta actividad se puede ver en la Figura 25.

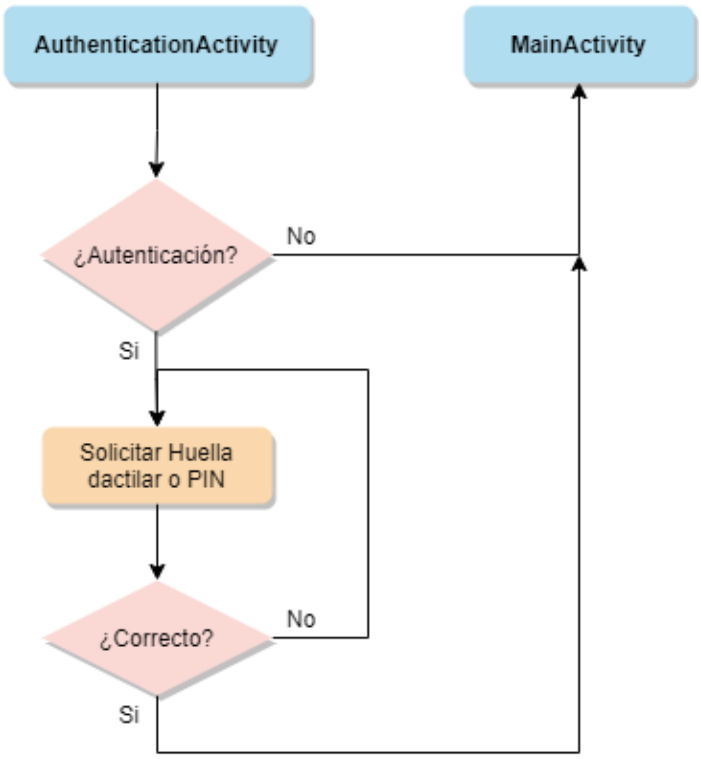

Figura 25. MyHeartFitness. Funcionamiento de AuthenticateActivity.

Para poder usar un autenticador al inicio de la aplicación, el usuario debe tener configurado un PIN, patrón, contraseña y/o huella digital. Si el usuario aún no tiene uno, el proceso de registro biométrico le pedirá que cree uno.

La implementación del autenticador se ha realizado usando la biblioteca Biométrica[23], la cual autentica con biometría o credenciales de dispositivo y realiza operaciones de encriptación.

Por lo tanto, si el usuario activa la opción de autenticarse al inicio de la aplicación, AuthenticateActivity primero comprobará si estos métodos de autenticación están disponibles en el dispositivo. Si es así, el diálogo de la Figura 26 aparecerá al iniciar la aplicación.

Hay dos maneras de verificar la identidad del usuario llegados a este punto: huella digital o PIN.

[23] (Librería Biométrica, s.f.)

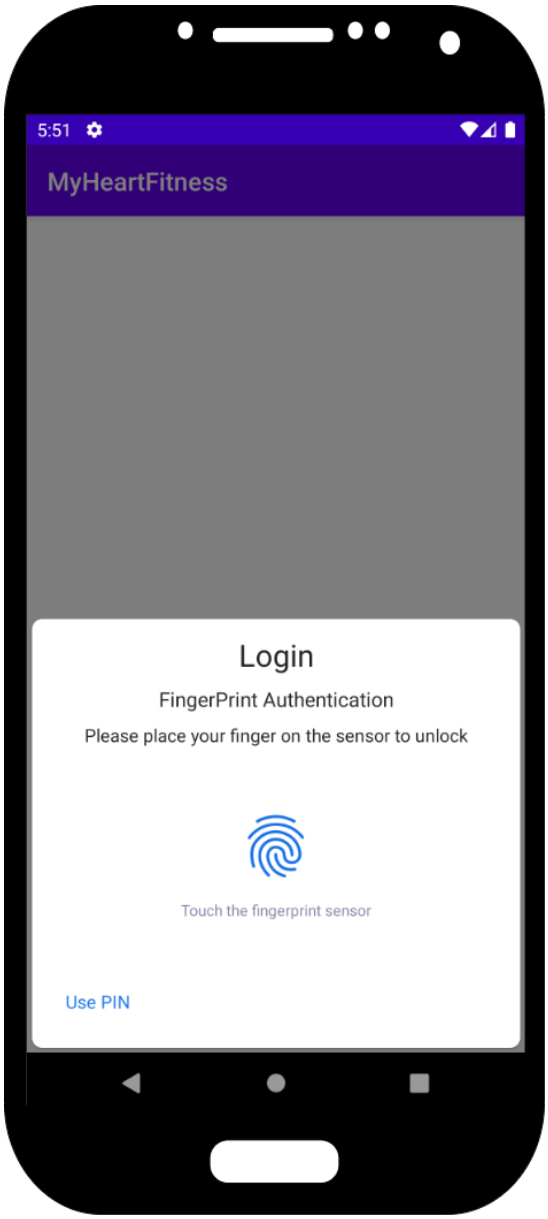

Figura 26. MyHeartFitness. Autenticación por huella digital.

Si el dispositivo del usuario no tiene huella digital ya sea porque no lo soporta o porque no hay ninguna configurada, tendremos la opción de autenticarnos con PIN como se muestra en la Figura 27.

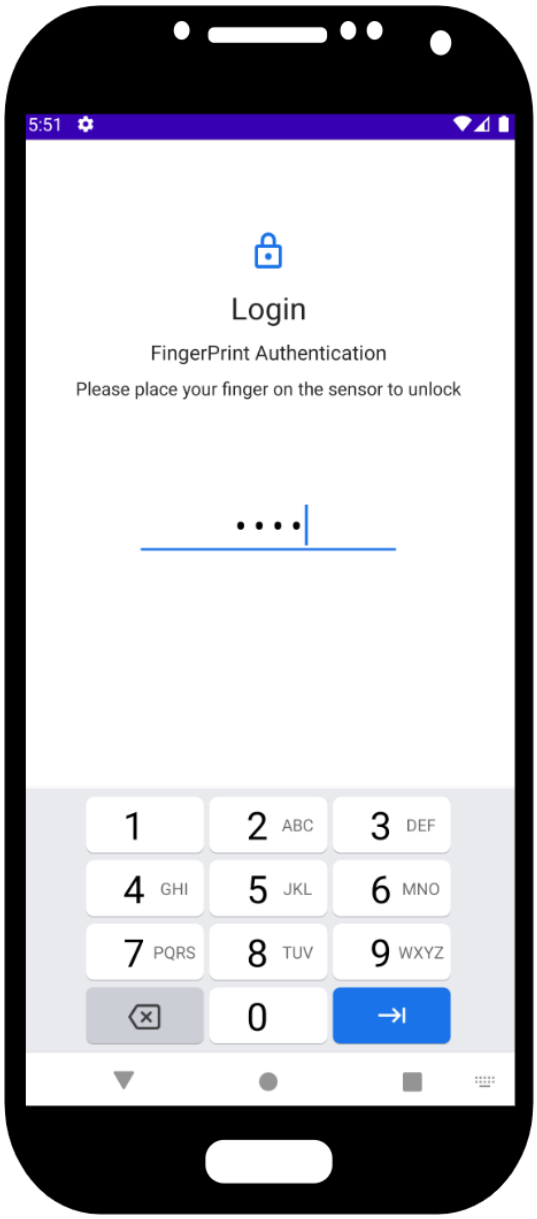

Figura 27. MyHeartFitness. Autenticación por PIN.

Como todas las actividades, AuthenticateActivity tiene asociado un archivo en formato XML que define su diseño. Esta actividad solo nos muestra el dialogo para la autenticación o pasa directamente a MainActivity por lo que el archivo activity_authenticate.xml no tiene ningún componente gráfico.

- **MainActivity**

Como se ha explicado al comienzo de esta sección, la aplicación va a contener tres secciones distintas con las que el usuario podrá interactuar. La Actividad MainActivity va a alojar estas tres vistas que van a ser manejadas por una clase Fragment cada una.

Los Fragmentos representan partes reutilizables de la interfaz de usuario de la aplicación. Cada Fragmento define y administra su propio diseño, tiene su propio ciclo de vida y puede manejar sus propios eventos de entrada. Sin embargo, estos no pueden existir por sí solos si no que están alojados en una Actividad y son controlados por esta. En este caso MainActivity alojará los fragmentos correspondientes a cada una de las vistas de la aplicación.

Dado que los Fragmentos no son una vista, estos se añaden dentro de un grupo de vistas dentro de la Actividad. El fragmento se muestra dentro del grupo de vistas como se puede ver en la Figura 28.

