WEARABLE DE MEDIDA DE RENDIMIENTO EN RUNNING

Autor: María del Camino Ayuso González-Montagut.

Tutor: Marina Alonso Diaz.

Cotutor: Salvador Coll Arnau.

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación.

Curso 2018-19.

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

Resumen

El objetivo final de este trabajo de fin de grado es el análisis de los datos, obtenidos a través sensores, con el fin de mejorar el proceso de recuperación de un paciente lesionado o que ha sufrido una operación, prevenir futuras lesiones musculares y osteoarticulares y, por tanto, optimizar y garantizar el ejercicio del *running* de forma segura.

Los datos se obtienen a partir de un dispositivo con diversos sensores e implementaremos una aplicación que sea capaz de recibir y manejar los datos para proporcionar información que sea de utilidad para el objetivo ya mencionado. El dispositivo se comunicará con la aplicación a para la transmisión de datos mediante tecnología Bluetooth.

Para todo ello, se hace un estudio de los distintos dispositivos en el mercado que poseen sensores útiles y que son capaces transmitir dicha información a través de bluetooth. También se hará un estudio de los distintos datos que nos pueden ser útiles y como gestionarlos para llegar a conclusiones que puedan mejorar el rendimiento de los deportistas. Por último, se implementará el software, la aplicación, se decidirá que sistema operativo es mejor y se pondrá en práctica para demostrar que la aplicación es eficiente.

Resum

L'objectiu final d'aquest treball és el análisis del dades, obtingudes per sensors amb la finalitat de millorar el próces de recuperació d'un paciente lesionat o que n'hi ha patit una operació, la prevenció de futures lesions musculars i osteoarculars i, per tant, optimizar y garantir el ejercici del atletisme d'una forma segura.

Les dades s'obtinguen a partir d'un dispositiu amb diverses sensors e implementarem una aplicación capaç de rebre y manejar les dades per a proporcionar información que siga de utilitat per a l'objetiu que hem mencionat. El dispositiu es comunicarà amb la aplicació per a la transmisió de dades mediante Bluetooth.

Per a tot això, es fa un estudi del diverses dispositius del mercat que poseeixen sensor útils i que són capaços de transmetre l'informació a través de Bluetooth. També es fará un estudi dels dades que ens poden ser útils i de com gestionarlos per a arribar a conclusions que puguen millorar el rendiment dels esportistes. Per últim, s'implementarà el software, la aplicació, on es decidirà quin sistema operatiu es millor i es posarà en práctica per a demostrar que la aplicación es eficiente.

Abstract

The main objective of this Project is the analysis of the data that are obtained by sensors with the goal of impoving the revory of the injured pacientes or those who has suffered an operation, or to prevent future muscle damages, in conclusión, to improve and guarantee the correct and safe practise of the running.

Data are obtained by a device which have some sensor and we will implement an application capable of receive and manipulate this data to provide useful information for the goal which I have just mentioned. The device will communicate with the application by Bluetooth technology.

For all of that, I have done an study about the differents devices that are in the market which have useful sensors os the Project and that can transmit the data by Bluetooth. I have also studied the differents data and how use them to obtaine conclusions which can help to the runners. Finally, I will implement the software, the application, choosing the best operating system and I will demostrate the functionality of the project.

Índice general

Índice de figuras

Índice de tablas

1. Introducción.

Este proyecto ha sido elegido y llevado a cabo por mi como trabajo de fin de grado debido a las razones y siguiendo la forma de proceder que a continuación voy a explicar.

1.1 Motivación

En primer lugar, este trabajo ha sido interesante para mi debido a mi interés en el *running* dado que se trata de uno de los hobbies que tanto como personas cercanas a mi practican cada día.

En segundo lugar, se trata de una práctica muy extendida en todo el mundo, y aquí en Valencia cada vez debido a la importancia que se le está dando a carreras cada vez más multitudinarias. Por ello, dado el incremento en la práctica del running, este se debe realizar de la mejor forma posible ya que, el realizarlo mal puede traer consecuencias muy perjudiciales.

Para ello, con este trabajo de fin de grado, se me daba la oportunidad de desarrollar y llevar a cabo alguna metodología que permitiese a todos realizar esta práctica de forma un poco más segura y beneficiosa.

Además, el desarrollo del proyecto también me iba a permitir ampliar mis conocimientos a cerca de distintos temas: tanto de los distintos factores y la importancia de realizar todo ejercicio físico de forma segura como del desarrollo de aplicaciones y de su programación, unos conocimientos muy útiles en un mundo en el que todo se controla mediante aplicaciones móviles.

Por último y no menos importante, el gran crecimiento del mercado de dispositivos '*wearables*: Los estudios realizados por IDC (), principal proveedor mundial del mercado de servicios de consultoría y eventos para los mercados tecnológicos, afirman a cerca del mercado de los '*wearables'* que en el mercado europeo liderará el crecimiento en estas tecnologías durante el próximo lustro y, qué en el mercado español, este crece a pasos agigantados.

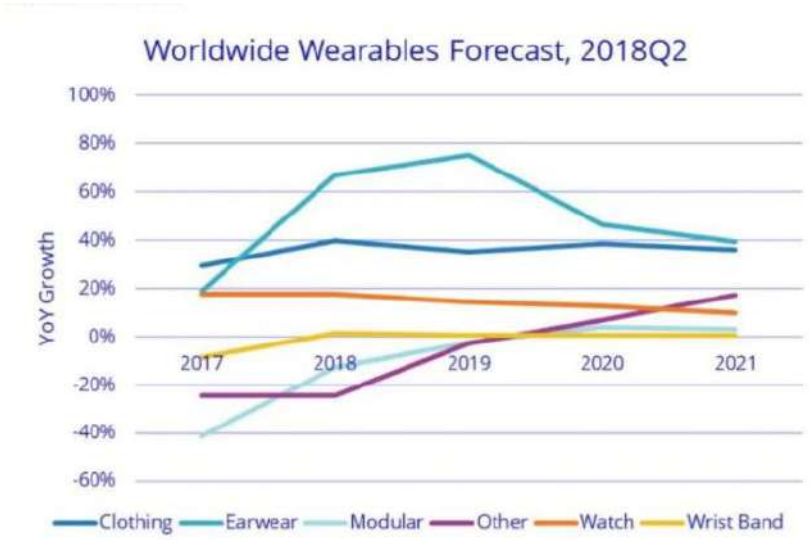

Figura 2.1. Evolución de los distintos dispositivos 'wearables' en el mercado.

Dado que nuestra aplicación va a ser diseñada con la idea de que sea utilizada por fisioterapeutas, me lleva a destacar el gran auge del uso de 'wearables' en aplicaciones médicas, donde cada vez más dispositivos 'wearables' que se usan cada día nos dan datos muy valiosos a

cerca de la salud que mejoran las condiciones de vida y contribuyen a prevenir numerosas enfermades y lesiones fundamentalmente a aquellos que prácticas deportes.

1.2 Objetivos.

Este trabajo tiene varios objetivos. En primer lugar, el estudio de los distintos dispositivos del mercado, su comparación y elección del más óptimo para el proyecto, para ello se analizan tanto los sensores, como las distintas facilidades que nos aportan sus fabricantes. Tras ello, el siguiente objetivo, es el estudio de lo sistemas operativos para desarrollar la aplicación má eficiente, que nos permita llegar a más usuarios y que mayores libertades de código y de funcionalidades deje a los programadores.

El objetivo fundamental sería el análisis y estudio de como los datos que obtenemos pueden ser útiles para distintas aplicaciones para mejorar la actividad física y reducir el riesgo de lesiones e, incluso, ayudar a mejorar y seguir el desarrollo de un paciente tratado por un fisioterapeuta.

1.3 Metodología empleada.

Para realizar dicho proyecto en un espacio de tiempo considerablemente pequeño, se ha requerido tener muy claro los objetivos y establecer un plan que contemplara los estudios de todos los sectores convenientes para el desarrollo del mismo.

1.3.1. Gestión del proyecto.

Siendo muy interesante, contaba con muchos campos, siendo el que más formación e investigación requería el de la implementación de la aplicación; gracias a la ayuda de mis tutores, antes de la implementación de la aplicación se realizaron todos los estudios previos: a cerca tanto de los diferentes dispositivos o de la evolución del mercado de los mismo como de todas las bases matemáticas que requería el análisis de los datos, puesto que al centrarnos en la medida de la aceleración se debía restar la aceleración de la gravedad que aparecía, o la interpretación de los datos pasándolos de cuaterniones a vectores Euler.

Es decir, para llevar el proyecto a cabo había que empezar por una fase de aprendizaje e investigación sobre toda la base teórica. También sobre las distintas medidas que podrían ser interesantes para los fisioterapeutas, para ello entreviste a varios a cerca de que medidas podrían serles útiles para tratar a pacientes que son aficionados a la práctica del *runnining* para poder realizar una aplicación para el proyecto lo más útil posible. Así pues, antes de implentar la aplicación y durante el desarrollo de la misma fue necesaria mucha formación a cerca de la programación en el sistema operativo elegido.

1.3.2. Distribución de tareas.

Siendo conscientes de todo el trabajo que conllevaba dicho proyecto, realicé una distribución de tareas según la cual voy a ordenar capítulos del trabajo:

Así pues, en primer lugar, investigue a cerca de la importancia de los wearable en los distintos campos del deporte y sanitario, así como de sus principales fabricantes; posteriormente, realice un estudio a cerca de los diferentes dispositivos con sensores a partir de los cuales iba a poder diseñar un wearable sencillo.

Una vez elegido el wearable y conociendo todos sus sensores y características, me dispuse al diseño de la aplicación: en primer lugar, me informé a cerca de los distintos sistemas operativos (iOS y Android los principales), una vez escogido el sistema operativo en el cual iba a desarrollar la aplicación, me dispuse a realizar un esquema de la lógica y estructura que tendría la aplicación. Teniendo claro que los puntos fuertes de la misma serían las bases de datos y la conexión bluetooth, me dispuse a aprender sobre la programación en estos dos campos, para poder comprenderla antes de implementarla.

Posteriormente, comencé el desarrollo de la aplicación. Además, hablé con diferentes fisios, deportistas y leí artículos a cerca de la practica del atletismo y sus peligros, y de que aplicación les podría ser útil y diferente al resto de los que existen. Así pues, fui desarrollando la aplicación y enfrentándome a distintos retos, explicados extensamente en el trabajo.

Una vez realizada la aplicación, tome varias medidas con ayuda de personas que sabían la forma correcta y segura de realizar ejercicio, y establecí la gráfica estándar. De esta forma, cuando más similar sea tu gráfica a la estándar, mejor estarás realizando la actividad. En caso de que varíe mucho, deberías consultar a un fisioterapeuta a cerca de como corres para evitar lesiones.

2. Bases teóricas.

En este capítulo voy a resumir todo el conocimiento que es necesario para llevar a cabo el proyecto; comenzaré con el estudio de las diferentes tecnologías disponibles en el mercado, veremos sus pros y contrar para poder, en el siguiente capítulo tomar la mejor decisión.

Continuaré con el estudio de los sistemas operativos, muy importante su elección para el diseño y desarrollo de la aplicación; y como más técnico y en el siguiente capítulo, veremos teorías matemáticas que son muy necesarias para poder llevar a cabo el análisis de los datos en la aplicación.

En este nos centraremos en las bases teóricas que nos van a ayudar al diseño del proyecto, las bases teóricas posteriores son fruto de la elección tomada a partir del siguiente estudio-

2.1. Estudio de los distintos dispositivos existentes en el mercado.

Vamos a analizar y comparar cuatro dispositivos diferentes que me han parecido los más útiles para este proyecto y competitivos entre ellos: en primer lugar, el SensorTag CC2650, SensorTag 3200, Mikroe 2026 y STEVAL WESU1.

2.1.1. *SensorTag CC2650*

Este dispositivo se conecta a otros mediante Bluetooth low energy y obtiene los datos de sus 9 sensores de forma online en tan solo 3 minutos. El SensorTag permite su uso en una aplicación implementada tanto en Android como en iOS sin tener mucha experiencia en programación para empezar.

Dicho SensorTag está basado y le da nombre el MCU inalámbrico CC2650, el cual ofree un consumo de energía un 75% inferior que el de otros productos que también son Bluetooth *low energy* pero anteriores; esta característica permite que el dispositivo SensorTag sea alimentado por batería.

El contenido basado en datos recibidos del SensorTag se puede personalizar, lanzar desde aplicaciones y conocer su ubicación física desde tu teléfono gracias a a que el Bluetooth *low energy* incluye tecnología iBeacon.

A continuación, voy a describir el SensorTag CC3200 y posteriormente veremos sus características comunes.

2.1.2. SensorTag CC3200

El Wi-Fi SensorTag se basa en el low power SimpleLink CC3200 wireless MCU, se trata del primer MCU industrial con certificado Wi-Fi con conectividad Wi-Fi, lo cual permite al dispositivo cargare con baterías AAA. Este dispositivo SensorTag posee 9 sensores low power MEMS .

Se conecta a la nube a través del Wi-Fi y obtiene los datos online de los sensores en tan solo unos minutos. El SensorTag configura la conexión Wi-Fi con una aplicación implementada a través de iOS o Android sin necesidad de experiencia en programación para empezar.

2.1.3. Características SensorTag

Algunas características comunes a los dos dispositivos que acabamos de ver, debido a que las piezas que los componen son iguales al ser fabricadas ambas por Texas Instruments. Las características las veremos a través de imágenes sacadas de la página de Texas Instrumen en la que se puede visualizar físicamente el dispositivo en cuestión.

Además, cabe destacar, que estos dispositivos permiten modificar su firmware a partir del DevPack que ofrece su fabricante que se conecta al ordenador mediane el DevPack Expansion Connector.

Figura 2.1: Vista frontal del dispositivo SensorTag de Texas Instruments.

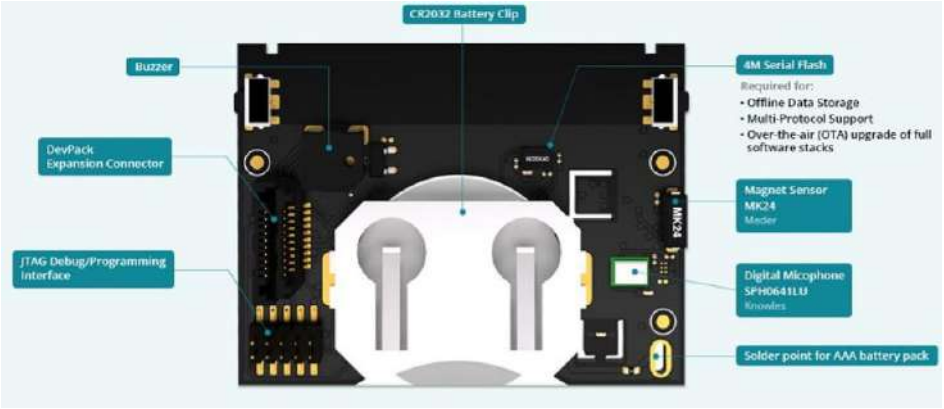

Figura 2.2. Vista interna frontal del dispositivo SensorTag de Texas Instrument.

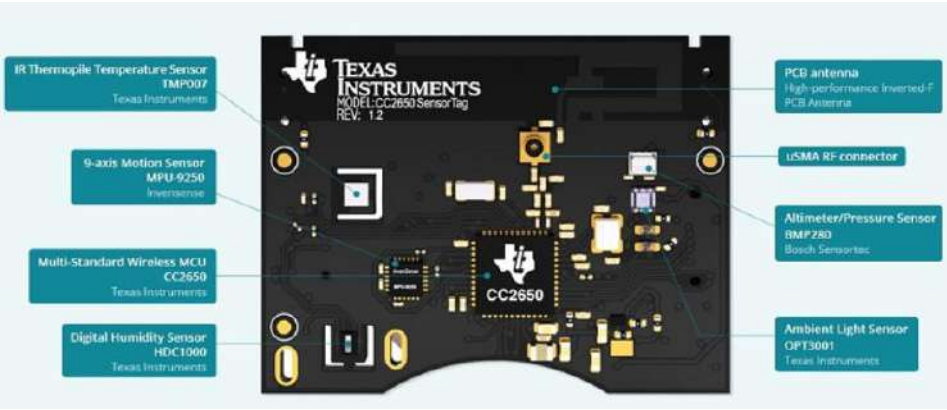

Figura 2.3. Vista interna trasera del dispositivo SensorTag de Texas Instrument.

