Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Prototipo de dispositivo de medida de rendimiento en *running* basado en acelerómetro triaxial y comunicación a dispositivo móvil

Trabajo final de grado

Grado en Ingeniería Electrónica Industrial y Automática

Autor: Melià de la Asunción, Sergi

Tutor: Coll Arnau, Salvador

Prototipo de dispositivo de medida de rendimiento en *running* basado en acelerómetro triaxial y comunicación a dispositivo móvil

Resumen

Debido al creciente número de practicantes del deporte conocido de forma vulgar como '*running'*, se ha abierto un gran mercado para las aplicaciones y dispositivos relacionados con la monitorización de esta actividad. La aplicación desarrollada mediante Android, establecerá comunicación con los sensores alojados en el 'CC2541 SensorTag' para estudiar su viabilidad como posible puente para la fabricación de posteriores sistemas de medida de rendimiento.

Palabras clave: Android, running, dispositivos móviles, bluetooth, BLE, bluetooth low energy, MySQL, MySQLite, sensorTag, sensortag, CC2541

Índice

Tabla de contenidos

Figuras

Tablas

Ecuaciones

1. Introducción

1.1. Entorno

El número de practicantes del running como deporte incrementa cada vez más; esto ha abierto la puerta a la creación de dispositivos y/o aplicaciones que ayudan a medir el rendimiento del corredor.

No obstante, solo se monitorizan cosas triviales como los pasos dados, distancia recorrida o calorías quemadas. Si bien, dichas mediciones suponen una ayuda para el practicante de dicho deporte no ofrecen datos sobre su estado físico particular.

1.2. Objetivos

El primer objetivo de este trabajo de fin de grado es el desarrollo y la creación de una aplicación Android que mida el rendimiento del usuario como corredor mediante la conexión a un dispositivo externo que el corredor lleve consigo mientras realiza la actividad.

El segundo y último objetivo es la validación como dispositivo externo del CC2541 Sensortag, testear el hardware que lleva incorporado y comprobar que es posible usarlo en un futuro junto a la aplicación.

1.3. Descripción del documento

Este documento está dividido en nueve apartados y un anexo. A continuación, se pasa a enumerarlos y a hacer una breve descripción de ellos:

- **Introducción:** se hace un breve planteamiento de la problemática que este proyecto ha de resolver, seguido de ellos, se presentan los objetivos a los que se quiere llegar. Por último, se encuentra una breve descripción del documento.

- **Análisis del *running* en España:** contiene una breve introducción al aparatado seguida de un análisis del crecimiento de este deporte en España. Se indican cuáles son las lesiones más comunes que se sufren al practicar este deporte y cuales son registradas por la aplicación. Después se hace un análisis de mercado para saber qué características tiene que poseer el dispositivo externo. Por último, se explica el tipo de conexión inalámbrica que se va a usar y el

entorno de programación elegido para el desarrollo de la aplicación.

- **Entorno de desarrollo:** apartado donde se explica en mayor detalle el porqué de la elección del entorno de desarrollo. De forma posterior, se pasa a explicar cómo tiene que estructurarse la aplicación en este entorno de desarrollo.

- **Especificaciones de requisitos:** utilizando el modelo IEEE830, se explica con detalle los diferentes requisitos que ha de cumplir la aplicación.

- **Diseño:** mediante los modelos estandarizados de entidad-relación y UML se explica la implementación de los requisitos en la aplicación.

- **Implementación:** en este apartado se comenta en profundidad como se ha construido el código para dar forma a la aplicación, mostrando pequeños fragmentos de código cuando es necesario.

- **Validación del dispositivo externo:** contiene todo el segundo objetivo de este trabajo. Se comprueba que el dispositivo CC2541 SensorTag puede usarse como dispositivo externo mediante la valoración de sus características, conectividad y testeo de *hardware*.

- **Conclusiones:** breve resumen del trabajo realizado y la descripción de posibles mejoras para la aplicación y el dispositivo externo.

- **Bibliografía:** apartado donde se muestra toda la información consultada para la elaboración del *software*, del *hardware* y de la memoria.

- **Anexo:** este apartado contiene una guía de usuario simple que intenta satisfacer la necesidad de conocimientos de un nuevo usuario de la aplicación.

2. Análisis del *running* en España

2.1. Introducción

Debido a varios factores, el *running* como deporte ha visto incrementar su número de participantes exponencialmente en poco tiempo. En este apartado se realizará un análisis del porque este crecimiento y se detallarán las lesiones más comunes de este deporte.

2.2. Crecimiento del *running*

El crecimiento de este deporte se puede separar en varios factores. El primero de ellos es la crisis económica que ha visto España en estos últimos años. Debido a la ingente cantidad de parados, la capacidad de adquisición del español medio fue mermando cada vez más y su tiempo libre, por otra parte, se vio incrementado.

Correr no cuesta poco o nada más que un buen par de zapatillas; así que no es de extrañar que la gente acabase sustituyendo las altas tarifas de los gimnasios o los gastos de equipamiento de otros deportes por el *running*.

Aparte del poco gasto que supone, correr es una disciplina fácil y accesible para todo aquel que esté dispuesto a realizarla. Pese a todo, requiere de un mínimo entrenamiento y no es aconsejable que nadie corra un maratón para comenzar en este deporte. Pero es esta simple logística lo que ha llevado a que el *running* ganase adeptos. Además, en poco tiempo el corredor medio siente que progresa rápidamente en este deporte y eso les lleva a seguir practicándolo, buscando mejores resultados.

El segundo factor que ha afectado al *running* es el 'boom' social. Hoy en día, con el fácil y rápido acceso a las redes sociales como Twitter y Facebook, la gente puede compartir sus progresos en este deporte de forma sencilla y rápida. No sólo eso, si no la creación de una increíble cantidad de blogs relacionados con este deporte y de foros no han hecho sino más por incrementar el número de corredores. Muchos comienzan a correr por probar algo que 'todo el mundo hace' y si bien, no todos terminan enganchados a practicarlo, se dan cuenta de su sencillez y de lo fácil que es practicarlo.

El tercer y último factor se puede describir como la propia 'ropa' de los corredores. Está bien visto parecer que corres, te hace parecer una persona saludable y constante. Así que ha nacido toda una colección de ropa solo para *runners*: zapatillas de *running*, ropa de *running*, pulseras de *running*… Ver toda la cantidad de accesorios y ropa creados solo para este deporte, atrae gente. Hace parecer el *running* como un deporte con tendencia entre la gente y hace que quieras unirte a él. Además, la propia gente llevando dicha ropa, están haciendo

publicidad del deporte, haciendo ver por las ciudades y pueblos que hay gente practicándolo y que cada vez son más.

2.3. Lesiones más comunes

Para el desarrollo de la aplicación tendremos en cuenta las lesiones que de forma más común pueden afectar a los corredores. De forma más concreta, a continuación, se detallará las diferentes pisadas realizadas por el usuario.

Las pisadas en carrera o como debe apoyarse el pie mientras se corre ha sido un tema de discusión bastante importante en el *running*. Se pueden discernir tres tipos diferentes de apoyo en las pisadas:

- **Apoyo con el retropié:** dado cuando el contacto inicial del pie contra el suelo se produce con el talón o el tercio posterior de la planta del pie.
- **Apoyo con la parte media del pie:** dado cuando el talón y la cabeza del primer metatarsiano contactan casi simultáneamente.
- **Apoyo con el antepié:** dado cuando el apoyo se realiza con la mitad anterior del pie, después del cual generalmente se produce el inmediato apoyo del talón.

Aunque haya diversos tipos de apoyo, la mayoría de gente suelen apoyar con el talón a la hora de correr (ver **Fig.1.**), tanto si son profesionales, rápidos, lentos o gente que comienza a correr.

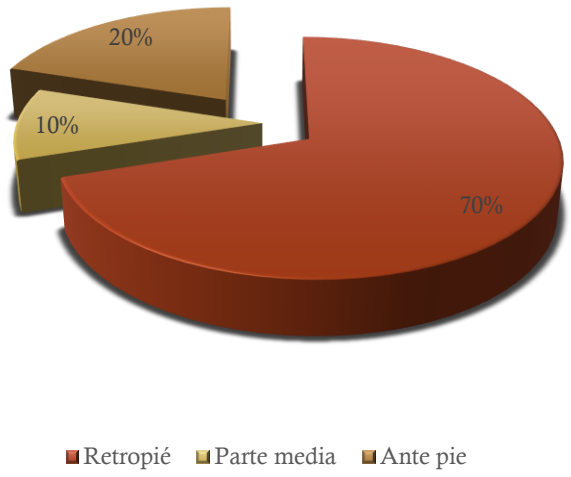

Fig.1. Gráfica de porcentajes de los diferentes apoyos de pie.

Sin embargo, se han realizado estudios a corredores descalzos que muestra que la mayoría acaban adoptando una posición de apoyo con el antepié o de

'puntillas'. Esto ha llevado a pensar a los entrenadores y entusiastas del deporte que la mejor forma de correr sería esta.

No obstante, unas investigaciones realizadas por la Universidad Tempere de Finlandia, descubrieron que ambas formas de apoyar el pie pueden causar lesiones. Para dichas investigaciones se dividieron a corredores en dos grupos; el primero utilizaría el apoyo con el retropié y el segundo con el antepié. Se decidió medir el estrés soportado por las rodillas, tobillos y tendón de Aquiles de cada participante.

En el grupo del retropié se informó de dolencias en las rodillas debido, en parte, a que estas tenían que soportar un 16 % más de fuerza que cuando el apoyo estaba próximo a la zona anterior del pie. Las otras zonas vulnerables también vieron incrementado su estrés.

El grupo de antepié no fue una excepción, si bien no fue en la zona de las rodillas, los corredores absorbían de manera diferente el estrés y llegaron a acumular casi un 20 % más de lo recomendado en las zonas de los tobillos y el tendón de Aquiles que los corredores de retropié.

En resumen, las dos formas de apoyo presentan ventajas e inconvenientes. El apoyo con retropié puede derivar en problemas como el síndrome del dolor rotuliano; el apoyo con el antepié podría producir a la larga lesiones en el tendón de Aquiles, inflamaciones o fracturas por estrés. En la aplicación Android se tomará registro de ambas formas de pisada y, además, de si esta se ha hecho con demasiada fuerza. Aterrizar con demasiada fuerza el pie puede agravar las lesiones producidas por el posicionamiento del pie.

2.4. Tecnología disponible

Debido al gran auge del *running* como deporte han surgido diferentes dispositivos y/o aplicaciones para la medición del rendimiento del usuario. Si bien, no son necesarias para practicar el deporte en sí, vienen bien como complemento para el desarrollo de una buena actividad. A continuación, se exponen los diferentes dispositivos que están disponibles en el mercado y los servicios que ofrecen.

Fitbit

Compañía fundada en los primeros meses de 2007 por James Park and Eric Friedman cuya actividad principal consistía en la implementación de sensores en dispositivos portátiles. Sin embargo, no recibían los ingresos esperados así que crearon su primer prototipo de pulsera *fitness*. (ver **Fig.2.**).

Fig.2. Primer prototipo que lanzó la compañía Fitbit.

Estos primeros dispositivos, constando no mucho más que de un podómetro, tuvieron un éxito arrollador en el mercado y a los pocos meses lanzaron modelos más avanzados que incluían, entre otras cosas, altímetro, reloj digital y cronómetro. En mediados de 2012, Fitbit lanzó el primer dispositivo *fitness* que empleaba la tecnología Bluetooth para la comunicación inalámbrica.

A su vez, distribuyo las aplicaciones para su dispositivo en Android e iOS y abrió su página web. En esta puedes subir tus progresos y compartirlos con otros usuarios de la pulsera de *fitness*. Esto ayudó a la compañía a consolidarse como la principal empresa de distribución en el mercado.

El último modelo lanzado por la empresa, Surge, puede realizar mediciones como la distancia recorrida, calorías quemadas, las plantas subidas, el ritmo cardíaco y también ofrece monitorización GPS. Además, de la conexión inalámbrica a cualquier dispositivo móvil y el envío de datos.

Sony

Empresa nipona fundada en 1945 y conocida mayoritariamente por su distribución de productos electrónicos. Esta empresa también ha lanzado al mercado varias pulseras de *fitness* aprovechando la moda del *running*.

Desde modelos que van de los más sencillo que podría ser una pulsera, hasta pulseras 'inteligentes'; todas ellas se encuentran en el catálogo de Sony. Destacar entre ellas la Sony Smartband SWR10 (ver **Fig.3.**), que es la que más se acerca al prototipo que se busca realizar.

Fig.3. Pulsera Sony Smartband SWR10.

Esta pulsera práctica, sin ningún elemento que destaque en ella o que no sean necesarios para la función que ha de realizar, puede realizar conexiones inalámbricas a dispositivos móviles, actúa como dispositivo de manos libres y lleva un control de actividad física básico.

El control realizado cuenta los pasos dados, la distancia recorrida y las calorías quemadas.

Xiaomi

Empresa china dedicada al desarrollo y venta de productos tales como teléfonos inteligentes, apps (aplicaciones) y productos electrónicos varios. Fue fundada en 2010 y hoy en día es una de las empresas que rivalizan con Samsung y Apple en la venta de móviles. Si bien no es del todo conocido aún en Europa, Xiaomi lleva a sus espaldas grandes éxitos en ventas de productos electrónicos y no se dejó amedrentar por la oportunidad de vender pulseras de *fitness*.

Los productos que ofrece esta empresa en el mercado destacan por su sencillez y su eficacia. De los cuales, cabe destacar la Xiaomi Mi Band (ver **Fig.4.**) por ser el producto más próximo al prototipo que se desea probar. Dicho producto dispone de medidor de actividad física básica: calorías quemadas, distancia recorrida, pasos dados… Pero además incorpora un medidor de sueño que controla el tiempo que el usuario pasa dormido y las distintas fases de sueño.

Fig.4. Pulsera *fitness* Xiaomi Mi Band

Este medidor de sueño ha llamado la atención de los consumidores por su gran exactitud. A parte, el hardware que posee es bastante fiable y su batería puede durar incluso un mes sin necesidad de ser recargada.

A continuación, se muestra una tabla que contiene, en resumen, todos los servicios y capacidades que las pulseras previamente analizadas poseen (ver **Tabla.1.**).

	Pulseras de *fitness*		
Características de la pulsera	Surge de Fitbit	Sony Smartband SWR10	Xiaomi Mi Band
Distancia	SÍ	SÍ	SÍ
Pasos	SÍ	SÍ	SÍ
Calorías quemadas	SÍ	SÍ	SÍ
Pisos subidos	SÍ	NO	NO
Horas de actividad	SÍ	SÍ	SÍ
Monitorización GPS	SÍ	NO	NO
Ritmo cardíaco	SÍ	NO	SÍ
Manos libres	SÍ	SÍ	NO
Comunicación inalámbrica	SÍ	SÍ	SÍ
Control del sueño	NO	SÍ	SÍ

Tabla.1. Tabla comparativa de los diferentes servicios ofrecidos por las pulseras de *fitness*.

En conclusión, se puede observar que, como mínimo, la pulsera prototipo necesita medir la actividad física básica (pasos, calorías quemadas y distancia) y, además, poseer comunicación con el dispositivo móvil. Por otra parte, en lugar de ser una pulsera se busca realizar una tobillera, para así medir, mediante acelerómetro y giroscopio la posición del pie del corredor, dando medidas de cómo es su pisada.

2.5. Conexión Bluetooth Low Energy (BLE)

En este apartado se explicará que es la conexión BLE, como funciona y cómo afectará en el desarrollo tanto de la aplicación para Android, como en el prototipo de dispositivo a testear.

Historia

Bluetooth low energy o BLE es una tecnología de conexión inalámbrica diseñado y vendido por Bluetooth Special Interest Group. Esta tecnología está especialmente enfocada para nuevas aplicaciones en los ámbitos de la salud, del deporte, radiobalizas, seguridad y domótica. En comparación con su hermano, este intenta mantener sus capacidades con un menor consumo de energía.

Los investigadores de la compañía Nokia determinaron en 2001, que una tecnología similar al Bluetooth tradicional, pero de menor consumo energético, era necesario. Partiendo de la base del Bluetooth tradicional, nació lo que se conoce comercialmente como Bluetooth Smart o BLE. Se completó su programación en 2010 y el primero dispositivo en usarlo fue el iPhone 4S, lanzado en 2011, siendo este el primer paso para el lanzamiento en el mercado de más dispositivos compatibles con esta tecnología.

Funcionamiento

Para explicar el funcionamiento de esta tecnología se enfocará el problema en vistas a la aplicación y su relación con el dispositivo externo a conectar, siendo el móvil el cliente y el dispositivo el servidor.

A partir de Android 4.3 (Nivel de Api 18), se introdujo la plataforma de apoyo para el desarrollo de BLE, permitiendo a los dispositivos móviles descubrir servicios, registrarlos y leer o escribir sus características. Para entender cómo funciona el sistema hay que entender los siguientes conceptos clave:

- **GATT (del inglés *Generic Attribute Profile*):** el GATT se basa en perfiles pre-programados en el servidor; estos perfiles tienen como trabajo enviar pequeñas piezas de datos conocidos como 'atributos' mediante la unión cliente-servidor del BLE.
- **ATT (del inglés *Attribute Protocol*):** el ATT es un protocolo construido sobre el GATT que transporta un identificado llamado UUID (del inglés *Universally Unique Identifier*) además de las características y servicios que ofrece el servidor BLE.
- **Característica:** dato que contiene un solo valor y de 0 a n descriptores que especifican el valor de la característica. Por ejemplo, si un servicio es el monitor cardíaco, la característica enviada será el ritmo cardíaco.
- **Descriptor:** los descriptores son definiciones de la característica leída en un servidor BLE. Pueden indicar que es el valor que se lee, cuál es su rango optimo o en que unidades ha de medirse.
- **Servicio:** un servicio es una colección de características. Es la forma de clasificar los diferentes sensores en un dispositivo y las características que contienen.

Cada conexión realizada con un dispositivo BLE está sujeta a unos roles y responsabilidades. Primero, hay que asignar los roles de central y periférico, siendo el central el que escanea en busca del dispositivo y el periférico el dispositivo a buscar. En este caso, el dispositivo móvil con la aplicación tendría el rol de central y el dispositivo prototipo sería el periférico.

Los otros dos roles son el servidor GATT y el cliente GATT. Cuando una conexión se establece entre dos dispositivos, empieza la transmisión de información GATT. Si del dispositivo externo al que se realiza la conexión BLE solo se quieren leer datos, se le asignará el rol de servidor y si se le quieren enviar datos al dispositivo, a éste se le asignará el rol de cliente. En este caso, el dispositivo móvil mediante la aplicación Android tendrá un papel de cliente GATT y el dispositivo prototipo a testear tendrá el rol de servidor GATT.

En conclusión, cuando se realice una conexión BLE entre dos dispositivos primero se tiene que asignar los roles respectivos a cada uno, cliente y servidor. Una vez realizada la asignación de roles, se pasará a enviar los datos

correspondientes en forma de servicio o característica, según lo que se requiera del dispositivo.

2.6. Entorno de programación

Android ha sido elegido como el entorno de desarrollo para la programación de la aplicación del dispositivo móvil. Android, es un sistema operativo de móvil desarrollado por Google; está basado en Linux y se diseñó principalmente para dispositivo portátiles con pantalla táctil.

Se ha elegido Android como entorno de desarrollo debido a su accesibilidad, es de código abierto, y la mayoría de gente posee dispositivos móviles con sistema operativo de Android. No obstante, su uso para desarrollar la aplicación requiere definir unos niveles mínimos de versión con los cuales ha de ser compatible.

2.7. Conclusiones

En esta sección se ha explicado la situación del running en España y cómo este deporte arraigó y germinó rápidamente. También se ha detallado qué características tiene que tener la aplicación Android y el dispositivo prototipo a testear. Esto se ha concluido mediante la comparación de las diferentes lesiones básicas que comúnmente sufren los corredores, además de la comparación de las diferentes características que los diferentes dispositivos en el mercado. Por último, el entorno de desarrollo para la aplicación ha sido seleccionado y comentado los motivos que han llevado a su selección.

3. Entorno de desarrollo

3.1. Introducción

En este apartado se especificarán y mostrarán las diferentes tecnologías y herramientas que se usarán para llevar a cabo el desarrollo de la aplicación Android para el dispositivo móvil. Además, se va a realizar una breve introducción de las más destacadas.

3.2. Herramientas

Android-SDK:
Paquete de software que incorpora todas las herramientas y librerías necesarias para el desarrollo de aplicaciones Android.

Android Studio:
Entorno de desarrollo proporcionado por Android Inc. para la programación de aplicaciones para todo tipo de dispositivos móviles; su lanzamiento el 8 de diciembre de 2014 supuso la eliminación de Eclipse como entorno de desarrollo principal para Android.

Gradle:
Incorporado dentro de Android Studio, Gradle permite de manera fácil la automatización de procesos de compilación, administración interna de dependencias y configuraciones específicas de compilación.

Bluetooth Smart:
Tecnología de conexión inalámbrica que se permite la conexión a dispositivos externos y la lectura de datos de los mismos. De código abierto, intenta reemplazar al Bluetooth tradicional como forma de conexión inalámbrica.

MySQL:
Gestor de bases de datos relacional con licencia GNU GLP, su uso está muy extendido debido al fácil acceso de los distintos lenguajes de programación a través de interfaces propias. En caso de Android, se usará la interfaz de SQLite.

SQLite permite el desarrollo de bases de datos sin necesidad de cliente o servidor, ya que la propia base de datos esta embebida en el código. Debido a no disponer de la relación de cliente y servidor, la conexión es casi instantánea ofreciendo los datos e interacciones en tiempo real.

Utiliza el mismo lenguaje de programación que MySQL y su uso se ha extendido a aplicaciones en dispositivos móviles y de web.

A diferencia de MySQL, SQLite tiene sus limitaciones y no puede interactuar de forma directa con algunos elementos de las aplicaciones a las que está destinado. Tampoco dispone de la capacidad de borrar o alterar columnas completas.

3.3. Tecnologías

La interfaz gráfica de una aplicación Android depende de una jerarquía de vistas y grupos de vistas (*Views* y *ViewGroups*) (ver **Fig.11.**). Un grupo de vistas sería el *layout* o diseño que contendría las diferentes vistas, pudiendo ser estas botones o cualquier método de comunicación con el usuario.

Una vista tiene un padre (*parent*) definido, excepto el primer grupo de vistas, cuyo padre serían las características del móvil. Esta vista hija (*child*), tiene acceso a algunas características del padre y solo se pueden cambiar sus atributos dentro del grupo de vistas padre de donde proviene. Usando como ejemplo el diagrama de bloques (ver **Fig.5.**), la vista declarada en el segundo grupo de vistas que se ha creado bajo el primer grupo de vistas, no podría interactuar de ninguna forma con la vista declarada en el mismo nivel que el grupo de vistas al que pertenece. No obstante, el grupo de vistas sí que puede interactuar con las vistas declaradas a su nivel porque son vistas hermanas.

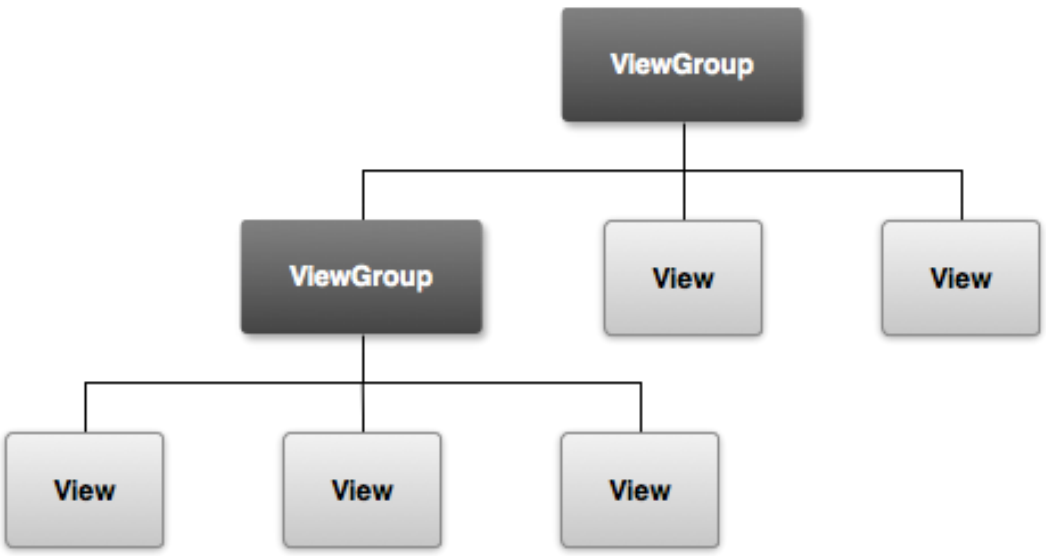

Fig.5. Jerarquía de la interfaz gráfica de Android.