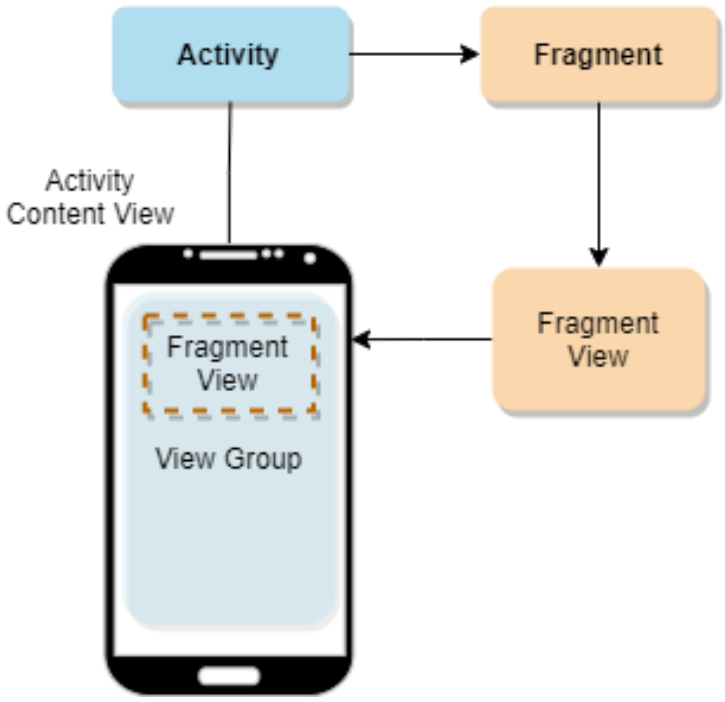

Figura 28. Funcionamiento de un Fragment.

La vista vinculada a MainActivity se llama activity_main.xml y contiene una barra de navegación inferior donde podemos ir cambiando entre las distintas secciones de la aplicación. Dentro de esta vista será donde se muestren los diferentes Fragmentos creados para las tres vistas.

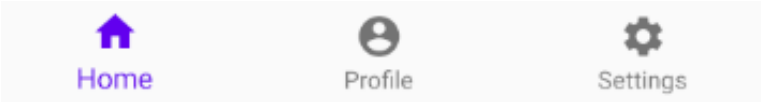

Figura 29. MyHeartFitness. Barra de Navegación.

Para crear la interfaz de usuario cada sección consta de un Fragmento y su archivo de diseño correspondiente. Además, haremos uso de una clase ViewModel para cada sección con tal de proporcionar los datos para cada componente de la interfaz gráfica.

Las clases ViewModel permiten almacenar y administrar datos relacionados con la IU de manera optimizada para los ciclos de vida de la aplicación. Con la clase ViewModel podremos conservar los datos después de que se realicen cambios en la aplicación, como las rotaciones de pantalla.

Por lo tanto, para cada sección definimos los siguientes archivos:

1. Los archivos de diseño de cada sección que definen el diseño de la interfaz gráfica para la pantalla. En el caso de Home y Profile tendrán asociado un archivo en el directorio res/layouts. Por otro lado, Settings tendrá asociado un archivo en res/xml ya que es una pestaña de preferencias.

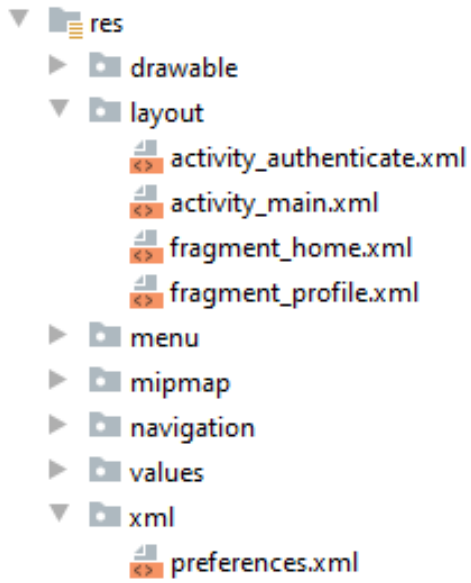

Figura 30. MyHeartFitness. Archivos de diseño.

2. Las clases Fragment y ViewModel que nos permiten controlar la interfaz gráfica y proporcionar los datos para la visualización respectivamente.

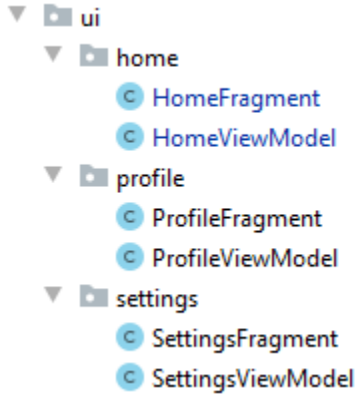

Figura 31. MyHeartFitness. Archivos para controlar la GUI.

A continuación, se va a hacer una descripción más detallada de cada una de las vistas dentro de MainActivity, sus componentes y funcionalidades.

- **Home**

Esta sección es la que se muestra por defecto cuando abrimos la aplicación y la podemos ver en la Figura 32. En ella podemos encontrar un resumen de los datos del usuario, subir archivos a la aplicación y ver los resultados que nos proporciona el algoritmo.

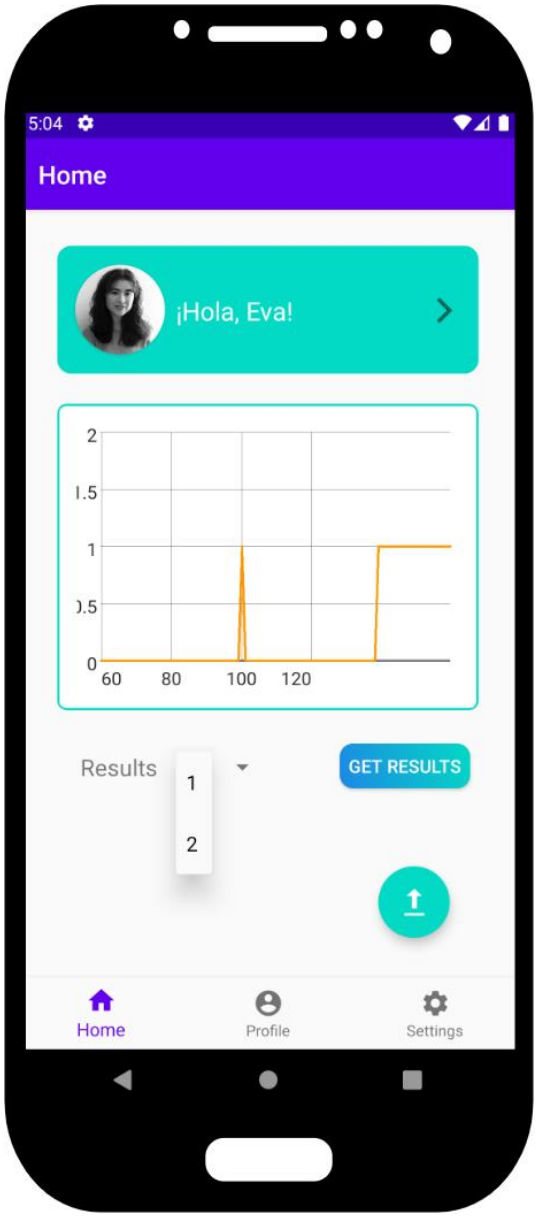

Figura 32. MyHeartFitness. Home.

Si el usuario quiere subir un archivo PDF o JPEG[24] con un electrocardiograma para obtener resultados del algoritmo lo hará desde esta vista, haciendo clic en el botón que se encuentra en la esquina inferior derecha.

Para ejecutar el algoritmo ReAD-AF, el usuario tiene que hacer clic en el botón "Get Results" después de haber subido el archivo PDF o JPEG a la aplicación, que hará que el algoritmo comience a ejecutarse mediante el uso de un WorkManager.

WorkManager es una API que facilita la programación de tareas asíncronas y diferibles que van a ejecutarse incluso cuando la aplicación esté cerrada. Esto es muy importante ya que de esta manera la aplicación no se quedará congelada mientras se ejecuta el algoritmo. Además, el algoritmo no dejará de ejecutarse, aunque la aplicación se cierre por lo que el usuario podrá seguir usando el dispositivo mientras se obtienen los resultados.

Los resultados se mostrarán en esta vista en forma de gráfica en la que el usuario podrá desplazarse si el tamaño de los resultados es muy grande. Además, se puede disponer de varios resultados en la aplicación por lo que con el selector de debajo de la gráfica se pueden seleccionar resultados a visualizar.

[24] La extracción de datos de un archivo PDF o JPEG para el algoritmo no está implementada, pero si está implementado el proceso de subir y guardar el archivo en la aplicación para después procesarlo.