2.1.4. Mikroe 2026.

Se trata de un dispositivo eficiente de bajo consumo de energía. Utiliza un microcontador Kinetis K64 de 32 bit ARM Cortex M4 MCU y posee sensores avanzados. Está equipado con Bluetooth low energy compatible con aplicaciones tanto de Android como de iOS y se puede mejorar con numerosos sensores. Otra ventaja que posee este dispositivo es que permite modificar su firmware a través de Kinetis Design Studio y posee fuente libre.

Figura 2.4: Vista inferior y lateral externa del dispositivo Mikroe 202

Figura 2.5: Vista frontal interna del dispositivo Mikroe 2026.

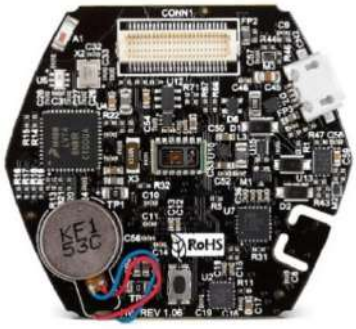

Figura 2.6: Vista frontal inferior del dispositivo Mikroe 2026.

2.1.5. STEVAL-WESU1.

Algunas de las características de este dispositivos serian:

-Es una solución compacta de wearable con sensores y que posee aplicaciones que permiten modificar el firmware.

-Disponible tanto en aplicaciones iOS como en Android.

-Sus principales componentes y los más destacables serian:

-MCU de 32 *bits ultra-low power*.

-Acelerómetro 3D y 3D giros.

-Magnetómetro de 3 ejes.

-Sensor de presión MEMS.

-Procesador de red BLE .

-Resistencia de 50 Ohmios balun con filtro armónico integrado (se denomina balun, 'Balanced-Unbalanced Lines Transformer', a un dispositivo conductor que convierte líneas de transmisión desequilibradas en líneas de transmisión equilibradas.).

-Bateria de Li-On.

-Conector SWD con capacidad de almacenamiento y programación.

Figura 2.7: Vista frontal tanto interna como externa del dispositivo STEVAL-WESU1.

2.2. Estudio de los distintos sistemas operativos

Un sistema operativo móvil es un conjunto de programas de bajo nivel que permite la abstracción de las peculiaridades del hardware específico del teléfono móvil y provee servicios a las aplicaciones móviles que se ejecutan sobre él. Al igual que en los PC's que usan Windows, Linux o Mac OS, los sistemas operativos en los dispositivos móviles serian Android, iOS, Windows Phone o Firefox OS; los sistemas operativos son muchos más simples que los de los ordenadores.

A continuación, vamos a analizar las diferentes características, ventajas y desventajas de los principales sistemas operativos móvil para, posteriormente, poder decidir cuál usaremos para realizar nuestra aplicación.

2.2.1. iOS.

Se trata de un sistema operativo cerrado. Apple no permite que se modifiquen las características internas y las opciones en los ajustes están limitados. Posee funciones que incluyen atajos para el envío de multimedia, ubicación, para mejorar la gestión de conversaciones grupales y una opción para el silencio.

Mas características de este sistema operativo seria la sensación de velocidad cuando se utiliza, esto se logra gracias a algunos trucos de programación; reciben constantemente actualizaciones y no da licencia del software iOS a terceros.

Su ventaja seria que se trata de un sistema operativo muy estable, intuitivo y sencillo de usar. En cuanto a las desventajas: depende de un ordenador que debe tener instalado iTunes para la configuración inicial, para intercambiar contenidos multimedia o para las actualizaciones y, además, decide por ti la ubicación en la que se almacenan los elementos.

2.2.2. Android

Se trata del sistema operativo con mayor popularidad. Es el sistema operativo de Google y su principal característica es la de ser abierto, es decir, poderse modificar, y estar disponible para cualquier fabricante interesado en utilizarlo para sus dispositivos. Está basado en Linux; se diseñó originalmente para móviles pero ha sido modificado para poder ser usado también en las tablets, notebooks y PC's.

Sus ventajas serian: su facilidad para transferir mensajes al ordenador, puesto que, basta con conectar el puerto USB al PC y arrastrar los ficheros. Destaca también por su navegador web, el cual es compatible con Flash Player.

Su desventaja seria su incapacidad para sincronizarse con Outlook y que las actualizaciones se reciben dependiendo del fabricante.

2.2.3. Windows Phone

El sistema operativo de Windows se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas. Su diseño es similar a las versiones de escritorio de Windows. Posee menos apps disponibles que Android o iOS pero se encuentra en crecimiento.

Su ventaja principal sería que posee una pantalla de inicio personalizable que ofrece las notificaciones de las apps de una manera mas sencilla y limpia.

Sus desventajas serían su incompatibilidad con Outlook, la necesidad de instalar el programa Zune en el PC para poder transferir archivos y las pocas apps disponibles para dicho sistema operativo.

2.2.4. Firefox OS

Se trata de un sistema operativo basado en HTML5, de código abierto, que posee núcleo Linux y está desarrollado por Mozilla Corporation. Las aplicaciones web de este sistema operativo pueden ser aplicaciones de servidor y aplicaciones empaquetadas:

-Las aplicaciones de servidor corren vía web, es decir, se requiere conexión a internet para acceder a ellas ya que se tratan de páginas webs con apariencia de aplicación.

-Las aplicaciones empaquetadas requieren la descarga de un paquete comprimido y se cargan desde la fuente local cada vez que se accede a la aplicación.

Así pues, ya estamos listo para elegir tanto el hardware como el software teniendo en cuenta la compatibilidad entre los mismos. Dicha solución la describiré en el siguiente capítulo.

3. Solución escogida.

En este capítulo se va a decidir tanto el hardware como el software que se va a utilizar para llevar a cabo el proyecto y, también, se va a decir la forma en la que se van a manipular los datos para obtener resultados finales útiles puesto que los diferentes datos que captamos de los sensores, según en cuales nos centremos o en cómo nos enfoquemos, pueden tener una finalidad u otra.

3.1. Hardware.

A partir del estudio sobre los diferentes dispositivos que nos podían valer para el desarrollo del proyecto, en el cual se han descrito las principales características, pros y contras de cada desde un lado mas técnico. Por último, dado que poseen características similares, la mayoría de fabricantes nos permiten la modificación del firmware y otras ventajas que poseen en común, he realizado un estudio de los precios de cada uno, que también es un aspecto a tener en cuenta en el presupuesto del proyecto.

Dispositivos	SensorTag CC2650	SensorTag CC3200	Mikroe 2026	STEVAL-WESU1
Precios	25$	39.99$	39.00$	50$

Tabla 3.1:Tabla comparativa de precios los posibles hardware para el proyecto.

3.1.1. *Elección de hardware.*

Por diversos motivos, el dispositivo escogido para llevar a cabo el proyecto ha sido el SensorTag CC2650 del fabricante Texas Instrument. El motivo fundamental de la elección es que la Universidad Politécnica dispone de ellos y me ha podido prestar uno para el desarrollo del trabajo. A continuación, voy a explicar el resto de motivos que me han llevado a escoger este dispositivo.

Se trata de uno de los dispositivos más modernos del mercado, gracias a ello y a que su fabricante posee numerosos dispositivos, hay disponible numerosa documentación, ayuda y explicaciones sobre los sensores que posee y todas las posibles utilidades que tiene.

Además, permite la modificación del firmware para adaptarlo a las necesidades de cada posible aplicación, posee sensores que nos pueden aportar información más que suficiente para desarrolla el proyecto.

Otra de sus principales, de las más importantes, es que se trata de un dispositivo de Bluetooth *low energy* (BLE), lo cual permite una conexión inalámbrica de bajo consumo. A través de este bluetooth se realiza el envío de los datos y nosotros podemos elegir qué información es la que queremos recibir.

Como es evidente que posee el precio más competitivo de todos ellos aun ofreciendo todas las características que acabamos de describir. Y, por último, posee una aplicación tanto para Android como para iOS que permite visualizar los resultados de los diferentes sensores del dispositivo.

3.1.2. Sensores del hardware.

Voy a realizar un pequeño resumen de los diferentes sensores que posee el dispositivo escogido.

3.1.2.1. Sensor de temperatura IR Thermopile TMP007.

Mide la temperatura de un objeto sin contacto directo debido a que absorbe la energía infrarroja pasiva de una longitud de onda entre 4y 16 micrómetros que desprende un objeto. El TMP007 también proporciona memoria no volátil para almacenar los datos. El bajo consumo de energía lo hace adecuado para aplicaciones que requieren uso de energía.

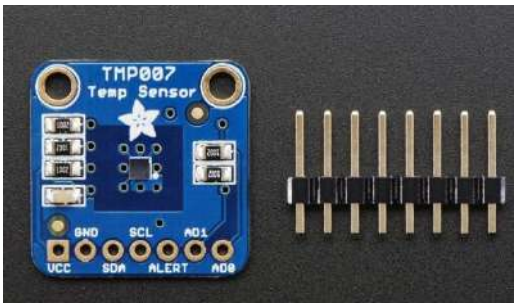

Figura 3.1: Vista frontal del sensor de temperatura IR Thermopile TMP0007.

3.1.2.2. Sensor de presión y altímetro BMP280.

Se trata de un sensor barométrico de precisión basado en el BMP280 de Bosch, que es evolución de los conocidos BMP085 y BMP180, siendo mejor solución para medir la presión barométrica y la temperatura.

A diferencia de los BMP ya mencionados, el sensor de nuestro dispositivo admite tanto conexión por SPI (perfecto para evitar conflictos con las direcciones) y mejora sustancialmente la precisión siendo de +/-1hPa en ppresión y 1ºC en temperatura.

Figura 3.2: Vista frontal del sensor de presión y altímetro BMP280.

3.1.2.3. Sensor digital de humedad HDC1000

Se trata de un sensor digital con sensor de temperatura integrado que provee una gran exactitud de las medidas con muy poca energía. Están calibrados de fabrica. El innovador WLCSP (Wafer Level Chip Scale Package) simplifica el diseño con un paquete compacto. El sensor se encuentra en la parte inferior del dispositivo, lo que lo hace más robusto frente a la suciedad, el polvo y otros agentes de contaminación. El sensor funciona en todo el rango de temperatura entre los -40ºC y los +125ºC.

3.1.2.4. Sensor de luz ambiental OPT3001

Es un sensor que mide la intensidad de luz visible. La respuesta espectral del sensor se ajusta perfectamente a la respuesta fotópica del ojo humano e incluye rechazo importante al infrarrojo. El OPT3001es un medidor de luz de un solo chip, que mide la intensidad de la luz como es visible por el ojo humano.

3.1.2.5. Sensores de movimiento de 9 ejes MPU 9520

Es el dispositivo de detección de 9 ejes más pequeño del mundo que incorpora los últimos diseños más innovadores de InvenSense (proveedor), permitiendo chips de reducido tamaño y de bajo consumo de potencia a la vez que mejora su actuación y coste.

Estos sensores son capaces de procesar complejos algoritmos de movimiento: en concreto el algoritmo de Madgwick, y son los líderes del mercado en brújulas digitales de 3 ejes.

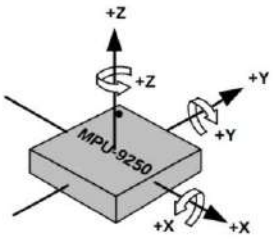

Figura 3.3:Orientación de los ejes de rotación del acelerómetro y giroscopio.

Los 9 sensores serian: 3 sensores de giroscopio, 3 sensores del acelerómetro y 3 sensores del magnetómetro: todos los sensores mencionados del modelo MPU-2950.

Como hemos mencionado, nuestros sensores son capaces de procesar complejos algoritmos usados para captar la información necesaria para los movimientos. Para los Sistemas de Referencia de Orientación y Rumbo, sistemas denominados AHRS debido a sus siglas en inglés, los algoritmos de procesamiento de datos más usados son: el Filtro de Kalman, el de Mahony y el algoritmo de Madgick.

Los sensores IMU o magnetómetros son dispositivos electrónicos que a partir de los sensores del acelerómetro, los cuales detectan la aceleración, y el giróscopio, los cuales deectan los movimientos rotacionales, permiten obtener la velocidad, la orientación y la fuerza de gravedad. El principal problema de estos dispositivos sería que acumula los errores y, al estar continuamente modificando su orientación, pasado un tiempo, la orientación calculada se diferente a la orientación real. Este problema se soluciona gracias al uso de los magnetómetros (AHRS).

El uso de varios sensores tiene distintas ventajas, como el aumentar la información sobre el espacio y el tiempo de las variables estudiadas, pero también mejora la precisión de las mediciones al disminuir el ruido.

A continuación, vamos a explicar cada uno de los complejos algoritmos que hemos mencionado anteriormente y que nuestros sensores son capaces de procesar.

3.1.3. Fundamentos teóricos para el hardware.

Me dispongo a resumir los complejos algoritmos que nuestros sensores son capaces de procesar y que son necesarios para poder comprender como son capaces de captar magnitudes como la orientación.

3.1.3.1. Filtro Kalman

Es un algoritmo recursivo de procesamiento de datos eficiente mediante los mínimos cuadrados, que toma en cuenta todos los datos de los que se dispone y, además, estima el valor de posibles variables que nos interesan. El mismo algoritmo usa las características, tanto de las señales como del ruido, teniendo en cuenta la dinámica tanto del proceso de estimación como del proceso de medición. El modelo debe ser lineal y el ruido gaussiano ya que calcula un estimador lineal y óptimo y se va actualizando para cada momento. El filtro de Kalman asume que los estados del sistema en un momento determinado se pueden calcular a partir de los estados anteriores del sistema.

3.1.3.1.1. Calculo de Roll, Pitch, Yaw usando el giróscopio.

Para calcular los ángulos roll, pitch y yaw, se realiza la siguiente integral:

$$\int_{t_0}^{t} \varphi(\mathrm{t})dt + \varphi(t_0)dt_0 \tag{3.1}$$

La velocidad angular de los ejes de rotación del sistema de referencia fijo se puede medir con el giroscopio, pero en general los ángulos obtenidos a partir de estas mediciones no se corresponden con los ángulos calculados a partir de las mediciones del acelerómetro y del magnetómetro. Así pues, se usa una matriz de transformación que relaciona las derivadas de los ángulos de Euler XYZ con las velocidades medidas con el giroscopio. En dicha ecuación los valores de pitch y roll (φ y θ) se calculan a partir de las mediciones del acelerómetro:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} sen(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ cos(\phi) & -sen(\phi) \\ \frac{sen(\phi)}{cos(\theta)} & \frac{cos(\phi)}{cos(\theta)} \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{3.2}$$

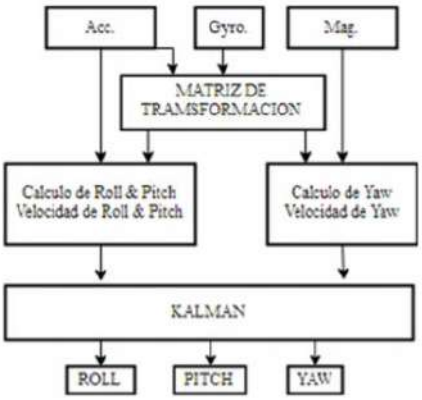

Figura 3.4: Implementación del filtro Kalman para el calculo de los ángulos Euler.