Tal y como se observa en el diagrama de bloques (ver **Fig.6.**), el Sistema operativo de Android basa la vida de sus aplicaciones en diferentes ciclos. Cada aplicación tiene que tener una actividad principal (fichero .java) que sea la primera en ejecutarse nada más abrirse la aplicación.

Cuando la aplicación se abre, se ejecuta el onCreate() de la actividad. Este es un método necesario en todas las actividades de Android y solo se ejecuta una vez, cuando la actividad es creada. Se recomienda que sea aquí donde inicialices las variables necesarias y las vistas que se vayan a usar en la interfaz con el usuario.

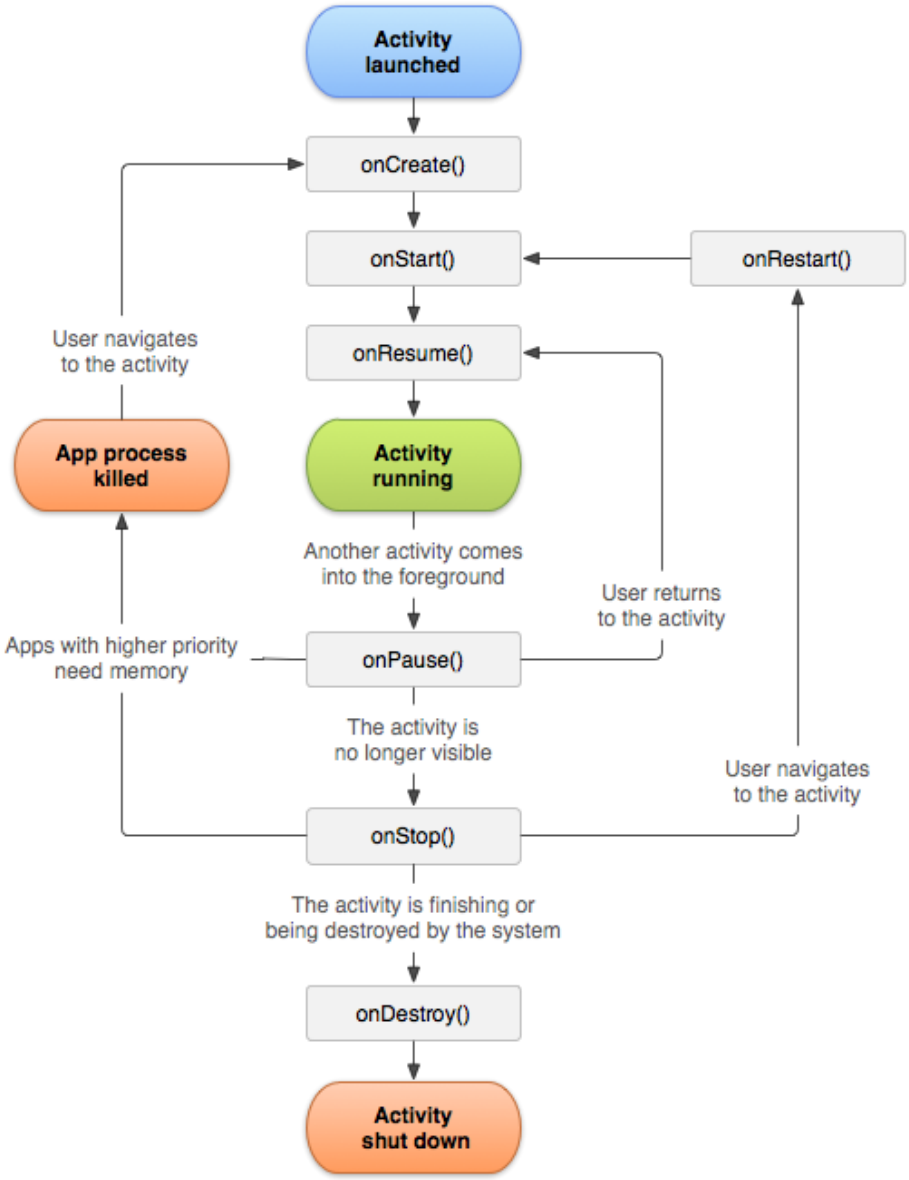

Fig.6. Diagrama de bloques del ciclo de vida de una aplicación Android.

Después sigue el método onStart(), este método se activa cuando la aplicación comienza a ser visible para el usuario. El siguiente método en ser ejecutado es onResume(), que es cuando la aplicación permite la interacción con el usuario; es donde se recomienda que vayan las interacciones con el usuario.

Alcanzado este punto la actividad está completamente visible y en total comunicación con el usuario, permitiendo un intercambio de datos entre esta y el usuario de la misma. Se considera que esta actividad está en primer plano y tiene prioridad sobre las demás. Cuando el usuario decide cambiar la aplicación o minimizarla pasa a un segundo plano y llama al método onPause().

Este método se llama cuando la actividad pasa a un segundo plano, ya que se va a proceder a llamar a otra actividad o a terminar con la aplicación. La recomendación dicta que es aquí donde se deben guardar los datos necesarios para la aplicación (*Ej. Puntuaciones en un videojuego o campos de texto rellenados por el usuario en un formulario*). También se tiene que parar el uso de actividades que consuman CPU sean éstas, la cámara del dispositivo móvil, animaciones o videos. No obstante, se ha de intentar que contenga la menor cantidad de código posible, ya que la siguiente actividad no comenzará hasta que se ejecute por completo el método. Si el usuario decide volver a la actividad, se llamará de nuevo al método onResume(), de lo contrario se pasará a llamar al método onStop().

El método onStop() es llamado cuando la aplicación ya no es visible para el usuario, ya sea porque otra actividad tiene prioridad en el dispositivo o porque esta superpuesta a ésta; dado el último caso, podría significar que la aplicación quiere terminar con la actividad. En este estado, se recomienda desconectar todo dispositivo externo al móvil o cualquier hardware del propio dispositivo, como la cámara. Si el usuario no llama de nuevo a la aplicación, se pasará al método onDetroy(), sino al método onRestart().

Cuando el método onDestroy() es ejecutado puede deberse a que la actividad ha sido destruida por el usuario, o que el propio sistema operativo necesita eliminarla por motivos de memoria. Desde este método no puedes volver a ejecutar la aplicación y tendrá que ser lanzada de nuevo desde la interfaz de usuario. El método onRestart() solo puede ser llamado desde el método onStop() y simplemente sirve como enlace con el método onStart().

Cada aplicación tiene que tener mínimo una actividad, que es la necesaria para que cuando la aplicación sea llamada se empiecen a ejecutar los métodos mencionados previamente. Esta actividad consiste de un fichero de clase, escrito en el lenguaje de programación JAVA y de un *layout* o contenedor de vistas que sirve de comunicación con el usuario. Si bien, este último no es del

todo necesario, Android ofrece unos *layout* predefinidos básicos de comunicación con el usuario, sean listas o simples inputs.

El fichero layout estará escrito en XML (del inglés *Extensible Markup Language*), que es un lenguaje de marcas o *tags* desarrollado por el W3C (del inglés *World Wide Web Consortium*) y se utiliza para almacenar datos de forma legible. XML facilita la comunicación con otros lenguajes, incluyo bases de datos, pudiendo integrarse fácilmente en ellos. Es de código abierto y el objetivo hoy en día es convertirla en el estándar para el intercambio de información estructurada entre diferentes plataformas.

A la aplicación se le pueden asignar unos recursos o *values* predefinidos o creados por el usuario mediante ficheros XML. Estos recursos pueden ser imágenes de vectores o de mapas de bits, recursos de *Strings*, que son palabras o frases definidas para un idioma o varios, colores y/o estilos. Todos estos recursos pueden ser llamados en cualquier momento de la actividad y es recomendable que, en ciertos casos, dado por ejemplo las imágenes o los recursos *String*, que estén pensados para varias plataformas. Es decir, las imágenes que estén en varias resoluciones y los *String* en varios idiomas.

Esta aplicación además ha de tener un manifiesto escrito en XML donde se declaren varios aspectos de la misma. Entre ellos, la actividad principal, los recursos que utiliza, tanto de *software* como de *hardware* y todos los servicios y actividades que estén incluidos en la aplicación.

Para terminar, toda aplicación está ligada a unos archivos GRADLE que ayudan a la compilación y configuración de la misma, siendo estos los que se encargan que la aplicación este en modo desarrollo o de lanzamiento. Los archivos GRADLE incorporarán el nivel de SDK mínimo al que va dirigido la aplicación.

3.4. Conclusiones

En este apartado se han descrito tanto las tecnologías como las herramientas que se utilizarán en la elaboración de la aplicación Android para el dispositivo móvil.

Utilizando el entorno de desarrollo que es Android Studio y mediante el Android-SDK se desarrollará la aplicación. Para la comunicación con el prototipo del dispositivo *runner* se empleará el Bluetooth Smart o BLE. Esta comunicación se basará en que el móvil es el cliente del prototipo, que será el servidor.

La propia aplicación empleará la tecnología de SQLite para la gestión de los datos del usuario y de su rendimiento. La aplicación se desarrollará en el lenguaje de Android que incluye tanto JAVA como XML.

4. Especificación de requisitos

4.1. Introducción

En esta sección se va a explicar las necesidades que va a tener la aplicación planteada, para describirlas, se va a emplear el estándar IEEE 830.

4.1.1. Propósito

En este apartado se va a pasar a detallar tanto los requisitos como las funcionalidades que debe satisfacer la aplicación de Android que, como se ha especificado en anteriores apartados, tiene que medir el rendimiento de un *runner* en función de la distancia recorrida, pasos dados y calorías quemadas; también controlará las posibles lesiones que pueda sufrir el corredor controlando la posición de su pie a la hora de correr.

Además, la aplicación tiene que ser capaz de realizar una conexión inalámbrica mediante BLE con un dispositivo externo, en este caso el CC2541 Sensortag, el cual se va a testear para la posibilidad de la elaboración de un prototipo más elaborado a partir del mismo.

Este documento ha de servir como referencia para la mejora o la ampliación del software ligado a la aplicación, además del entendimiento del funcionamiento para el posible desarrollo y su posterior implementación en futuros dispositivos móviles.

4.1.2. Ámbito del sistema

La aplicación **Pro Run**, que es el nombre dado a la aplicación a desarrollar, tiene que tener para su correcto funcionamiento un CC2541 SensorTag, con el cual ha sido desarrollado, o un prototipo con similares características.

Una vez se tiene este dispositivo externo, la aplicación permitirá la creación de un usuario y el posterior registro del mismo en la base de datos. Una vez creado, el usuario podrá acceder a la interfaz principal donde se le permitirá dar comienzo a la actividad que mientras esta misma actividad, controlará el rendimiento físico del corredor.

La interfaz de usuario principal dará al usuario la opción de modificar sus datos en la base de datos, y el borrado de algunos registros. También se le proporcionará al corredor un reto diario para probar sus habilidades como corredor. Por último, el usuario puede acceder a un registro

completo de todas sus actividades donde se muestra su rendimiento total desde que se creó el usuario o desde el último borrado de memoria.

4.1.3. Definiciones, acrónimos y abreviaturas

API: del inglés *application programming interface*; es un conjunto de definiciones de subrutinas, protocolos y herramientas para la elaboración de *software* y aplicaciones.

***Null*:** o nulo, se utiliza normalmente en los lenguajes de programación como valor nulo. Se da cuando no se le asigna valor alguno a una variable o dato; se puede utilizar como referencia de puntero.

Aplicación: programa o conjunto de programas informáticos destinados a realizar un trabajo específico, diseñado para satisfacer las necesidades de un usuario final.

Aplicación móvil: programa o conjunto de programas desarrollados con objetivo de ser usados en dispositivos móviles.

Dispositivo móvil: dispositivo electrónico de pequeño tamaño, pensado para que el usuario pueda transportar fácilmente de un lado a otro y teniendo una capacidad de procesamiento similar a una computadora.

BBDD: bases de datos, colección de datos de información organizadas para que un sistema externo pueda seleccionar fragmentos específicos de la misma.

SQL: del inglés *structured query language*, es un lenguaje de programación diseñado para la relación racional de sistemas de bases de datos.

SQLite: sistema de gestión de bases de datos relacional; compatible en mayor parte con los lenguajes de programación para dispositivos móviles, pero, que a diferencia de MySQL, contiene algunas restricciones. No hace falta tener una conexión con un servidor externo ya que el propio código esta embebido dentro de la aplicación.

JAVA: lenguaje de programación orientado a objetos que en la actualidad es el más usado entre los sistemas operativos de los diferentes dispositivos móviles. De código abierto y fácil acceso, permite una fácil difusión de los programas y aplicaciones creados con el mismo.

BLE: del inglés *Bluetooth low energy*, es un sistema de conexión inalámbrico entre dispositivos con similares capacidades que el Bluetooth tradicional, pero con menor consumo de energía.

GATT: del inglés *generic attribute profile*, es un perfil de especificación general para el envío y la lectura de datos en una conexión BLE. Para más información consultar el apartado 2.6.

UUID: del inglés *Universally Unique IDentifier*, es un código identificador estándar empleado en la creación de software y aplicaciones para el registro de elementos en el mismo.

Cliente: programa que necesita la conexión a otro programa que actúa como servidor. El papel del cliente es la lectura de datos de la conexión BLE.

Servidor: programa que proporciona los datos a leer en una conexión BLE. El servidor no puede recibir datos para su lectura ni escritura.

Corredor: o *runner* es la persona que dedica su tiempo a practicar el *running* cómo deporte principal.

***Running*:** también conocido como jogging, es un deporte que arraigado últimamente en la población debido a su facilidad y accesibilidad. Para más información consultar el apartado 2.1.

Vista: parte visible de una interfaz de usuario que sirve para la interacción con el mismo.

Grupo de vistas: conjunto de vistas que definen la relación padre-hijo entre las diferentes vistas de la aplicación.

XML: lenguaje de marcas extensible que permite la organización y etiquetado de documentos.

***Layout*:** documento XML considerado coloquialmente como un 'croquis' de donde debe ir cada elemento en una interfaz de usuario, siendo estos elementos vistas o grupos de vistas.

Manifiesto: documento XML donde se especifican las características de la aplicación, así de los elementos que esta utiliza, sean físicos o no.

Metadato: dato sobre un dato, información que no es relevante para el usuario de la aplicación, pero si para la misma.

4.1.4. Visión general del documento

El documento consta de tres partes definidas con claridad, siendo la primera una breve introducción general a la aplicación y sus definiciones, la segunda es donde se explica el contexto de la aplicación y una tercera donde se detallan las funcionalidades que se requiere que tenga esta.

4.2. Descripción general

La aplicación tiene que permitir la medición de rendimiento de un corredor mediante la conexión a un dispositivo externo y los propios sensores del dispositivo móvil. A su vez, tiene que garantizar la creación de perfiles de usuario y su posterior modificación; también tiene que asegurar que los rendimientos de cada usuario sean guardados para su posterior uso.

4.2.1. Perspectiva del producto

Por una parte, la aplicación a desarrollar tiene que al menos, proporcionar al usuario las mediciones mínimas de actividad física que otros dispositivos y aplicaciones en el mercado ofrecen, siendo éstas el cálculo de la distancia recorrida, pasos dados y calorías quemadas.

Por otra parte, la aplicación tiene que conectarse a un dispositivo externo, en este caso el CC2541 Sensortag, del que leerá los datos proporcionados por un acelerómetro y un giroscopio para saber en un mapa 3D cuál es la posición del pie del usuario.

La aplicación se desarrollará casi en su totalidad con el lenguaje de programación Android, el cual incluye JAVA y XML; haciendo uso de las bibliotecas públicas del SDK de Android para su correcto funcionamiento.

El otro lenguaje a utilizar será el SQL. Este lenguaje se usará para la elaboración de una base de datos interna en el dispositivo mediante SQLite. Esta base de datos tiene que almacenar los datos del dispositivo externo, un registro de usuario y un registro sobre las actividades del usuario.

4.2.2. Funciones del producto

La aplicación que, de una manera cómoda y fácil, tiene que realizar las siguientes funciones:

- Conexión con un dispositivo externo mediante BLE.
- Creación de un usuario ligado al dispositivo externo.
- Modificación de los datos del usuario una vez creado.
- Manejar actividades físicas para medir el rendimiento del corredor
- Recopilación de todos los datos de las actividades físicas realizadas por el usuario.

- Interacción con la base de datos para la posible eliminación de registros de actividades pasadas.
- Retos diarios que pongan a prueba al usuario.
- Persistencia de los datos del usuario después del apagado o reinicio de la aplicación.
- Garantizar la seguridad de conexión del puerto Bluetooth del dispositivo móvil, permitiendo que solo se pueda conectar el dispositivo externo señalizado.

4.2.3. Características de los usuarios

Los usuarios objeto de esta aplicación tienen que ser practicantes del deporte conocido como *running* y que busquen una alternativa para el control de su rendimiento como corredores. Si bien, la aplicación no requiere un nivel de estudios ni experiencia previa con otras aplicaciones, se ha de tener precaución a la extrema acumulación de actividades en la base de datos, que puede llegar a producir una corrupción de datos.

La aplicación se podrá usar por diferentes usuarios y conectarse a diferentes dispositivos, debido a que los registros se guardan en la base de datos. No obstante, con un dispositivo sólo se puede registrar a un usuario, pero si ya hay varios usuarios registrados, el mismo dispositivo se puede usar con diferentes usuarios.

4.2.4. Restricciones

Primero de todo será necesario un dispositivo externo habilitado con conexión inalámbrica Bluetooth Smart® o BLE, o un dispositivo de prototipo como el CC2541 SensorTag. Sin ninguna de estas dos cosas, no se podrá iniciar la aplicación como es debido, debido a que al menos, se necesita un dispositivo para registrar al primer usuario.

La aplicación requiere del uso de la tecnología Bluetooth dispuesta en el dispositivo móvil del usuario, si este no tiene ningún tipo de tecnología Bluetooth no se podrá realizar la conexión. Dado el caso de que dicho dispositivo móvil contenga tecnología Bluetooth, tiene que tener compatibilidad con la tecnología Bluetooth Smart® o BLE. Si no es compatible con esta tecnología no se podrá realizar la conexión.

También se requiere el uso de la geo localización del dispositivo móvil, dado que es necesario en la búsqueda y lectura de la radio baliza del dispositivo externo con conexión BLE. Si la geo localización no está activada, no se podrá localizar al dispositivo incluso si ya se ha activado el Bluetooth del dispositivo móvil.

La aplicación tiene que permanecer en primer plano mientras se quiera realizar alguna actividad; si se da el caso de que se pasa a segundo plano tendrá que reiniciar la aplicación y realizar de nuevo la conexión con su dispositivo externo para continuar con su actividad. Esto se debe a la naturaleza de la conexión BLE, ya que es necesario que ambos dispositivos estén conectados entre sí y se les hayan asignado los roles pertinentes. Si el dispositivo hace como servidor y pierde al cliente, el dispositivo móvil, pasa a inhabilitar toda conexión y deshabilita todos los sensores hasta que se realice la siguiente conexión acabando así, en la perdida de toda información enviada al dispositivo móvil.

Al usuario se le ha dado interacción restringida con la base de datos de la aplicación, permitiéndoles actualizar datos tales como: peso, altura y zancada. Dado que las condiciones físicas del usuario varíen a lo largo del uso de la aplicación es recomendable que estas se actualicen con los datos más recientes para una medición más exacta del rendimiento del corredor. En caso de no saber calcular su zancada, o de desconocimiento total de ésta, la aplicación pasa a usar unas constantes globales asignadas como valores default para todos los usuarios, pero que dan los resultados de forma imprecisa respecto a las condiciones físicas del usuario.

El usuario también dispone de la capacidad de borrar algunos datos de la base de datos, siendo estos: usuarios registrados, borrado de actividades realizadas y el borrado total de la base de datos. Debido a la relación que hay entre las diferentes tablas de la base de datos, borrar todos los usuarios está prohibido por la posibilidad de causar un fallo total en la misma; por tanto, la aplicación está restringida a tener, por lo mínimo, un usuario. Respecto a las actividades, se considera que como máximo, un usuario podrá realizar a lo largo del día, cinco actividades físicas. Dado que éstas no se pueden auto descartar con el tiempo, se pueden acumular de manera excesiva en la memoria del dispositivo móvil del usuario, causando a la larga fallos de memoria y de corrupción de datos; debido a esto, se le proporciona al usuario un botón de borrado total de actividades. Como elemento depurador en caso de corrupción de datos, el usuario podrá eliminar todo dato guardado en la base de datos, pero se recomienda no hacer un uso excesivo del mismo, ya que podría corromper la aplicación, haciendo necesaria una reinstalación completa del mismo.

Cada interacción con la base de datos, sea para añadir un usuario o el borrado de alguna instancia, supondrá el reinicio de la aplicación para evitar futuras corrupciones de datos.

La seguridad del puerto Bluetooth se garantiza mediante la programación, ya que se ha diseñado para que solo pueda conectarse a

un tipo específico de datos, en este caso, del CC2541 SensorTag. La aplicación no detectará ningún otro dispositivo que no lleve las marcas del puerto GATT necesarias para la conexión.

Si bien la aplicación permite la creación de varios usuarios y del registro de varios dispositivos, solo podrá utilizarse un dispositivo a la vez por aplicación. Es decir, con un solo prototipo solo te puedes conectar a una aplicación y una aplicación solo podrá conectar a un dispositivo. La conexión de otra aplicación al dispositivo conectado en otra aplicación puede causar la desconexión de este y, por ende, la perdida de datos del corredor.

Por último, debido a la naturaleza del prototipo y de la conexión BLE, se han detectado problemas de incompatibilidad en algunos dispositivos móviles. Estos problemas varían desde la imposibilidad de conectar con el dispositivo externo hasta la no lectura de los datos enviados por el servidor GATT.

4.2.5. Suposiciones y dependencias

Dada la naturaleza de SQLite, una vez creada la base de datos no se puede modificar; tan solo se puede borrar y volver a crear. Esto inhabilita la posibilidad de simples actualizaciones. Si se quisiera añadir nuevas funcionalidades a la base de datos para la lectura de datos del dispositivo externo se tendría que modificar el código fuente de la aplicación en sí.

No obstante, la parte de la interfaz de usuario está programada de forma modular, permitiendo la incorporación de esas nuevas características de forma fácil y simple. También pueden añadirse de forma sencilla nuevos retos diarios para el usuario sin necesidad de alterar la base de datos.

4.2.6. Requisitos futuros

Debido a que la tecnología avanza rápidamente, se debería adaptar la aplicación en base al dispositivo externo que mejor venga para recabar datos sobre el usuario. Es decir, si en algún momento dado se desarrollan nuevos sensores que se puedan utilizar con el fin de medir el rendimiento del usuario, se debería adaptar la aplicación a esos nuevos sensores.

En un futuro también se podría implementar, mediante la conexión a un servidor web externo, una red social en base a la aplicación y los diferentes usuarios de esta; permitiendo así el intercambio de progreso en los diferentes retos diarios y el incremento de sus estadísticas de rendimiento. También se podrían ligar la tabla de resultados de fin de

actividad con la red social Twitter para difundir fácilmente los datos de la actividad entre los usuarios.