- **Profile**

En la vista de Profile podemos ver los datos personales del usuario si este los ha introducido, ya que como se ha indicado antes no son necesarios para el uso de la aplicación.

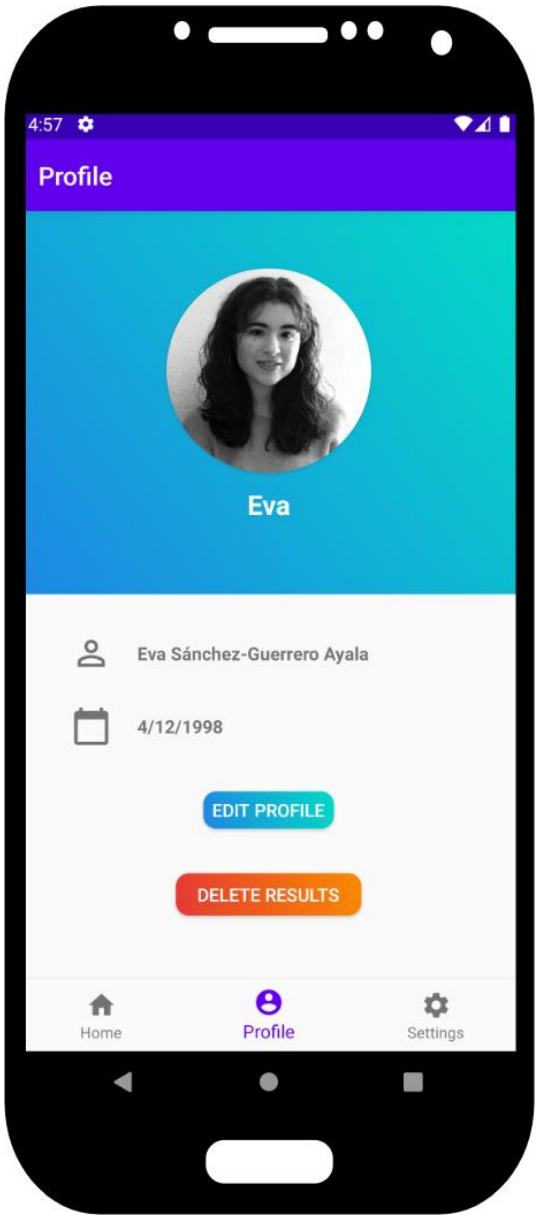

Figura 33. MyHeartFitness. Profile.

Observando la Figura 33 podemos ver que el usuario puede modificar los datos si lo desea haciendo clic en el botón “Edit Profile”. Este botón nos llevará a la vista de preferencias donde el usuario podrá modificar sus datos personales.

Por otro lado, desde esta vista el usuario también podrá realizar un borrado total de todos los resultados obtenidos por el algoritmo haciendo clic en el botón “Delete results”. De esta manera se liberará espacio si los resultados almacenados ya no son necesarios para el usuario.

- **Settings**

En esta vista el usuario podrá hacer algunas configuraciones de la aplicación. Para implementar esta sección se ha usado una clase que extiende PreferenceFragmentCompat. PreferenceFragmentCompat es el punto de entrada para usar la biblioteca de preferencias de Android. Este fragmento muestra al usuario las diferentes opciones que el usuario puede configurar.

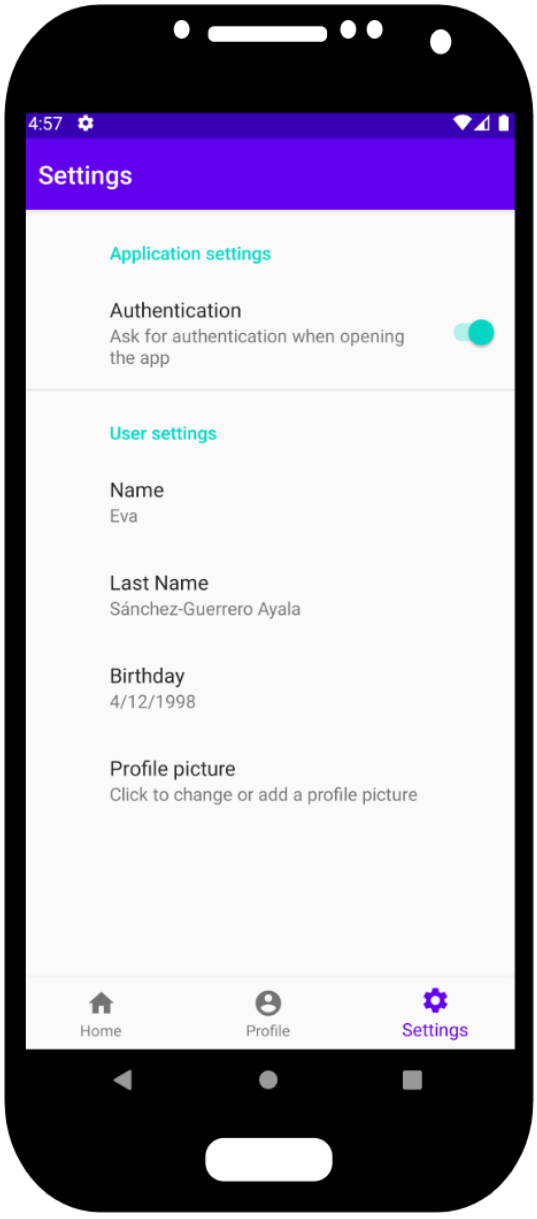

Figura 34. MyHeartFitness. Settings.

Como se puede apreciar en la Figura 34, esta vista permite al usuario introducir una serie de datos personales en la aplicación, aunque, como se ha indicado antes, estos datos no son necesarios para el uso de la aplicación. Además, en esta vista el usuario puede seleccionar si quiere usar autenticación en la aplicación o no.

Para obtener todas estas preferencias, hacemos uso de la instancia SharedPreferences que es una manera de almacenamiento en aplicaciones Android y nos permite obtener la configuración que el usuario ha realizado.

5.3.2. Algoritmo

La funcionalidad principal de esta aplicación es ejecutar el algoritmo ReAD-AF que se ha introducido anteriormente en el punto 4. Para ejecutar el algoritmo ReAD-AF en la aplicación, se han creado una serie de clases que lo implementan, así como una clase para manejar su ejecución en segundo plano. Esto es muy importante ya que para ejecutar el algoritmo tenemos que asegurarnos que lo hacemos de manera que el resto de la aplicación no se quede bloqueada ya que la ejecución del algoritmo puede llevar de unos segundos a minutos, según el tamaño de los datos de entrada.

Para realizar la ejecución del algoritmo en segundo plano se ha hecho uso de una clase que extiende la clase Worker y se llama AlgorithmWorker. Los Workers funcionan de manera que ejecutan un método llamado `doWork()` de forma asíncrona en un subproceso proporcionado por un WorkManager. En nuestra aplicación, AlgorithmWorker (Anexo D) llama a una función que comienza la ejecución del algoritmo ReAD-AF en su método `doWork()`.

Una vez definido el trabajo que ejecuta el algoritmo, hay que programarlo mediante el servicio WorkManager para que se ejecute. WorkManager ofrece mucha flexibilidad para la programación de tu trabajo. Con WorkManager podemos programar una tarea para que se ejecute de forma periódica en un intervalo de tiempo o una única vez.

Para ejecutar el trabajo, tenemos que usar un objeto WorkRequest que define cómo y cuándo se debe ejecutar el trabajo.

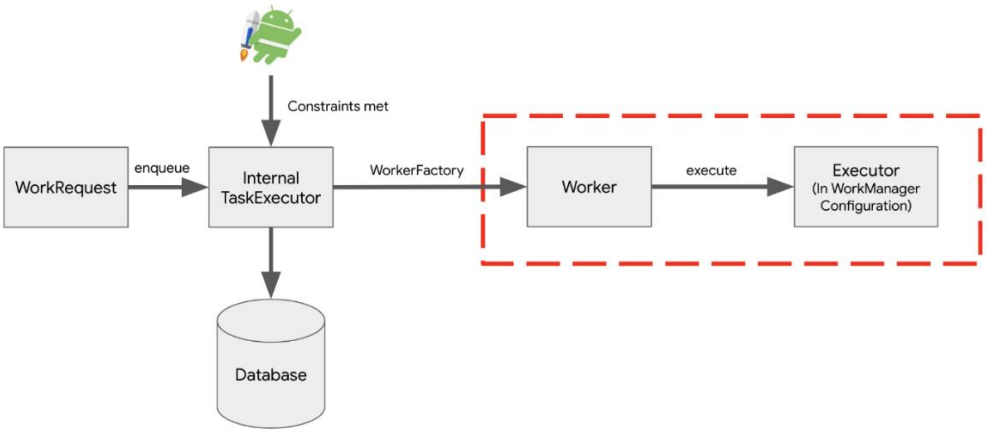

Figura 35. Ejecución en segundo plano con un Worker.