A continuación, y dada su importancia, voy a proceder a explicar la representación de los ángulos mediante los grados Euler:

Los ángulos de Euler determinan la orientación a partir de tres coordenadas angulares en sus sistema de referencia móvil frente a otro fijo y ambos ortogonales. La ventaja de Euler frente a la representación cuaternion es su fácil interpretación gráfica:

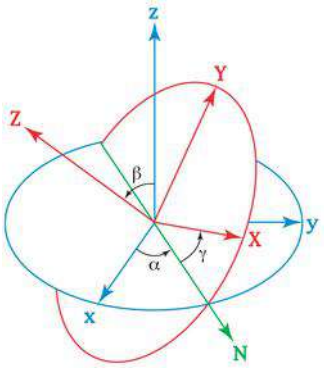

Figura 3.5: Representación de los ángulos de Euler.

3.1.3.2. Algoritmo de Madgick.

Este algoritmo fue desarrollado por Sebastián Madgwick a partir del filtro Kalman. Este filtro también se puede aplicar a IMUs y a MARGs (giróscopios, acelerómetros y magnetómetros triaxiales). Además, dicho filtro se trata de una representación de la orientación por medio de cuaterniones (que explicaré a continuación) y, por ello, no está sujeto a los problemas de singularidad que se dan en aquellas representaciones basadas en matrices de cosenos. El algoritmo de Madgwick se basa en el el algoritmo del gradiente descendente que utiliza para calcular, a partir de los datos recopilados por el giroscopio, la dirección del error de medición. El algoritmo se dividide en cuatro partes:

1. Se calcula la orientación a partir de las velocidades angulares medidas por el giroscopio

2. Se calculan las orientaciones a partir de los vectores medidos del campo gravitacional y del campo magnético.

3.Se fusionan las dos estimaciones anteriores.

4. Se normaliza el cuaternión de la medición. A continuación, detallaremos cada una de las partes del algoritmo.

Su principal ventaja es el bajo coste computacional que supone, lo que nos permite, en nuestro caso, que se lleve a cabo desde el teléfono móvil. Además, este filtro sirve con niveles de muestreo bajos y nos permite ajustar este y otros parámetros como la ganancia.

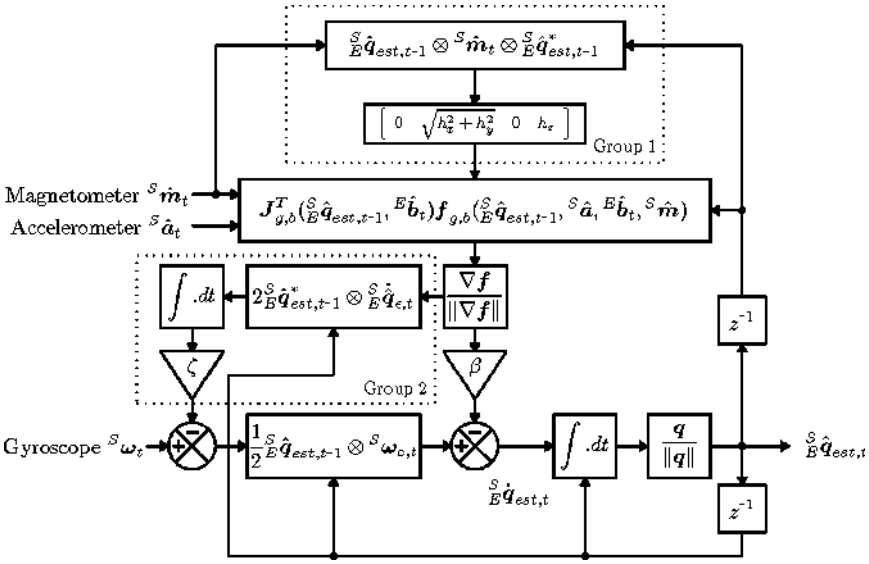

Figura 3.6: Diagrama de bloques del algoritmo de Madgick.

A continuación, y dada su importancia voy a explicar la representación de la orientación a partir de cuaterniones que usa este filtro:

Un cuaternión se basa en un número complejo de cuatro dimensiones a partir del cual se puede representar la orientación de un cuerpo en 3 dimensiones. Su representación seria:

Q=a+bi+cj+dk siendo a,b y c reales e i,j y k unidades cuaternarias.

Se suelen utilizar los cuaternarios para calcular rotaciones tridimensionales, tienen ventajas sobre Euler como la de que evitan el *gimbal lock* cuando rota el eje y 90 grados.

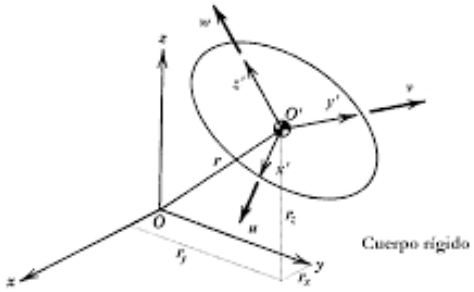

Figura 3.7: Representación de los cuaterniones.

3.1.4. Biomecánica aplicada al 'running'.

Dado que el objeto de nuestra aplicación va a ser medir la aceleración con la que los pies chocan con el suelo cuando corremos, es importante tener claro el concepto de la aceleración y como se produce el mismo aplicado a esta actividad:

Para ello hemos de tener en cuenta las leyes de Newton, en especial la tercera ley de Newton según la cual, la fuerza que ejerce el suelo sobre nuestro pie es la misma que la de nuestro pie al caer pero opuesta. Esta fuerza recibe el nombre de: fuerza de reacción del suelo.

Al correr, cada persona pisa de una forma diferente y genera diferentes tipos de fuerza por la dirección y duración. En general, se dividen en dos tipos de corredores:

1) Los que apoyan primero el 'retropié' , es decir, el talón, en los cuales, el pico de aceleración se produce al principio de la zancada, descendiendo posteriormente y ascendiendo en el momento de impulso.
2) Los que apoyan primero el 'mesopié', es decir, la punta del pie, en los cuales no se diferencia tanto la diferencia de la aceleración.

En la grafica de la Figura 3.8 podemos observar dichas diferencias. Así sería, de forma normalizada, la gráfica que debería devolvernos nuestra aplicación si todo funciona correctamente.

En caso de algún tipo de lesión, la gráfica tomada será igual o muy parecida, solamente se diferenciará en un descenso casi constante de la aceleración. En caso de que la gráfica dibuje una gráfica muy diferente, el fisioterapeuta deberá analizar si la lesión afecta de manera que el corredor desvía el pie al correr o no pisa de forma correcta.

Es decir, cualquier anomalía de la grafica normalizada permite detectar que hay algún causante por el cual el paciente no corre de forma correcta, lo cual puede llegar, a la larga, a generar lesiones más graves.

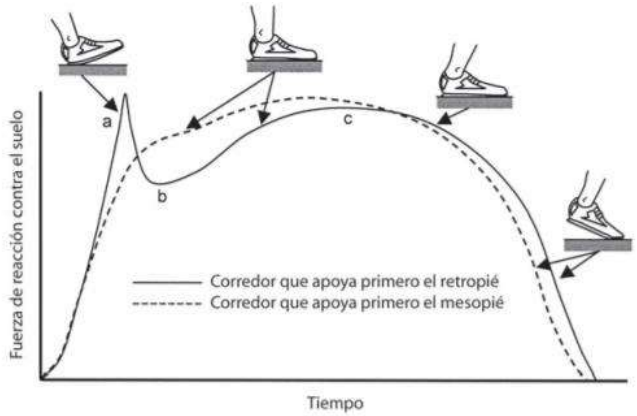

Figura **3.8: Evolución de la fuerza en una zancada.**

3.2. Software

Tras el anterior estudio realizado sobre los distintos sistemas operativos posibles en los cuales podemos realizar nuestra aplicación, a continuación, explicaré la decisión tomada y, posteriormente, mostraré mediante un diagrama un esquema general que muestra como funciona la sencilla aplicación.

3.2.1. Elección del software.

El software elegido ha sido Android ya que posee numerosas ventajas: es el más extendido del mercado, cuenta con más usuarios y más aplicaciones que el resto de sistemas operativos. Gracias a ello y a que posee código abierto, es relativamente fácil encontrar foros y cursos a cerca de como programar y de como usar las numerosas clases y funcionalidades que posee dicho sistema operativo.

Para llevarlo a cabo se ha utilizado Android Studio para Windows, un entorno de desarrollo de aplicaciones bastante sencillo que permite el desarrollo de aplicaciones para Android a partir de archivos en lenguaje Java y xml, un sencillo lenguaje de programación basado en etiquetas.

3.2.2. Idea y diseño de la aplicación.

En primer lugar, voy a realizar una explicación de la idea inicial de aplicación antes de implementarla, con su correspondiente diagrama que plasmará a modo de esquema la idea inicial sobre la que empecé el desarrollo de la aplicación. En algún capitulo posterior veremos y realizaremos la comparativa de si se ha podido realizar según lo esperado y si no es el caso, veremos los problemas que han surgido y las soluciones que se han podido llevar a cabo.

Así pues, como inicio de la aplicación, mi idea sería realizar un menú principal en el que a partir de tu nombre, apellido y edad puedas elegir crear usuario o iniciar sesión. Ambos dos te llevarían a lo que realmente sería el menú principal para el usuario.
Una vez el usuario ha sido creado o se ha iniciado sesión, el menú principal daría la opción de tomar nuevas medidas o de visualizar resultados de actividades anteriores a través de gráficas, de esta forma cada usuario podría llevar un seguimiento de como está realizando la actividad física.

En cuanto a la parte técnica de como implementarlo, habría que realizar una base de datos inicial en la que se guarden los usuarios que se van creando cada vez que se pulsa el botón de registrar usuario. En esta misma base de datos habría que buscar al usuario según su nombre/apellido o edad cuando un usuario se dispone a iniciar sesión.
A continuación, el siguiente reto, sería lograr que una vez selecciona la opción de tomar nuevas medidas, se conecte el dispositivo móvil a nuestro SensorTag a través de Bluetooth; es decir, lograr establecer una comunicación vía Blueetooth basada en la recepción de datos que nos envía el sensor.
Posteriormente, se deberían almacenar los datos recibidos del usuario con otra base de datos, una base de datos para cada usuario en el que se almacenen los datos del sensor que queremos en cuestión y finalmente, habría que manipular los datos de dicha base de datos con el fin de representarlos en una gráfica con un resumen de la actividad realizada durante un tiempo limitado.
Dado que nuestra aplicación tiene como fin ser usada por fisioterapias que ayudan a sus pacientes tanto a mejorar sus marcas, como a correr de forma eficiente o recuperarse de una lesión, las carreras medidas no serán de más de 30 segundos, puesto que no se requieren más medidas para analizar el cómo realiza la actividad un individuo.

4. Implementación del software

En el siguiente capítulo voy a explicar la programación llevada a cabo en Android Studio para el desarrollo de la aplicación. Se ha programado toda la aplicación en lenguaje Java y he necesitado adquirir conocimientos sobre bases de datos SQLite y sobre conexión de Bluetooth.

Voy a explicar las clases más características e importantes, necesarias para el desarrollo del trabajo. Así pues, en primer lugar desarrollaré las bases de datos, y, posteriormente, la programación de cada clase y layout principales conforme se va usando la aplicación.

Para finalizar, mediante un diagrama de la aplicación, mostraré las imágenes que van saliendo en la misma conforme avanzamos.

4.1 Base de datos SQLite

En este trabajo se han realizado dos bases de datos diferentes, las cuales se relacionan gracias a una variable ID. Esta variable era necesaria para poder identificar y mostrar, para cada usuario, las diferentes carreras que realizaba, dado que las bases de datos son independientes y requeríamos de alguna forma de que se relacionarán. Ambas variables que identifican las tablas se rellenan de forma aleatoria.

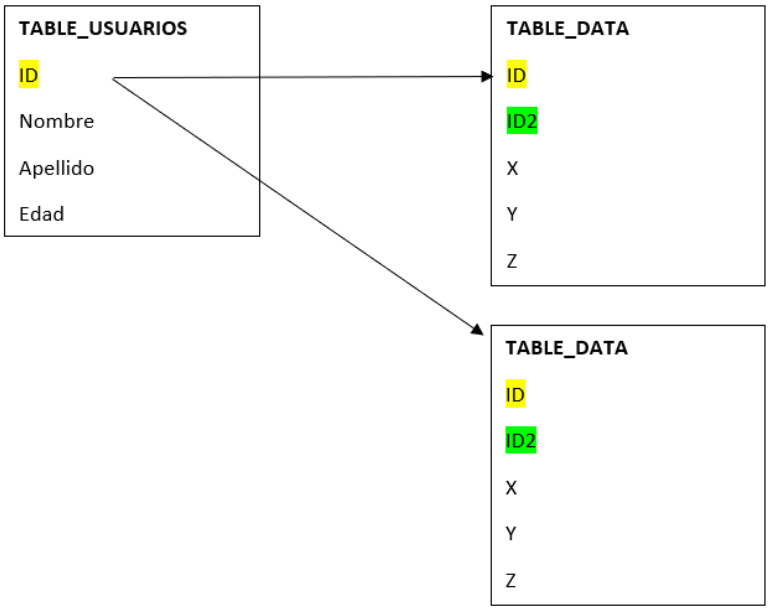

Figura **4.1 Estructura de las bases de datos.**

La clase en la que crean las bases de datos se denomina: BaseDeDatos.class, se usa programación SQLite básica como podemos observar a continuación, en la Figura 4.2.

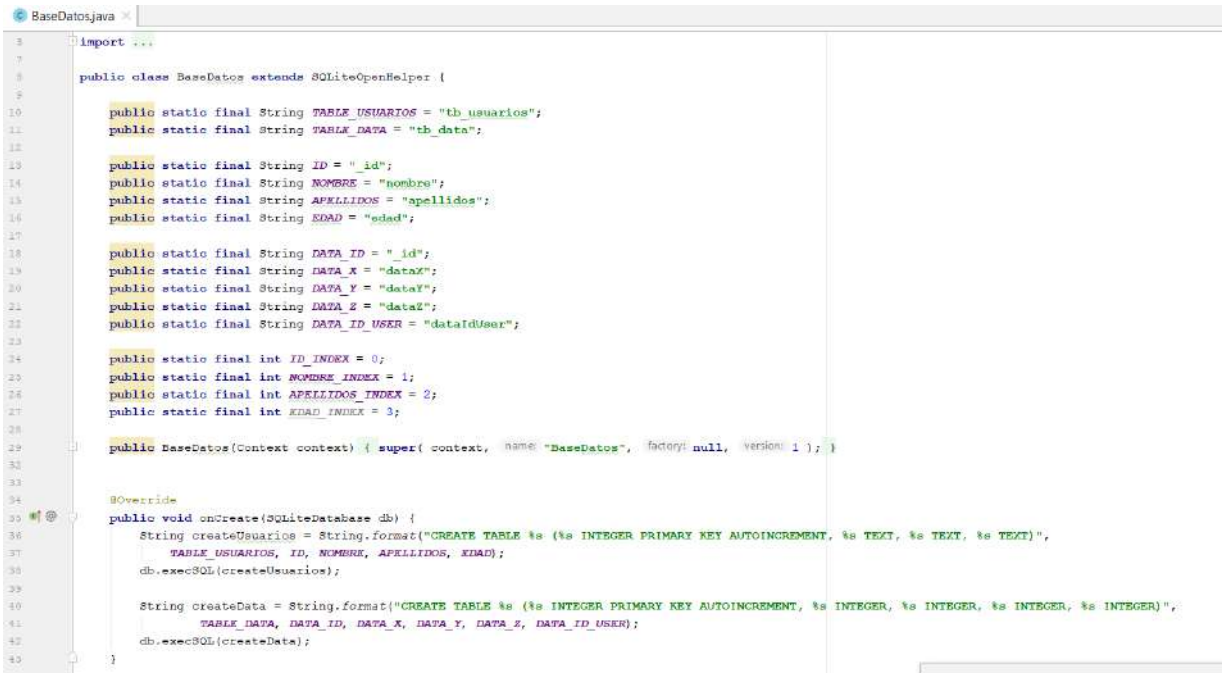

Figura **4.2 Clase BaseDatos.class con programación SQLite.**

En el código se puede observar cómo se crean las variables String con cada columna de cada una de las columnas. Las variables int son útiles para recorrer posteriormente la tabla puesto que en la misma clase se encuentra un método usado posteriormente para realizar búsquedas dentro de la base de datos.