Se puede plantear la incorporación de un sistema de recompensa en base a los retos completados en base al usuario. Pudiendo ser moneda virtual para ser gastada de alguna forma por el usuario o simplemente experiencia o puntos que sirvan como ayuda para crear una tabla de clasificación entre los diferentes usuarios de la aplicación.

Debería implementarse un logaritmo que sea capaz de identificar las actividades más antiguas y menos visitadas por el usuario para así poder eliminarlas, evitando así el borrado sistemático de todas.

4.3. Requisitos específicos

A continuación, se van a describir con detalle todos los requisitos que debe de cumplir el sistema para que, como mínimo, permita planificar, implementar, diseñar las pruebas y validar el cumplimiento de dichas funciones. También se detallarán los requisitos necesarios que debe cumplir el dispositivo externo prototipo a testear.

4.3.1. Interfaces externas

La aplicación a desarrollar está diseñada para funcionar en cualquier dispositivo móvil que incorpore Android como sistema operativo. El sistema operativo debe estar actualizado a Android 6.0 (*Marshmallow*) para su correcto funcionamiento. No obstante, dada la naturaleza del dispositivo prototipo externo, se ha detectado incompatibilidad de conexión con algunos móviles. Se recomienda comprobar la compatibilidad antes de usar la aplicación.

El usuario de la aplicación debe de aceptar, en el momento de ejecutar la aplicación, el uso de la geo-localización y del puerto Bluetooth; se le notificará con un mensaje de aviso en caso de no estar activadas estas opciones en el móvil y, si continúan sin ser activadas, se pasará a finalizar la ejecución de la aplicación.

Para el correcto funcionamiento de la aplicación es necesario un dispositivo prototipo externo o un equipo prototipo como el CC2541 SensorTag. Si no se dispone de ninguna de las dos cosas no se podrá iniciar la aplicación con éxito, impidiendo la continuación de la misma.

Si se dispone de un dispositivo prototipo externo, el usuario ha de crear un perfil de usuario ligado a ese dispositivo para continuar con la ejecución de la aplicación. Un usuario puede registrar varios dispositivos

a su nombre, pero un dispositivo solo puede registrar a un usuario. No obstante, si ya se han registrado más de un usuario, estos podrán intercambiar dispositivos y seguirá funcionando de forma correcta.

La conexión BLE se basa en el establecimiento de roles cliente y servidor. Si se intenta establecer conexión con otro cliente, dispositivo móvil, al mismo servidor, dispositivo prototipo externo, en el que esté conectado otro usuario, este último se desconectará del primer cliente y establecerá como cliente al nuevo usuario.

Es imperativo que la aplicación este en primer plano; si se da el caso de que pase a segundo o tercer plano, la aplicación detendrá toda conexión realizada a dispositivos externos, impidiendo así el correcto envío de datos entre cliente y servidor.

El dispositivo móvil del usuario de la aplicación tiene que disponer de sensores de movimiento básico. Estos son necesarios en el correcto funcionamiento del sistema, y de la medición del rendimiento. En caso de no disponer de ellos, se producirá una pérdida de información en el cálculo del rendimiento.

4.3.2. Funciones

Se ha elegido que el orden de aparición de las diferentes funciones de la aplicación siga una jerarquía funcional. Es decir, a continuación, se mostrarán las diferentes funciones llevadas a cabo por la aplicación en el mismo orden en el que aparecen en la aplicación. Se ha elegido este método porque es una forma sencilla y fácil de explicar para el funcionamiento de las diversas actividades que se han de realizar.

Aplicación

Pantalla inicial

Descripción:

Zona inicial que presentará una lista vacía y un botón para el escaneo.

Entrada:
(Ninguna)

Salida:

Selección de un ítem en la lista de dispositivos localizados.

Restricciones:
Conectividad Bluetooth y geo-localización activadas.

Descripción:

El usuario puede visualizar fácilmente el botón de escaneo, con el cual podrá iniciar la búsqueda de dispositivos BLE en la zona. Si ha facilitado los permisos requeridos y encendido el dispositivo, cuando el escaneo haya finalizado se podrá observar en una lista los diferentes dispositivos localizados. Los datos del dispositivo, siendo estos: el código de radiobaliza del dispositivo externo conectado, UUID del dispositivo externo conectado, nombre del dispositivo externo conectado y nombre dado por el usuario al dispositivo externo conectado; se registrarán automáticamente en la base de datos, si no existe ya en ella. Se continuará mediante la selección de un ítem de la lista de dispositivos conectados

Nuevo dispositivo localizado

Descripción:
Diálogo de alerta notificando que se ha encontrado un nuevo dispositivo

Entrada:
Ítem seleccionado en la lista de dispositivos encontrados.

Salida:
Botón de aceptar.

Restricciones:
Datos de radio baliza del dispositivo externo.

Descripción:
Alerta para el usuario que describe la necesidad de registrar el nuevo dispositivo encontrado y así, registrarlo en la base de datos.

Registro de nuevo dispositivo

Descripción:
Actividad que pide al usuario los distintos datos necesarios para el correcto registro del dispositivo.

Entrada:
Botón de aceptar del dialogo de alerta de dispositivo localizado.

Salida:
Botón de aceptar o cancelar.

Restricciones:
No está permitido dejar como campos vacíos el nombre del dispositivo o el nombre de usuario.

Descripción:
En esta actividad se le pedirá al usuario los datos necesarios para el registro del dispositivo en la base de datos. Esto consta de:

- Un nuevo nombre para el dispositivo. No se admite ni el nombre pre establecido ni dejarlo en blanco.
- Nombre de usuario. Puede ser un nuevo nombre o utilizar los datos de un usuario ya establecido en la base de datos.
- Peso en kilogramos del usuario. Puede ser un nuevo peso o utilizar los datos de un usuario ya establecido en la base de datos.
- Altura en centímetros del usuario. Puede ser una nueva altura o utilizar los datos de un usuario ya establecido en la base de datos.

Se procederá a salir de esta actividad mediante el botón de aceptar, guardando todos los datos proporcionados, o mediante el botón cancelar, no guardando los datos proporcionados.

Alerta de inscripción completada

Descripción:
Diálogo de alerta que muestra al usuario que se ha completado el registro.

Entrada:
Botón de aceptar del registro de nuevo dispositivo.

Salida:
Botón de aceptar.

Restricciones:
El campo de nombre del dispositivo y del usuario no deben ser *null* o ser campos vacíos.

Descripción:
Si el usuario ha rellenado correctamente los campos y ha pulsado el botón de aceptar, este diálogo le muestra que sus datos han sido registrados correctamente en la base de datos.

Diálogo de selección de usuario

Descripción:
Alerta que insta al usuario de la aplicación a elegir un usuario registrado.

Entrada:
Ítem seleccionado en la lista de dispositivos localizados.

Salida:
Botón de seleccionar usuario.

Restricciones:
(Ninguna)

Descripción:
Si el dispositivo externo localizado ya está registrado en la base de datos se le pide al usuario que se identifique como alguno de los usuarios de la base de datos. Esto se consigue mediante una lista desplegable con los diferentes usuarios registrados en la aplicación.

Diálogo de reto diario

Descripción:
Diálogo de alerta que muestra al usuario el reto diario que ha de completar.

Entrada:
Usuario seleccionado.

Salida:
Botón de aceptar.

Restricciones:
No se ha completado un reto en el mismo día.

Descripción:
Diálogo que muestra al usuario el reto que debe de completar en ese día para medir su capacidad como corredor. El diálogo consta de una imagen identificativa, el nombre del reto a cumplir, una breve descripción de éste y el progreso que lleva el usuario en este día.

Interfaz de usuario principal

Descripción:
Zona de interacción del usuario con la aplicación.

Entrada:
Usuario seleccionado.

Salida:
Botón de reto, botón de configuración, botón de borrado, botón de estadísticas o botón de detener actividad.

Restricciones:
(Ninguna)

Descripción:
Se muestra al usuario un menú desde donde puede: comenzar una actividad física que desea ser medida, configurar los datos del usuario, borrar información de la base de datos y acceder a sus estadísticas globales. El botón de detener esta deshabilitado normalmente para evitar fallos, se activará en cuanto el usuario indique que comenzará una actividad.

Diálogo de explicación de reto diario

Descripción:
Diálogo de alerta que muestra la información del reto que el usuario ha de completar.

Entrada:
Botón de reto diario en la interfaz principal de usuario.

Salida:
Botón de aceptar.

Restricciones:
No se ha completado un reto en el mismo día.

Descripción:
Diálogo que muestra al usuario una descripción más detallada del reto diario a completar en ese día. También muestra su progreso en el mismo.

Diálogo de estadísticas generales

Descripción:
Diálogo de alerta que muestra al usuario sus estadísticas globales desde el último borrado.

Entrada:
Botón de estadísticas de la interfaz de usuario principal.

Salida:
Botón de aceptar, botón de información de pasos en puntillas, botón de información de pasos de talón o botón de información de pasos pesados.

Restricciones:
(Ninguna)

Descripción:
Diálogo que muestra al usuario todas las estadísticas registradas en la base de datos relacionada con las actividades físicas que ha realizado con la aplicación. Entre ellas constan:

- Pasos totales realizados.
- Porcentaje de pasos de puntillas sobre los pasos totales realizados.
- Porcentaje de pasos de talón sobre los pasos totales realizados.
- Porcentaje de pasos pesados sobre los pasos totales realizados.
- Distancia total recorrida.
- Número de calorías quemadas por el usuario.

Al usuario se le proporciona unos botones de información en caso de no saber que significan los pasos de puntillas, los pasos de talón o los pasos pesados.

Diálogo de información de pasos en puntillas

Descripción:
Diálogo de alerta que muestra al usuario información sobre los pasos en puntillas.

Entrada:
Botón de información sobre pasos en puntillas de estadísticas generales.

Salida:
Botón de aceptar.

Restricciones:
(Ninguna)

Descripción:
Diálogo que muestra al usuario la información básica necesaria sobre cómo se registran los pasos en puntillas y de cómo le pueden afectar como corredor. También muestra su porcentaje actual.

Diálogo de información de pasos de talón

Descripción:
Diálogo de alerta que muestra al usuario información sobre los pasos de talón.

Entrada:
Botón de información sobre pasos de talón de estadísticas generales.

Salida:
Botón de aceptar.

Restricciones:
(Ninguna)

Descripción:
Diálogo que muestra al usuario la información básica necesaria de como la aplicación registra los pasos de talón; además de cómo pueden afectar a la larga a su rendimiento como corredor. También muestra el porcentaje actual de pasos de talón.

Diálogo de información sobre los pasos pesados

Descripción:
Diálogo de alerta que muestra al usuario información sobre los pasos pesados.

Entrada:
Botón de información sobre pasos pesados de estadísticas generales.

Salida:
Botón de aceptar.

Restricciones:
(Ninguna)

Descripción:
Muestra información básica al usuario sobre como el sistema registra los pasos pesados y de cómo puede afectar a su rendimiento; además su porcentaje actual de pasos pesados.

Diálogo de configuración de usuario

Descripción:
Diálogo de alerta que permite al usuario una modificación básica de sus datos.

Entrada:
Botón de configuración de la interfaz de usuario principal.

Salida:
Botón de aceptar, cancelar o de información sobre zancadas.

Restricciones:
(Ninguna)

Descripción:
Este diálogo permite al usuario la modificación de algunos de sus datos personales registrados previamente en la base de datos. Siendo estos datos:

- La altura en centímetros.
- El peso en kilogramos.
- La zancada en centímetros.

En caso de que el usuario no sepa que es la zancada se le proporciona un botón de información. Todos los datos previamente registrados son mostrados al usuario para su cotejamiento. Puede guardar los nuevos datos mediante el botón de aceptar, o cancelar los cambios con el botón del mismo nombre.

Diálogo de información sobre la zancada

Descripción:
Diálogo de alerta que muestra al usuario información básica sobre la zancada.

Entrada:
Botón de información de zancada en la configuración de usuario.

Salida:
Botón de aceptar.

Restricciones:
(Ninguna)

Descripción:
Muestra al usuario información básica de como la aplicación gestiona la calzada e instrucciones para la inserción de la misma en sus datos. También le muestra como devolver la zancada a su valor de fábrica.

Diálogo administrador de datos

Descripción:
Diálogo de alerta que permite al usuario control sobre algunos datos registrados.

Entrada:
Botón de borrado en la interfaz de usuario principal.

Salida:
Botón de cancelar, botón de borrar todas las actividades, botón de borrar todos los datos o botón de borrar al usuario.

Restricciones:
Mínimo de dos usuarios para permitir el borrado de los mismos.

Descripción:
Este diálogo le muestra al usuario una breve descripción de lo que puede hacer en el mismo, además de permitirle una interacción directa con botones que permiten el borrado de diferentes datos registrados. Tal y como se le indica al usuario, cualquier acción que no sea la de cancelar los cambios, repercutirá en un reinicio de la aplicación.

Diálogo fin de actividad

Descripción:
Diálogo de alerta que muestra al usuario los resultados de la actividad física realizada.

Entrada:
Botón de detener actividad de la interfaz de usuario principal.

Salida:
Botón de aceptar.

Restricciones:
Se tiene que haber iniciado previamente una actividad.

Descripción:
En este diálogo el usuario podrá ver los resultados de su entrenamiento, mostrándole por pantalla los siguientes datos:

- Fecha de finalización de la actividad.
- Distancia recorrida en kilómetros.
- Pasos realizados.
- Pasos en puntillas realizados.
- Pasos de talón realizados.
- Pasos pesados realizados.
- Calorías quemadas durante la actividad.
- Tiempo de duración de la actividad.

Cada texto a mostrar debe de disponer de unas unidades que especifiquen que se está midiendo. Los datos aquí mostrados son los que se guardan en la base de datos para su posterior uso.

4.3.3. Requisitos de rendimiento

El sistema está desarrollado para un solo dispositivo externo que se conecte mediante BLE y viceversa, el dispositivo externo solo está diseñado para una conexión instantánea con un cliente.

Las transacciones entre cliente y servidor mediante conexión BLE se realizan mediante una máquina de estados, minimizando el tiempo que los sensores están conectados y reduciendo el número de conexiones realizadas por el dispositivo para leer los datos. Es decir, cada vez que el usuario de un paso, el dispositivo móvil leerá los datos en el dispositivo externo que sirve de servidor.

No hay una manera fácil de calcular cuantas conexiones se pueden llegar a hacer durante una actividad física completa, ya que depende del propio rendimiento de usuario.

Partiendo de la base de que la distancia media, en kilómetros, recorrida en un entreno de running son cinco y que la altura media en España para hombres son 174 cm y para mujeres 163 cm, se puede despejar un número de pasos medio mediante la **ecuación uno.**

$$Kilometros\ recorridos = \frac{zancada\ (m) \cdot pasos}{1000} = \frac{altura\ (m) \cdot K \cdot pasos}{1000}$$

Ec.1. Ecuación base para el cálculo de la distancia recorrida en base de los pasos dados.

Siendo K la constante utilizada por la aplicación para la conversión de altura a zancada, se puede despejar de esa fórmula los pasos realizados en base a los kilómetros recorridos (ver **Ec.2.**).

$$Pasos = \frac{Kilometros\ recorridos \cdot 1000}{altura\ \ (m) \cdot K}$$

Ec.2. Ecuación despejada mostrando los pasos en base a los kilómetros recorridos y la altura del usuario.

Ahora se sustituyen los valores, siendo kilómetros recorridos, 5 km; la altura será la media para cada sexo, 1,74 m y 1,63 m; K será la constante del dispositivo: 0.414 (ver **Fórmula.3.**).

$$Hombre:$$

$$Pasos = \frac{5000}{1.74 \cdot 0.414} \cong 6941\ \ pasos$$

$$Mujer:$$

$$Pasos = \frac{5000}{1.63 \cdot 0.414} \cong 7410\ \ pasos$$

Ec.3. Ecuación con los diferentes valores para hombre y mujer, y su cálculo de pasos respectivo.

En base a este cálculo se obtiene que el hombre, da de media 6941 pasos y la mujer 7410 pasos. Esto significa que la interacción media de envío de datos entre cliente y servidor que ha de soportar el dispositivo móvil, serán de entre [6941 – 7410] interacciones por actividad.

4.3.4. Restricciones de diseño

El diseño de la aplicación está limitado por los puertos GATT necesarios para realizar la conexión inalámbrica con el dispositivo externo. Los puertos GATT son necesarios para identificar los diferentes servicios disponibles en el dispositivo externo, y sin ellos no es posible recibir ningún dato del mismo.

En este caso, como se va emplear el CC2541 SensorTag como dispositivo prototipo para la comprobación y testeo como dispositivo externo, se emplearán los puertos GATT especificados por Texas Instruments para la conexión a sus servicios.

En caso de desarrollar otro prototipo de dispositivo externo, bien habría que cambiar el código de la propia aplicación o bien, asegurarse de que los puertos del dispositivo externo sean los mismos.

4.3.5. Atributos del sistema

Por un lado, la aplicación no requiere de un mantenimiento extensivo por parte del usuario, simplemente es recomendable que este no acumule un excesivo número de actividades y que vaya limpiando el registro de tanto en tanto.

Por otro lado, la seguridad de la misma está garantizada. Esto se debe a que está diseñada para conectar con un tipo concreto de dispositivos BLE y no debería permitir la entrada de otros tipos de dispositivos externos en el sistema. Además, cada usuario tiene su propio registro separado del resto, impidiendo cualquier alteración de los datos del mismo.

Por último, la interacción con la base de datos por parte del usuario se ha limitado a la modificación de algunos atributos del mismo y del borrado de pequeñas partes de la misma. Solo se podrán borrar usuarios si ya hay más de uno registrado y los dispositivos no se podrán volver al registro pre-fabricado después de registrarlos en la base de datos.

4.3.6. Otros requisitos

Para el total y completo funcionamiento de la aplicación es necesario que el usuario posea un dispositivo externo prototipo con conectividad BLE y los adecuados puertos GATT. Si no se da el caso, la misma queda reducida a una pantalla de búsqueda de dispositivos que no da pie a más interacción con el usuario. En caso de no disponer de un dispositivo prototipo externo funcional se puede emplear un CC2541 SensorTag, al ser éste en el que se ha basado el diseño de la aplicación.

5. Diseño

5.1. Introducción

A continuación, se explicará el diseño tanto de la base de datos como de la misma aplicación Android.

5.2. Diseño de la base de datos

El modelado de la base de datos sigue el modelo entidad-relación. Este, consiste en buscar las entidades que describan los objetos que intervienen en el problema y las relaciones entre las entidades. Toda esta relación se ha de registrar en un esquema gráfico que tiene por objetivo, por una parte, de ayudar al programador durante la codificación y, por otra, al usuario para comprender el funcionamiento del programa.

Se ha dividido el diseño en varios sub apartados para ayudar a la comprensión lectora. Primero se expondrán las definiciones de las palabras usadas que estén relacionadas de forma directa con la creación de bases de datos.

En segundo lugar, se estipulan las especificaciones de desarrollo de la base de datos, es decir, que ofrece para el correcto funcionamiento y como lo debe hacer.

En tercer lugar, se muestra el desarrollo de un diagrama de bloques para la ilustración del modelo de la base de datos.

En cuarto y último lugar, se explica la asignación de tipos de dato a cada una de las características de la base de datos y como se ha llegado a esa conclusión.

5.2.1. Definiciones

Entidad: representación de un objeto individual concreto del mundo real.

Conjunto de entidades: conjunto de entidades con características similares o comunes.

Atributo: cada una de las características que posee una entidad individual y, que agrupadas, permiten la distinción de la misma de otras entidades del mismo conjunto.

Dominio: conjunto de valores posibles para un atributo.

Interrelación: la asociación o conexión entre diferentes conjuntos de entidades.

Grado: número de conjuntos de entidades que intervienen en una interrelación.

Clave: conjunto de atributos que identifican de forma univoca una entidad.

Clave principal: también conocida como primaria, es una clave candidata que se elige de forma arbitraria; usada siempre para identificar una entidad.

Tipo de dato: tipo de variable que cada atributo posee. Esto sirve para identificar si el atributo tiene que interpretarse como un número, como un texto o de otra forma.

NULL: tipo de dato que solo puede contener el valor *null*.

INTEGER: tipo de dato entero cuyo tamaño varía de 1 a 8 bytes.

REAL: tipo de dato decimal real.

TEXT: tipo de dato de texto que sigue los códigos de encriptación UTF-8, UTF-16BE y UTF-16LE.

BLOB: tipo de datos sin especificar. Se leen de la misma forma que se introdujeron en la base de datos.

UNIQUE: atributo que se le asigna a un tipo de dato para identificarlo como único, es decir, que no se puede encontrar duplicado dentro de la tabla.

NOT NULL: atributo que se le asigna a un tipo de dato para indicar que no puede contener un valor número como dato.

5.2.2. Especificaciones de desarrollo

Debido a que la naturaleza de este proyecto no es la de un pedido de un cliente, se detallan las diferentes especificaciones según la relación que se espera que tenga el usuario con la aplicación Android.

La base de datos tiene que ser capaz de administrar los datos personales del usuario del dispositivo externo, los datos del mismo dispositivo externo y ser capaz de guardar un registro de cada actividad física realizada por el usuario de la aplicación. Se podrán crear nuevos usuarios y registrar nuevos dispositivos, y cada usuario será libre de crear tantas actividades como le sean necesarias. No obstante, un dispositivo solo podrá registrar un usuario y un usuario sólo estará ligado a un dispositivo externo.

Los datos personales del usuario son: nombre, altura, peso y zancada. Además, tendrán que incluir que reto debe realizar el usuario ese día,

cuando fue la última vez que accedió al sistema y si ha completado o no el reto diario.

Del dispositivo externo se necesita almacenar el código de radio balizamiento, su propia UUID, el nombre dado por el fabricante y el nombre dado por el usuario de la aplicación.

Cada rendimiento físico realizado por el usuario va a constar de: la fecha en la que se ha realizado, los pasos realizados, los pasos de puntillas y de talón realizados, los pasos pesados, la distancia recorrida en kilómetros y las calorías quemadas.

5.2.3. Modelo entidad-relación

Según lo descrito en las especificaciones podemos encontrar tres conjuntos de entidades, siendo estos: dispositivos, usuarios y registros. Ya que se requiere la inserción de nuevos dispositivos, nuevos usuarios y nuevos registros, y la posterior capacidad para eliminarlos, es la forma más sencilla de agruparlos.

La identificación de las interrelaciones es sencilla ya que, según las especificaciones, un dispositivo define a un solo usuario. La interrelación entre dispositivos y usuarios es de grado 1:1. Como cada usuario puede registrar varias actividades, su relación es de un claro grado 1: N. A continuación, se muestra el primer esbozo del modelo (ver **Fig.7.**).

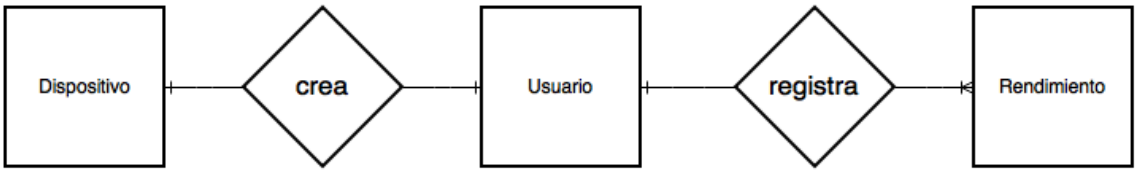

Fig.7. Diagrama de bloques representando el primer esbozo de la base de datos.

A continuación, han de identificarse los atributos correspondientes a cada conjunto de entidades. En las especificaciones se detallan de forma muy clara todas ellas, así que se va a proceder a numerarlas.

Para el dispositivo tenemos: código de la radio-baliza (a partir de ahora pasará a llamarse mDevice), su UUID (mServiceUUID), su propio nombre (mDeviceName) y el nombre dado por el usuario (GivenName).