En la aplicación MyHearFitness, cuando el usuario inicia la ejecución del algoritmo, lo hace mediante un botón en la vista "Home" como podemos ver en la Figura 36. Este botón llama a un método en la Actividad MainActivity que se llama `startAlgorithm()`. La función `startAlgorithm()` programa y define la ejecución del algoritmo ReAD-AF mediante el servicio WorkManager y el objeto WorkRequest. En el método, usamos solicitud de tipo OneTimeWorkRequest para que el algoritmo se ejecute una única vez.

Además, en la solicitud se indican dos restricciones:

1. Que la batería del dispositivo no sea baja.
2. Que el almacenamiento del dispositivo no sea bajo ya que tras ejecutar el algoritmo tenemos que almacenar los datos resultantes en dicho dispositivo.

Si estas condiciones se cumplen, se comenzará a ejecutar el algoritmo. El código que implementa esta funcionalidad se encuentra en el Anexo E.

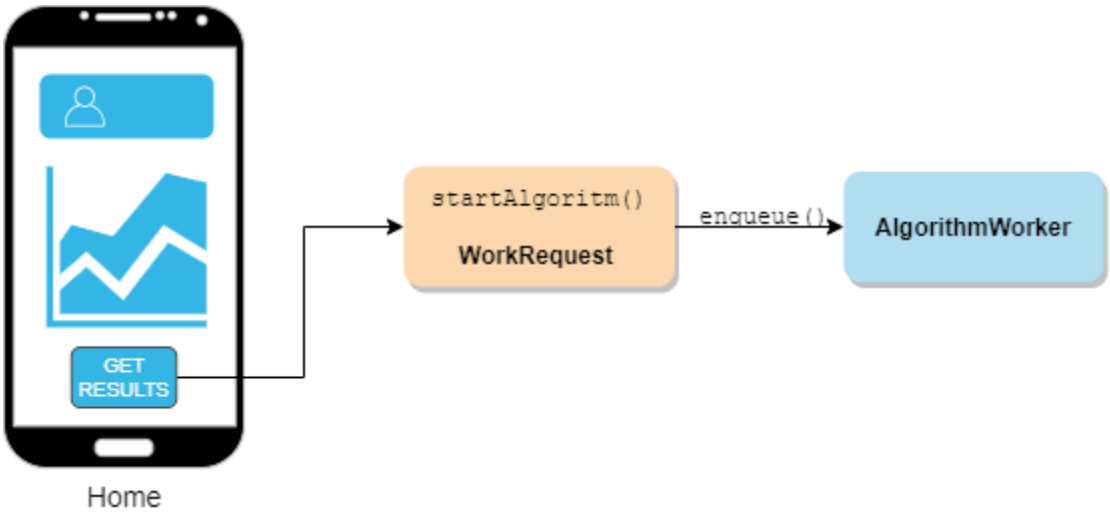

Figura 36. Ejecución del algoritmo mediante AlgorithmWorker.

Como se ha indicado anteriormente, el código original de este algoritmo se encuentra escrito en MATLAB y el objetivo era traducirlo a Java para una aplicación Android en un dispositivo móvil. Durante el desarrollo del código Java se ha tratado de traducir con una estructura lo más similar posible al código original[25]. Para la implementación del algoritmo se ha hecho uso de varias clases que se van a detallar a continuación.

Las clases que implementan el algoritmo se encuentran en el paquete `com.myheartfitness.app.srqa`.

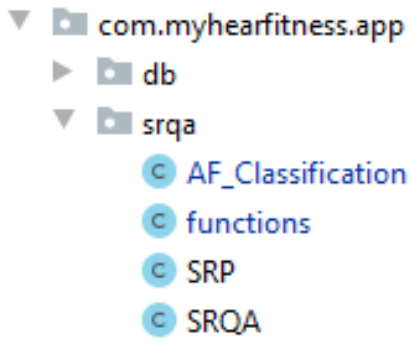

Figura 37. MyHeartFitness. Archivos que implementan el algoritmo.

- **AF_Classification**

Esta clase tiene dos métodos principales que son llamados para iniciar el algoritmo:

`readCSV()`

Cuando el usuario quiere que sus datos sean procesados por el algoritmo, automáticamente se ejecuta la función `readCSV()` que lo que hace es leer los intervalos RR de un electrocardiograma de un archivo almacenado en la aplicación en formato CSV.

`loadTimeSeries()`

Cuando se han leído los datos del archivo CSV, entonces se llama a la siguiente función con nombre `loadTimeSeries()` pasándole los datos leídos, que será la que comience el algoritmo ReAD-AF. Esta función es la que implementa los pasos indicados en el pseudocódigo del Anexo A.

[25] (MyHeartFitnesApp, 2020)

Este método divide los intervalos RR en ventanas iguales y para cada una de estas ventanas, calcula una probabilidad para determinar si el paciente se clasifica como AF o NS. Cada una de estas probabilidades se compara con un umbral y si lo supera, la ventana será clasificada como AF. Si por el contrario, la probabilidad no supera el umbral, la ventana será clasificada como NS. Esto nos dará como resultado un vector que contiene 0 y 1 siendo 1 las ventanas clasificadas como AF y 0 las ventanas clasificadas como NS.

Estos datos se mostrarán al usuario mediante un gráfico como el de la Figura 38 en la sección "Home" que tiene como eje x las ventanas y como eje y su clasificación.

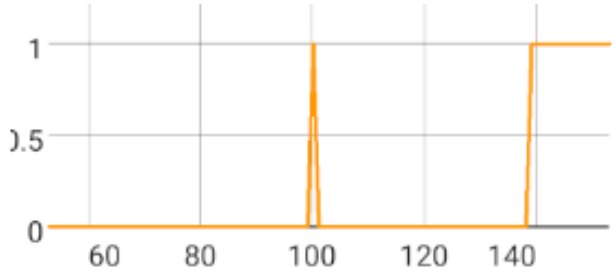

Figura 38. MyHeartFitness. Resultados obtenidos del algoritmo.

- **functions**

Dado que el algoritmo hace uso de muchos bucles y de datos en forma de array y vector, algunas de las funciones usadas en MATLAB de manera nativa no existen en Java.

Esta clase contiene una serie de funciones de las que hace uso el algoritmo y que no se encontraban implementadas en Java. Estas funciones principalmente realizan operaciones con listas y arrays.

- **SRP y SRQA**

Estas clases implementan una serie de funciones del algoritmo que se usan en la clase AF_Classification.

5.3.3. Almacenamiento

En la aplicación se almacenan datos de distintas fuentes y de distinto tipo, por lo que hacemos uso de dos tipos de almacenamiento que se exponen a continuación.

- **SharedPreferences**

SharedPreferences almacena datos en formato de clave/valor en un documento XML. Por lo que para estos datos es necesario conocer su clave. Este formato hace que sea muy sencillo tanto leer como almacenar los datos. Normalmente este almacenamiento se usa para guardar pequeñas cantidades de datos que no ocupan mucho espacio. En MyHeartFitness se ha usado para justamente eso, en SharedPreferences almacenamos datos como el nombre del usuario, apellido, fecha de nacimiento y otras variables tales como si el usuario quiere autenticación o no.

En el Anexo C se encuentra un ejemplo del formato del archivo donde se guardan los datos introducidos en SharedPreferences

- **SQLite**

Hacemos uso de SQLite cuando queremos almacenar grandes cantidades de datos o datos más complejos. Dado que los datos están compuestos y administrados por una base de datos, podemos usar lenguajes de consulta como SQL para examinar los datos y obtener un subconjunto que cumpla ciertas condiciones.

En esta aplicación hacemos uso de SQLite para almacenar datos que pesan más y son más complejos como por ejemplo los resultados obtenidos por el algoritmo.

Para la interacción con la base de datos vamos a usar la API Room ya que proporciona una capa de abstracción sobre SQLite, que permite un acceso sin problemas a la base de datos y aprovechar toda la potencia de SQLite.

En Room existen tres componentes principales:

- **Base de datos:** Contiene el titular de la base de datos y sirve como punto de acceso principal para la conexión subyacente a los datos persistentes y relacionales de la aplicación.

  La clase anotada con `@Database` debe cumplir con las siguientes condiciones:

  - Ser una clase abstracta que extiende RoomDatabase.
  - Incluir la lista de entidades asociadas con la base de datos dentro de la anotación.
  - Contener un método abstracto que tenga 0 argumentos y muestre la clase anotada con `@Dao`.
- **Entidad:** Representa una tabla dentro de la base de datos.
- **DAO:** Contiene los métodos utilizados para acceder a la base de datos.