Cabe resaltar que, debido a que la variable id se genera de forma autoincremental (INTEGER PRIMARY KEY AUTOINCREMENTAL) para cada usuario. Se ha creado una clase especial para poder guardar está variable aleatoria y poderla usar en la tabla de los datos recibidos con identificador que une el usuario con sus datos. Dicha clase se denomina: MyApplication.class.

Figura **4.3 Código Java de MyApplication.class**

Hay que tener en cuenta que la tabla de los usuarios se llama TABLE_USUARIOS y en la que se almacenaran los resultados medidos se denomina TABLE_DATA. Dado que recopilamos los

datos de los tres sensores que componen el acelerómetro (X, Y, Z), estos serán los datos recibidos y que guardaremos para posteriormente poder representarlos.

La clase en la que se rellenan dichas bases de datos los veremos a continuación. Vamos a comenzar a ver las clases por orden de uso de la aplicación en el siguiente apartado.

4.2 Aplicación paso a paso

En este apartado, describiré de la forma más ordenada y resumida posible las principales *activities* y métodos por las que está compuesta la aplicación.

MainActivity.class

Lo primero que se encuentra el usuario al abrir la aplicación es un menú en el que debe elegir si crear usuario o iniciar sesión, según lo que elija, la aplicación debe rellenar una línea en la tabla de los usuarios (TABLE_USUARIOS) o, si escoge la segunda opción, se realizará una búsqueda en la misma tabla.

A continuación, veremos el código necesario para implementar lo ya descrito.

```
import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_main );


        Button BotonCrearUsuario = (Button) findViewById( R.id.BotonRegistrarUsuario );
        BotonCrearUsuario.setOnClickListener( (view) -> {

                BotonCrearUsuario() ;
        } );

        final Button BotonIniciarSesion = (Button) findViewById( R.id.BotonIniciarSesion );
        BotonIniciarSesion.setOnClickListener( (view) -> {
                BotonIniciarSesion();
        } );



    }


    public void BotonIniciarSesion(){
        Intent Inicio = new Intent( packageContext: this, IniciarSesion.class );
        startActivity( Inicio );
    }

    public void BotonCrearUsuario (){
        Intent CrearUsuario = new Intent( packageContext: this, com.example.followyourrun.CrearUsuario.class );
        startActivity( CrearUsuario);
    }


}
```

Figura **4.4 Código del MainActivity.class.**

Tal y como se puede observar, para crear usuario nos dirigirá al activity CrearUsuario.class, y, para iniciar sesión, nos dirigirá al activity IniciarSesion.class.

Antes de mostrar cada una de ellas, voy a mostrar la programación del layout que ven los usuarios, el cual se encuentra en la carpeta res/layour/activity_main.xml. Al final del capitulo se mostrarán capturas de dichas pantallas

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">


    <Button
        android:id="@+id/BotonRegistrarUsuario"
        android:layout_width="232dp"
        android:layout_height="wrap_content"
        android:text="Registrar usuario"
        android:translationX="75dp"
        android:translationY="220dp" />

    <Button
        android:id="@+id/BotonIniciarSesion"
        android:layout_width="232dp"
        android:layout_height="wrap_content"
        android:text="Iniciar sesión"
        android:translationX="75dp"
        android:translationY="240dp" />

</LinearLayout>
```

Figura **4.5 Código del layout inicial.**

Como se puede observar, está realizado de la forma más sencilla posible. Cabe destacar la configuración del 'id' gracias al cual se relaciona el botón con el activity correspondiente y el texto que muestran los botones, el resto, son simples configuraciones de posición, tamaño, etc.

CrearUsuario.class

En este activity, el usuario introducirá manualmente sus datos, los cuales se le solicitan. Con los datos que el usuario introduce en la pantalla, se rellena una fila de la tabla de los usuarios ('TABLE_USUARIOS'), código que veremos posteriormente en la Figura 4.6.

```java
public class CrearUsuario extends AppCompatActivity {
    private EditText nombre, apellidos, edad;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_crear_usuario );

        nombre = (EditText) findViewById( R.id.nombre );
        apellidos = (EditText) findViewById( R.id.apellidos );
        edad = (EditText) findViewById( R.id.edad );

        Button BotonGuardarUsuario = (Button) findViewById( R.id.BotonGuardarUsuario );
        BotonGuardarUsuario.setOnClickListener( (view) -> {

                guardarUser();
        } );
    }

    public void guardarUser () {
        BaseDatos baseDatos = MyApplication.getInstance().getBaseDatos();

        String name = nombre.getText().toString();
        String surname = apellidos.getText().toString();
        String age = edad.getText().toString();


        if (!name.isEmpty() && !surname.isEmpty() && !age.isEmpty()) {

            try {
                baseDatos.insertIntoUsuarios(name, surname, age);
                Toast.makeText( context: this, text: "Registrado correctamente", Toast.LENGTH_SHORT ).show();

                Intent MainMenu = new Intent( packageContext: this, MainMenu.class );
                startActivity( MainMenu );
            } catch (Exception ex){
                ex.printStackTrace();
                Toast.makeText( context: this, text: "Error insertando", Toast.LENGTH_SHORT ).show();
            }
```

Figura **4.6 Código del CrearUsuario.class.**

En la Figura 4.5 se observa como para guardar en la base de datos de los usuarios, se ha de llamar a la activity que contiene el método 'insertIntoUsuarios, mostrado a continuación. De la misma forma al iniciar sesión se ha de llamar a dicha activity para que lance la el método '' como vemos en la Figura 4.7.

```
public void insertIntoUsuarios(String nombre, String apellidos, String edad) {
    SQLiteDatabase db = getWritableDatabase();
    String insert = String.format("INSERT INTO %s (%s, %s, %s) VALUES ('%s', '%s', '%s')",
        TABLE_USUARIOS, NOMBRE, APELLIDOS, EDAD, nombre, apellidos, edad);
    db.execSQL(insert);
}

public int checkIfUserExists(String nombre, String apellidos) {
    SQLiteDatabase db = getReadableDatabase();

    String select = String.format("SELECT * FROM %s WHERE %s LIKE '%s' AND %s LIKE '%s'", TABLE_USUARIOS, NOMBRE, nombre,
    Cursor fila = db.rawQuery(select,  selectionArgs: null);

    if (fila.moveToFirst()) {
        String name = fila.getString(NOMBRE_INDEX);
        String surname = fila.getString(APELLIDOS_INDEX);

        int id = fila.getInt(ID_INDEX);

        fila.close();
        return id;
    } else {
        fila.close();
        return -1;
    }
}
```

Figura **4.7 Código para insertar línea en la tabla y realizar búsquedas.**

Además, como se puede observar en la Figura 4.5, una vez creado el usuario, se pasa automáticamente a un activity denominado MainActivity que veremos posteriormente. Además, se tiene en cuenta el fallo de dejarse algún dato sin rellenar, en cuyo caso saltaría un mensaje informándonos de que debemos rellenar todos los datos.

El layout, que se encuentra en res/layout/activity_crear_usuario.xml, lo muestro a continuación:

```
    <EditText
        android:id="@+id/nombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Nombre"
        android:translationY="60dp" />


    <EditText
        android:id="@+id/apellidos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Apellidos"
        android:translationY="120dp" />

    <EditText
        android:id="@+id/edad"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Edad"
        android:translationY="180dp" />

    <Button
        android:id="@+id/BotonGuardarUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Guardar usuario"
        android:translationY="200dp"
        android:translationX="120dp"/>


</LinearLayout>
```

Figura **4.8 Código del layout activity_crear_usuario.xml.**

IniciarSesion.class

Este activity nos envía a un método que se encuentra en la clase de BaseDatos.class, y que realiza una búsqueda dentro de la tabla 'TABLE_USUARIOS'. El método para ello se denomina checkIfUserExist, cuyo código se muestra en la Figura 4.6. El código del activity IniciarSesion.class se encuentra a continuación:

```
public class IniciarSesion extends AppCompatActivity {
    private EditText NombreUsuario, ApellidoUsuario;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_iniciar_sesion );

        NombreUsuario = (EditText) findViewById( R.id.NombreUsuario );
        ApellidoUsuario = (EditText) findViewById( R.id.ApellidoUsuario );

        Button BotonIniciarSesion = (Button) findViewById( R.id.BotonIniciarSesion );
        BotonIniciarSesion.setOnClickListener( (view) -> {

                IniciarSesion();
        } );
    }

    public void IniciarSesion() {
        BaseDatos baseDatos = MyApplication.getInstance().getBaseDatos();

        String nombre = NombreUsuario.getText().toString();
        String apellidos = ApellidoUsuario.getText().toString();

        try {
            int id = baseDatos.checkIfUserExists(nombre, apellidos);
            if (id!=-1) {
                MyApplication.getInstance().setId(id);
                Toast.makeText( context: this, text: "Usuario encontrado", Toast.LENGTH_SHORT).show();
                Intent MainMenu = new Intent( packageContext: this, MainMenu.class );
                startActivity( MainMenu );
            } else {
                Toast.makeText( context: this, text: "Usuario NO encontrado", Toast.LENGTH_SHORT).show();
            }
        } catch (Exception ex){
            ex.printStackTrace();
            Toast.makeText( context: this, text: "Error buscando usuario", Toast.LENGTH_SHORT ).show();
        }

    }
```

Figura **4.9 Código del activity IniciarSesion.class.**

En el caso de que se haya encontrado en dicha tabla, entonces te manda a la pantalla del activity MainMenu.class, si no encuentra el nombre y apellido, que el usuario ha introducido, en la tabla de la base de datos, mandará un mensaje informando al usuario, el cual podrá modificar la búsqueda fallida.

El layout en el cual el usuario realiza la búsqueda a partir del nombre y del apellido posee el siguiente código y se encuentra dentro de Android Studio en la carpeta res/layout/activity_iniciar_sersion.xml:´

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent" android:orientation="vertical"
    android:layout_height="match_parent"
    tools:context=".IniciarSesion">

    <EditText
        android:id="@+id/NombreUsuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Nombre"
        android:layout_marginTop="30dp" />


    <EditText
        android:id="@+id/ApellidoUsuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Apellidos" />

    <Button
        android:id="@+id/BotonIniciarSesion"
        android:layout_marginTop="20dp"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:text="Iniciar sesión"/>

</LinearLayout>
```

Figura **4.10 Código del layout activity_iniciar_sesion.xml.**

MainMenu.class

En este activity, el usuario debe elegir entre capturar unas nuevas medidas o ver los resultados de medidas anteriores. Si selecciona tomar nuevas medidas pasaría a la clase Captura.class, en la cual se iniciará el proceso de conexión de Bluetooth (se debe conectar al Bluetooth llamado 'SensorTag CC2650'). En el caso de seleccionar la opción de visualizar capturas anteriores, le saldrá una lista con las últimas medidas tomadas por ese usuario. Cada una de dichas medidas posee la representación de una tabla de datos, todas las tablas tienen un mismo ID que lo identifican con el usuario y un ID2 diferente, como hemos visto anteriormente.

```
public class MainMenu extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_main_menu );


        Button ComenzarCaptura = (Button) findViewById( R.id.ComenzarCaptura );
        ComenzarCaptura.setOnClickListener( (view) -> {

                ComenzarCaptura();
        } );

        final Button Resultados = (Button) findViewById( R.id.Resultados );
        Resultados.setOnClickListener( (view) -> { Resultados(); } );

        TextView usuario = (TextView) findViewById( R.id.Usuario );
        usuario.setText(String.format("USUARIO: %d", MyApplication.getInstance().getId()));
    }

    public void ComenzarCaptura(){
        Intent Captura = new Intent( packageContext: this, com.example.followyourrun.Captura.class );
        startActivity( Captura );

    }

    public void Resultados(){

    }
```

Figura **4.11 Código del actitivity MainMenu.class.**

Las siguientes clases, enfocadas a la conexión de Bluetooth y representación de datos son explicadas en siguientes capítulos, dando en ellos una explicación más extensa a cerca del

Bluetooth BLE, por haberlo considerado una de las partes más costosas e importantes del proyecto.

A continuación, en la Figura 4.10, represento un diagrama donde se puede ver la lógica que sigue la aplicación.

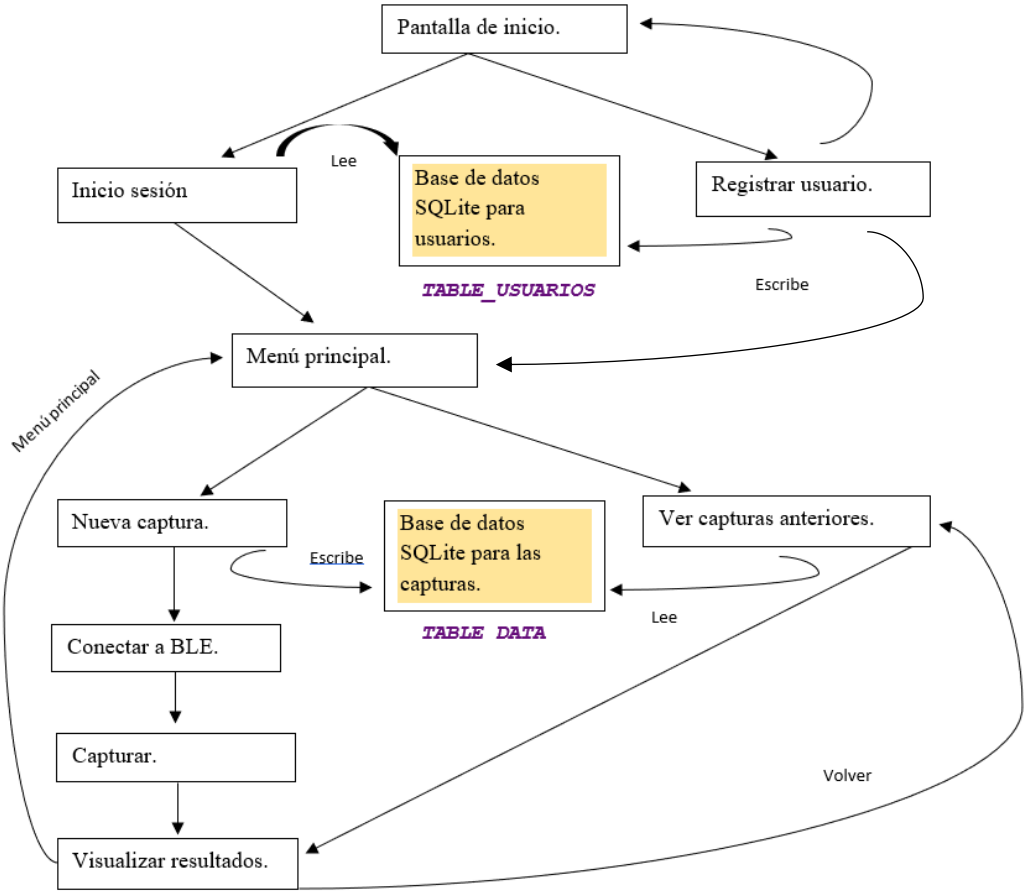

Figura **4.12. Diagrama de la lógica de la aplicación.**

A continuación, voy a mostrar la pantalla que se visualiza en la aplicación en cada momento del diagrama, de forma que, con el código ya mostrado de cada clase y las capturas posteriores, se pueda ver de forma clara como se ha ido programando toda la aplicación. Como aclaración, el problema de la gravedad no se refleja a continuación puesto que lo datos que se muestran ya han sido tratados por diferentes clases, pero se ha de tener en cuenta que no se representan directamente, si no que son modificados para compensar el efecto de la gravedad.