Por parte del usuario sabemos que necesitamos los siguientes datos para crearlo: su nombre (Nombre), su peso (Peso), su altura (Altura), su

zancada (Zancada), el código de reto diario (RetoDiario), si se ha completado (RetoCompletado) y su último registro en la aplicación (LastLog).

También se ha especificado que cada rendimiento ha de tener: la fecha en la que se ha realizado (Fecha), los pasos realizados (Pasos), los pasos de puntillas (PasosPuntillas), los pasos de talón (PasosTalón), los pasos pesados (PasosPesados), distancia recorrida en kilómetros (Distancia) y calorías quemadas (CaloríasQuemadas)

Una vez agrupado cada conjunto con sus respectivos atributos, se ha de proceder a la asignación de claves primarias y foráneas. Para los datos del dispositivo externo, se emplea mDevice como la clave primaria. En el usuario, se dispondrá de un identificador propio proporcionado por SQLite como clave primaria y tendrá como llave foránea el mDevice. Esto se debe al requisito de que cada usuario solo puede registrarse con un dispositivo, de esta forma se pueden asociar ambas tablas y llevar un control más específico.

Los registros de actividad física de cada usuario tendrán como llave primaria un identificado proporcionado por SQLite y como llave foránea el nombre del usuario que realiza la actividad.

Ahora que tanto los atributos como las claves han sido asignadas, se puede hacer un diagrama de bloques completo de las relaciones en la base de datos (ver **Fig.8.**).

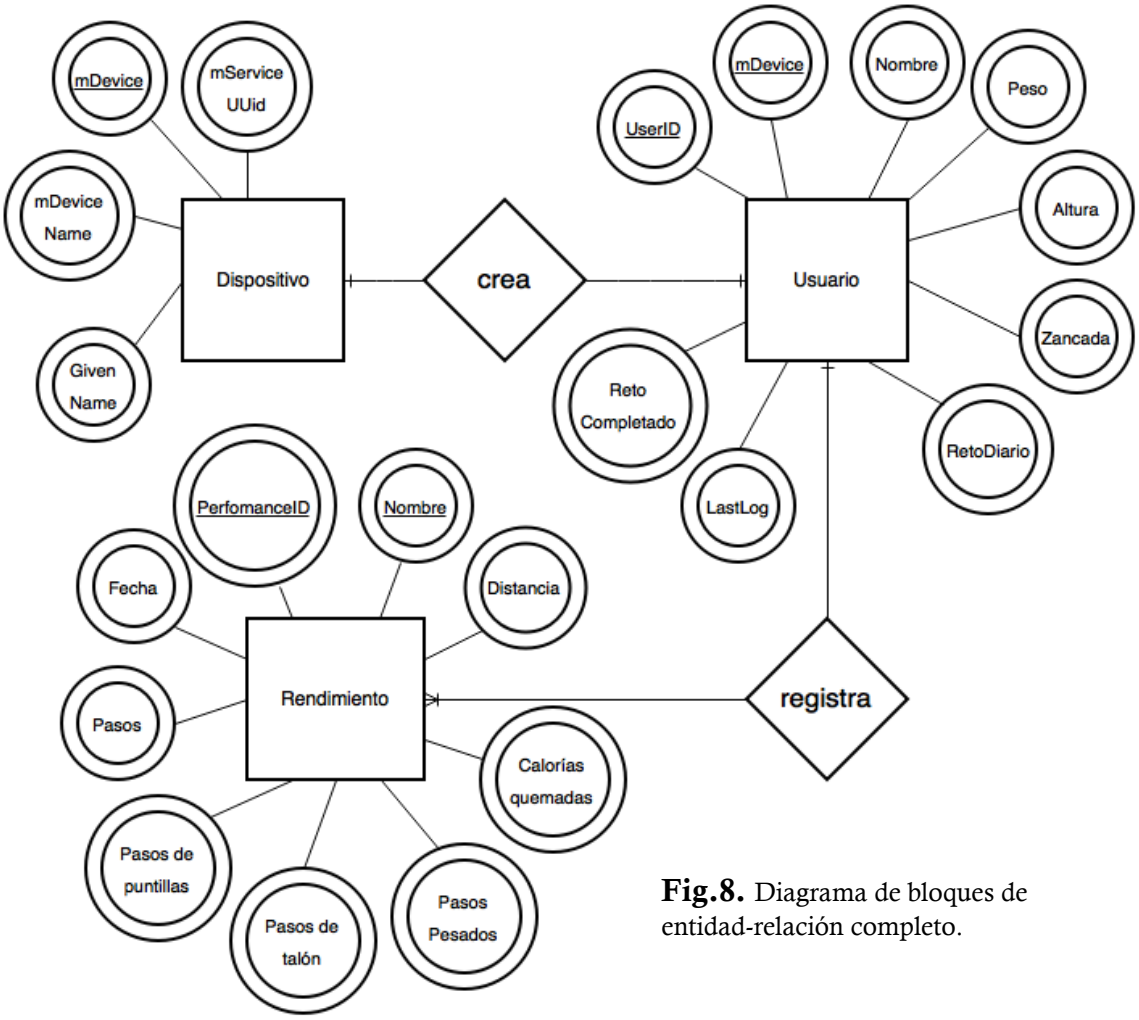

Fig.8. Diagrama de bloques de entidad-relación completo.

También se podría modelizar de la siguiente forma más sencilla (ver **Fig.9**):

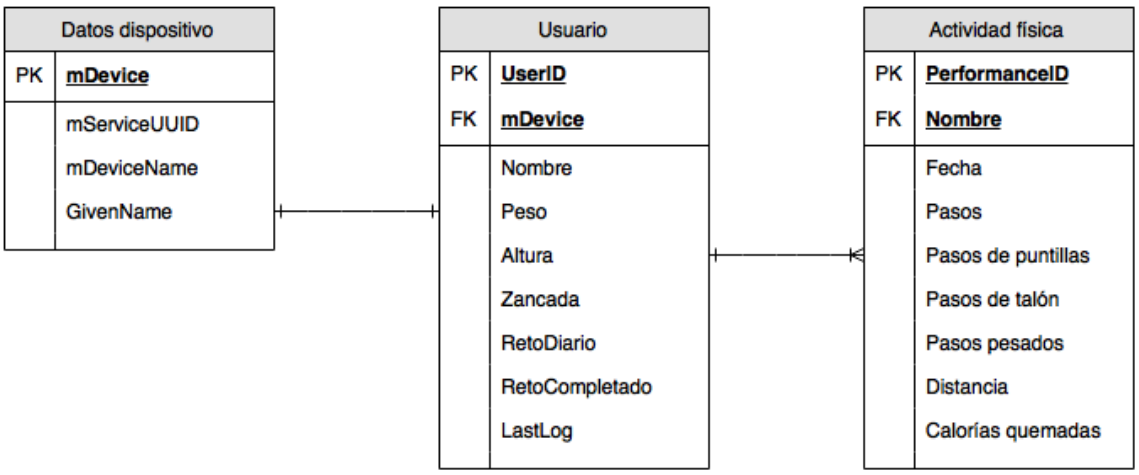

Fig.9. Modelo de la base de datos mediante tablas organizadas.

5.2.4. Asignación de valores a las columnas

Cada atributo relacionado a un conjunto de entidades necesita tener un tipo de valor asignado a la hora de ser programado (char, String, Integer…).

A continuación, se explica la asignación de dichos valores a los atributos previamente mencionados y como se ha llevado a cabo. Cabe comentar que SQLite ofrece unos valores deprecados comparado con MySQL, solo existiendo: NULL, INTEGER, REAL, TEXT y BLOB.

Datos del dispositivo:

mDevice: código de la radio baliza situada en el dispositivo externo. Se le asigna en la base de datos como TEXT UNIQUE NOT NULL.

mServiceUUID: código identificado propio del dispositivo externo. Se le asigna en la base de datos como TEXT NOT NULL.
mDeviceName: nombre que el fabricante ha proporcionado al dispositivo. Se le asigna en la base de datos como TEXT NOT NULL.

GivenName: nombre que el usuario ha proporcionado para identificar al dispositivo en la aplicación. Se le asigna en la base de datos como TEXT NOT NULL.

Todos los atributos del dispositivo pueden ser creados como entidades texto que no pueden adoptar el valor nulo (o *null*). Esto se debe a que los datos proporcionados por la radio baliza del dispositivo externo ya vienen en este formato y no hace falta cambiar el tipo de dato.

Usuario:

UsuarioID: código identificador propio de cada usuario. Se le asigna en la base de datos como INTEGER PRIMARY KEY NOT NULL. Con esta asignación la clave se irá autoincrementando cada vez que un nuevo dispositivo sea registrado en la base de datos, ofreciendo una numeración básica de interpretar que comienza en el 1 y va ascendiendo según el número de dispositivos registrados.

mDevice: código de la radio baliza del dispositivo externo. Se le asigna en la base de datos como TEXT UNIQUE NOT NULL. Es un identificador foráneo que relaciona esta tabla con los datos del dispositivo.

Nombre: nombre que el usuario ha seleccionado para identificarse en el sistema. Se le asigna en la base de datos como TEXT NOT NULL.

Altura: altura que ha proporcionado el usuario al sistema. Se le asigna en la base de datos como TEXT NOT NULL. El formato TEXT ha sido elegido sobre el formato INTEGER o BLOB debido a la facilidad en Android de extraer información de Strings, además de facilitar las cosas en el cuestionario de creación de usuario.

Peso: peso que ha proporcionado el usuario al sistema. Se le asigna en la base de datos como TEXT NOT NULL por los mismos motivos que la altura.

Zancada: zancada que ha proporcionado el usuario al sistema. Se le asigna en la base de datos como TEXT NOT NULL por los mimos motivos que la altura.

LastLog: última vez que el usuario registró actividad en la aplicación. Se le asigna en la base de datos como TEXT NOT NULL. Esto se debe a que es una codificación interna basada solamente para la aplicación consistente en dos números que indican día y mes separados mediante ':'. Ej.: Si el día 8 de octubre fuese la última vez que el usuario registró actividad, el código sería: *'8:10'*.

Reto diario: el reto que se le ha asignado en este día al usuario. Se le asigna en la base de datos como INTEGER NOT NULL. Los retos tienen claves identificativas dentro del sistema que van desde el 1 hasta el número de retos programados.

Reto completado: dato auxiliar para la aplicación que sirve para identificar si el usuario ya ha completado su reto o no. Dado que en SQLite no existe BOOLEAN como tipo de dato se le asignará un valor INTEGER que puede varias entre 0 o 1. Se le asigna en la base de datos como INTEGER NOT NULL.

Rendimiento de usuario:

PerformanceID: identificador propio de cada rendimiento. Se le asigna en la base de datos como INTEGER PRIMARY KEY AUTOINCREMENT. Esto proporciona a la base de datos un índice claro para relacionar el número de actividades físicas registradas.

Nombre: identificador foráneo empleado para relacionar cada usuario con sus diferentes actividades físicas. Se le asigna en la base de datos como TEXT UNIQUE NOT NULL.

Fecha: día en el que se ha realizado la actividad, además de en qué hora, minuto y segundo a finalizado. Se le asigna en la base de datos como TEXT NOT NULL. Se utiliza el formato TEXT debido a que la fecha se guardará siguiendo un formato de: 'dd/mm/yyyy – hh:mm'.

Esto permite realizar a posteriori consultas diferentes sobre la fecha y sobre la hora a la que se ha realizado la actividad.

Pasos: número de pasos que el usuario ha dado en la actividad física registrada. Se le asigna en la base de datos como INTEGER NOT NULL. Ya que no puede haber medios pasos, no se requiere de un número decimal para almacenar el número de pasos.

Pasos de puntillas: número de pasos en puntillas que el usuario ha realizado en la duración de la actividad física. Se le asigna en la base de datos como INTEGER NOT NULL por las mismas razones que los pasos.

Pasos de talón: número de pasos acabados con el talón que el usuario ha realizado en la duración de la actividad física. Se le asigna en la base de datos como INTEGER NOT NULL por las mismas razones que los pasos.

Pasos pesados: número de pasos que el usuario ha realizado con demasiada fuerza mientras estaba activa la actividad física. Se le asigna en la base de datos como INTEGER NOT NULL por las mismas razones que los pasos.

Distancia: kilómetros que el usuario ha recorrido mientras realizaba la actividad física registrada. Se le asigna en la base de datos como REAL NOT NULL. Ya que se pueden correr cinco kilómetros y medio, se tiene que asignar un valor de coma flotante a la distancia recorrida.

Calorías quemadas: número de calorías que el usuario ha quemado en el transcurso de la actividad física. Se le asigna en la base de datos como REAL NOT NULL. Esto se debe a que se pueden quemar cien calorías con veinticinco, por ende, hay que asignar un valor de coma flotante a las calorías quemadas.

5.3. Diseño de la aplicación

Aquí se detalla el proceso de diseño llevado a cabo para la aplicación Android. Para llevarlo a cabo se empleará el lenguaje de modelado UML (del inglés *Unified Modeling* Lenguage), qué es el estándar para modelar sistemas. Mediante el uso de varios tipos de diagrama para describir diferentes aspectos del sistema, UML describe de forma específica y visual el modelado de la aplicación.

5.3.1. Diagrama de casos de uso

Este diagrama describe los pasos que tiene el proceso mediante una secuencia iniciada por un actor. Dicho actor recorre los distintos pasos que da el sistema para proporcionar una respuesta deseada.

El actor es la entidad externa usuario de la aplicación, el caso de uso se entiende como cada actividad o función que el sistema realiza para satisfacer las necesidades del usuario y, las relaciones, son la comunicación entre actividades, incluyendo relaciones entre actividad y usuario.

Se destacan cuatro tipos de relación distintas: la relación de comunicación, la relación de inclusión, la relación de extensión y la generalización. El primero es la relación estándar que tienen los casos de uso.

El segundo, la relación de inclusión, es una forma de interacción donde un caso de uso dado, puede incluir a otro distinto, donde el primero depende del segundo para conseguir un resultado. Se representa mediante una flecha de punta abierta con línea discontinua que incluye una etiqueta '«*include*»'.

El tercero, la relación de extensión, indica que el comportamiento del primer caso de uso referenciado se extiende a otro distinto. Se representa mediante una flecha de punta abierta con línea discontinua, con la etiqueta '«*extend*»'.

El cuarto y último, la generalización es una manera de construir clasificaciones entre conceptos previamente representados en diferentes jerarquías de clases. Se representa mediante una línea sólida terminada en un triángulo dibujado.

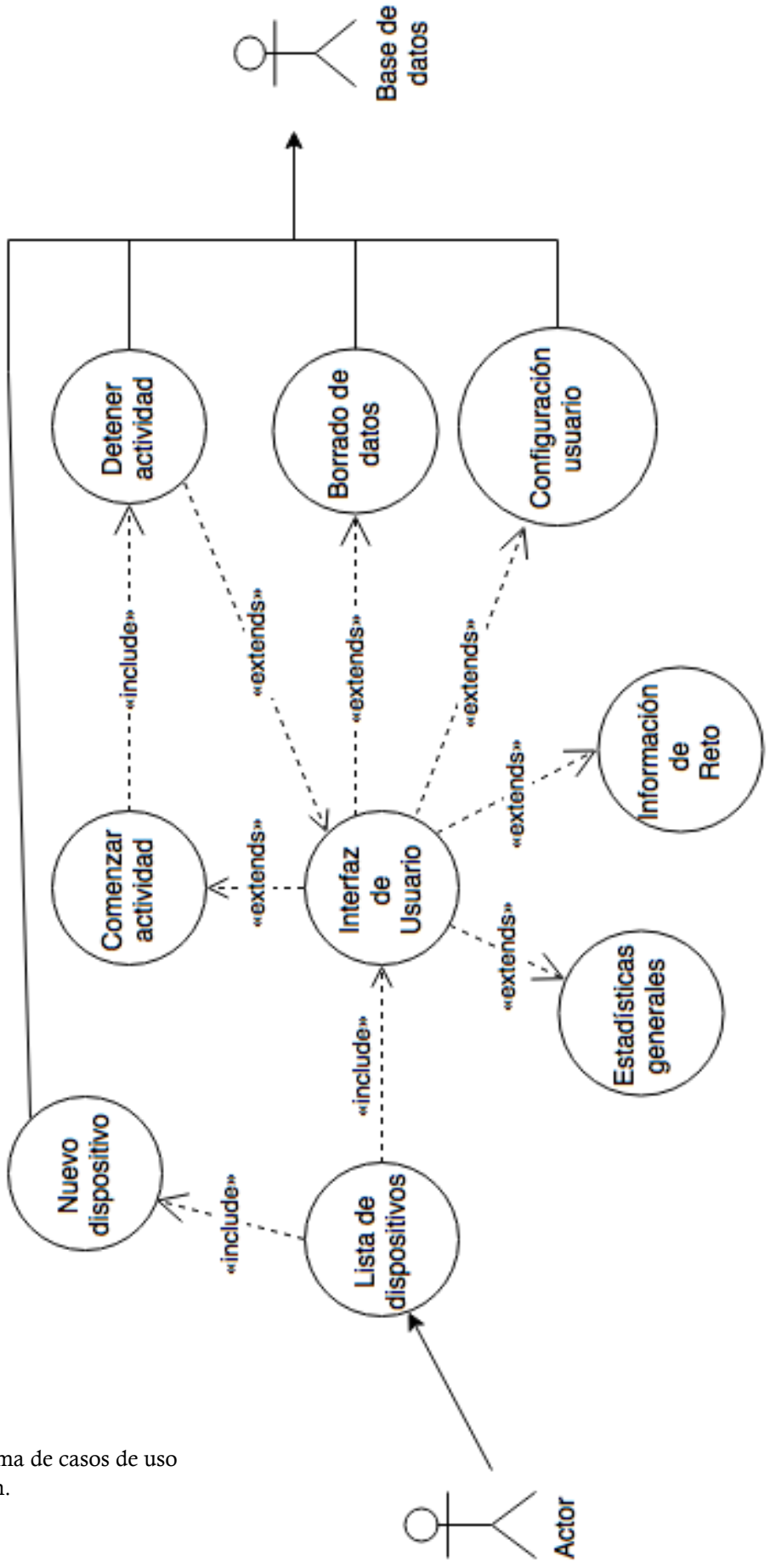

Fig.10. Diagrama de casos de uso para la aplicación.

Como se puede observar en el diagrama (ver **Fig.10.**), se considera que el usuario quiere empezar mediante la selección de su dispositivo en la lista correspondiente. Una vez se ha seleccionado el dispositivo correspondiente, la aplicación detecta un nuevo dispositivo o bien, le permite al usuario acceder a la interfaz principal. Al registrar un nuevo dispositivo, se notifica a la base de datos del cambio y se reinicia la aplicación.

Cuando el usuario accede a la interfaz principal, puede interactuar con ella de diversas formas. Cuando el usuario decide comenzar una actividad, la aplicación guarda unos registros que han de ser pasados para que "detener actividad" funcione. Al finalizar la actividad los registros se guardan en la base de datos y se vuelve al punto de inicio de la interfaz de usuario. No pasa lo mismo con el borrado de datos y la configuración de usuario, que acaban con el reinicio de la aplicación.

La interacción del usuario con la información del reto o las estadísticas generales no desembocan a otros casos; permitiendo así la continuidad de la interfaz de usuario principal.

5.3.2. Diagrama de clases

El diagrama de clases en UML es una estructura estática que describe el funcionamiento de un sistema mostrando las clases del sistema, sus atributos, métodos y sus relaciones. Debido a que la aplicación gira entorno a una sola actividad para evitar la pérdida de conexión del dispositivo, el diagrama se mostrará en seis partes.

MainActivity

La primera clase creada por la aplicación y la actividad principal de la misma (ver **Fig.11.**), esta extiende la clase 'Activity' proporcionada por el SDK de Android, y es necesaria para el correcto funcionamiento de la aplicación, ya que contiene los métodos del ciclo de vida. También implementa la interfaz SensorEventListener, que permite la comunicación con los sensores disponibles en el dispositivo móvil.

MainActivity tiene una relación con la base de datos conocida como asociación, debido a que sus respectivos tiempos de vida no están relacionados entre sí. El mismo caso ocurre con la actividad conocida como NewDeviceActivity.

La actividad principal tiene una dependencia con la clase SensorTagData, ya que es la que le permite extraer los datos de los sensores localizados en el prototipo de dispositivo externo CC2541 SensorTag. También tiene dependencia de los diferentes diálogos que se

le lanzan al usuario, esto se debe a que son la manera principal de comunicación entre ambos.

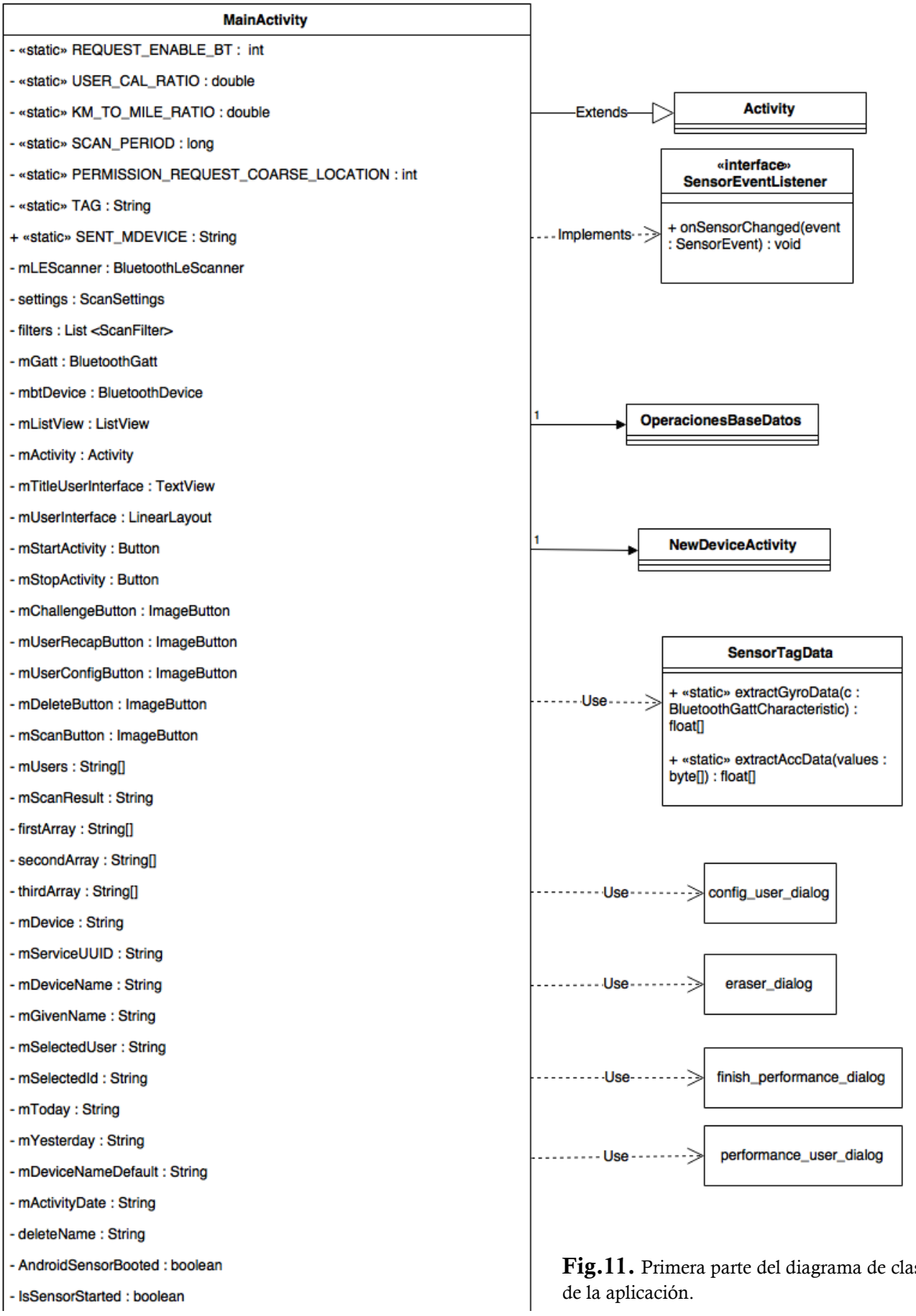

Fig.11. Primera parte del diagrama de clases de la aplicación.