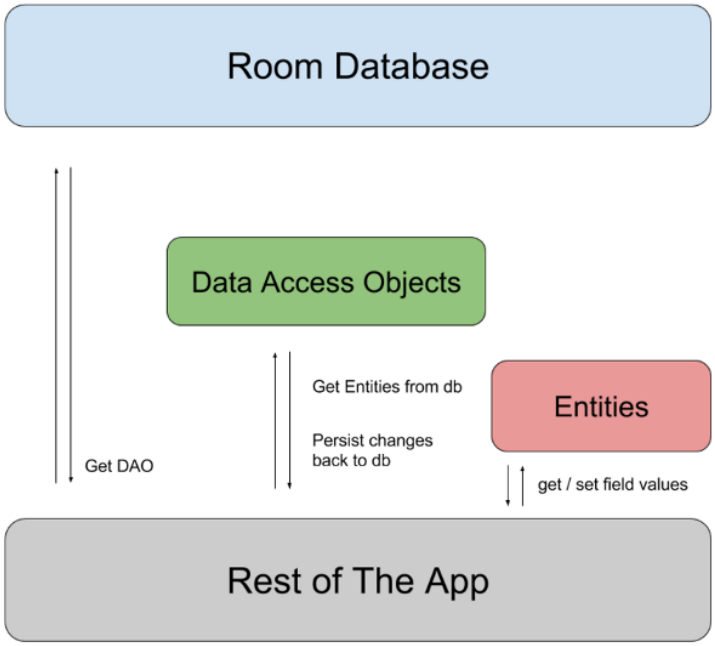

Figura 39. Arquitectura de Room.

En MyHeartFitness el uso de Room nos permite hacer peticiones a la base de datos de manera sencilla para escribir y leer los resultados que nos proporciona el algoritmo. En la Figura 40 podemos distinguir las distintas clase creadas para interactuar con la BBDD SQLite.

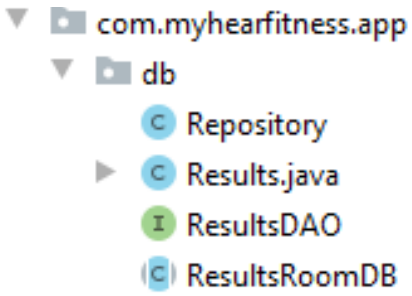

Figura 40. MyHeartFitness. Archivos para la BBDD.

Estas clases se corresponden con los tres componentes principales de Room explicados anteriormente.

- Results: representa una tabla dentro de la base de datos que se llama results_table.
- ResultsDAO: esta clase contiene los métodos usados para acceder a la base de datos.
- ResultsRoomDB: es la base de datos con nombre results_database donde vamos a almacenar los datos

A parte de estas tres clases principales, se ha añadido una clase llamada Repository que abstrae el acceso a la base de datos para el resto de la aplicación. Se puede observar la relación de todos estos elementos en la Figura 41.

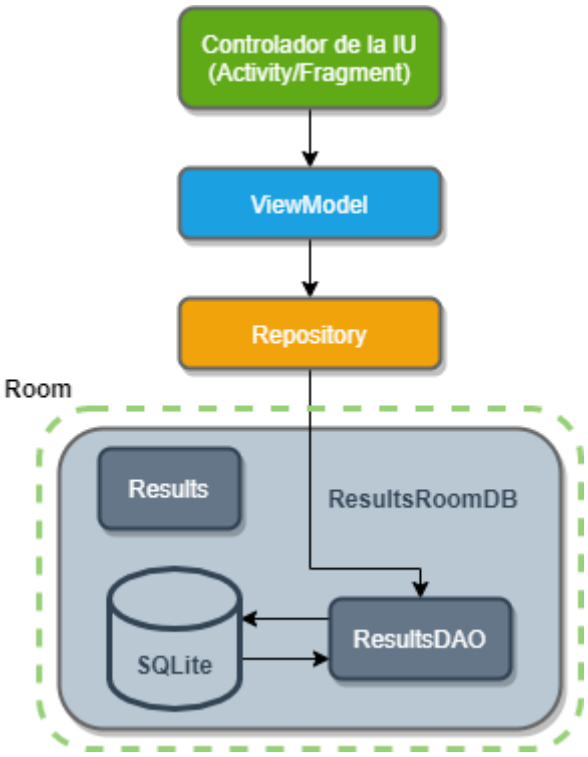

Figura 41. Arquitectura de componentes Room.

La tabla creada para almacenar los datos obtenidos por el algoritmo tiene los campos indicados en la Figura 42. Donde el resultado del algoritmo se almacena en formato JSON para después mostrarlo en la gráfica de la página principal y el resto de los datos son métricas obtenidas a cerca del algoritmo.

results_table					
_ID	results	accuracy	sensitivity	FPR	specifity

Figura 42. MyHeartFitness. Tabla de resultados.

5.3.4. Notificaciones

Cuando el algoritmo haya terminado de ejecutarse y los resultados estén disponibles, el usuario recibirá una notificación para informarle de que puede visualizar los resultados.

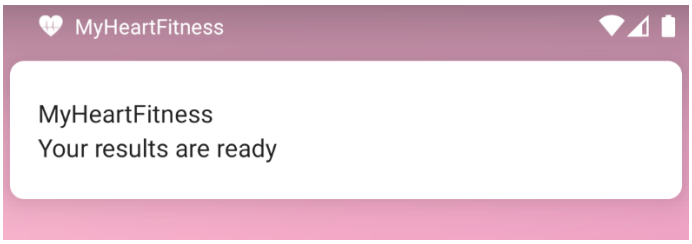

Figura 43. MyHeartFitness. Notificaciones.

Si el usuario hace clic en esta notificación se le llevará a la página principal de la aplicación para mostrarle los resultados.

5.4. Simulación

Para hacer pruebas con la aplicación se ha optado por usar un emulador dado que nos permite probar la aplicación fácilmente en un dispositivo Android de las características que elijamos. El Emulador de Android simula dispositivos Android en nuestro equipo para que podamos probar la aplicación que está siendo desarrollada en diferentes dispositivos y niveles de API de Android sin necesidad de contar con los dispositivos físicos. Además, nos proporciona casi todas las funciones de un dispositivo Android real como probar sensores del dispositivo, acceder a Google Play Store, simular llamadas y mucho más.

Para poder hacer uso del emulador en Android Studio nos dirigimos a la barra de herramientas → AVD Manager y vemos los dispositivos disponibles como en la Figura 44. Por defecto se ha instalado un dispositivo con Android 11.0 (API 30).

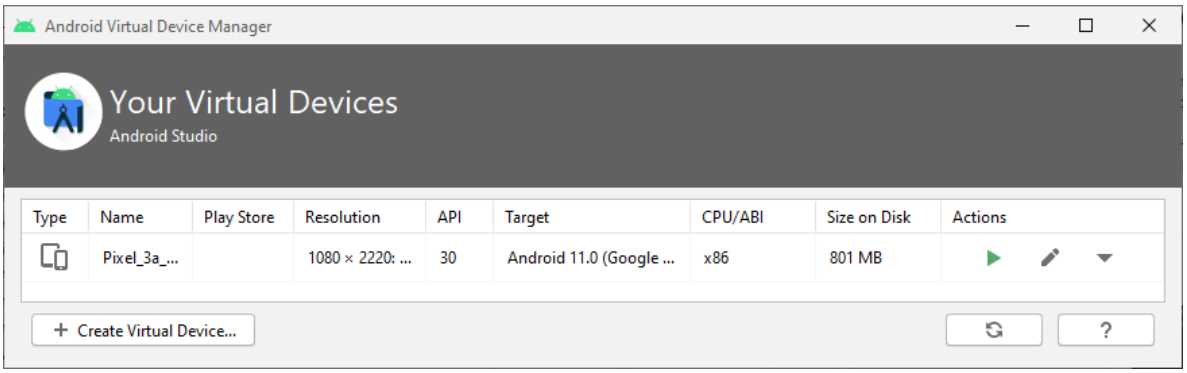

Figura 44. AVD Manager.

Una vez elegido el dispositivo que queremos emular, simplemente tenemos que ejecutar la aplicación que estamos desarrollando en la barra de navegación superior (Figura 45) y se iniciará un dispositivo virtual con nuestra aplicación instalada. El resultado de ejecutar el emulador se muestra en la Figura 46.

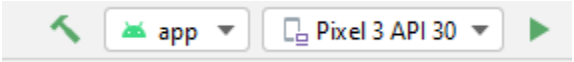

Figura 45. Botón para ejecutar un emulador.