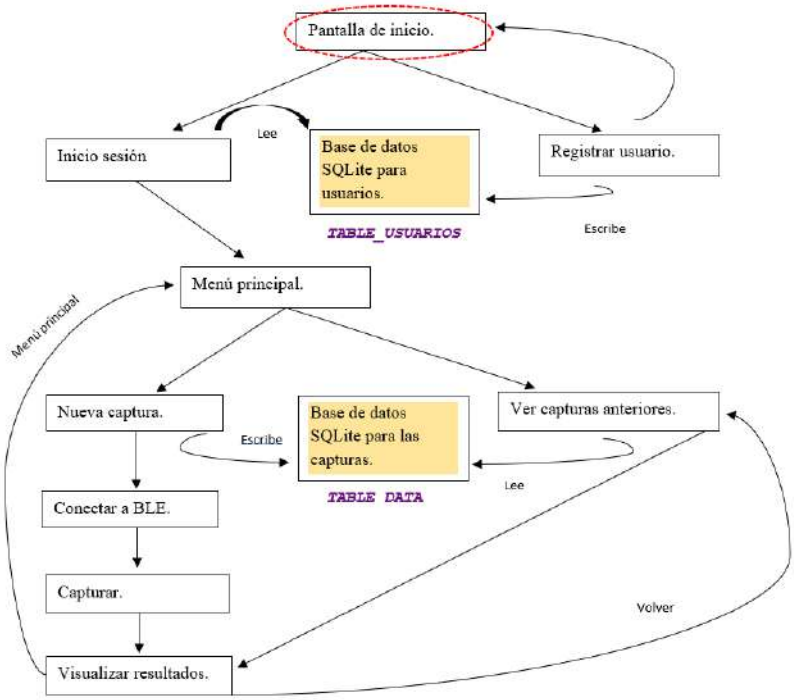

Figura **4.13. Diagrama de la lógica de la aplicación.**

Figura **4.14. Pantalla de inicio de la aplicación.**

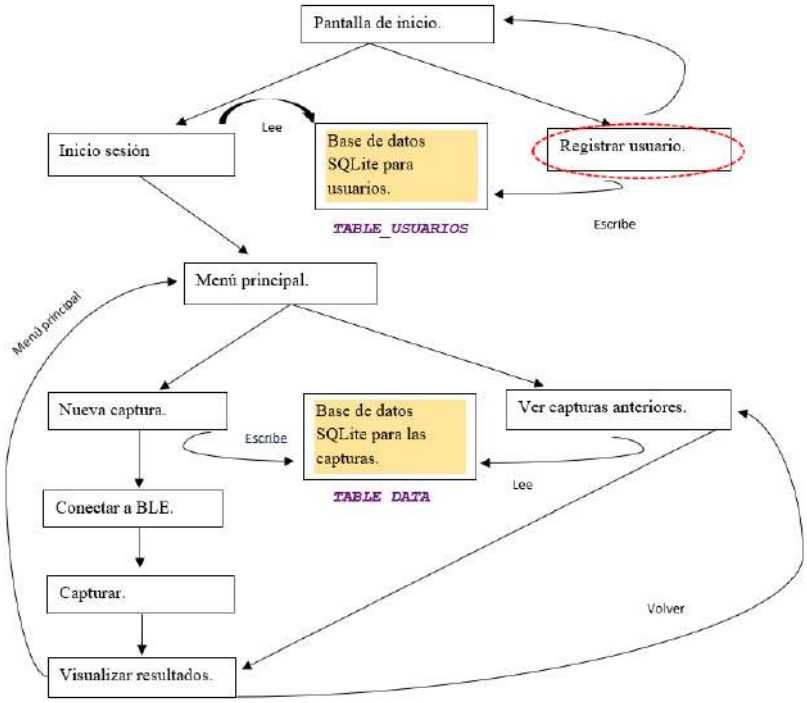

Figura **4.15 Diagrama de la lógica de la aplicación.**

Figura **4.16. Pantalla del registro de usuario.**

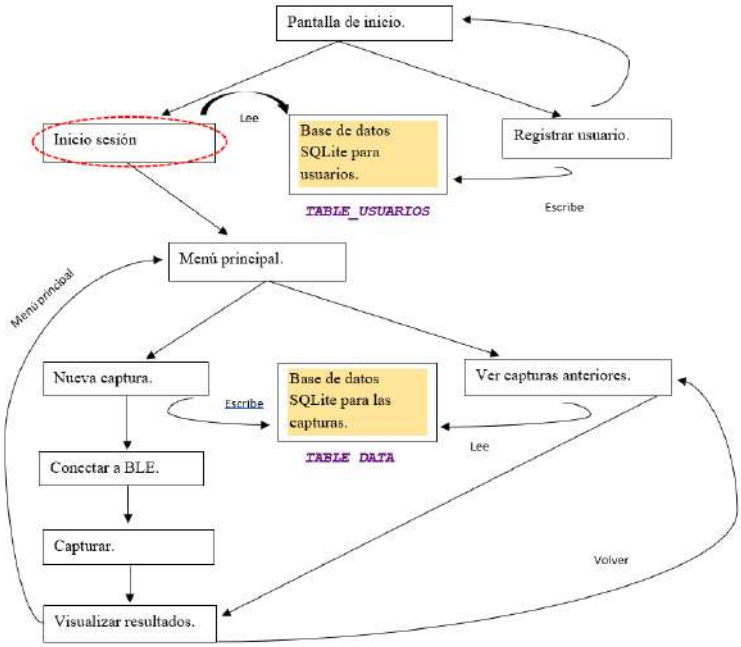

Figura **4.17. Diagrama de la lógica de la aplicación.**

Figura **4.18. Pantalla de inicio de sesión.**

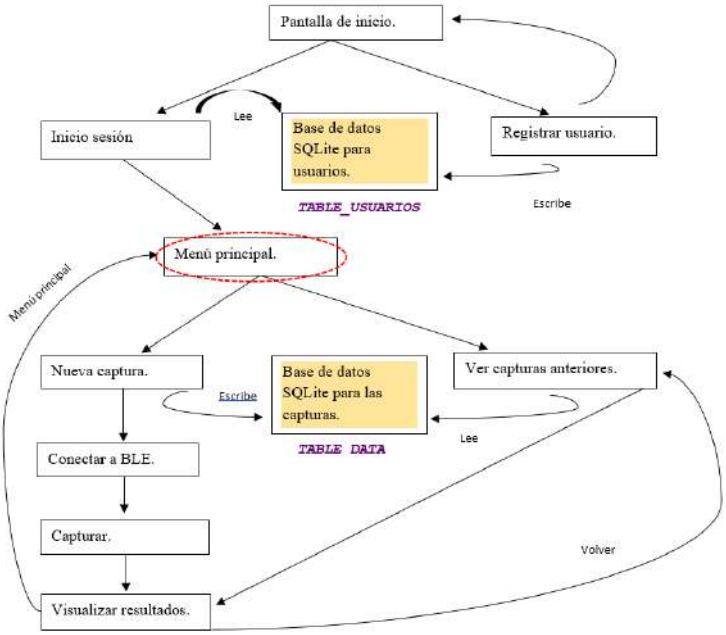

Figura **4.19. Diagrama de la lógica de la aplicación.**

Figura 4.20. Pantalla del menú principal de la aplicación.

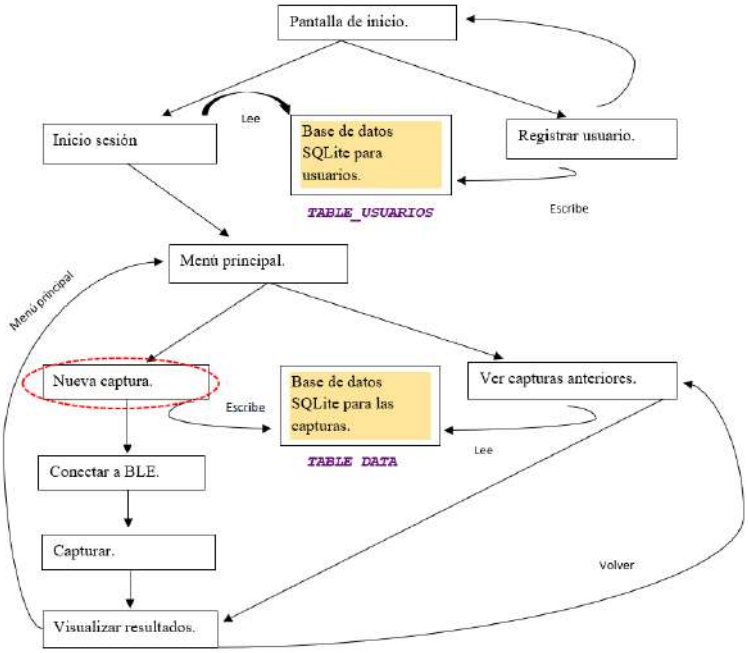

Figura **4.21. Diagrama de la lógica de la aplicación.**

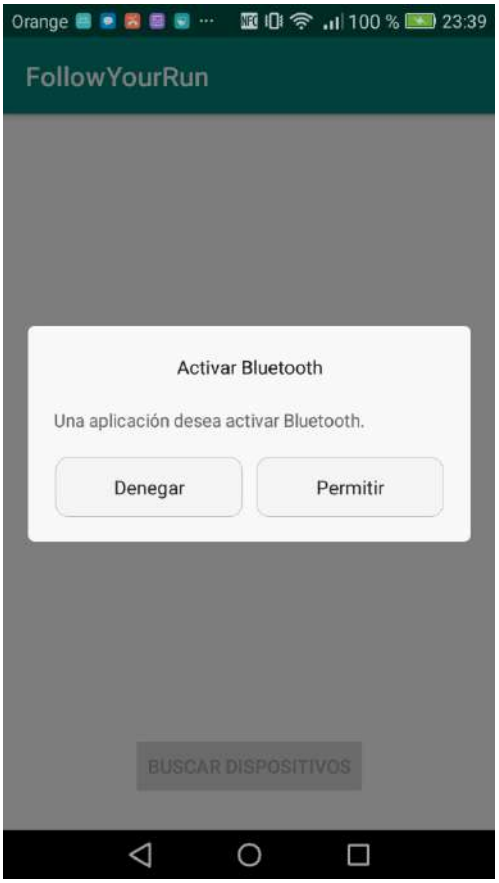

Figura 4.22. Pantalla activación Bluetooth.

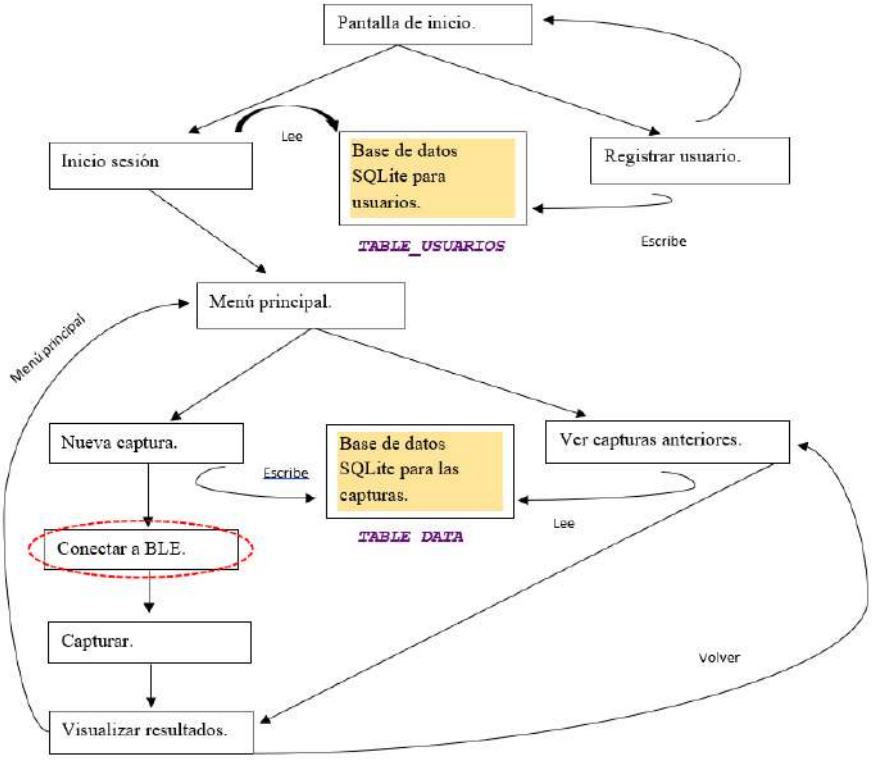

Figura 4.23. Diagrama de la lógica de la aplicación.

Figura 4.24. Pantalla detección del Bluetooth.

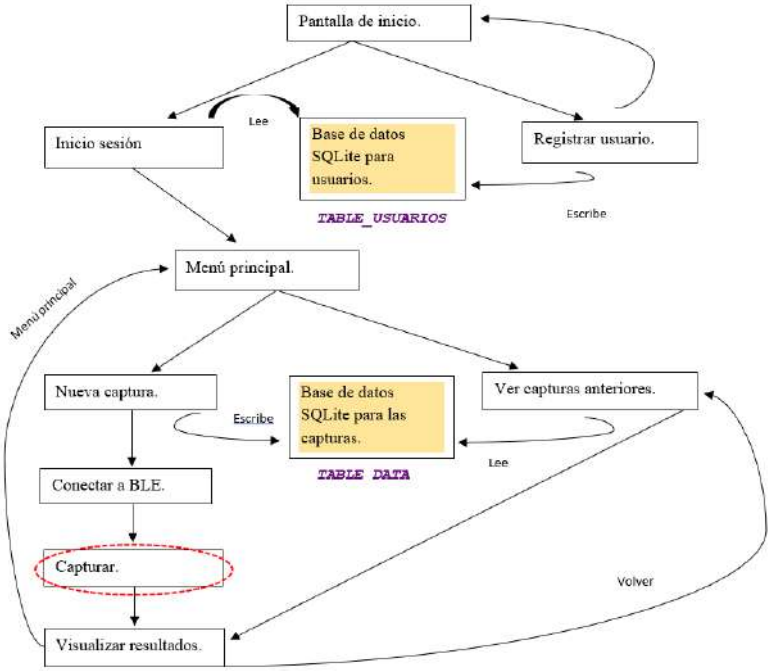

Figura 4.25. Diagrama de la lógica de la aplicación.

Figura 4.26. Pantalla de la conexión del Bluetooth.