- mAccX : float
- mAccY : float
- mAccZ : float
- mGyroX : float
- mGyroY : float
- mGyroZ : float
- mInclinationAngle : double[]
- mStartTime : double
- mFinishTime : double
- mCountBadSteps : int
- mCountBadSteps2 : int
- mCountHeavySteps : int
- mCountSteps : int
- mOffsetCountStep : int
- howManyUsers : int
- howMuchIds : int
- howManyDevices : int
- mTodaysChallenge : int
- datos : OperacionesBaseDatos
- mBase : BaseDatosDispositivos
+ mCalendar : Calendar
- «static» «final» GYROSCOPE_SERVICE : UUID
- «static» «final» GYROSCOPE_DATA_CHAR : UUID
- «static» «final» GYROSCOPE_CONFIG_CHAR : UUID
- «static» «final» GYROSCOPE_PERIOD_CONFIG : UUID
- «static» «final» ACCELEROMETER_SERVICE : UUID
- «static» «final» ACCELROMETER_DATA_CHAR : UUID
- «static» «final» ACCELEROMETER_CONFIG_CHAR : UUID
- «static» «final» ACCELROMETER_PERIOD_CONFIG : UUID
+ «static» «final» CONFIG_DESCRIPTOR : UUID
- «static» «final» MSG_GYRO : int
- «static» «final» MSG_ACC : int
- «static» «final» MSG_PROGRESS : int
- «static» «final» MSG_DISMISS : int
- «static» «final» MSG_CLEAR : int
- mBluetoothAdapter : BluetoothAdapter
- mConnectedGatt : BluetoothGatt
- mProgress : ProgressDialog
- mScanCallback : ScanCallback
- «final» gattCallback : BluetoothGattCallback

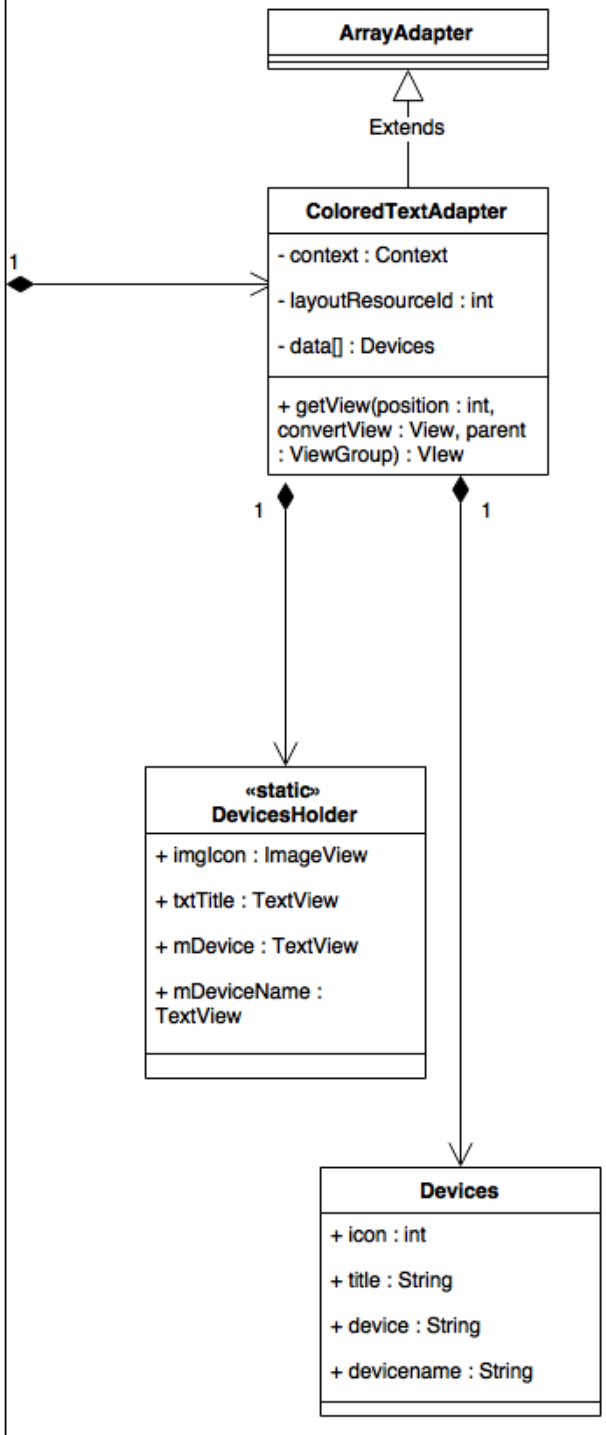

Fig.12. Segunda parte del diagrama de clases de la aplicación.

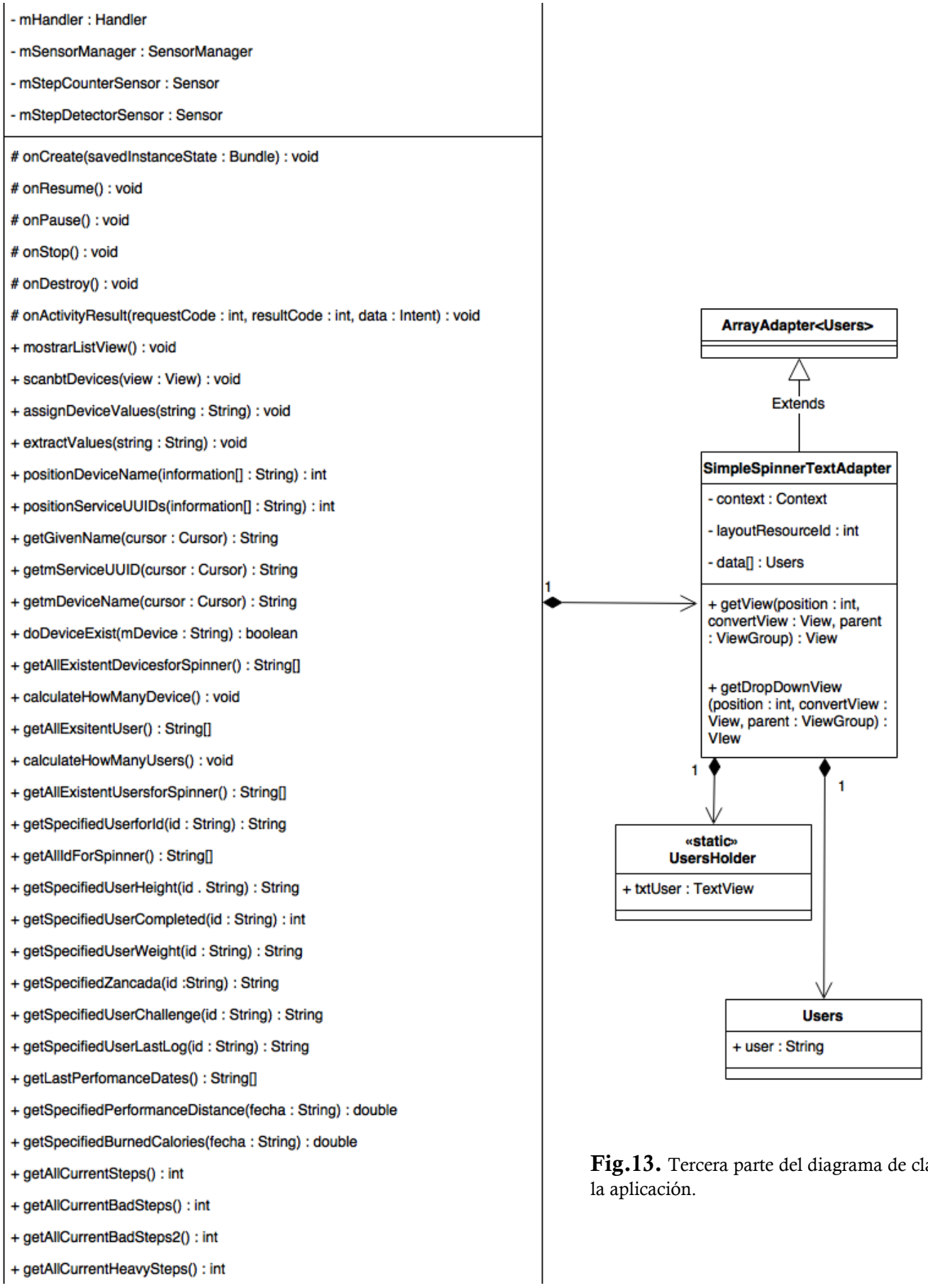

Fig.13. Tercera parte del diagrama de clases de la aplicación.

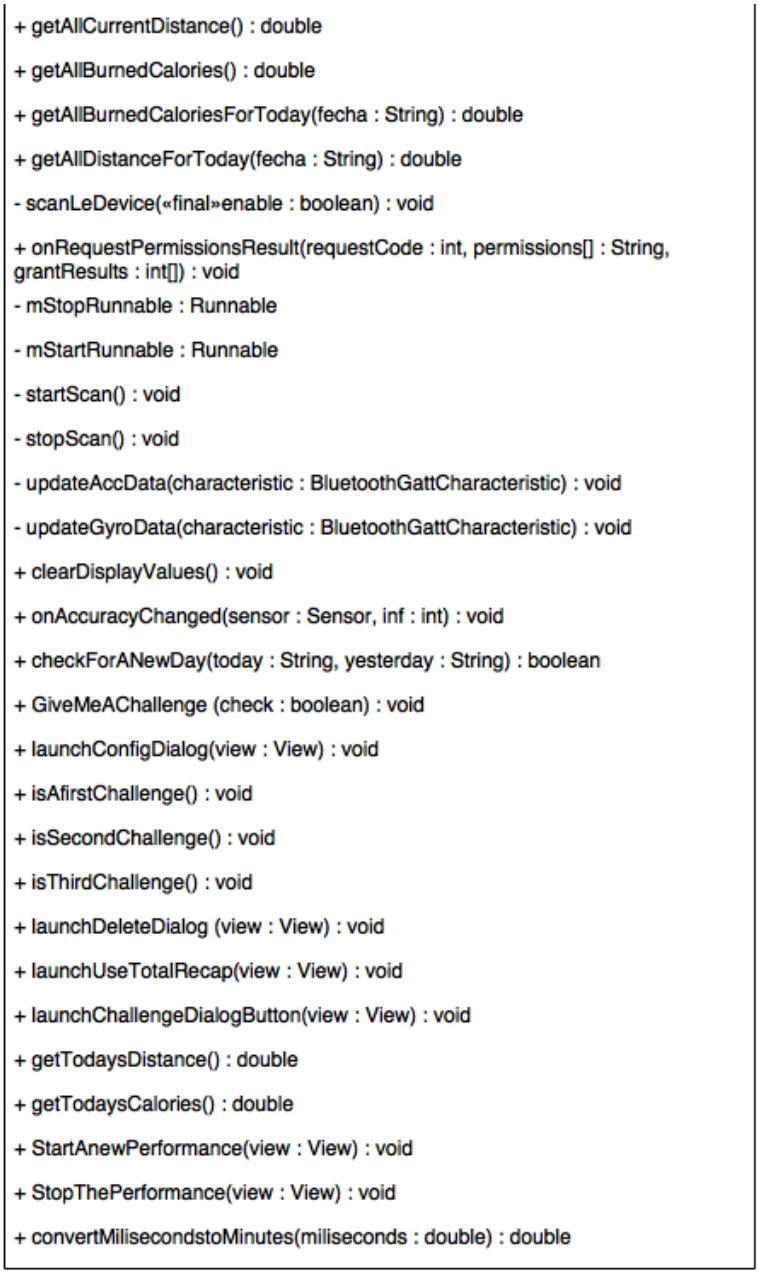

Fig.14. Cuarta parte del diagrama de clases de la aplicación.

Se pueden observar varias relaciones de relación de agregación por valor (ver **Fig.12.**) (ver **Fig.13.**); estas relaciones se forman entre clases que dependen la una de la otra para la creación de tipos de datos no básicos o formatos no estandarizados. En el caso de una relación de agregación por valor, el tiempo de vida del objeto incluido está ligado al objeto que lo crea.

En la aplicación, cuando se necesita mostrar al usuario la lista de dispositivos encontrados, se crea una instancia del ColoredTextAdapter. Mediante esta instancia, se dará formato al ListView donde aparecerá la información para el usuario, mostrando un icono relacionado con el dispositivo, el nombre dado por el usuario al dispositivo, el código de radio baliza del mismo y el nombre dado por el fabricante al producto. También se creará una instancia de SimpleSpinnerTextAdapter cuando la aplicación muestre la lista desplegable con los usuarios ya registrados.

Ambas actividades instanciadas en la principal extienden ArrayAdapter; ésta es una clase proporcionada por el SDK de Android y nos permite sobrescribir la misma para editar una interfaz visual customizada por los desarrolladores.

La actividad principal contiene un alto número de variables declaradas; la misión principal de éstas es la comunicación entre la base de datos, el dispositivo y la propia actividad. Además, se han declarado todas las vistas usadas para mostrar información al usuario para poder inicializarlas en el método onCreate() y poder referenciarlas en cualquier otro método de la actividad.

Tanto en la **Fig.11.** como en la **Fig.12.** se pueden observar variables estáticas. Estas variables son globales para toda la aplicación y son usadas para definir identificadores exclusivos. Entre estos identificadores exclusivos se encuentran los identificadores de puertos GATT necesarios para la conexión con dispositivo prototipo externo.

En MainActivity también se han establecido métodos que ayudan a: conectar con el dispositivo externo, comunicación con la base de datos, gestión de la base de datos, gestión de alertas de usuario y métodos auxiliares (ver **Fig.13.**) (ver **Fig.14.**).

NewDeviceActivity

Esta clase contiene la actividad que es llamada cuando en MainActivity se detecta un nuevo dispositivo sin registrar. Dicha actividad (ver **Fig.15.**) muestra al usuario un formulario donde debe rellenar los campos que se le piden: nombre para el dispositivo, nombre de usuario, peso y altura. También tiene métodos relacionados con la base de datos que son empleados cuando se selecciona un usuario previamente registrado para añadir el nuevo dispositivo.

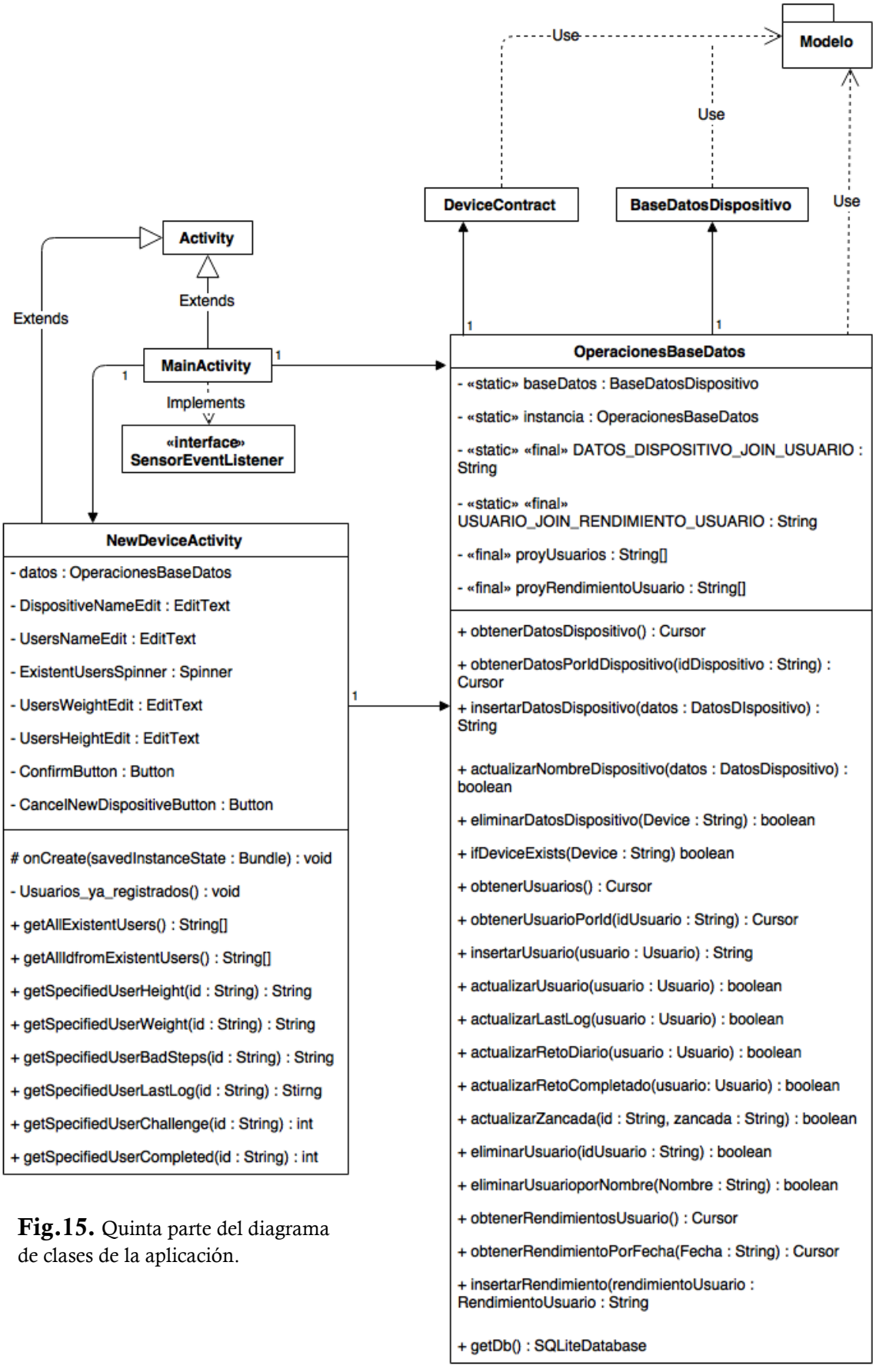

Fig.15. Quinta parte del diagrama de clases de la aplicación.

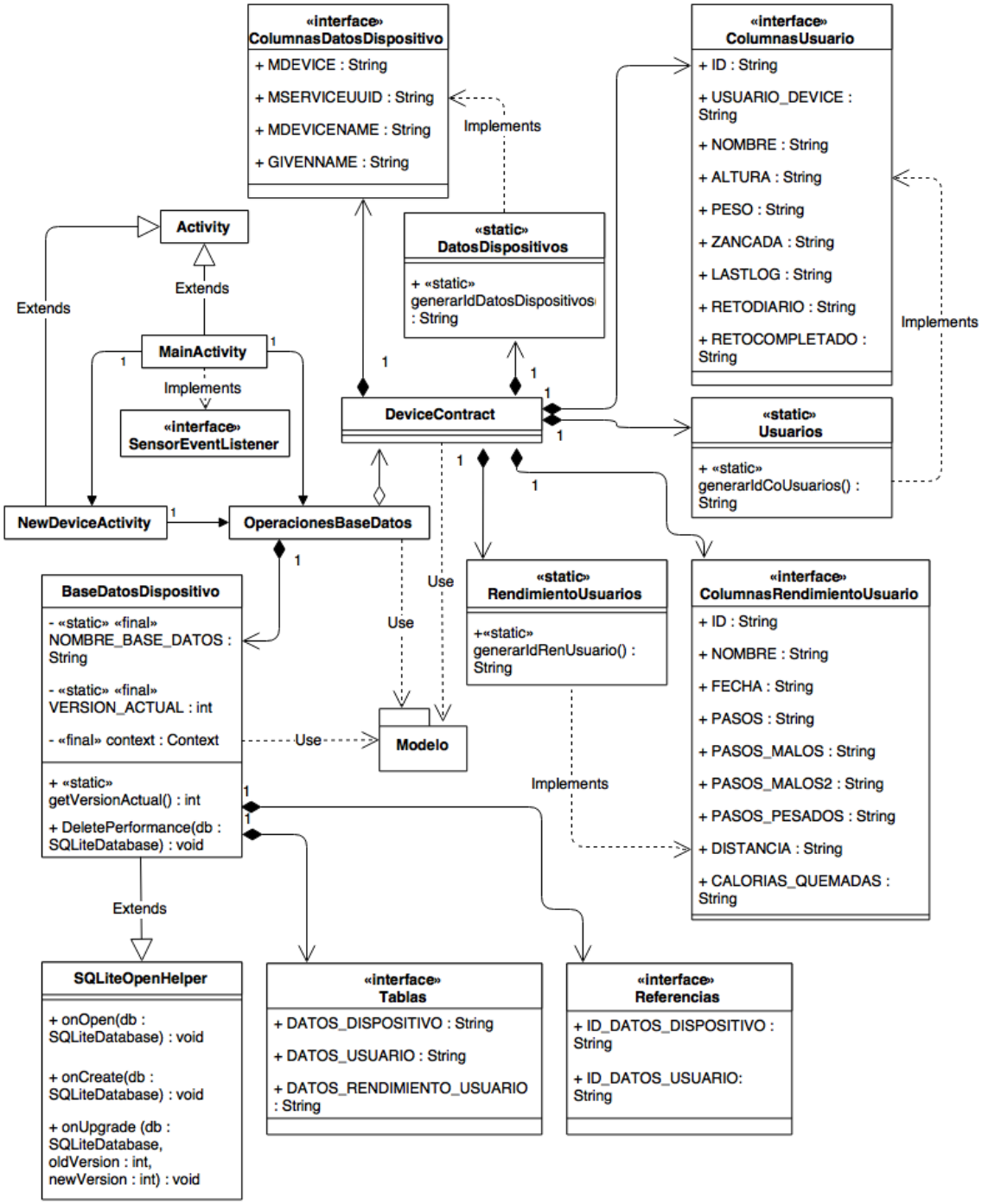

Fig.16. Sexta parte del diagrama de clases de la aplicación

OperacionesBaseDatos

Cuando MainActivity y NewDeviceActivity son ejecutadas, crean una asociación con esta clase (ver **Fig.16.**). OperacionesBaseDatos se encarga de organizar el CRUD (del inglés Create Read Update and Delete) para la base de datos de la aplicación. Es decir, crea los métodos que permiten a las demás clases la creación de datos en las tablas de la base de datos, su lectura, los métodos que permiten modificarlas y los que permiten borrarlas.

Dado que a la hora de modelar la base de datos se crearon relaciones con llaves foráneas entre las diferentes tablas, esta clase proporciona unas constantes que indicas las relaciones entre tablas y llaves foráneas. Además, indica cómo deben ser las proyecciones de las tablas relacionadas cuando se pidan los datos pertenecientes a estas.

Esta clase tiene una asociación directa con DeviceContract y BaseDatosDispositivo (ver **Fig.16.**). Esto se debe a que el CRUD de la base de datos necesita usar las interfaces creadas en estas clases para la interacción con la misma. Estas tres clases usan un paquete llamado Modelo; dicho paquete contiene los constructores necesarios para cada tabla de la BBDD.

DeviceContract

Su función principal es de contenedor de metadatos mediante el establecimiento de interfaces y de clases estáticas. Las interfaces establecen los nombres que ha de tener cada columna de cada tabla y las clases estáticas implementan dichas interfaces para poder interactuar directamente con ellas. Las clases estáticas también contienen generadores de UUID aleatoria por si son necesarias en el registro de algún usuario, dispositivo o actividad.

DeviceContract tiene una relación de agrecación por referencia con OperacionesBaseDatos. Esta relación viene dada por el hecho de que esta última clase, emplea a la primera como generador de interfaces para la rápida consulta de datos en la base de datos; además, el ciclo de vida de la primera no está ligada al ciclo de vida de la segunda. Las relaciones de DeviceContract con las interfaces y clases estáticas es de agregación por valor. Como su relación con la base de datos, DeviceContract solo sirve como contenedor de metadatos y una vez su ciclo de vida llegue a su fin, el ciclo de vida de las interfaces y clases estáticas también llegará a su fin.