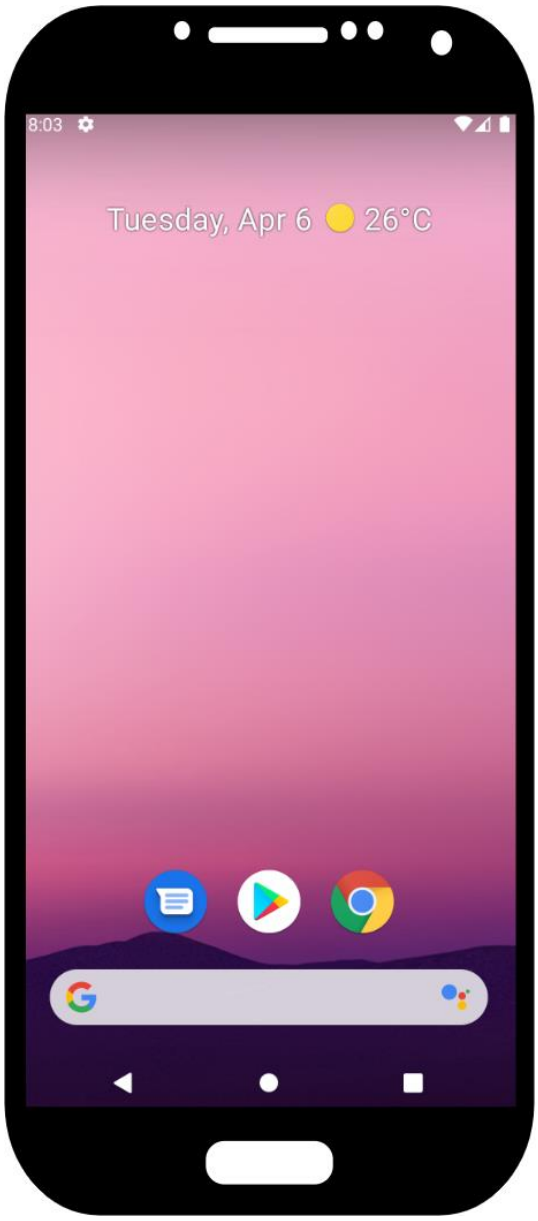

Figura 46. Emulador Android.

De esta manera, se ha podido probar MyHeartFitness en varios dispositivos con distinto nivel de API para comprobar su correcto funcionamiento. Además, también se ha podido experimentar la autenticación mediante huella dactilar ya que gracias a los controles extendidos del emulador podemos configurar huellas digitales para el dispositivo en cuestión.

5.5. Librerías

Para el desarrollo de la aplicación se ha hecho uso de distintas librerías que han permitido mejorar y complementar aspectos que ya nos proporciona el SDK de Android. Las librerías usadas son las siguientes:

- **Preference**

Se ha hecho uso de la librería `androidx.preference` para la creación de la vista Settings en nuestra aplicación. Esta librería nos ayuda a crear vistas de configuración interactivas sin necesidad de interactuar con el almacenamiento del dispositivo ni de administrar la interfaz.

- **Room**

La librería `androidx.room` nos proporciona una capa de abstracción para SQLite que nos permite acceder a una base de datos sin problemas aprovechando toda su potencia. Esta librería se ha usado para almacenar los resultados del algoritmo en una BBDD SQLite.

- **Biometric**

La biblioteca `androidx.biometric` se ha usado para autenticar al usuario en la aplicación MyHeartFitness con credenciales biométricas o del dispositivo.

- **Work**

La librería `androidx.work` se ha usado para la programación de tareas asíncronas diferibles que deben ejecutarse de manera fiable. Esta librería nos permite crear una tarea y pasársela a un WorkManager para que se ejecute cuando se cumplan ciertas restricciones. Dado que la ejecución del algoritmo debe de realizarse de manera fiable, se ha hecho uso de esta librería para dicho objetivo.

- **OpenCSV**

Mediante la librería `com.opencsv:opencsv` se ha podido realizar la acción de leer de archivos CSV que es el formato con el que se almacenan los datos de entrada del algoritmo.

- **GraphView**

Esta librería ha permitido crear diagramas flexibles para la representación de los resultados del algoritmo. Es una librería sencilla de usar ya que en su página podemos encontrar ejemplos para la creación de gráficos de distinto tipo.

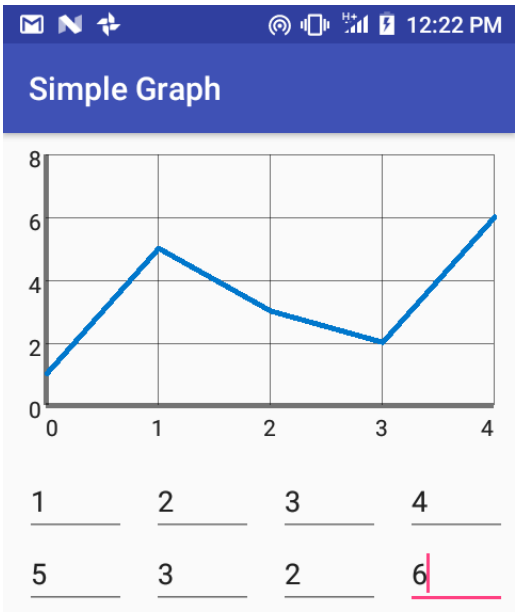

Figura 47. Gráfico creado con GraphView

- **Smile**

Smile (Statistical Machine Intelligence and Learning Engine) es una librería de Machine Learning para Java. Esta librería cubre todos los aspectos del aprendizaje automático, incluida la clasificación, regresión, clustering, selección de característica, búsqueda efectiva del vecino más cercano, etc.

En MyHeartFitness esta librería se ha usado para calcular el modelo Logístico y obtener las probabilidades de AF o NS en los intervalos RR.

6. Resultados

Para la obtención de resultados mediante el algoritmo ReAD-AF es necesario introducir los intervalos RR de un electrocardiograma como datos de entrada. Para ello, se ha hecho uso de la base de datos PhysioNet MIT-BIH Atrial Fibrillation Database[26] [27] al igual que en el artículo a cerca del algoritmo que ejecuta la aplicación[28]. Esta base de datos contiene electrocardiogramas de 25 pacientes con fibrilación atrial que nos podemos descargar y usar como entrada del algoritmo.

PhysioBank nos proporciona una herramienta llamada PhysioBank ATM[29] (Figura 48) para consultar las distintas bases de datos disponibles desde un navegador. De esta manera, se pueden consultar las bases de datos disponibles en PhysioBank como la mencionada anteriormente y observar los distintos electrocardiogramas disponibles (Figura 49). Además, también es posible descargar las señales de estos electrocardiogramas en distintos formatos (como *.csv* y *.txt*).

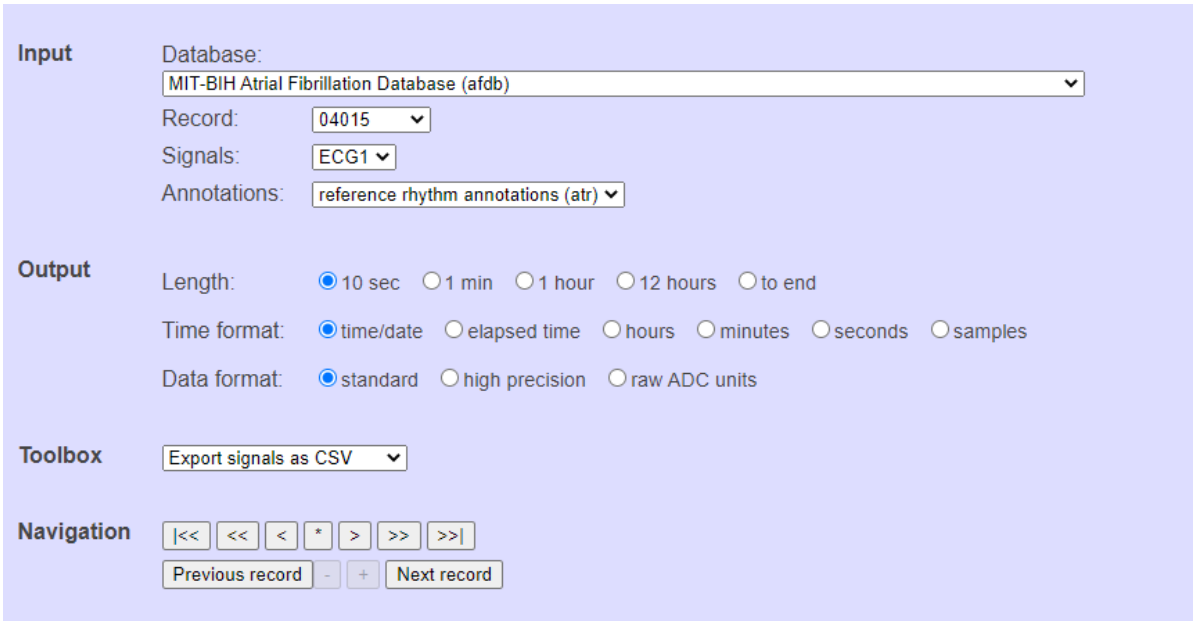

Figura 48. Herramienta PhysioBank ATM.