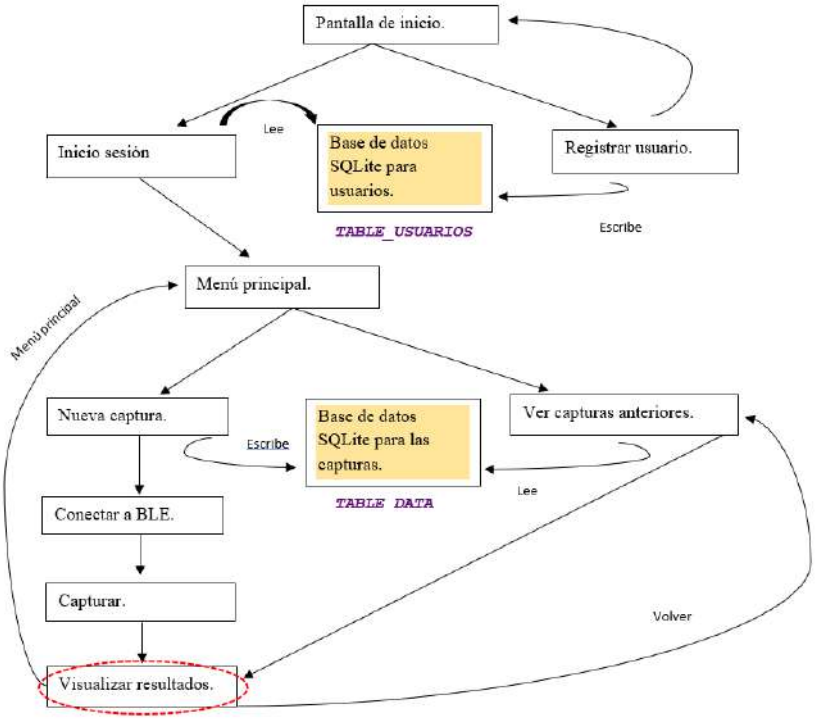

Figura **4.27. Diagrama de la lógica de la aplicación.**

Figura **4.28. Pantalla del resultado de la captura.**

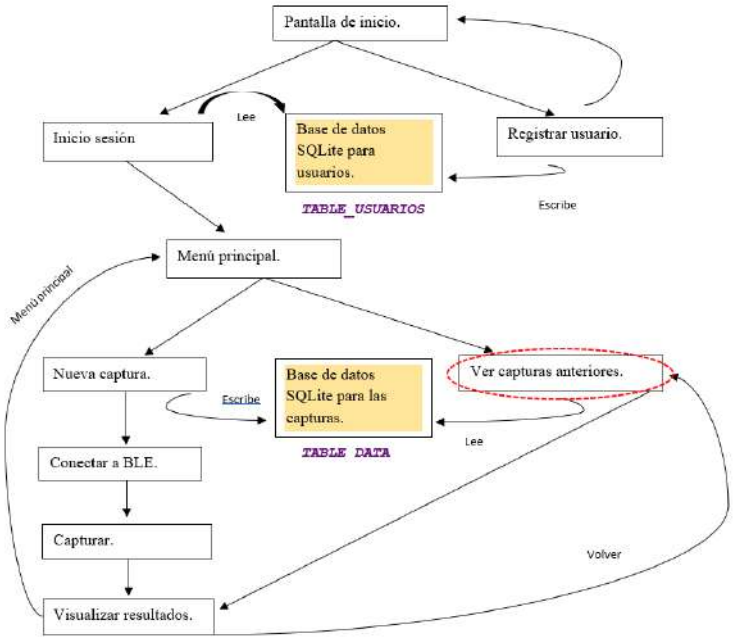

Figura **4.29. Diagrama de la lógica de la aplicación.**

Figura **4.30. Pantalla de elección de capturas anteriores.**

5. Puntualizaciones

Durante el desarrollo del trabajo han surgido una serie de cuestiones a tener en cuenta debido al estudio y aprendizaje de conceptos que han supuesto y que considero de suficiente relevancia como para destacarlos en este capítulo.

En primer lugar, explicaré la compensación de la gravedad que ha sido necesaria y el porqué me he tenido que enfrentar a esta cuestión. A continuación y como consecuencia del problema de la aceleración de la gravedad, debemos calcular la posición del dispositivo, lo cual requirió la programación del algoritmo de Madgwick para obtener los cuaterniones y nos permitió detectar y analizar, además, una inestabilidad que se produce en el z, la cual analizamos. Por último, me parece importante, realizar un pequeño resumen de como se ha llevado a cabo la conexión y recepción de datos vía Bluetooth por ser uno de los aspectos de la programación en los que más tiempo he invertido.

5.1 Compensación de la gravedad

Antes de comenzar a implementar el código, mientras analizaba el dispositivo SensorTag, gracias a mis tutores y observando todas las utilidades del mismo con la App para Iphone denominada: 'SensorTag' pude observar qué al ver la gráfica del acelerómetro, salía algo extraño, causado por la gravedad.

Así pues, mi aplicación iba a mostrar los valores distorsionados debido a la gravedad. Es decir, como veremos a continuación en la Figura 5.1, el valor del eje y saldría siempre por encima de su valor real debido al efecto de la gravedad. Para ello, se implementó una clase que eliminaba dicho efecto.

En la Figura 5.1 se pueden observar los tres ejes, siendo el eje rojo, el eje Z vertical, el afectado por la aceleración de la gravedad. Según la orientación del dispositivo SensorTag, si mira hacia arriba o hacia abajo, se ve que cambia la influencia de la gravedad. Hay que tener en cuenta como varía esta influencia para que nuestra aplicación pueda mostrar la representación sin esta influencia.

Para solucionar dicho problema es necesario conocer la orientación del sensor, por ello hemos de recibir también los datos del sensor del giroscopio y del magnetómetro para poder aplicar el algoritmo de Madgiwck. Además el giroscopio genera otras dificultades que desarrollaré mas adelante.

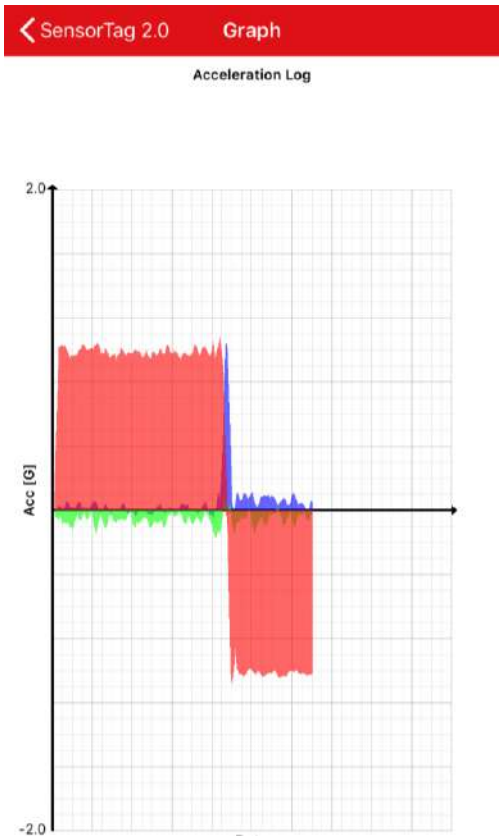

Figura 5.1: Gráfica aceleración en la aplicación SensorTag.

Así pues, el código resultante de solucionar el problema de la aceleración sería el siguiente:

```
public static double[] Gravitity(double[] q, double ace, int unit){
            double[] g = {0.0, 0, 0.0, 0.0};
           g[0]=2* (q[1] * q[3] - q[0] * q[2]);
           g[1]=2* (q[0] * q[1] - q[2] * q[3]);
           g[2]= q[0] * q[0] - q[1] * q[1] * q[2] * q[2]);

         double[] r = {ace[0] - g[0], ace[1] - g[1], ace[2] - g[2]};

         return r;
}
```

Figura 5.2: Código de la programación de la compensación de la gravedad.

En primer lugar, se calcula el valor de la gravedad en el sistema de referencia, calculando la orientación del sensor. Una vez conocemos dicho dato, se calcula la diferencia que produce la gravedad en la posición del cuaternión, dichos valores y operaciones se realizan mediante vectores. Posteriormente, se resta la aceleración de la gravedad: 9,8 $^{m}/_{s^2}$ a cada uno de los ejes y se devuelve este vector, que será el que se guardará en la tabla correspondiente y el que se representará para el usuario.

5.2 Datos del cuaternión.

Para poder resolver el problema de la gravedad, se requiere conocer la posición del sensor. Para poder conocer la posición del dispositivo SensorTag hemos de recibir dichos datos del giroscopio. Para poder recibir estos datos se usa como base la recepción de los datos del acelerómetro, sin embargo, la posición del dispositivo requiere de dos aspectos especiales: en primer lugar, la programación del algoritmo de Madgwick, explicado en la base teórica, y en segundo lugar, la resolución de un problema que surge debido al resultado del algoritmo de Madgwick ya que los cuaterniones obtenidos no se quedaban fijos sobre el eje z, vertical, si no que iba a variando.
A continuación, voy a resolver en dos subcapítulos diferentes la programación del algoritmo de Masgwick y, en el siguiente, la resolución de dicho problema, siendo la resolución del mismo mucho más complejo.

5.2.1 Programación del algoritmo de Madgwick.

El algoritmo de Madgwick lo necesitamos para calcular el valor de las unidades de medida conocidas como cuaterniones, también explicados con anterioridad, a modo de recordatorio, la definición de cuaternión es un número complejo que posee cuatro dimensiones y permite orientar un cuerpo en el espacio. He escogido este algoritmo por tener dicha funcionalidad y por tener como característica un coste computacional bajo, lo cual no disminuye el rendimiento del dispositivo ni produce retardos en la aplicación.

Como sabemos, este algoritmo se basa en el filtro de Kalman y se usará la versión MARG, también explicada. Se calcula que para actualizar el filtro se requieren alrededor de 380 operaciones aritméticas. Dado que nuestra frecuencia no es muy elevada, el filtro resulta mas efectivo todavía y, además, gracias a (RELLENAR CON LO DEL FIRMWARE) podemos modificar parámetros como la frecuencia o la ganancia del filtro para optimizar su uso, lo cual haremos en caso de ser necesario.

El algoritmo de Madgwick se implementa en el Android Studio mediante un constructor denominado MadgwickAHRS el cual require de tres valores: el periodo, que debe ser el mismo periodo con el que registren los datos del sensor anteriormente configurados. La ganancia del algoritmo, identificado con el nombre de ‘beta; está ganancia puede estar producida por diversos factores como errores de calibración, ruido u otros muchos y representa la variación del valor real al medir debido a diversos factores externos. Por último, el algoritmo requiere del parámetro del cuaternión de inicio, el cual se puede elegir según la aplicación.

Siguiendo las recomendaciones que se dan de dichos valores, para implementar el algoritmo Madgwick MARG se recomienda un valor de beta de 0,041 para un rendimiento óptimo, sin embargo, en nuestro caso, hemos tenido que aumentar dicho valor dado que la frecuenta es alta y, por tanto, necesitamos una convergencia rápida. Realizando diversas pruebas, decidimos fijar el valor de beta en 0.9, tardando alrededor de 7 segundos en lograr la convergencia. Como valor del cuaternión inicial, dejamos el que se recomienda para este caso: (1,0,0,0). A continuación, mostraré la parte del código más importante para el mismo, es decir, del código del constructor.

```
public MadgwickAHRS(double samplingPeriod, double beta, double[] quat) {
        this.samplingPeriod = samplingPeriod;
        this.beta = beta;
        this.quaternion = quat;
    }
```

Figura 5.3: Programación del algoritmo de Madgwick.

5.3 Estabilidad del eje Z.

En el dispositivo SensorTag que estamos utilizando se produce una inestabilidad en este eje. Se puede comprobar analizando los valores de los cuaterniones al dejar el dispositivo en los tres posibles ejes (yaw, pitch, roll).

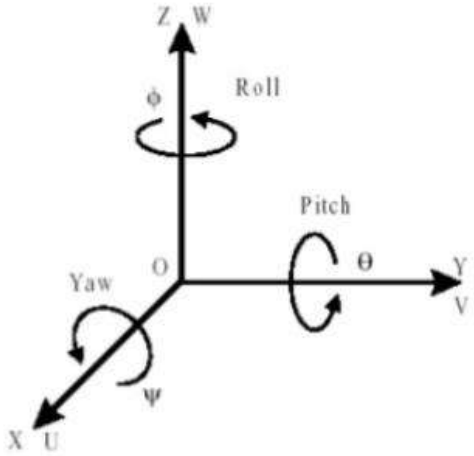

Figura 5.4: Representación teórica de los ángulos de navegación.

Al dejar el dispositivo quieto en los diferentes ángulos, los valores de los cuaterniones se quedaban quietos, excepto en el eje z, en el cual los valores de los cuaterniones varían hasta quedarse fijos.

No he intentado solucionar dicho problema, el cual hubiera intentado resolver mediante la compensación de la desviación que sufre fijándome en el tiempo que tarda en llegar a esos valores fijos y cuanto distan de los valores que debería tener. No he intentado solucionarlo debido a que, debido a la utilidad de la aplicación, esta desviación no iba a afectar en gran medida. No iba a afectar en gran medida dado que las medidas se toman siempre en movimiento y no se va a utilizar para analizar las posiciones que toma el usuario.

5.4 Conexiones Bluetooth.

La programación en Android para las conexiones de Bluetooth es un tanto especifica puesto que cuenta con unas clases de Android ya configuradas, una serie de métodos que se han de desarrollar y bastantes aspectos a tener en cuenta.

Intentaré desarrollar todo ello o, por lo menos, los que he considerado de mayor relevancia o dificultad a la hora de implementar el código, así como los conceptos más básicos. Así pues, para establecer una comunicación mediante Bluetooth, hay que realizar cuatro tareas: en primer lugar, configurar el bluetooth; en segundo lugar, buscar los dispositivos sincronizados,

posteriormente, conectar dichos dispositivos y, por último, la trasferencia de datos entre los dispositivos conectados.
Cabe aclarar que se trata de 'Bluetooth Low Energy' (BLE) , es decir, se caracteriza por proveer Bluetooth de bajo consumo de energía. Solamente se comunica con dispositivos que poseen también 'Bluetooth LowEnergy', lo cuales poseen especificaciones bastante estrictas.
Previamente a desarrollar estas cuatro fases, voy a describir las clases e interfaces más importantes que hay disponibles en el paquete android.bluetooth y que he utilizado para la realización del proyecto:
-BluetoothAdapter: Permite visualizar otros dispositivos Bluetooth, consultar una lista con todos los disposivos que se encuentran sincronizados o crear, a partir de una dirección MAC, un instancia de BluetoothDevice.
-BluetoothDevice: Permite acceder a información sobre un dispositivo como su dirección, nombre o estado de conexión.
-BluetoothManager: Nivel superior que permite realizar una instancia de BluetoothDevice.

Otros conceptos que están mas relacionados con el Bluetooth pero de baja energía, que es el que vamos a emplear, serían:
-GATT (Generic Attibute Profile): Especificación para enviar y recibir atributos a través de un enlance de BLE. Todas las aplicaciones de 'Bluetooth Low Enery' se basan en el GATT.
-Characteristic: Una característica posee un determinado valor y un número concreto de descriptores que describen el valor de la característica. Es una especie de tipo, parecido a una clase.
-Descriptor: Definen los atributos que describen el valor de una característica, como por ejemplo, el rango de dicha característica o la unidad de medida especifica.
-Service: Es una colección de características que realizan un servicio determinado.

En primer lugar, cuando en una aplicación se va a implementar conexión Bluetooth, hay que tener en cuenta que se ha de rellenar el archivo AndroidManifest.xml con las siguientes líneas:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Figura 5.5: Programación de los permisos para Bluetooth.

El permiso de 'BLUETOOTH' es necesario para cualquier comunicación Bluetooth. El permiso de 'ADMIN' es necesario en el caso de que tu aplicación inicie la detección de los dispositivos y el de 'COARSE_LOCATION', es decir, el de ubicación, permite activar el Bluetooth del dispositivo que posee la app.

5.4.1 *Configuración del Bluetooth*

Mediante el uso del BluetoothAdapter, se ha de comprobar que el Bluetooth sea compatible con el dispositivo y de que esté habilitado. En el caso de que no sea compatible la actividad deberá finalizar. En el caso de que sea compatible se procederá a comprobar si esta habilitado o no, en caso negativo, podremos activarlo desde la aplicación. Para todas estás posibilidades, al usuario le saldrá una pantalla en la que concederá o denegará el permiso.
En mi proyecto, esta configuración del Bluetooth se maneja desde una activity llamada Captura.class que contempla todas las posibilidades:

```
private void requestPermissions(){
    ActivityCompat.requestPermissions( activity: this, new String[] {Manifest.permission.ACCESS_COARSE_LOCATION}, REQUEST_ACCESS_COARSE_LOCATION
}

private boolean hasCoarseLocationPermission(){
    return ContextCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION)
        == PackageManager.PERMISSION_GRANTED;
}

private boolean isBluetoothEnable(){
    if (bluetoothAdapter == null){
        Toast.makeText( context: this, text: "Bluetooth no soportado", Toast.LENGTH_SHORT ).show();
        finish();
        return false;
    }

    return bluetoothAdapter.isEnabled();
}
```

Figura 5.6: Código de como se configura el Bluetooth en Android Studio.