BaseDatosDispositivo

Esta clase contiene los metadatos necesarios para la creación de la BBDD y su mantenimiento. Esto se consigue mediante la creación de valores globales que contienen el nombre deseado para la base de datos y su versión actual.

Se le han asignado dos métodos, uno para la actualización de la base de datos, con el cual borra todos los datos almacenados y crea de nuevo las tablas. El segundo es para el borrado de actividades que, en lugar de implementarlo en la CRUD, se considera que borrar una a una las actividades es un proceso tedioso y sólo se permite el borrado sistemático de las mismas.

BaseDatosDispositivo implementa SQLiteOpenHelper, esta es una clase localizada en el SDK de Android que permite la creación y modificación de bases de datos por aplicaciones Android. De ella, se implementan los métodos de onCreate(db : SQLiteDatabase), onUpdate(db : SQLiteDatabase) y onOpen(db : SQLiteDatabase).

5.4. Conclusiones

Respecto al modelado de la BBDD, se han especificado las pautas que debería seguir la BBDD para la aplicación y, mediante el modelo de Entidad-Relación, se ha podido diseñar con claridad la arquitectura de la BBDD a utilizar; además, de la asignación de tipos de datos de la misma.

Respecto al modelado de la Aplicación, mediante el diagrama de casos de uso se ha podido observar como interactúa la aplicación con un cliente desde su punto de vista, y con el diagrama de clases se ha podido ilustrar las diferentes interacciones entre las clases del programa.

6. Implementación

6.1. Introducción

En el apartado de implementación se muestra paso a paso como se ha diseñado la aplicación para cumplir con las especificaciones requeridas. Se muestra tan solo el código más relevante.

6.2. Desarrollo

Cliente

AndroidManifest.xml

Archivo de configuración necesario en cualquier aplicación Android. Contiene los parámetros básicos para la configuración de la misma. Entre estos parámetros se encuentran (ver **Fig.17.**):

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Pro Run"
    android:supportsRtl="true"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
    <activity
        android:name=".MainActivity"
        android:label="Pro Run"
        android:theme="@style/AppTheme.NoActionBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".NewDeviceActivity"></activity>
</application>
```

Fig.17. Parte del AndroidManifest.xml donde se aplica la configuración de la aplicación.

- **allowBackup:** permite que la aplicación sea utilizada en los archivos de restauración del dispositivo móvil objetivo.
- **icon:** permite declarar el icono usado por la aplicación.
- **label:** permite declarar el nombre usado por la aplicación.
- **supportsRtl:** atributo que permite o no a la aplicación el uso de *layouts* de derecha a izquierda.
- **theme:** atributo que declara el tema a usar por la aplicación.

Después de los parámetros de configuración general de la aplicación, se pasa a declarar las actividades que se encuentran en la aplicación y los atributos necesarios para su funcionamiento (ver **Fig.18.**).

Al principio del manifiesto se tienen que declarar los permisos que necesita la aplicación, así como las diferentes partes de *hardware* del dispositivo móvil que esta utilizará.

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

<uses-feature
    android:name="android.hardware.bluetooth_le"
    android:required="true" />
<uses-feature
    android:name="android.hardware.sensor.stepdetector"
    android:required="false" />
<uses-feature
    android:name="android.hardware.sensor.stepcounter"
    android:required="false" />
```

Fig.18. Parte del AndroidManifest.xml donde se declaran los permisos y las características que usa la aplicación.

Los permisos requeridos por la aplicación, en resumen, permite el uso del Bluetooth y la geolocalización del dispositivo móvil. Del mismo dispositivo, requiere el uso de los sensores de contador de pasos y de detención de pasos. Además, avisa de que necesita la característica del Bluetooth Smart.

MainActivity.java

Esta es la actividad principal de la aplicación y también la inicial. Desde ella se le permite al usuario escanear dispositivos, comenzar y detener actividades, información sobre retos, configurar su usuario, acceder a estadísticas generales y el borrado de datos. A través de la misma actividad se puede acceder al registro de los nuevos dispositivos.

La clase hereda de Activity.java, esta es una clase disponible en la SDK de Android que incorpora el ciclo de vida de la aplicación y algunos métodos necesarios para el funcionamiento de la aplicación.

En el método onCreate() se inicializan todas las variables y vistas necesarias; además, mediante condicionantes se revisa que el usuario ha activado las características necesarias para el correcto funcionamiento de la aplicación (ver **Fig.19.**). Se hacen dos condicionantes, una para la geo localización y otra para el puerto Bluetooth.

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){
    //Android M Permission check
    if(this.checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION)!=PackageManager.PERMISSION_GRANTED){
        //Creación de la alerta de dialogo
        final AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("This app needs location access");
        builder.setMessage(R.string.locationAccessDescription);
        builder.setPositiveButton(android.R.string.ok, null);
        builder.setOnDismissListener((dialog) -> {
                                        //Aunque no se acepte la localización la activamos, si no el dispositivo no iria
                                    ActivityCompat.requestPermissions(mActivity,
                                            new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                                            PERMISSION_REQUEST_COARSE_LOCATION);
        });
        builder.show();
```

Fig.19. Condición encontrada en el método onCreate() para controlar los permisos de la aplicación.

Cuando se ejecuta el método onResume() se comprueba si la conexión Bluetooth está activado. Si no está habilitado, se le insta al usuario su activación, si está habilitado, se configura la conexión Bluetooth a realizar.

El escaneo para buscar el dispositivo Bluetooth puede configurarse tres maneras diferentes. La primera, es el SCAN_MODE_LOW_POWER, siendo el default, consume muy poca energía del sistema, pero tarda en establecer conexión. La segunda, es el SCAN_MODE_BALANCE, está configuración tiene término medio de consumo de recursos y de tiempo de conexión. La tercera y última es SCAN_MOE_LOW_LATENCY, ordenado que la búsqueda consuma los recursos necesarios para que la conexión sea lo más rápida posible.

En el método onPause(), se comprueba si hay una conexión Bluetooth establecida, y si es así se cortará la comunicación con el dispositivo conectado. También pondrá en pausa los diferentes hilos usados a lo largo de la actividad. Se puede forzar la desconexión total con los dispositivos Bluetooth escribiendo scanLeDevice(false) fuera de la conexión.

Cuando la actividad pasa por completo a segundo plano se ejecuta el método onStop(), desconectado de forma inmediata cualquier conexión con un servidor GATT y si la aplicación es destruida del todo, el método onDestroy() cerrará el propio puerto GATT del móvil y asigna su valor a nulo.

Después de la configuración del ciclo de vida de la aplicación, aparece el método onActivityResult. Este método se encarga de notificar al sistema

el éxito de la conexión a un sistema de Bluetooth externo. El método mostrarListView() se encarga de añadir los dispositivos Bluetooth escaneados a la lista para mostrarlos al usuario. Dentro de este método se da formato a la lista y se le asigna un evento (ver **Fig.20.**) para redirigir al usuario al registro de un nuevo dispositivo o a la interfaz principal en base del nombre del dispositivo. Debido a la naturaleza de la comparación, se le prohíbe al usuario el registro del dispositivo con el nombre estándar de nuevo dispositivo o dejarlo en blanco.

```
//Designar un evento de OnItemClick para cada elemento de la ListView
mListView.setOnItemClickListener((parent, view, position, id) → {

        if (mGivenName.equals("New device")) {
            //Si el dispositivo contiene el nombre default, se instará al usuario a crear un usuar
            final AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setTitle("New device found!");
            builder.setMessage("It seems that we found a new device. Please, select a n...");
            builder.setPositiveButton(android.R.string.ok, null);
            builder.setOnDismissListener((dialog) → {
                    //Comenzar actividad NewDeviceActivity, para la creación de usuario
                    Intent intent = new Intent(MainActivity.this, NewDeviceActivity.class);
                    intent.putExtra(SENT_MDEVICE, mDevice); //Envio de datos a la otra actividad
                    scanLeDevice(false); //Desactivar el escaneo de dispositivos
                    startActivity(intent);
            });
            builder.show();
```

Fig.20. Asignación de un evento de escucha de ítem seleccionado a la lista de dispositivos.

Si se redirige al usuario a la interfaz principal, el mismo método asigna los valores pertinentes a cada variable para el correcto funcionamiento de la interfaz. Esto se consigue habilitando botones, mostrando vistas ocultas y dándole al usuario un reto diario si este no dispone de uno completado para el mismo día.

En MainActivity.java también se implementan los métodos necesarios para el registro del dispositivo en la BBDD. Para este fin, la aplicación comprueba si el dispositivo ya existe, si es así llama al método extractValues(String string), que asigna a las variables lo datos almacenados. En el caso contrario, ejecuta el método assignDevicesValues(String string); este extrae la información del dispositivo al que se ha conectado la aplicación y la almacena en la BBDD.

Se han declarado muchos métodos auxiliares cuyo objetivo es la localización de datos en la base de datos de forma concreta o la

extracción de los mismos. También se implementan los eventos de los botones con los que interactúa el usuario.

La conexión con el dispositivo prototipo externo se ha diseñado mediante cuatro máquinas de estados distintas (ver **Fig.21.**). La primera se encarga de la activación de los sensores, la segunda de la lectura de los datos, la tercera de la activación del a notificación de cambios y la cuarta y última, de las notificaciones entre las diversas máquinas de estados.

Al realizar la implementación mediante máquinas de estados, el sistema se adapta a la arquitectura GATT, ya que los sensores están en un estado de deshabilitado o dormido cuando no son llamados para su uso; esto facilita la aplicación de un sistema que no consume casi energía para la lectura de datos externos.

```
private void enableNextSensor(BluetoothGatt gatt){
    BluetoothGattCharacteristic characteristic;
    switch (mState){
        case 0:
            Log.d(TAG, "Enabling Gyroscope");
            characteristic = gatt.getService(GYROSCOPE_SERVICE)
                    .getCharacteristic(GYROSCOPE_CONFIG_CHAR);
            characteristic.setValue(new byte[]{7});
            break;
        case 1:
            Log.d(TAG, "Enabling gyroscope period");
            characteristic = gatt.getService(GYROSCOPE_SERVICE)
                    .getCharacteristic(GYROSCOPE_PERIOD_CONFIG);
            characteristic.setValue(new byte[]{0x0A});
            break;
        case 2:
            Log.d(TAG, "Enabling Accelerometer");
            characteristic = gatt.getService(ACCELEROMETER_SERVICE)
                    .getCharacteristic(ACCELEROMETER_CONFIG_CHAR);
            characteristic.setValue(new byte[]{0x01});
            break;
        case 3:
            Log.d(TAG, "Enabling accelerometer period");
            characteristic = gatt.getService(ACCELEROMETER_SERVICE)
```

Fig.21. Máquina de estados para la activación de los diversos sensores.

NewDeviceActivity.java

Esta actividad es llamada, cuando la aplicación detecta que el dispositivo que el usuario ha seleccionado no ha sido previamente registrado en la BBDD. Su objetivo es actuar como un formulario

simple, que permite al usuario introducir los diferentes datos que se le piden y su posterior guardado en la BBDD.

Como MainActivity.java, hereda de Activity.java para tener su ciclo de vida. A su vez, recibe un Intent de MainActivity.java que contiene el código de radio baliza del dispositivo localizado para su posterior uso por la misma actividad.

Es una actividad de transición e, de forma independiente a la interacción con el usuario, acaba redirigiendo a este a la actividad principal. Implementa métodos auxiliares para la interacción con la base de datos, pero ningún método que sea empleado en otra clase.

SensorTagData.java

Es una clase auxiliar de MainActivity.java que se encarga de la extracción de datos del dispositivo prototipo externo conectado a la aplicación. Esto se consigue mediante dos métodos que, a partir de la característica, convierten la información que ha sido dada en bytes, en datos que pueden ser usados directamente por la aplicación (ver **Fig.22.**).

```
public class SensorTagData {

    /**
     * Método que convierte los bytes leídos de la característica en parámetros que se pueden
     * almacenar en un array de float
     * @param c característica a leer
     * @return array float con datos del gyroscopio
     */
    public static float[] extractGyroData(BluetoothGattCharacteristic c){
        //byte[] value = c.getValue();
        float gyroData[] = new float[3];
        gyroData[0] = shortSignedAtOffset(c, 0) * (500f / 65536f) * -1;
        gyroData[1] = shortSignedAtOffset(c, 2) * (500f / 65536f);
        gyroData[2] = shortSignedAtOffset(c, 4) * (500f / 65536f);

        return gyroData;

    }
```

Fig.22. Método que convierte los valores en bytes a formato float para su posterior uso.

Los datos de conversión empleados son los del dispositivo externo a validar, el CC2541 SensorTag, y Texas Instruments los proporciona de forma gratuita. El envío de bytes de la característica es enviado y convertido de tal forma que se pueden separar mediante offset los valores diferentes de los ejes XYZ. También implementa los métodos que usa la propia clase para declarar los offset necesarios en la conversión.

OperacionesBaseDatos.java

Esta clase contiene una colección de métodos que establecen el CRUD de la BBDD. También implementa una serie de constantes globales que indican la relación de las distintas llaves foráneas y sus tablas; además de un método que devuelve la BBDD con la que se está trabajando. Los métodos relacionados con el CRUD devuelven:

- Si el método es para leer, devuelve un cursor que contiene los datos de la tabla seleccionado.
- Si el método es para la creación o inserción de datos, tiene que devolver la ID del nuevo elemento.
- Si el método es para la actualización de datos tiene que devolver un booleano que compruebe que se ha realizado con éxito la operación (ver **Fig.23.**).
- Si el método es para el borrado de datos, tiene que devolver un booleano por los mismos motivos que la actualización.

```
/**
 * Se actualizan los datos a través de nuevos valores que vienen en una instancia
 * DatosDispositivo. Esta solo se lleva a cabo si el registro tiene el mismo id que los datos y
 * mismo mDevice.
 */

public boolean actualizarNombreDispositivo(DatosDispositivo datos){
    SQLiteDatabase db = baseDatos.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(DeviceContract.DatosDispositivos.GIVENNAME, datos.GivenName);

    String selection = String.format("%s=?",
            DeviceContract.DatosDispositivos.MDEVICE);
    final String[] whereArgs = {datos.mDevice};

    int resultado = db.update(Tablas.DATOS_DISPOSITIVO, values, selection, whereArgs);
    return resultado > 0;
}
```

Fig.23. Método que actualiza el nombre del dispositivo y devuelve un booleano con el resultado de la transacción.

DeviceContract.java

Aquí se declaran como interfaces (ver **Fig.24.**) las diferentes columnas de cada tabla de la BBDD. Las columnas se definen como String ya que solo son un nombre identificativo y no afectan directamente a la creación de esta.

Esta clase contiene otras clases estáticas, cada una relacionada con una tabla de la BBDD y que implementan las interfaces correspondientes. Con ellas, otras actividades pueden llamar los datos guardados en las interfaces de esta clase.

```
interface ColumnasUsuario{
    String ID = "id";
    String USUARIO_DEVICE = "Usuario_Device";
    String NOMBRE = "Nombre";
    String ALTURA = "Altura";
    String PESO = "Peso";
    String ZANCADA = "Zancada";
    String LASTLOG = "Lastlog";
    String RETODIARIO = "Reto_Diario";
    String RETOCOMPLTADO = "Reto_Completado";
}
```

Fig.24. Interfaz que declara las diferentes columnas de la tabla Usuario.

BaseDatos.java

Clase encargada de la creación de las diferentes tablas de la BBDD, y de la implementación de los métodos requeridos por la clase implementada SQLiteOpenHelper.java.

Como se observa en la **Fig.25.**, la creación de la tabla se hace mediante un execSQL. Este es un método importado de SQLiteOpenHelper que permite el envío de comandos SQL para interactuar con la base de datos. Para la creación se utiliza un comando con argumentos para garantizar la seguridad de la base de datos. Estos argumentos son las interfaces previamente declaras en DeviceContract.java y en los diferentes modelos.

En SQLite, al estar mermada su capacidad para las llaves foráneas, es necesario la creación de un índice que indique que elemento sirve como llave foránea en otra tabla. En el ejemplo de abajo, se declara el nombre del usuario como llave foránea.

```
db.execSQL(String.format("CREATE TABLE %s (%s INTEGER PRIMARY KEY AUTOINCREMENT," +
                "%s TEXT UNIQUE NOT NULL, %s TEXT NOT NULL %s, %s TEXT NOT NULL, %s TEXT NOT NULL," +
                "%s TEXT NOT NULL, %s TEXT NOT NULL, %s TEXT NOT NULL, %s INTEGER NOT NULL," +
                "%s INTEGER NOT NULL)",
        Tablas.DATOS_USUARIO, BaseColumns._ID,
        DeviceContract.Usuarios.ID, DeviceContract.Usuarios.USUARIO_DEVICE,
        Referencias.ID_DATOS_DISPOSITIVO,
        DeviceContract.Usuarios.NOMBRE, DeviceContract.Usuarios.ALTURA,
        DeviceContract.Usuarios.PESO, DeviceContract.Usuarios.ZANCADA,
        DeviceContract.Usuarios.LASTLOG, DeviceContract.Usuarios.RETODIARIO,
        DeviceContract.Usuarios.RETOCOMPLTADO));

/**
 * Indices tabla usuario
 */

db.execSQL(String.format("CREATE UNIQUE INDEX i1 ON %s(%s)",
        Tablas.DATOS_USUARIO, DeviceContract.Usuarios.NOMBRE));
```

Fig.25. Creación de la tabla Usuario en BaseDatos.java

En ella se declaran dos interfaces, una para la identificación de los nombres de las tablas y otra con las referencias entre las distintas llaves foráneas y sus tablas. Se han implementado dos métodos auxiliares, uno para el borrado sistemático de los datos de la tabla de rendimiento del usuario, y el segundo, para conseguir la versión actual de la BBDD.

Usuario.java

Modelo creado para contener los datos necesarios para la adicción de un nuevo usuario a la BBDD. Esto se realiza mediante el constructor del propio modelo y unas variables asignadas a este. Es aquí donde se define el tipo de dato para cada columna de la tabla Usuario.

RendimientoUsuario.java

Modelo creado para contener los datos necesarios para la adicción de un nuevo rendimiento a la BBDD. Esto se realiza mediante el constructor del propio modelo y unas variables asignadas a este. Es aquí donde se define el tipo de dato para cada columna de la tabla RendimientoUsuario.

DatosDispositivo.java

Modelo creado para contener los datos necesarios para la adicción de un nuevo dispositivo a la BBDD. Esto se realiza mediante el constructor del propio modelo y unas variables asignadas

a este. Es aquí donde se define el tipo de dato para cada columna de la tabla DatosDispositivo.

Users.java

Modelo creado para el SimpleSpinnerTextAdapter.java que contiene los parámetros necesarios para la elaboración de una lista desplegable. Esto se consigue mediante el constructor del propio modelo y unas variables asignadas al mismo.

Devices.java

Modelo creado para el ColoredTextAdapter.java que contiene los parámetros necesarios para dar forma a cada elemento de la lista que muestra los dispositivos externos encontrados. Esto se consigue mediante el constructor propio del modelo y unas variables asignadas al mismo.

SimpleSpinnerTextAdapter.java

Clase que hereda de ArrayAdapter.java. Esta es una clase incorporada en el SDK de Android que permite a los desarrolladores la elaboración de formatos propios de adaptadores, usados en los diferentes elementos de la interfaz de usuario.

En concreto, esta clase es un nuevo adaptador para la vista Spinner, que le muestra al usuario, las personas que ya están registradas en la BBDD. En ella se declaran los métodos necesarios para el ArrayAdapter.java. También se declara un constructor, este es quien recibe los valores que el adaptador debe usar para crear el elemento de la vista.

Los métodos declarados son getView y getDropDownView. El primero es usado para la elaboración de cada ítem de la lista, y el segundo muestra al sistema como tiene mostrar la lista desplegable al usuario.

Por último, se declara una clase estática ligada a SimpleSpinnerTextAdapater.java que solo sirve como contenedor de los diferentes datos necesarios para la elaboración de cada ítem de la lista.

ColoredTextAdapter.java

Clase que hereda de ArrayAdapter.java, como se ha explicado previamente, esta clase nos permite la elaboración de elementos personalizados para las vistas.

El objetivo de esta clase, es la creación de un adaptador para la lista de dispositivos externos escaneados. En ella, se declaran los

métodos necesarios para la clase heredada y las variables necesarias para la creación de cada ítem de la lista.

En este caso, el método heredado es solo getView. Esto se debe a que la ListView muestra todo su contenido de forma inmediata al usuario, sin necesidad de clicar a ningún desplegable. Mediante este método se da forma a cada ítem de la lista.

Se declara una clase estática ligada a ColoredTextAdapter.java que sirve como contenedor de los diferentes datos necesarios, para la elaboración de cada ítem. En el caso particular de la lista de dispositivos, estos son: icono del dispositivo, nombre dado por el usuario, código de radio baliza y nombre del fabricante.

activity_main.xml

Archivo layout escrito en formato xml. Un archivo layout, es un contenedor de vistas que declara que componentes se visualizan y sus respectivos atributos. Este fichero está ligado a MainActivity.java y, a diferencia de un layout simple, este está compuesto por dos layouts diferentes superpuestos.

El primero de ellos (ver **Fig.26.**), se encarga de facilitar al usuario el entorno de localización de dispositivos mediante la declaración de un botón de escaneo, y una lista donde mostrar los dispositivos encontrados. Este layout es el principal y será el mostrado al usuario de forma predefinida.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/List_menu"
        android:background="@drawable/background">
```

Fig.26. Declaración del primer layout de activity_main.xml

El segundo layout (ver **Fig.27.**), es el superpuesto y permanecerá invisible e inactivo al usuario hasta que la aplicación reconozca que el usuario ha activado un dispositivo previamente registrado en la BBDD. Se encarga de facilitar una interfaz de usuario, mediante un GridLayout en el que se disponen los diferentes botones de interacción.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    android:id="@+id/UserInterfaceLayout"
    android:visibility="invisible"
    android:orientation="vertical">
```

Fig.27. Declaración del segundo layout de activity_main.xml

Usar dos contenedores de objetos superpuesto, permite el reutilizamiento del fichero .java detrás de la actividad. Esto nos permite mantener la conexión con el dispositivo externo después de haberlo localizado y a su vez, ofrecerle al usuario una interfaz distinta con la que se ha encontrado previamente.

activity_new_device.xml

Fichero layout XML que contiene el formulario que el usuario necesita rellenar con datos, a la hora de registrar un nuevo dispositivo y usuario en la BBDD. Contiene declarados diferentes TextViews y EditText para informar y obtener del usuario dicha información. Además, implementa un botón para guardar los datos y otro para cancelar el proceso.

config_user_dialog.xml

Este contenedor de vistas, sirve para dar forma a una alerta de diálogo personalizada, en este caso, de la configuración de usuario. Cuando el usuario haga uso del botón correspondiente en la interfaz de usuario, un diálogo de alerta será activado, y este layout será utilizado para darle forma.

Contiene los TextView necesarios para informar al usuario de sus datos actuales, EditText que permiten la recogida de los nuevos valores de los parámetros del usuario, un botón de ayuda en caso del que usuario no sepa que es una zancada y dos botones, uno para cancelar los cambios y otro para guardarlos.

eraser_dialog.xml

Fichero layout XML empleado para dar forma a la alerta de dialogo para el borrado de datos. En él se declara un TextView con información relevante al borrado de datos, una lista desplegable con los diferentes usuarios registrados, tres botones que permiten tres tipos diferentes de borrado (el de usuario, el de todas las actividades realizadas y el borrado de toda la base de datos). También contiene un botón que permite salir al usuario de la alerta sin tener que borrar nada.

finish_perfomance_dialog.xml

Fichero layout XML empleado para dar forma a la alerta de diálogo mostrada al usuario, cuando termina una actividad física. Esta alerta de diálogo muestra al usuario, mediante el uso de varios TextView, todas las estadísticas relacionadas con la actividad física realizada. Contiene un botón de aceptar para cerrar el dialogo cuando el usuario haya visualizado el contenido.

first_challenge_dialog.xml

Fichero layout XML empleado para dar forma a la alerta del primer reto diario. Esta alerta de diálogo será enviada cuando al usuario se le haya asignado el reto diario número uno. Contiene una imagen ilustrativa para llamar la atención del usuario, un título, el nombre del reto, una breve descripción y el estado actual del mismo. Implementa un botón mediante el cual el usuario puede despachar la alerta.

second_challenge_dialog.xml

Fichero layout XML empleado para dar forma a la alerta del segundo reto diario. Esta alerta de diálogo será enviada cuando al usuario se le haya asignado el reto diario número dos. Contiene las mismas vistas que el primer reto.

third_challenge_dialog.xml

Fichero layout XML empleado para dar forma a la alerta del tercer reto diario. Dicha alerta será enviada cuando al usuario se le haya asignado el reto diario número tres. Contiene las mismas vistas que el primer reto.

listview_header_row.xml

Fichero XML empleado junto a listview_item_row.xml por ColoredTextAdapter.java para dar forma a cada ítem de la lista. Este en concreto, contiene la vista que dará forma al icono del dispositivo externo en la lista.

listiview_item_row.xml

Fichero XML empleado junto a listview_header_row.xml por ColoredTextAdapter.java para dar forma a cada ítem de la lista. El layout que contenido en el fichero da forma al nombre dado por el usuario al dispositivo, el código de radio baliza y el nombre dado por el fabricante.

listiview_simple_user_format.xml

Fichero XML empleado por SimpleSpinnerTextAdapter.java para dar forma a cada elemento de la lista desplegable de usuarios.

Este layout contiene el formato que tiene que tener cada nombre de usuario en la lista desplegable.

performance_user_dialog.xml

Fichero layout XML empleado para dar forma a la alerta de diálogo que aparece cuando el usuario pide sus estadísticas globales. Aquí se declaran las diferentes TextView que dan forma a la tabla de datos y se les asignan sus identificadores.

También se añaden tres botones diferentes de ayuda, el primero para explicar al usuario que son los pasos en puntilla, el segundo muestra información sobre los pasos de talón y el tercero explica que son los pesos pesados. Se ha añadido un botón con el cual el usuario puede cerrar el diálogo.

selected_user_dialog.xml

Fichero layout XML empleado para dar forma a la alerta de diálogo que muestra la lista desplegable con los diferentes usuarios registrados en la base de datos. Este diálogo es lanzado cuando el usuario conecta con un dispositivo ya registrado en la base de datos. Dentro del fichero solo se declara una lista desplegable y un botón para seleccionar el usuario.

colors.xml

Fichero XML dentro del paquete de valores que permite la rápida identificación por parte de la aplicación de los colores especificados por el desarrollador (ver **Fig.28.**).

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="golden">#F7A802</color>
    <color name="sligthy_golden">#F5D563</color>
</resources>
```

Fig.28. Declaración de los colores usados por la aplicación.