[26] (Goldberger, y otros, 2000)
[27] (MIT-BIH Atrial Fibrillation Database, 2000)
[28] (Pérez-Valero, y otros, 2019)
[29] (PhysioBank ATM, s.f.)

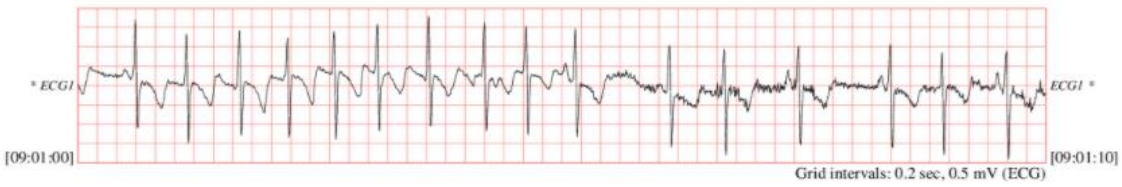

Figura 49. Señal de Electrocardiograma de MIT-BIH Atrial Fibrillation Database.

Por lo tanto, para obtener los intervalos RR usados como entrada del algoritmo ReAD-AF, se han usado señales de electrocardiograma de la base de datos MIT-BIH Atrial Fibrillation Database[30].

6.1. Comparación de resultados

Para comprobar la similitud de los datos obtenidos en el algoritmo escrito en MATLAB y el algoritmo traducido a Java para la aplicación, se ha hecho uso del mismo conjunto de datos de intervalos RR y se ha observado el resultado para ambos. En este caso, el paciente tiene varias ventanas de fibrilación atrial por lo que tiene un alto porcentaje de clasificación AF.

Como se representa en la Figura 50, el porcentaje de ventanas clasificadas como NS y el de ventanas clasificadas como AF en MATLAB (izquierda) y Java (derecha) es muy cercano para ambos algoritmos.

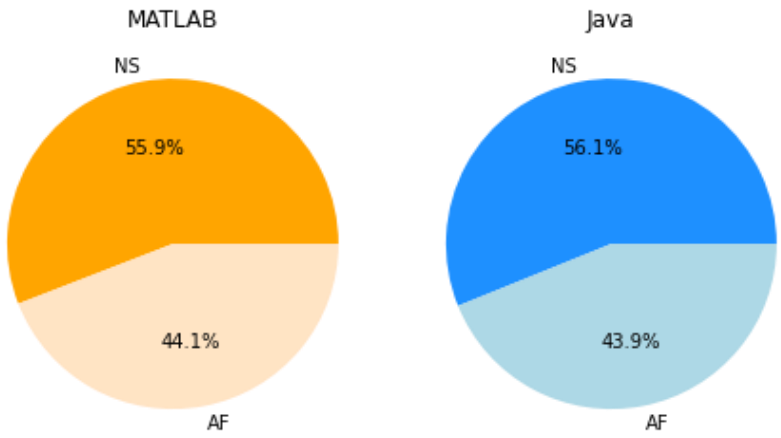

Figura 50. Clasificación en MATLAB y Java.

[30] (Goldberger, y otros, 2000)

6.2. Comparación de métricas

Como punto final de este proyecto, es importante hacer una comparativa de métricas entre el algoritmo escrito originalmente en MATLAB y el que ejecuta la aplicación Android para ver si existen diferencias entre ambos.

Para mostrar el poder de clasificación del algoritmo calculamos una serie de métricas que nos permiten evaluarlo:

- **Especificidad**: es la tasa de verdaderos negativos. Con ella medimos la proporción de negativos identificados correctamente.
- **Sensibilidad**: es la tasa de verdaderos positivos. Mide la proporción de positivos identificados correctamente, es decir, la proporción de ventanas de fibrilación atrial que son clasificadas correctamente.
- **Precisión**: determina la precisión del modelo.

Consultando los resultados obtenidos en el modelo del artículo donde se desarrolla el funcionamiento del algoritmo[31], vemos que las tasas del modelo de MATLAB son las mostradas en la Tabla 2. Donde w es el tamaño de ventana, τ es el umbral, Se la sensibilidad, Sp la especificidad y ACC la precisión.

	w = 30	w = 60	w = 120	w = 200
τ	0.414	0.448	0.513	0.510
Se	0.961	0.970	0.976	0.979
Sp	0.948	0.960	0.971	0.971
ACC	0.954	0.964	0.973	0.977

Tabla 2. Métricas de ReAD-AF.

Para poder hacer una comparativa entre las métricas en ambos algoritmos, se han calculado también en el algoritmo de la aplicación Android para un tamaño de ventana 30 y al igual que en el punto anterior, con los mismos intervalos RR de entrada al algoritmo. Los resultados se exponen en la Figura 51 y en ella se puede apreciar y la comparación de resultados.

[31] (Pérez-Valero, Garcia-Sanchez, Ruiz Marín, & Garcia-Haro, 2020)

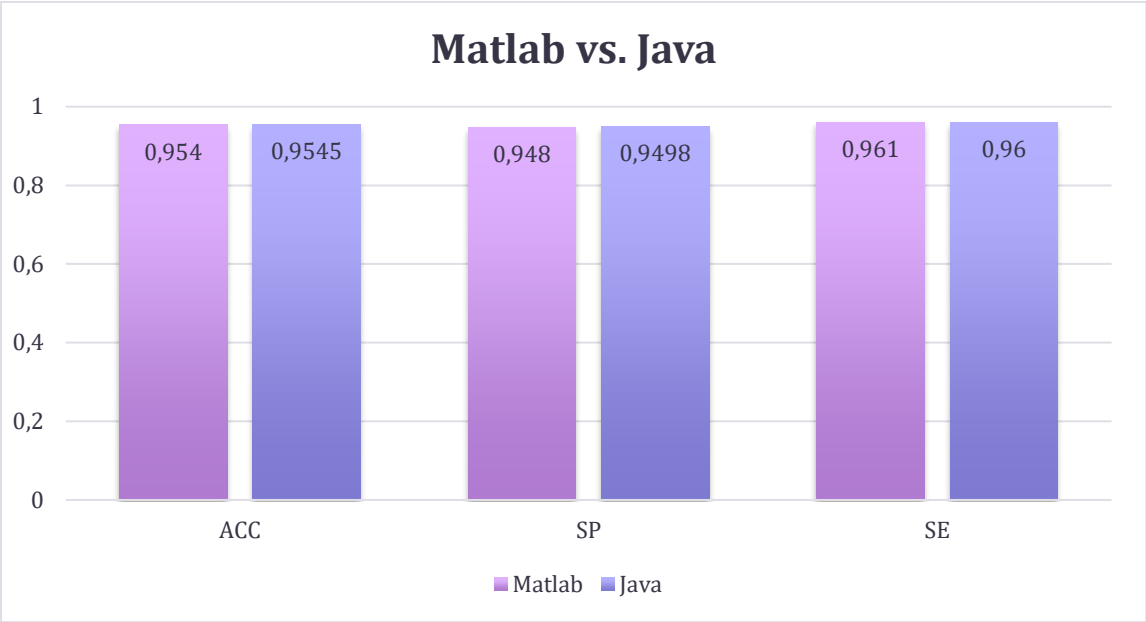

Figura 51. Comparación de métricas de ReAD-AF en MATLAB y Java

7. Conclusiones y líneas futuras

7.1. Conclusiones

Tras haber finalizado el proyecto, se puede afirmar que se han cumplido la gran mayoría de objetivos marcados al inicio.

Al comienzo de este proyecto, partía de una capacidad limitada para realizarlo, dado que no había trabajado con muchas de las tecnologías usadas en este trabajo. Durante el desarrollo de este proyecto se han adquirido una serie de conocimientos que han hecho posible el desarrollo de MyHeartFitness.