Se observa como como se comprueba si es compatible el Bluetooth con el dispositivo, en caso de serlo pasa a comprobar si está activo, si no lo está, lo activa.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult( requestCode, resultCode, data );

    if (requestCode==REQUEST_ENABLE_BLUETOOTH && resultCode == RESULT_OK) {
        manageBluetoothState();
    } else {
        Toast.makeText( context: this, text: "Activa el bluetooth",Toast.LENGTH_SHORT ).show();
        scanningBtn.setEnabled(true);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult( requestCode, permissions, grantResults );

    if (requestCode == REQUEST_ACCESS_COARSE_LOCATION) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            manageBluetoothState();
        } else {
            Toast.makeText( context: this, text: "Son necesarios los permisos", Toast.LENGTH_SHORT).show();
            scanningBtn.setEnabled(true);
        }
    }
}
```

Figura 5.7: Código de como se configura el Bluetooth en Android Studio.

5.4.2 Búsqueda de dispositivos sincronizados.

La búsqueda de los dispositivos sincronizados se realiza mediante el BluetoothAdapter. Este realiza un escañero de todos los dispositivos 'detectables', los responderán a la petición de visibilidad aportando información sobre su nombre, clase o su dirección MAC; dichos datos son los necesarios para poder realizar la conexión del dispositivo.

Dos dispositivos están sincronizados cuando se reconocen mutuamente, además, han de tener una clave compartida que sirve para la autenticación y deben poder establecer entre ellos una conexión segura. Es diferente de estar conectados, puesto que, para estar conectados, los dispositivos deben compartir un canal RFCOMM y deben poder intercambiar datos entre ellos. Para que dos dispositivos puedan conectarse, deben estar, en primer lugar sincronizados.

Poseemos dos clases, la anteriormente mencionada: Captura.class, en la cual capta el estado de los dispositivos:

```
private final BroadcastReceiver devicesFoundReceiver = (context, intent) → {
        String action = intent.getAction();

        switch (action) {
            case BluetoothDevice.ACTION_FOUND:
                final BluetoothDevice device =
                    intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                listAdapter.add(device.getName() + "\n" + device.getAddress());
                break;
            case BluetoothAdapter.ACTION_DISCOVERY_FINISHED:
                scanningBtn.setText("Buscar dispositivos");
                scanningBtn.setEnabled(true);
                break;
            case BluetoothAdapter.ACTION_DISCOVERY_STARTED:
                scanningBtn.setText("Buscando...");
                break;
        }
};
```

Figura 5.8: Código de como se realiza la búsqueda de los dispositivos sincroinizados.

Y la clase CapturaBLE.class en la cual se ejecuta la selección del dispositivo, y el establecimiento de conexión con el dispositivo escogido, así como el intercambio de los datos.

```
private void startScan() {
  bluetoothAdapter.getBluetoothLeScanner().startScan(new ScanCallback() {
    @Override public void onScanResult(int callbackType, ScanResult result) {
      super.onScanResult(callbackType, result);
      BluetoothDevice device = result.getDevice();
      listAdapter.add(device.getName() + "\n" + device.getAddress());

    }

    @Override public void onBatchScanResults(List<ScanResult> results) {
      super.onBatchScanResults(results);
      scanningBtn.setEnabled(true);
    }

    @Override public void onScanFailed(int errorCode) {
      super.onScanFailed(errorCode);
      scanningBtn.setEnabled(true);
    }
  });
}
```

Figura 5.9: Código de como se muestra la lista de dispositivos sincronizados.

5.4.3 Conectar al GATT server.

Para conectarse a un dispositivo BLE se será el método conectGatt() y se ha de emplear también el método BluetoothGattCallback, que sirve para recibir datos del cliente como su estado. El método conectGatt() nos permitirá conctarnos al servidor del dispositivo. Nuestra aplicación sería el GATT cliente en este caso.

```
52      //crear lista que saldra por pantalla
53      listAdapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1);
54      devicesList.setAdapter(listAdapter);
55      devicesList.setOnItemClickListener((adapterView, view, position, l) → {
58          String item = listAdapter.getItem(position);
59          String name = getDeviceName(item);
60          String address = getDeviceAddress(item);
61          Toast.makeText( context: CapturaBLE.this, String.format("Conectando a %s", name), Toast.LENGTH_SHORT)
62              .show();
63          connectDevice(address);
64      });
66
67      //get bluetoothAdapter
68      bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
69      bluetoothAdapter = bluetoothManager.getAdapter();
70
71      scanningBtn.setOnClickListener((view) → { manageBluetoothState(); });
76  }
77
```

Figura 5.10: Código de cómo se selecciona el dispositivo al que se quiere conectar

En la Figura 5.10 podemos observar como se conecta al dispositivo qe el usuario escoge seleccionándolo a partir de una listAdapter en la que van apareciendo los distintos dispositivos sincronizados que se encuentran. Al presionar sobre uno de los dispositivos, se llama a la clase connectDevice() que conectará los mismos y quedará prepara para la recepción de datos.