Esto se consigue mediante la declaración de un identificador para el color Ej.: “golden” y a continuación asignarle el valor del color en HTML5. Es recomendable identificar los colores que se repitan a lo largo de la aplicación para un desarrollo más limpio y claro.

dimens.xml

Fichero XML dentro del paquete de valores que permite la rápida identificación por parte de la aplicación de las dimensiones para vistas especificadas por el desarrollador. Su asignación sigue el mismo patrón que colors.xml simplemente teniendo que cambiar la etiqueta de color por dimen.

string.xml

Fichero XML dentro del paquete de valores que permite la rápida identificación por parte de la aplicación de las String especificadas por el desarrollador (ver **Fig.29.**).

```
<!-- Strings used in MainActivity -->
<string name="locationAccessTitle">This app needs location access</string>
<string name="locationAccessDescription">Please gran location access so this app can detect
    beacons.</string>
<string name="BLEisNotSupported">BLE not supported.</string>
<string name="newDeviceDetectedAlertTitle">New device found!</string>
<string name="newDeviceDetectedAlertDescription">It seems that we found a new device.
<!-- Strings used in MainActivity -->
<string name="locationAccessTitle">Esta aplicación necesita acceder a tu localización</string>
<string name="locationAccessDescription">Por favor, concédele permisos para que pueda localizar
    los datos necesarios.</string>
<string name="BLEisNotSupported">BLE no admitido.</string>
<string name="newDeviceDetectedAlertTitle">¡Nuevo dispositivo encontrado!</string>
```

Fig.29. Declaración de parte de las Strings usadas por la aplicación en distintos idiomas.

Android permite la elaboración de varios ficheros strings.xml que compartan el mismo nombre, pero utilicen diferentes identificadores de país; creando de forma sencilla una plantilla para la traducción simultánea a distintos idiomas de la aplicación. Los idiomas implementados por la aplicación son el inglés y el español.

El único requisito para emplear la traducción es que, cada recurso, debe compartir el mismo identificador. En el ejemplo (ver **Fig.29.**) se declara, tanto en español como en inglés, el recurso que da nombre al título de la alerta de localización con el mismo identificador: "locationAccessTitle".

Estos recursos pueden ser llamados tanto desde fichero .java como ficheros .xml, permitiendo definir el texto de vistas o de Strings de código de forma universal.

styles.xml

Fichero XML dentro del paquete de valores que permite la declaración, por parte del desarrollador, de diferentes estilos para la

aplicación. La forma de implementación sigue la misma que los demás ficheros de valores, exceptuando que hay que cambiar el identificador a style.

Ficheros.png

A la aplicación se le han asignados recursos gráficos para ilustrar la interfaz de usuario, tanto en el formato de mapa de bits como en formato de vector. Para una mayor extensión del producto, se han implementado diferentes resoluciones y tamaños a cada imagen. El propio sistema operativo del dispositivo móvil se encarga de buscar y asignar la imagen que más le convenga a la resolución de pantalla del usuario.

6.3. Conclusiones

En este apartado se ha explicado cómo ha sido implementado el código en la aplicación y las diferentes de cada fichero .java o .xml. Cada fichero ha sido comentado de forma breve y en algunos casos se ha adjuntado un código ilustrativo para ejemplificar la descripción.

Después de analizar cada fichero .java, se identifica como el núcleo de la aplicación el MainActivity.java. Esto se debe a que es el que contiene las variables que afectan a todo el sistema, y a que es la actividad principal con la que interactúa el usuario. Si bien, lo que parece una ventaja, acaba convirtiéndose en el punto débil de la aplicación ya que no delega en actividades hija o fragmentos para la creación de una interfaz de usuario más dinámica.

7. Validación del dispositivo externo

7.1. Introducción

A continuación, se va a proceder a la validación como dispositivo externo del CC2541 Sensortag, explicar cómo se ha testeado el hardware que incorpora y analizar su posible uso como dispositivo externo para la aplicación.

7.2. Validación

En este apartado se presentan y explican las diferentes capacidades que el dispositivo a testear como prototipo de medición de rendimiento de running posee. Además de detallar los diferentes servicios a utilizar por la aplicación Android para la medición y recogida de datos del usuario del dispositivo.

Fig.30. CC2541 SensorTag de la empresa Texas Instruments.

El dispositivo CC2541 SensorTag (ver **Fig.30.**), fue desarrollado por la empresa americana Texas Instruments. Con ello se busca dar pie al desarrollo de más dispositivos portátiles que estén conectados a internet y que leen y reciban datos de diferentes fuentes.

Según la filosofía de la empresa, en el año 2020 habrá millares de aparatos conectados a internet y que nos den información real de su actividad sean éstas: lavadoras inteligentes, aspiradoras, bicicletas, collares inteligentes… Las posibilidades son tantas como la imaginación de la gente que las lleve a cabo.

Este dispositivo recibe su nombre del microprocesador de Texas Instruments, que lleva implementado en su circuito impreso, el CC2541. Este microprocesador lleva a cabo la gestión de diferentes sensores dispuestos por todo el dispositivo entre los cuales se encuentran: sensor de temperatura sin contacto IR (TMP006 de Texas Instruments), sensor de humedad (Sensirion SHT21), giroscopio (Invensense IMU-3000)(ver **Fig.33.**), acelerómetro (Kionix KXTJ9) (ver **Fig.32.**), magnetómetro (Freescale MAG33110), sensor de presión barométrica (Epcos T5400), sensor de temperatura en contacto con el chip (Construido en el microprocesador CC2541) y sensor de voltaje y batería (implementado en el microprocesador CC2541).

Como elementos aparte, dispone de un botón de encendido/apagado en el lateral izquierdo y de un diodo LED (Light-emitting diode) que indica el estado del dispositivo.

Todos los sensores son pequeños y fáciles de montar en placa, e implementan la interfaz I2C. Las conexiones internas de interfaz se realizan con el mismo *bus,* pero con señales de activación separadas (ver **Fig.31.**). Esto implica una importante disminución en el consumo de corriente del dispositivo, ya que cada sensor está en un estado de deshabilitado de forma natural y pasan a un estado de *sleep mode* entras las distintas mediciones.

El Sensortag utiliza como fuente de alimentación una pila de botón CR2032 de 3V, hecha de Litio. Comúnmente utilizada en otros dispositivos electrónicos como ordenadores o calculadoras, es de pequeño tamaño y de larga duración.

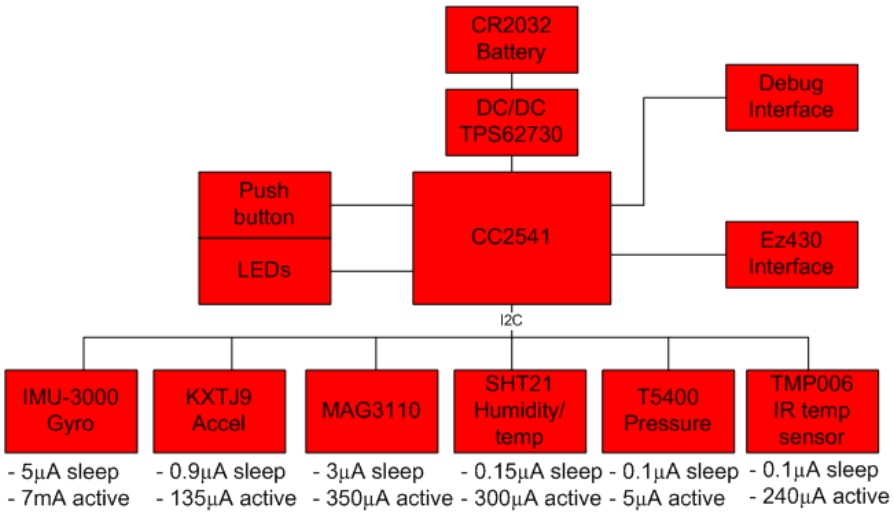

Fig.31. Diagrama de bloques de CC2541 SensorTag.

Para el desarrollo de las pruebas del CC2541 SensorTag como posible dispositivo de fitness tan sólo se emplearán dos de los sensores que contiene. El acelerómetro y el giroscopio. Ya que todos los sensores están deshabilitados por naturaleza, no existirá interferencia alguna en las pruebas por parte del resto de sensores.

Acelerómetro Kionix KXT J9

El acelerómetro mide la aceleración en los 3 ejes (X, Y, Z) con una resolución programable de hasta 14 bit. Este acelerómetro permite, a diferencia del giroscopio, la medición de la dirección de la gravedad.

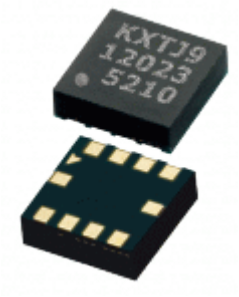

Fig.32. Acelerómetro KXTJ9-1007.

Originalmente diseñado por Kionix para su uso en aplicaciones móviles, este acelerómetro ofrece gran capacidad de medición con bajo coste de energía. Sus medidas son de 3x3x0.9 mm.

Su funcionamiento se basa en el principio del incremento de la capacidad diferencial, debido a la inducción de la aceleración del movimiento del elemento a medir.

Giroscopio Invensense IMU-3000

El giroscopio mide la ratio de rotación en los 3 ejes (X, Y, Z), con una resolución de hasta 16 bits. Se puede emplear junto al acelerómetro para determinar la posición del dispositivo CC2541 Sensortag en un mapa 3D.

Fig.33. Giroscopio Invensense IMU-3000

Desarrollado por Invensense como su respuesta a la necesidad de un sensor de 6 ejes, dispone de un giroscopio 3 ejes que, junto a un procesador digital de moción, combina las medidas de un acelerómetro externo para completar las medidas de los 6 ejes.

El sensor combina movimiento lineal y rotacional en tan solo un envío de datos para la aplicación. Esto lo convierte en el producto ideal para dispositivos que requieren sensores de bajo coste y de bajo consumo, pero que ofrecen gran rendimiento a cambio. Su única problemática podría deberse a la necesidad de un acelerómetro de terceros para completar los datos que envía, con el cual tiene una relación de maestro-esclavo.

Microprocesador CC2541

Fig.34. Microprocesador CC2541 de Texas Instruments.

El microprocesador de Texas Instruments CC2541 (ver **Fig.34., Fig.35.**), es el cerebro del SensorTag y se dedica al control y procesamiento de los diferentes sensores incorporados en el dispositivo.

A su vez implementa un sensor de temperatura propia y un sensor de consumo de voltaje. También contiene todo lo necesario para la conexión inalámbrica del dispositivo mediante el sistema BLE (Bluetooth Low Energy). Anunciado por Texas Instruments como un procesador de gran rendimiento y poco consumo, está preparado para su instalación en circuitos integrados pequeños.

Contiene programas de software que le permiten depurar y compilar conexiones de BLE, optimizar consumo de energía tanto del controlador como del usuario, Cliente y servidor GATT para la conexión BLE. Disponibilidad de diferentes perfiles de programación y aplicación.

En conclusión, es el microprocesador que Texas Instruments dió como respuesta al nuevo tipo de conexión BLE, permitiendo conexiones seguras y de alta conectividad.

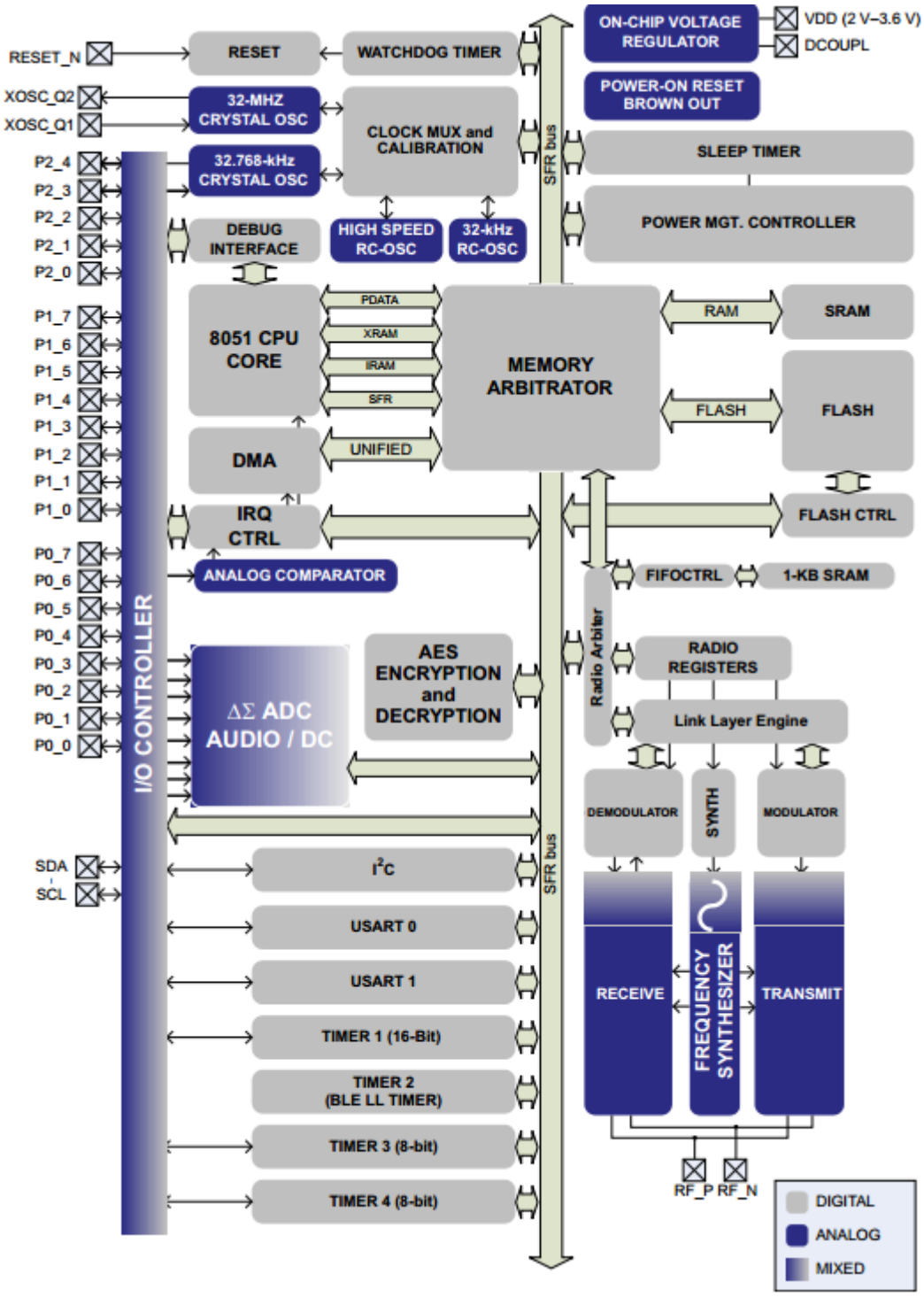

Fig.35. Diagrama de bloques del microprocesador CC2541.

Después de realizar un análisis exhaustivo de las diferentes propiedades que ofrece el CC2541 Sensortag y sus sensores, se puede determinar que es compatible como dispositivo externo para la aplicación. Esto se debe a sus sensores de bajo consumo y gran resolución, y a la capacidad de establecer una conexión BLE segura.

Cabe destacar que se tiene que utilizar como un elemento de prototipo y no como un dispositivo de lanzamiento al mercado, ya que contiene sensores y otros elementos que nunca serán empleados por la aplicación y, a la larga, pueden suponer una carga para la misma.

7.3. Testeo de hardware

Como se ha mencionado en el apartado anterior, el CC2541 SensorTag posee un acelerómetro y un giroscopio que son compatibles con la aplicación. En este apartado se analiza en más detalle su conectividad con la aplicación y su rendimiento.

7.3.1. Análisis de la conectividad

La conexión mediante Bluetooth Smart® o BLE es una conexión de bajo consumo, enfocada a la comunicación con elementos inalámbricos como sensores y periféricos. Esto requiere del establecimiento de unos roles o papeles entre los dispositivos que establecen la conexión, el rol de cliente y el rol del servidor. Para esta prueba se le asigna el papel de servidor al CC2541 SensorTag y el rol de cliente a la aplicación.

El servidor tiene que proporcionar un perfil de atributos genérico o GATT (ver **Fig.36.**). Esto permite que los dispositivos que lo localicen encuentren la información de forma organizada y clara.

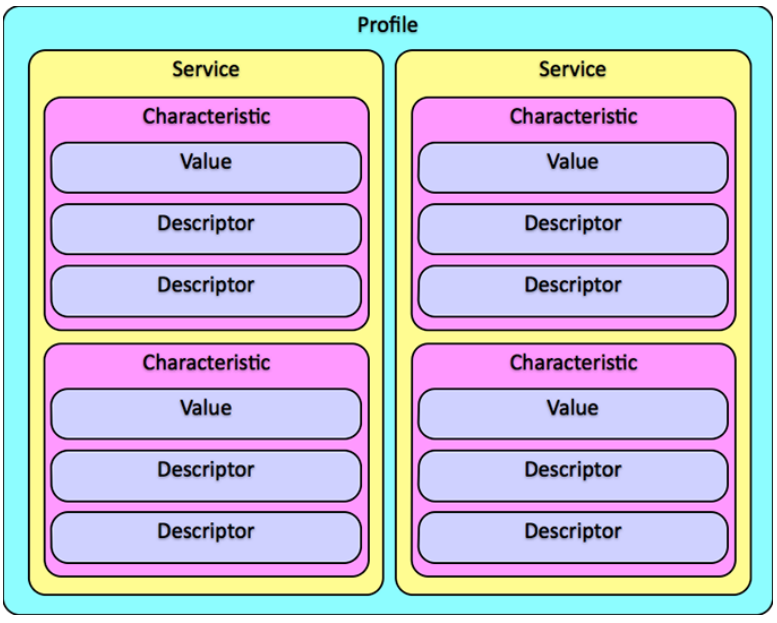

Fig.36. Esquema simplificado de un perfil genérico de atributos o GATT

La información que generan los periféricos del dispositivo que actúa como central, se organiza en colecciones llamadas servicios. Cada servicio identifica de forma individual un sensor, en esta prueba de validación un servicio será el giroscopio y otro el acelerómetro. Estos servicios engloban una colección de características que permiten la transferencia de datos discretos entre dispositivos, es decir, los datos que el sensor registra y son leídos por la aplicación.

Las características están organizadas de forma simple, una variable *flag* y un valor. Con la variable *flag* describen, mediante información básica, las capacidades de la característica (si es de lectura, escritura o si admite notificaciones). De forma opcional, pueden contener descriptores que indican información como tipo de unidad leída o configuración adicional.

A diferencia del sistema clásico de comunicación inalámbrica por Bluetooth, BLE necesita saber de antemano estas características para poder escanearlas. Por tanto, el primer paso para testear el *hardware* es localizar el perfil GATT de los servicios del CC2541 SensorTag. Estos datos los proporciona de manera gratuita Texas Instruments por ser el fabricante del producto a testear.

El primer código necesario es el del servidor GATT incorporado en el dispositivo. Texas Instruments emplea un UUID en base a 128-bit para cada puerto, filtrando así cualquier dispositivo BLE que no contenga dicho código en los escaneos de búsqueda. Este código es: **F0000000-0451-4000-B000-000000000000.** Cada servicio individual utiliza parte de ese código, más un identificador propio de 16-bit que sustituye a la cabecera del código.

Acelerómetro

El acelerómetro KXT J9 @ U7 de Kionix, está identificado mediante los códigos mostrados en la siguiente tabla (ver **Tabla.2.**):

Acelerómetro				
Tipo	**UUID**	**Manejador**	**Lectura/Escritura**	**Descripción**
<Data>	F000AA11	0x2D	Lectura/Notificación	
<Data Notification>		0x2E	Lectura/Escritura	
<Configuration>	F000AA12	0x31	Lectura/Escritura	
<Period>	F000AA13	0x34	Lectura/Escritura	Period = [input * 10] ms

Tabla.2. Características del servicio acelerómetro

Como se ha mencionado previamente, cada servicio consta de un conjunto de características con descriptores. En el acelerómetro, que es el servicio, hay anidadas cuatro características:

- Dato leído por el sensor con código F000AA11 y descriptor indicando que puede ser de lectura o de notificación.
- Notificación de datos con el mismo código que el dato y descriptor indicando que puede ser leído o escrito.
- Configuración del servicio con código F000AA12 y descriptor indicando que se puede leer o escribir.
- Período de lectura del sensor con código F000AA13 y con descriptores que indican que es posible leerlo o escribirlo, y que el período será igual al dato de configuración metido multiplicado por diez, siendo el resultado en unidades de milisegundos.

El manejador no tiene relevancia a la hora de desarrollar la aplicación. Los códigos una vez agregados al primero, tendrán este aspecto (ver **Fig.37.**):

```
/* Accelerometer Service */
private static final UUID ACCELEROMETER_SERVICE = UUID.fromString("f000aa10-0451-4000-b000-000000000000");
private static final UUID ACCELEROMETER_DATA_CHAR = UUID.fromString("f000aa11-0451-4000-b000-000000000000");
private static final UUID ACCELEROMETER_CONFIG_CHAR = UUID.fromString("f000aa12-0451-4000-b000-000000000000");
private static final UUID ACCELEROMETER_PERIOD_CONFIG = UUID.fromString("f000aa13-0451-4000-b000-000000000000");
```

Fig.37. Códigos UUID de las características del acelerómetro.