Finalmente, la aplicación ofrece las siguientes características:

- Interfaz sencilla y amigable para el usuario.
- Ejecución del algoritmo ReAD-AF a partir de los intervalos RR obtenidos de un electrocardiograma.
- Almacenamiento de los resultados de salida del algoritmo en una base de datos SQLite.
- Representación de los resultados obtenidos por el algoritmo de manera que puedan ser interpretados por el usuario.
- Almacenamiento de datos del usuario como su nombre, apellido y fecha de nacimiento, si el usuario lo desea.

7.2. Líneas futuras y mejoras

Aunque la gran mayoría de objetivos iniciales han sido cumplidos, hay una implementación que no se ha llevado a cabo todavía. Para el funcionamiento pleno de la aplicación, todavía es necesario implementar la extracción de datos de un electrocardiograma en formato PDF o JPEG.

Además, esto es una primera versión de MyHeartFitness por lo que siempre se podrán introducir nuevas funcionalidades y mejoras a esta. Alguna de las posibles mejoras que podría incluir MyHeartFitness en líneas futuras son:

- Opción de introducir más datos personales del usuario, como por ejemplo su género u otros parámetros biológicos que puedan servir para estudios futuros y mejora del algoritmo.
- Permitir la descarga de los datos obtenidos por el algoritmo en formato PDF.
- Mejorar y optimizar el código.

8. Anexos

Anexo A. Pseudocódigo del Algoritmo ReAD-AF.

```
Input: RR interval data set of a patient
raw_data ← load(RR)
data ← splitRR(raw_data, w)            //Split RR intervals in windows of size w
meanRR ← mean(data)                    //Mean
medianRR ← median(data)                //Median
VRR ← std(data)/mean(data)             //Pearson coefficient of variation
VmeRR ← sum(abs(data-medianRR))/medianRR      //Mean absolute dispersion
SRR: symbolic recurrence rate
SRP: symbolic recurrence plots
Vinc(1,3): Shannon entropy prob. distrib. vertical line lengths; SRP for
increasing symbol {i}
Vinc(1,4): Mean length of vertical lines; SRP for increasing symbol {i}
Vdec(1,3): Shannon entropy prob. distrib. vertical line lengths; SRP for
decreasing symbol {i}
Vdec(1,4): Mean length of vertical lines; SRP for decreasing symbol {i}
DETt: The percentage of recurrence points which form diagonal lines; SRP for all
symbols
ENTRt: Shannon entropy of the probability distribution of the diagonal; SRP for
all symbols
Matrix ← func_SRP(data, m)             //Matrix SRR for all symbols
[Vinc(1, 3), Vinc(1, 4)] ← recu_SRQA(Matrix {i})
[Vdec(1, 3), Vdec(1, 4)] ← recu_SRQA(Matrix {i})
[DETt, ENTRt] ← recu_SRQA(Matrix)
measure: Storage of all metrics in a matrix
measure ←[meanRR,medianRR,VRR,VmeRR,DETt,
ENTRt,Vinc(1,3),Vdec(1,3),Vinc(1,4),Vdec(1,4)]
pos_af: The boolean vector identifying the AF windows
mod ← fitglm(measure,'Distribution','binomial',pos_af) //Compute the logistic model
prob ← mod.Fitted.Probability          //Compute the estimated probability of AF
Tlist ←[0.001:0.001:1]
```

```
for j=1 to j=lenght(Tlist) do
       ths ← Tlist(j) . Compute the threshold
       TP ← length(find(prob(find(pos_af>0))>ths))                    //True Positive
       TN ← length(find(prob(find(pos_af==0))0))ths))                 //True Negative
       FN ← length(find(prob(find(pos_af>0))ths))                     //False Positive
       FP ← length(find(prob(find(pos_af==0))>ths))                   //False Positive
       TPR ← TP/(TP+FN)                                          //Sensitivity or TPR
       FPR ← FP/(FP+TN)                                          //False Positive Rate
       SPC ← 1-FPR                                               //Specificity
       ACC ← (TP+TN)/(TP + FN + FP + TN)                         //Accuracy
end for
τ: Optimal threshold parameter
τ ← min{FPR2 τ + (1 – Seτ) 2}                  //optimal threshold for AF
classification
if prob ≥ τ do
       patient classified as AF
else
       patient classified as NS
end if
Output: Classified patient
```

Anexo B. Archivo AndroidManifests.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.myhearfitness.app">

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.USE_BIOMETRIC"
        android:requiredFeature="false"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MyHeartFitness"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".AuthenticateActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:label="MyHeartFitness">
        </activity>
    </application>

</manifest>
```

Anexo C. SharedPreferences.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="birthday">4/12/1998</string>
    <boolean name="auth" value="true" />
    <string name="name">Eva</string>
    <string name="pic">/data/user/0/com.myhearfitness.app/app_files</string>
    <string name="lastname">Sánchez-Guerrero Ayala</string>
</map>
```

Anexo D. Pseudocódigo de AlgorithmWorker.

```
//algorithm Worker class
class AlgorithmWorker:
	function doWork:
		Input: nothing
		try:
			AF_Classification.readCSV()	//try to start algorithm
			return success			// algorithm started
		catch:
			return failure			//algorithm could not start
		Output: nothing
	end function
end class
```

Anexo E. Pseudocódigo de startAlgorithm.

```
//function to start algorithm
function startAlgorithm:

	Input: nothing

	constraints ← battery not low, storage not low

	manager ← create WorkManager

	request ← execute AlgorithmWorker one time	//request algorithm execution

	manager.enqueue(request)			//enqueue request for execution

	manager.getInfo(request)			//get info of the request


	if (sucess):

		notify()	//notify the user when results are ready

	end if

	Output: nothing

end function
```

9. Bibliografía

Android. (2021). Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Android

Anexo:Historial de versiones de Android. (s.f.). Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android

Apple Heart Study. (12 de 03 de 2021). Obtenido de Apple Newsroom: https://www.apple.com/newsroom/2017/11/apple-heart-study-launches-to-identify-irregular-heart-rhythms/

Bollepalli, S., Challa, S., Jana, S., & Patidar, S. (2017). Atrial Fibrillation Detection Using Convolutional Neural Networks. *2017 Computing in Cardiology Conference (CinC)*. doi:10.22489/cinc.2017.163-226

Censi, F., Corazza, I., Reggiani, E., Calcagnini, G., Mattei, E., Triventi, M., & Boriani, G. (2016). P-wave Variability and Atrial Fibrillation. *Scientific Reports*. doi:10.1038/srep26799

Crecimiento de Sensores Portátiles en la Atención Médica. (11 de 12 de 2020). Obtenido de WhatNext: https://www.whatnextglobal.com/post/wearable-sensors-in-healthcare

Cuota de Mercado en España de Sistemas Operativos Móviles. (2021). Obtenido de StatCounter Global Stats: https://gs.statcounter.com/os-market-share/mobile/spain

Cuota de Mercado Global de Sistemas Operativos Móviles. (2021). Obtenido de StatCounter Global Stats: https://gs.statcounter.com/os-market-share/mobile/worldwide

Cuota de Mercado Global de Versiones Android. (2021). Obtenido de StatCounter Global Stats: https://gs.statcounter.com/android-version-market-share/mobile/worldwide/#yearly-2018-2021

Enfermedades Cardiovasculares (ECVs). (2017). Obtenido de Wold Health Organization: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

evasga98/MyHeartFitness. (2021). Obtenido de GitHub: https://github.com/evasga98/MyHeartFitness

Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P., Mark, R., . . . Stanley, H. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation, 101*(23), 215-220. doi:10.1161/01.cir.101.23.e215

Guía para Desarrolladores Android. (s.f.). Obtenido de Android Developers: https://developer.android.com/guide

Librería Biométrica. (s.f.). Obtenido de Android Developers: https://developer.android.com/jetpack/androidx/releases/biometric

MIT-BIH Atrial Fibrillation Database. (04 de 11 de 2000). Obtenido de PhysioNet: https://physionet.org/content/afdb/1.0.0/

MyHeartFitnesApp. (2020). Obtenido de https://github.com/v4lerO/MyHeartFitnessApp

Pérez-Valero, J., Caballero Pintado, M., Melgarejo, F., García-Sánchez, A., Garcia-Haro, J., García Córdoba, F., . . . Ruiz Marín, M. (2019). Symbolic Recurrence Analysis of RR Interval to Detect Atrial Fibrillation. *Journal of Clinical Medicine, 8*. doi:10.3390/jcm8111840

Pérez-Valero, J., Garcia-Sanchez, A., Ruiz Marín, M., & Garcia-Haro, J. (2020). A Prototype Framework Design for Assisting the Detection of Atrial Fibrillation Using a Generic Low-Cost Biomedical Sensor. *Sensors, 20*. doi:10.3390/s20030896

PhysioBank ATM. (s.f.). Obtenido de https://archive.physionet.org/cgi-bin/atm/ATM