```
180 private void connectDevice(String address) {
181
182     BluetoothDevice device = bluetoothAdapter.getRemoteDevice(address);
183     device.connectGatt( context: this,  autoConnect: true, new BluetoothGattCallback() {
184       @Override public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
185         super.onConnectionStateChange(gatt, status, newState);
186         if (newState == BluetoothProfile.STATE_CONNECTED) {
187           Log.d( tag: "BLE ---> ",  msg: "Conectado!");
188
189           gatt.discoverServices();
190         }
191       }
```

Figura 5.11: Código del método connectDevice().

En la Figura 5.11 podemos observar cómo se realiza la conexión/enlace al dispositivo seleccionado gracias a las clases de onConnectionStateChange o connectGatt y mediante el uso del getRemoteDevice(address), en donde la variable ‘address’ guarda la dirección del dispositivo seleccionado por el usuario para establecer la conexión y la recepción de datos.

Los estados están definidos por AndroidStudio como podemos ver en la Figura 5.12. De esta clase, una vez se ha establecido la conexión, se prepara para la recepción de datos.

int	HEARING_AID Hearing Aid Device
int	HID_DEVICE HID Device
int	SAP
int	STATE_CONNECTED The profile is in connected state
int	STATE_CONNECTING The profile is in connecting state
int	STATE_DISCONNECTED The profile is in disconnected state
int	STATE_DISCONNECTING The profile is in disconnecting state

Figura 5.12: Posibles estados del dispositivo al que deseamos conectarnos mediante BLE.

5.4.4 Recibir datos del dispositivo.

Para la recepción de datos, hay que tener claro el concepto de UUID y como conectarse. UUID (Universal Unique ID) es un identificador universal de los sensores que informa de que servicio transmite.

Para averiguar los UUID que nos interesaban usé la aplicación para iOS llamada 'nrf Connect' el cual, conectándose mediante Bluetooth a nuestro dispositivo, informa de las UUID de los distintos sensores del mismo.

CAPTURA DE LA APP

Guardamos las UUID que vamos a usar en variables, en nuestro caso necesitaremos la del acelerómetro y la del giroscopio, tanto de los servicios como de los datos de los mismos. Para conectarnos a cada uno y y recibir los datos usamos la clase gatt.getService(UUID_SERVICE) y gatt.getCharacteristic(UUID_CHARACTERISTIC).

Para guardar los valores obtenidos y poder utilizarnos, debíamos crear vectores y guardar en cada posición del vector cada valor se han empleado las siguientes instrucciones siendo el código:

BluetoothGattDescriptor =
characteristic.getDescriptor(UUID_CONFIGURATION_CHARACTERISTIC);

descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);

gatt.writeDescriptor(descriptor);

Posteriormente, se ha manipulado los datos para solucionar el problema de gravedad como se ha visto anteriormente. Por último, se han representado dichos valores, explicado posteriormente.

5.5 Modificar el firmware.

Una de las ventajas de usar el dispositivo SensorTag es la posibilidad de realizar modificaciones del firmware del mismo gracias al uso del Code Composer Studio de Texas Studio, el cual permite editar código, el depurador o el compilador de los distintos dispositivos del frabricante.

El objetivo de modificar el firmware del dispositivo seria modificar la frecuenta para lograr obtener mayor cantidad de muestras por segundo de los sensores de movimiento que vamos a utilizar: el giroscopio y el acelerómetro.

El dispositivo posee unos sensores que poseen unas frecuentas bastante buenas, sin embargo, Texas Instrument limita la frecuencia máxima de estos sensores a 1kHz, es decir, se capta una muestra cada 100ms. En el caso de que analizaremos el movimiento del pie del usuario al correr, sería muy conveniente modificar estar frecuenta para que fuera más elevada; sin embargo, dado que nuestra aplicación se limita a obtener la medida del acelerómetro, con usar solamente la máxima frecuencia que permite este dispositivo (0.01kHz) nos salen unos resultados y unas gráficas que resultan suficientes si entendemos bien cual es el objetivo y la finalidad de la aplicación, la cual se explicará convenientemente en el Anexo del proyecto.

5.6 Representación de la aceleración.

Para obtener las gráficas resultantes de los datos recibidos por el dispositivo, se han tratado estos datos recibidos para lograr la compensación de la gravedad, lo cual ha exigido tratar también los datos de posición recibidos por el sensor de movimiento.
Usaremos un fragment llamado 'fragment_accelerometer' en el cual, gracias al lineChart se muestra de forma lineal de los datos de los tres ejes. Los datos recibidos son guardados en una tabla de la base de datos, creada para almacenar la información recibida de los sensores, y manipulados para eliminar la gravedad. Se mostrarán directamente sin la gravedad, sin posibilidad de recibirlos con gravedad.

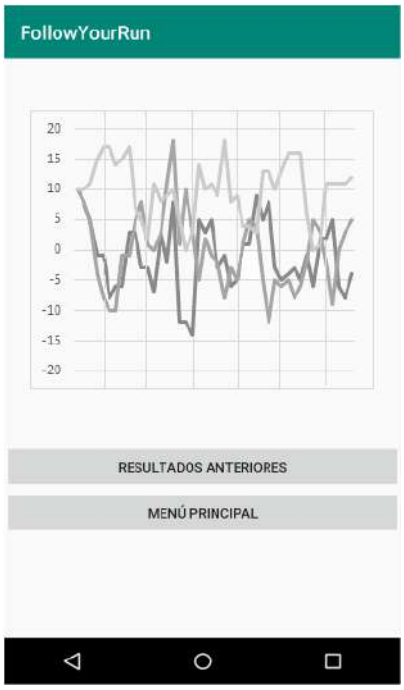

Figura 5.13: Resultados obtenidos por la aplicación.

Para mostrar los resultados anteriores, opción que tiene cada usuario dentro de la aplicación, se visualizará la Id de cada carrera en una list view, de modo que, cuando presionamos sobre la deseada, mostrará la gráfica correspondiente de dicha captura:

Figura 5.14: Listado de capturas realizadas.

6. Conclusiones.

Una vez realizada y finalizada la aplicación, habiendo realizado las pruebas necesarias para comprobar su correcto funcionamiento y su utilidad para el fin para el cual ha sido diseñada, procedo a realizar un resumen final del trabajo realizado. Posteriormente explicaré las condiciones bajo las que se ha de realizado el trabajo, presupuesto, conocimientos y los resultados obtenidos a nivel personas. Por último, daré algunas ideas sobre cómo este proyecto podría mejorarse en un futuro.

6.1 Resumen del trabajo.

Cada día más y más personas se suman a la práctica del 'running' debido al auge del deporte y a la promoción de numerosas carreras en todos los municipios con el fin de aumentar la afición de todos por el deporte. Como todos los deportes, tiene numerosos beneficios, sin embargo, una mala práctica, sin preparación ni conocimiento, puede provocar numerosas lesiones, las más comunes en la rodilla.

Con la finalidad de lograr que aquellos que practican este deporte puedan observar si lo hacen correctamente, o, de que aquellos que han sufrido alguna lesión puedan comprobar su mejoría, he desarrollado una aplicación que mida la aceleración con la que el usuario mide la aceleración con la que golpea el suelo, la cual afecta directamente a su rodilla, en mayor o menos medida según su forma de correr.

Para ello, hemos usado el dispositivo SensorTag de Texas Instrument, que posee diversos sensores de movimientos muy útiles para el estudio del movimiento en la realización de deportes. Además, Texas Instrument permite la modificación del firmware para adapatarlo a distintas aplicaciones. También se ha usado Android Studio para el desarrollo de la aplicación, programando en lenguaje Java.

A lo largo de la realización de la aplicación han surgido diversos problemas y dilemas que se han tenido que ir solucionado buscando la manera más practica de realizar el trabajo, quedan, por tanto, numerosas posibles mejoras. Entre los problemas y dilemas destacaría el de la compensación de la gravedad, ya que el sensor de movimiento del dispositivo proporciona datos afectados por la gravedad, lo cual nos hizo tener que identificar la posición del dispositivo mediante los cuaterniones y el algoritmo de Madgwick, para poder restar la aceleración de la gravedad en el eje afectado por el mismo, según la posición.

Para poder analizar las gráficas, se ha hablado contado con dos fisios, Javier Bonastre y Pilar Sanz, y un deportista de triatlón, Patxo Grau, con el cual he grabado las gráficas para llegar a concluir que la medida de carreras realizas con una correcta pisada tienen una gráfica similar a la Figura 5.17. Para poder deducir eso, podemos observar en la gráfica 6.1 y Figura 5.17 son similares y en cambio, en la Figura 6.2 se observa cómo, con una mala pisada, se obtiene una grafica que nada tiene que ver.

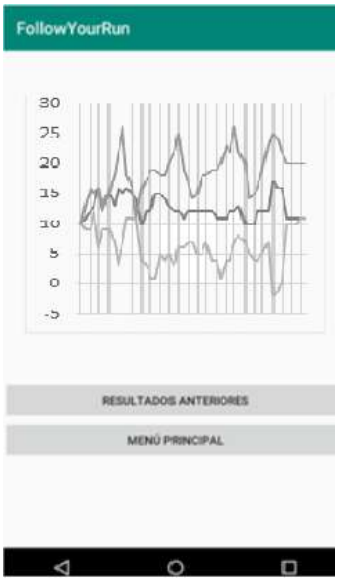

Figura 6.1: Otro ejemplo de captura.

Finalmente, se ha obtenido una aplicación que representa la aceleración a lo largo un tiempo limitado, pensado para medir la aceleración del usuario en un periodo de tiempo limitado.

6.2 Condiciones y resultados.

Para que los usuarios transporten el sensor, se ha diseñado la siguiente prenda que posee el sensor en su interior, y se colocaría en el tobillo, parte en la cual se van a tomar las medidas. Quedaría tal y como se muestra en la Figura 6.1.

Es muy importante colocárselo como se indicará posteriormente en el Apéndice, ya que el deporte se realizará correctamente o no según qué parte del pie se apoye primero, lo cual se puede observar a partir de las gráficas de la aceleración, pues al llevarlo en la parte trasera, los valores al pisar en primer lugar con el talón serán más elevados que al pisar primero con la punta.

Los resultados de una persona sana, que corre correctamente y no tiene ninguna lesión ni problema al apoyar el pie tiene la forma mostrada en la gráfica Figura 5.14.

El resultado de una persona lesionada o con peligro de lesión sería similar a la Figura 6.1. Muchas personas corren de forma no correcta sin saberlo y les mostraría una gráfica como la siguiente:

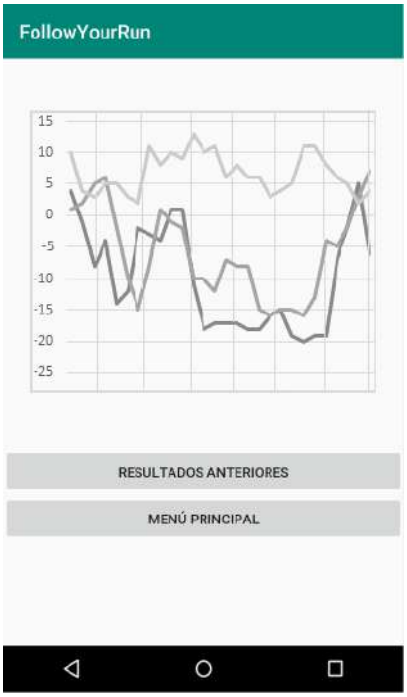

Figura 6.2: Resultado de una persona con peligro de lesionarse.

Otra utilidad es la capacidad de medir la corrección de la posición al pisar, por lo ya comentado de correr apoyando punta o talón. Lo correcto sería apoyar primero la punta, ya que, sobre el talón, todo el empuje del suelo al correr recae sobre nuestra rodilla. Una persona que corre apoyando primero la punta, lo cual tiene unas altas probabilidades de lesión de rodilla puesto que todo el empuje del suelo llega directamente a la rodilla, daría una gráfica similar a está:

La diferencia entre ambas gráficas, tal y como se puede observar, estaría en primer lugar en la diferencia que se produce en los valores de los ejes de las distintas gráficas, puesto que, cuando se realiza el ejercicio correctamente no se produce tanta diferencia entre los valores de los ejes.

Otra diferencia son los valores inferiores. La valores que se muestran están normalizados en una escala de 9.8, de ahí los valores que se muestran. Cuando hay valores tan bajos como los que se muestran en la gráfica 5.15 se debe a que no se está apoyando correctamente el pie al correr.

6.3 Posibles mejoras futuras

En primer lugar, darle a la aplicación más funcionalidades relacionadas con el análisis de los datos. También se podía añadir la funcionalidad de una gráfica de evolución, representando la aceleración media de cada carrera para ver la evolución o de otras posibles medidas que se hayan obtenido y/o analizado.

La más sencilla sería, con un estudio un poco más amplio, el de generalizar una gráfica patrón con la que se pudieran comparar los diferentes resultados obtenidos en cada captura. Esto es lo que más cerca se ah estado de lograr, aunque no se puede garantizar debido a la falta de comprobaciones.

Otra posible mejora podría ser, dada la gran cantidad de sensores que posee el SensorTag, ir sumando funcionalidad: tomar medidas de otros sensores, como la orientación, con cuyos datos se podría obtener más información y aplicarlo a más tipos de deportes y de ejercicios de

rehabilitación. Cuanta más información de cada sensor, siempre que sepamos la posible aplicación de cada uno en el campo del deporte y la rehabilitación, se podría mejorar la aplicación haciéndola cada vez más completa y útil.

Por ejemplo, la funcionalidad que diga qué tipo de corredor eres según como efectúas las zancadas al correr. Analizar la forma en la que se realizan las zancadas sería muy útil puesto que, como hemos visto anteriormente, este es uno de los principales causantes de lesiones; la mejor opción sería ser corredor pronador, apoyando el exterior del pie. Esto se podría analizar, pero hoy por hoy, a partir de los resultados recibidos, tan solo podemos interpretar y distinguir entre retropié o mesopié.

También, la más extendida pero también la más usada sería calcular la velocidad mediante los datos recibidos, se podría estimar la velocidad a la que va el usuario y la distancia recorrida, así como incluir Google maps en la aplicación para visualizar el recorrido.

Es decir, mediante la programación de la aplicación de Android Studio se podría hacer una aplicación bastante completa que no solo midiera carreras si no que tuviera como objetivo fundamental proporcionar al usuario la información necesaria sobre como esta realizando la actividad para evitar posibles lesiones o para recuperarse de ella; es decir, que la gente pueda aficionarse a este deporte y practicarlo de forma segura.

6.4 Estudio y conclusión final.

Para finalizar el trabajo, cabe destacar como conclusión la finalidad última de la aplicación. En un primer lugar, como hemos visto, la finalidad era desarrollar una aplicación que serviría a todos y a los fisioterapeutas a mejorar la forma de correr, a reducir el riesgo de lesión e, incluso, a curar lesiones.

Para lograr dicha finalidad y saber como enfocar el análisis de los datos, hablé con dos fisios: Javier Bonastre, posee una clínica fisiotepauta llamada Vitrum, y Pilar Sanz. A ambos les conté el proyecto y me dieron diferentes ideas que les podrían ser útil. Además, para tener el punto de vista de un deportista y paciente de los mismos, hablé con el deportista Patxo Grau, el cual entrena para realizar triatlones y el iron man, competiciones que ya ha realizado.

Con todo ello, había numerosas posibles aplicaciones posibles. Me dispuse a la que consideré más sencilla a la par que útil para un aficionado cualquiera a este deporte: ser capaz de mejorar y conocer la forma en la que corremos para evitar lesionarnos, puesto que un elevado porcentaje de personas aficionas a correr sufren lesiones de rodilla, debido a la fuerza que ejerce el suelo sobre la misma cuando no pisamos correctamente.

Junto con Patxo, he realizamos muchas pruebas, me ha ayudado a realizar pruebas corriendo correctamente y corriendo con peligro de lesión, lo cual me ha permitido comparar las graficas y ver las diferencias. De ahí que concluya con que las gráficas correctas tienen una forma mas parecida que cuando no corremos correctamente. De ahí que pueda concluir, pero dado que no esta probado con médicos, se puede estimar lo concluido, sin embargo, objetivamente, mi aplicación solamente es capaz de producir graficas de la aceleración al correr, con posibles aplicaciones futuras, que serian fruto de estudios más certificados.

Apéndice A: Manual de usuario.

En este apéndice, se va a explicar paso a paso como se ha de usar tanto el dispositivo como la aplicación, viendo las distintas funcionalidades de la misma.

A.1. Colocación del dispositivo.

Es muy importante colocarse correctamente el dispositivo ya que de ello depende las medidas que se van a representar, lo cual es el objetivo de la aplicación.

Así pues, se ha diseñado la siguiente banda con velcro, adaptable a todos, y que sjera el dispositivo SensorTag para la medición:

Es muy importante que es dispositivo se cómo de la siguiente manera, en la siguiente orientación para que las medidas se tomen correctamente según como se van a explicar los resultados.

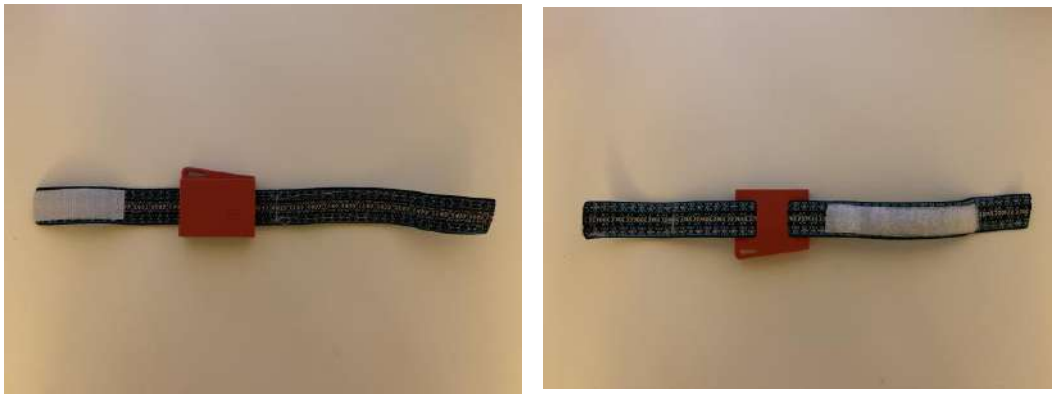

Figura A.1: Diseño por delante y por detrás.

Figura A.2: Forma de colocación.

FIGURA DEL DISPOSITIVO EN LA PIERNA

A.2. Pantalla principal

Al iniciar la aplicación, llamada ‘FollowYourRun’, se mostrara la pantalla de inicio con dos opciones. Si es la primera vez que se usa la aplicación, dado que no hay ningún usuario registrado, y, por tanto, no se poda iniciar sesión con ningún usuario, se seleccionará la opción de ‘REGISTRAR USUARIO’.

En caso de ser un usuario ya creado y que, incluso, ya ha realizado capturas de carreras, se seleccionará la opción de 'INICIAR SESIÓN', puesto que su usuario ya está registrado en la base de datos.

Figura A.3: Pantalla de inicio de la aplicación.

A.3. Crear usuario

La pantalla para registrar un usuario es la siguiente. Está programado para que sea obligatorio rellenar los campos de Nombre y Apellidos, ya que son los necesarios para iniciar sesión.

Una vez rellenados los campos se presiona Guarda usuario y la aplicación nos llevaría directamente al menú principal, se da por hecho que se inicia sesión.

Se recomienda poner un nombre y un solo apellido, dado que se inicia sesión con esos datos, es importante no liarse y no poner nombres o apellidos extraños que no vayamos a recordar.

Figura A.4: Pantalla de 'Registrar usuario'.

A.4. Iniciar sesión

Para iniciar sesión habrá que llenar el nombre y apellidos con los que el usuario se ha registrado anteriormente. En el caso de que el usuario deje vacío alguno de los campos, saldrá un aviso en la pantalla avisándole de que debe rellenar todos los campos.

Figura A.5: Pantalla de inicio de la aplicación.

A.5. Iniciar captura y uso del Bluetooth.

Para establecer la conexión Bluetooth, están previstos todos los casos posibles:

-En caso de incompatibilidad de Bluetooth, saldrá un cartelito avisando y la aplicación finalizará.

-En caso de que el Android tena el Bluetooth desactivado, solicitara los permisos para activarlo.

-En caso de no tener los permisos, los solicitará mediante solicitud de ubicación.

Figura A.6: Pantalla de permisos de activación de Bluetooth en el Android.

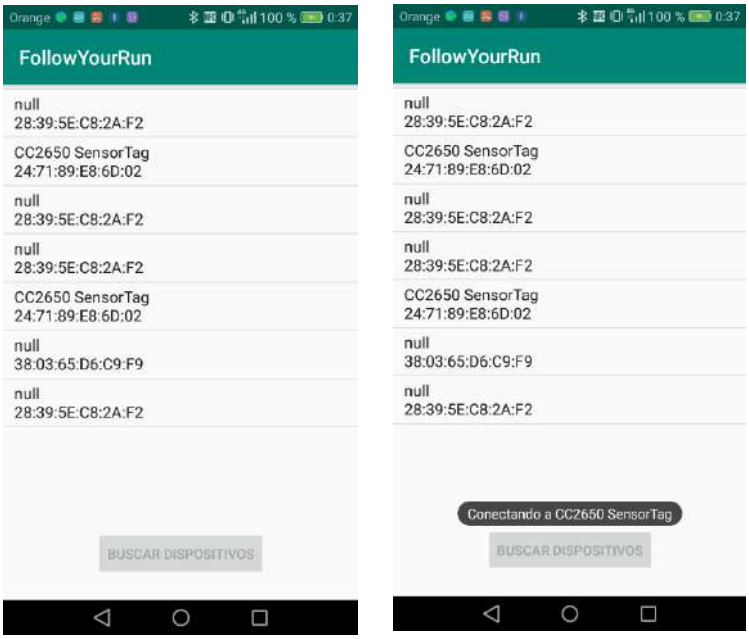

Figura A.7: Detección de dispositivos Bluetooth.

A.6 Visualizar resultados

Una vez terminada la captura de las muestras, tomada durante 30 segundos, aparecerán los resultados en gráficas. Según la interpretación de las gráficas, podemos interpretar si el usuario está realizando correctamente el ejercicio o habrá que analizar la forma en la que lo realiza por no se correcta.

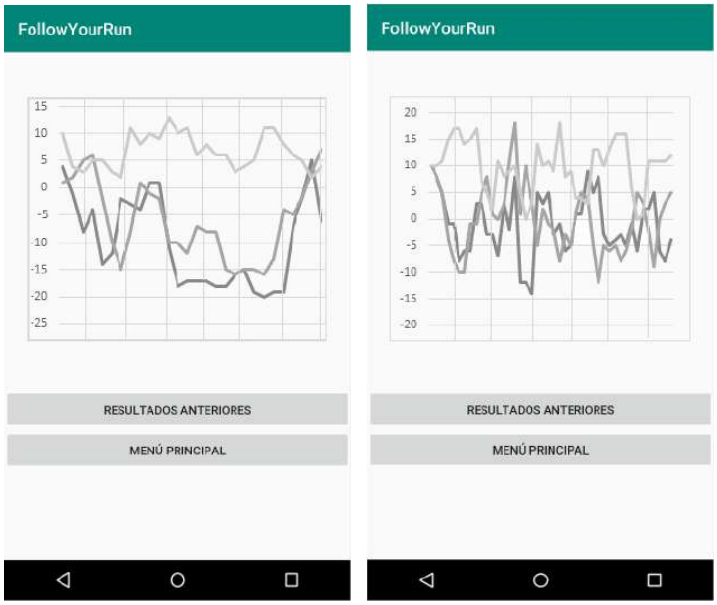

Figura A.8: Gráficas resultado.

A.6 Análisis de los resultados

Tras realizar numerosas pruebas, se llega a una gráfica estándar que define la buena realización de esta actividad, el ‘running’. Por tanto, al realizar el ejercicio, cuanto más parecido sea el resultado obtenido a la gráfica estándar, más correctamente estarás realizado el ejercicio. Si por lo contrario, los valores que obtienes son muy diferentes o se obtienen valores muy bajos de la aceleración, se debe consultar a algún fisioterapeuta para que analice como estas realizando el ejercicio y pueda corregirte. Se deberá a una mala pisada con el pie al correr.

La gráfica estándar la podemos observar en la gráfica derecha de la figura A.5.

Bibliografía

[1] Información sobre los dispositivos SensorTag. Consultado en: http://dev.ti.com/tirex/#/?link=Development%20Tools%2FKits%20and%20Boards%2FCC1350%20SensorTag.

[2] Información sobre el dispositivo Mikroe . Consultado en: https://www.mikroe.com/hexiwear.

[3] Información sobre el dispositivo STEVAL WESU1. Consultado en: https://www.st.com/en/evaluation-tools/steval-wesu1.html#overview.

[4] Sensor de temperatura. Consultado en: https://media.digikey.com/pdf/Data%20Sheets/Texas%20Instruments%20PDFs/TMP007.pdf.

[5] Sensor de presión y altímetro. Consultado en:

https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999334039&cPath=1343

[6] Sensor digital de humedad. Consultado en: http://www.ti.com/lit/ds/symlink/hdc1000.pdf

[7] Sensor digital de luz. Consultado en: http://www.ti.com/lit/ds/symlink/opt3001.pdf

[8] Sensor de movimiento de 9 ejes MPU-520. Consultado en: http://www.ti.com/lit/ds/symlink/opt3001.pdf

[9] Rafael Molina Sorian. 'Bases del filtro de Kalman'. Consultado en: 0 https://decsai.ugr.es › vip › doctorado › pvd

[10] Algoritmo de Madagwick en java para Android Studio. Consultado en: https://github.com/PaulStoffregen/MadgwickAHRS

[11] Teoría sobre cuaterniones. Consultado en: http://www.wag.caltech.edu/home/ajaramil/libro_robotica/transformaciones_espaciales.pdf

[12] Teoria sobre modificar el firmware. Consultado en: http://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html

[13] Programación de bases de datos SQLite en Android Studio. Consultado en: https://developer.android.com/guide/topics/connectivity/bluetooth-le#java

[14] Prgramación conexión BLE en Android Studio. Consultado en:

https://developer.android.com/guide/topics/connectivity/bluetooth-le#java