El UUID para el servicio del acelerómetro no está especificado por Texas Instruments, pero siempre toma el valor 0 del último bit del código. Mediante el UUID del servicio se puede declarar que servicio se está configurando en el sistema. Con el código UUID de CONFIG_CHAR se pasa al dispositivo externo los parámetros de configuración necesarios; para el acelerómetro hay que pasar un byte con valor 0x01 para habilitarlo. El periodo del acelerómetro se configura teniendo en cuenta que la resolución del mismo es de 10 ms. Texas Instruments permite un rango de periodo de entre 100 ms y 2.55 s; su valor por defecto es de 1 s. A continuación, se muestra la configuración del acelerómetro en la aplicación (ver **Fig.38.**):

```
case 2:
    Log.d(TAG, "Enabling Accelerometer");
    characteristic = gatt.getService(ACCELEROMETER_SERVICE)
            .getCharacteristic(ACCELEROMETER_CONFIG_CHAR);
    characteristic.setValue(new byte[]{0x01});
    break;
case 3:
    Log.d(TAG, "Enabling accelerometer period");
    characteristic = gatt.getService(ACCELEROMETER_SERVICE)
            .getCharacteristic(ACCELEROMETER_PERIOD_CONFIG);
    characteristic.setValue(new byte[]{0x0A});
    break;
```

Fig.38. Configuración de las características del acelerómetro.

Al periodo se le ha pasado un byte con valor 0x0A indicando que se establezca el periodo mínimo de 100 ms para la lectura de datos. Toda comunicación con el dispositivo externo conectado con BLE debe realizarse mediante una máquina de estados. Esto es debido a que no se permite más de una conexión a la vez con cada característica.

Mediante el código UUID de DATA_CHAR se pueden leer los datos enviados por los sensores, y además habilitar la lectura automática de estos cuando su valor varíe. Esto se consigue mediante un código UUID conocido como descriptor de configuración, y sirve de manera general para todos los servicios del dispositivo (ver **Fig.39.**). Dicho código es: **00002902-0000-1000-8000-00805f9b34fb**.

```
    case 1:
        Log.d(TAG, "Set notify Accelerometer");
        characteristic = gatt.getService(ACCELEROMETER_SERVICE)
                .getCharacteristic(ACCELEROMETER_DATA_CHAR);
        break;
    default:
        mHandler.sendEmptyMessage(MSG_DISMISS);
        Log.i(TAG, "All Sensors Enabled");
        return;
}
//Permitir notificaciones locales
gatt.setCharacteristicNotification(characteristic, true);
//Permitir notificaciones remotas
BluetoothGattDescriptor desc = characteristic.getDescriptor(CONFIG_DESCRIPTOR);
desc.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
gatt.writeDescriptor(desc);
```

Fig.39. Uso del código para habilitar la notificación automática del servicio.

Giroscopio

El giroscopio IMU-3000 @ U8 de Invensense, está identificado mediante los siguientes códigos (ver **Tabla.3.**):

Giroscopio			
Tipo	**UUID**	**Lectura/Escritura**	**Formato**
<Data>	AA51	Solo lectura	XLSB XMSB YLSB YMSB ZLSB ZMSB
<Data Notification>		Lectura/Escritura	2 bytes
<Configuration>	AA52	Lectura/Escritura	1 byte
<Period>	AA53	Lectura/Escritura	1 byte

Tabla.3. Características del servicio giroscopio

Como se ha mencionado previamente, cada servicio consta de un conjunto de características con descriptores. En el giroscopio, que es el servicio, hay anidadas cuatro características:

- Dato leído por el sensor con código F000AA51 y dos descriptores distintos. El primero indica que es un dato solo de lectura, el segundo indica como es el formato de salida de los datos.
- Notificación de datos con el mismo código que el dato y descriptor es indicando que puede ser leído o escrito y que el formato es en 2 bytes.
- Configuración del servicio con código F000AA52 y descriptor indicando que se puede leer o escribir y que el formato de entrada es de 1 byte.
- Periodo de lectura del sensor con código F000AA53 y con descriptores que indican que es posible leerlo o escribirlo y que el formato será de 1 byte.

Después de juntar este código de 16-bit con el primero proporcionado por Texas Instruments, aparece el código completo de la siguiente manera (ver **Fig.40.**):

```
/* Gyroscope Service */
 private static final UUID GYROSCOPE_SERVICE = UUID.fromString("f000aa50-0451-4000-b000-000000000000");
 private static final UUID GYROSCOPE_DATA_CHAR = UUID.fromString("f000aa51-0451-4000-b000-000000000000");
 private static final UUID GYROSCOPE_CONFIG_CHAR = UUID.fromString("f000aa52-0451-4000-b000-000000000000");
 private static final UUID GYROSCOPE_PERIOD_CONFIG = UUID.fromString("f000aa53-0451-4000-b000-000000000000");
```

Fig.40. Códigos UUID de las características del giroscopio.

El UUID para el servicio del giroscopio tampoco está especificado por Texas Instruments, pero ya se ha explicado que toma el valor 0 del último bit. A diferencia del acelerómetro, el giroscopio necesita un valor de activación compuesto en lugar de un 1 o un 0. Esto es debido a que puedes activar la lectura de cada eje de forma individual, una forma combinada Ej.: XY o la lectura de todos los ejes a la vez. Para habilitar dichas lecturas, se tiene que escribir un valor de byte entre 0 y 7 en el CONFIG_CHAR del giroscopio (ver **Fig.41.**). El rango del periodo es el mismo que del acelerómetro.

```
case 0:
    Log.d(TAG, "Enabling Gyroscope");
    characteristic = gatt.getService(GYROSCOPE_SERVICE)
            .getCharacteristic(GYROSCOPE_CONFIG_CHAR);
    characteristic.setValue(new byte[]{7});
    break;
case 1:
    Log.d(TAG, "Enabling gyroscope period");
    characteristic = gatt.getService(GYROSCOPE_SERVICE)
            .getCharacteristic(GYROSCOPE_PERIOD_CONFIG);
    characteristic.setValue(new byte[]{0x0A});
    break;
```

Fig.41. Configuración de las características del giroscopio.

El resto de la configuración de las características sigue el mismo patrón que las del acelerómetro y, por ende, no es necesario explicar cómo se ha llevado a cabo.

7.3.2. Análisis del hardware

En este apartado se analizan los datos que leen los sensores y que la aplicación recoge. También se explica cómo estos datos son tratados por la aplicación para conseguir datos fiables.

Acelerómetro

La lectura de la característica de datos del servicio del acelerómetro viene en formato de bytes, y esta tiene que ser parametrizada mediante un método. Este ha sido declarado en la clase SensorTagData.java (ver **Fig.42.**).

```
public static float[] extractAccData(byte[] value){

    final float SCALE = (float) 4096.0;
    float[] Accsent = new float[4];

    Accsent[0] = (value[0]<<8)/SCALE;
    Accsent[1] = (value[1]<<8)/SCALE;
    Accsent[2] = (value[2]<<8)/SCALE;

    return Accsent;
}
```

Fig.42. Conversión del byte leído por la característica del acelerómetro a valores útiles.

El byte de información viene sectorizado, esto implica que para recoger los valores hay que indicar cual queremos en el vector de bytes y hacer un desplazamiento a la izquierda de tamaño ocho. A estos parámetros hay que pasarlos por una escala para que sean correctos. El valor de escalado lo proporciona Texas Instruments y es de 4096.

El método da como resultado un vector float que, de forma ordenada, contiene los valores de los ejes X, Y y Z en el mismo orden. Este método es llamado en la actividad principal, donde se aginan los valores del vector a las variables correspondientes (ver **Fig.43.**).

```
private void updateAccData(BluetoothGattCharacteristic characteristic){

    byte[] value = characteristic.getValue();
    float[] resacc = SensorTagData.extractAccData(value);

    mAccX = (resacc[0]);
    mAccY = (resacc[1]);
    mAccZ = (resacc[2]);

    mInclinationAngle[0] = Math.atan(mAccX/Math.sqrt(Math.pow(mAccY,2) + Math.pow(mAccZ,2)))*(180.0/Math.PI);
    mInclinationAngle[1] = Math.atan(mAccY/Math.sqrt(Math.pow(mAccX,2) + Math.pow(mAccZ,2)))*(180.0/Math.PI);
}
```

Fig.43. Llamada al método extractAccData en MainActivity.java.

Cada vez que la conexión registre un cambio en los valores del sensor, el método updateAccData será ejecutado. Este lee los datos enviados por el sensor mediante el método explicado anteriormente; una vez leídos, los almacena en tres variables declaradas en la clase.

Para disponer de unos datos más claros con los que interactuar, los diferentes valores de los ejes del acelerómetro se emplean para calcular la inclinación del dispositivo en el eje X e Y.

Se tiene en cuenta que la única fuerza que actúa sobre el sensor es la de la gravedad. Debido a esto, se puede considerar que los valores obtenidos por el sensor están correlacionados con la gravedad y que representarán la inclinación del plano del sensor. En la **Fig.44.** se ilustra de forma sencilla el problema.

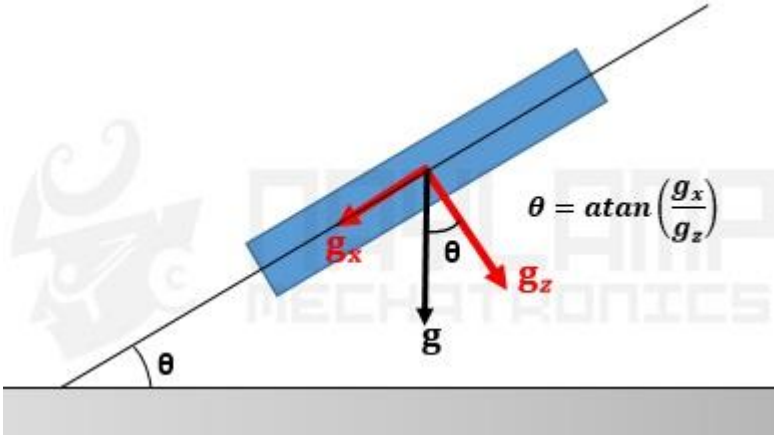

Fig.44. Representación gráfica de la interacción de la gravedad con el dispositivo.

El diagrama representa al dispositivo en un plazo X-Z inclinado un cierto ángulo theta. La fuerza de la gravedad que incide sobre el objeto puede dividirse en la fuerza de la gravedad del eje X y del eje Z. El ángulo theta de inclinación se puede calcular mediante el arco tangente de las dos fuerzas de gravedad distintas. Sin embargo, esto es solo usable en un plano 2D, para el cálculo de la inclinación en un mapa 3D se necesita la siguiente fórmula (ver **Ec.4.**):

$$\theta_x = \tan^{-1}\left(\frac{mAccX}{\sqrt[2]{mAccY^2 + mAccZ^2}}\right)$$

Ec.4. Fórmula para el cálculo de la inclinación en el eje X en un mapa 3D.

A la fórmula se le pasan los valores del acelerómetro almacenados en las variables de ManActivity.java, y da como resultado un ángulo de inclinación que oscila entre 90º y -90º. El cálculo del ángulo de inclinación del eje Y es el siguiente (ver **Ec.5.**):

$$\theta_y = \tan^{-1}\left(\frac{mAccY}{\sqrt[2]{mAccX^2 + mAccZ^2}}\right)$$

Ec.5. Fórmula para el cálculo de la inclinación en el eje Y en un mapa 3D.

El cálculo del ángulo de inclinación del eje Z no es necesario para la prueba de verificación de sensores. Por ahora tenemos un conjunto de datos para la posible situación en un mapa 3D del pie del corredor, pero falta saber cómo interpretarlos. Primero hay que hacer un esquema gráfico para situar los ejes de los cuales se calcula la inclinación (ver **Fig.45.**).

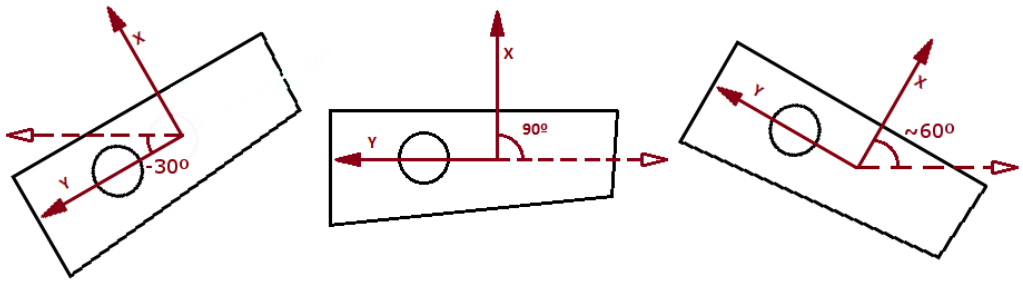

Fig.45. Representación de los ejes de gravedad del CC2541 SensorTag situado en un pie.

Debido a que se ha especificado que el dispositivo debe registrar cuando el usuario ha aterrizado de puntillas o de talón, hay que ver que ángulo es el más indicado para considerar que el pie está de puntillas o de talón. A través de una lectura de consola de los registros enviados por sensor se han sacado dichos valores de inclinación. Para el eje X, toda inclinación menor de 60º es considerada de puntilla; cuando el pie del corredor este en reposo, el dispositivo lee una inclinación del eje X de 90º, el levantamiento del pie hacia delante hace decrecer ese ángulo hasta 0º.

Para las pisadas de talón se empleará el eje Y el cual, en un estado de reposo, da como lectura 0º. Cuando el ángulo de inclinación de dicho eje sea inferior a 0º, el paso será considerado de talón.

Giroscopio

La lectura de la característica de datos del servicio del giroscopio viene en formato de bytes, y esta tiene que ser parametrizada mediante un método. Este ha sido declarado en la clase SensorTagData.java (ver **Fig.46.**).

```
public static float[] extractGyroData(BluetoothGattCharacteristic c){
    //byte[] value = c.getValue();
    float gyroData[] = new float[3];
    gyroData[0] = shortSignedAtOffset(c, 0) * (500f / 65536f) * -1;
    gyroData[1] = shortSignedAtOffset(c, 2) * (500f / 65536f);
    gyroData[2] = shortSignedAtOffset(c, 4) * (500f / 65536f);

    return gyroData;

}
```

Fig.46. Conversión del byte leído de la característica del giroscopio.

Al byte leído se le pasará un offset dado por Texas Instruments y multiplicado por una escala también proporcionada por Texas Instruments. El resultado de esta conversión es almacenado en un vector float que es enviado como resultado del método.

Este método es llamado en la actividad principal MainActivity.java (ver Fig.47.), guardando los datos recibidos en las variables correspondientes a cada eje. El giroscopio lee los °/s que se han detectado en cada eje, variando en un rango de entre -255 °/s a 255 °/s.

```
private void updateGyroData(BluetoothGattCharacteristic characteristic){
    float[] resgyro = SensorTagData.extractGyroData(characteristic);

    mGyroX = (resgyro[0]);
    mGyroY = (resgyro[1]);
    mGyroZ = (resgyro[2]);
}
```

Fig.47. Llamada del método extractGyroData en MainActivity.java

Los datos leídos son usados por la aplicación para medir cuando el usuario ha realizado un paso con demasiada fuerza. Cualquier lectura que supere los 200°/s en positivo se considera como un paso pesado por parte del corredor y se guarda en el registro.

7.3.3. Conclusión

El CC2541 Sensortag proporciona una conexión inalámbrica BLE que cumple con el requisito de poseer un perfil genérico de conexión GATT. Se han definido los códigos de acceso UUID a dicho perfil y sus características, dejando claro que se puede realizar una conexión inalámbrica con dicho dispositivo. El giroscopio y el acelerómetro han sido analizados y también se ha explicado cómo se extraen los datos de los mismos.

7.4. Análisis de su posible uso

El CC2541 SensorTag puede emplearse como dispositivo externo para la aplicación. Su capacidad de conexión inalámbrica mediante GATT es imprescindible, y los sensores que lleva incorporados ofrecen una medición de datos precisa y de bajo coste energético.

Si bien, para este fin, hay que adaptar la aplicación a las especificaciones del SensorTag, impidiendo un carácter generalizado como se ha explicado en la conectividad. No obstante, esto no es del todo una problemática concreta, pues todos los dispositivos con conexión BLE necesitan predefinir sus UUID en la aplicación antes de establecer la conexión.

8. Conclusiones

8.1. Resumen del trabajo desarrollado

Debido al incremento de deportistas que eligen el *running* como su deporte, se ha planteado la idea del desarrollo de un prototipo de dispositivo que mida el rendimiento de los corredores, el cual, tiene que estar comunicado mediante una aplicación al dispositivo móvil del usuario.

Mediante una comparativa con los diferentes productos del mercado, se ha decidido que características debía reunir la aplicación y el dispositivo externo. Con esas especificaciones delimitadas, se han estudiado las tecnologías disponibles con las que implementarlas.

Con el uso de Android Studio como herramienta de desarrollo, se ha realizado la aplicación necesaria para el dispositivo móvil. Para el dispositivo externo que debe medir el rendimiento del usuario, se ha comprobado la validez del CC2541 SensorTag. En el documento se ha explicado cómo se ha adaptado la aplicación para cumplir con las condiciones aportadas por dicho dispositivo.

Por último, el objetivo de realizar un dispositivo que mida el rendimiento del corredor se ha realizado con éxito. No obstante, esto se ha realizado con el CC2541 SensorTag como dispositivo prototipo y no estaría preparado para un lanzamiento al mercado.

8.2. Aportaciones

Al inicio del documento se ha explicado el crecimiento del *running* como deporte en España, y de cómo esto ha abierto las puertas a varios mercados centrados en esta temática. El mercado analizado en este documento es el de los dispositivos electrónicos de *fitness*.

Con el objetivo de analizar las diferentes características que ofrecen estos dispositivos, se ha realizado un estudio de mercado, donde se han comparado los dispositivos electrónicos que las diferentes empresas ofrecen en el mercado. De este análisis se han deducido las características básicas que deben tener tanto la aplicación, como el dispositivo externo prototipo.

Se ha hecho hincapié en las tecnologías que se han empleado para desarrollar la aplicación, describiéndolas y comentando de forma breve los elementos más importantes de esta.

Para explicar cómo se ha desarrollado la aplicación se ha recurrido a las normativas IEEE 830 y la UML. Dichos modelos han sido aplicados y publicados en esta memoria. En ellos se pueden encontrar: los objetivos a

desarrollar, sus especificaciones, las interacciones entre las diferentes partes de la aplicación.

Por último, se ha comprobado la viabilidad del CC2541 SensorTag como dispositivo externo a utilizar en la aplicación. Para ello se ha realizado un estudio de sus características y un análisis en profundidad sobre su capacidad de conexión. También se ha hecho un análisis de los sensores dispuestos por el mismo y de los datos leídos con ellos; también se ha explicado como la aplicación trata los datos leídos.

8.3. Trabajo futuro

A continuación, se exponen posibles mejoras y ampliaciones que se pueden llevar a cabo en la aplicación y el dispositivo externo; todas ellas en busca de mejorar el producto final y su calidad.

8.3.1. Aplicación

Respecto a la aplicación, se podría rediseñar la interfaz gráfica del usuario para que sea más dinámica. Esto se podría conseguir mediante el uso de fragmentos o añadir más actividades a la aplicación haciendo que, de paso, se merme la carga que soporta la actividad principal.

La base de datos del dispositivo podría rediseñarse para ofrecer un diseño más elegante y accesible para la aplicación. Si en un futuro se desea añadir más mediciones del rendimiento, se necesitará rediseñar la base de datos igualmente.

Para fomentar un uso prologando de la aplicación, se podría diseñar un sistema de puntos o monedas que los usuarios puedan usar para desbloquear, por ejemplo, más retos. También podría remodelarse la aplicación para permitir interacción con las diversas redes sociales usadas hoy en día, lo que permitiría a los usuarios compartir sus diferentes rendimientos entre sus contactos.

Por último, toda la aplicación podría rediseñarse como aplicación web. Esto permitiría construir el producto, la aplicación y el dispositivo externo, como una sola entidad alojada en la página web de la empresa. Dicha página de internet, podría usarse como base de datos, permitiendo al usuario el registro de sus datos on-line y el fácil acceso al resto de redes sociales, permitiendo también una alta conectividad entre dispositivos.

8.3.2. Dispositivo externo

Dado que el dispositivo externo usado para el desarrollo de la aplicación ha sido el CC2541 SensorTag, habría que fabricar un nuevo dispositivo desde cero. Dicho dispositivo tiene que tener una conexión inalámbrica BLE e incorporar los dos sensores necesitados por la aplicación.

Dicho dispositivo tendría que diseñarse pensando en un perfil general de atributos o GATT, para que pueda ser aplicado con facilidad a la aplicación. Si los puertos GATT no están identificados de manera correcta por sus respectivos UUID, no podrá establecerse una comunicación entre ambos dispositivos.

Por último, este dispositivo externo debe ser diseñado de manera elegante y ergonómica. Esto es debido a que el usuario principal, un practicante de *running*, debe llevarlo puesto en la zona del tobillo durante toda la actividad física realizada por el mismo. Si no se diseña con materiales que sean amigables con la piel, no podrá considerarse como un dispositivo de éxito.

9. Bibliografía

Título: Layouts
Autor: Android Inc.
Enlace: https://developer.android.com/guide/topics/ui/declaring-layout.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Supporting different devices
Autor: Android Inc.
Enlace: https://developer.android.com/training/basics/supporting-devices/index.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Providing Alternative Resources
Autor: Android Inc.
Enlace: https://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Localizing with Resources
Autor: Android Inc.
Enlace: https://developer.android.com/guide/topics/resources/localization.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Spinners
Autor: Android Inc.
Enlace:
https://developer.android.com/guide/topics/ui/controls/spinner.html?hl=es
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Adopted specifications
Autor: Bluetooth company
Enlace: https://www.bluetooth.com/specifications/adopted-specifications
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Tutorial de listas y adaptadores en Android
Autor: Revelo, James
Enlace: http://www.hermosaprogramacion.com/2014/10/android-listas-adaptadores/
Creado el: 12 de octubre de 2014 **Última consulta:** 25 de agosto de 2016

Título: Is your beacon app ready for Android 6.0?
Autor: G. Young, David
Enlace: http://developer.radiusnetworks.com/2015/09/29/is-your-beacon-app-ready-for-android-6.html
Creado el: 29 de septiembre de 2015 **Última consulta:** 25 de agosto de 2016

Título: Android Bluetooth Low Energy (BLE) example
Autor: Gupt, Mohit
Enlace: http://www.truiton.com/2015/04/android-bluetooth-low-energy-ble-example/
Creado el: 23 de abril de 2015 **Última consulta:** 25 de agosto de 2016

Título: Customizing Android ListView items with custom ArrayAdapter
Autor: Anwar, Waqas
Enlace: http://www.ezzylearning.com/tutorial/customizing-android-listview-items-with-custom-arrayadapter
Creado el: 16 de agosto de 2011 **Última consulta:** 25 de agosto de 2016

Título: Tutorial de bases de datos SQLite en aplicaciones Android
Autor: Revelo, James
Enlace: http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/
Creado el: 20 de octubre de 2014 **Última consulta:** 25 de agosto de 2016

Título: Bases de datos SQLite en Android con múltiples tablas
Autor: Revelo, James
Enlace: http://www.hermosaprogramacion.com/2016/01/base-de-datos-sqlite-en-android-con-multiples-tablas/
Creado el: 16 de enero de 2016 **Última consulta:** 25 de agosto de 2016

Título: GATT
Autor: Townsed, Kevin
Enlace: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Tutorial: intro to developing Bluetooth Smart applications for Androd
Autor: Smith, Dave
Enlace: https://newcircle.com/s/post/1553/bluetooth_smart_le_android_tutorial
Creado el: 3 de diciembre de 2013 **Última consulta:** 25 de agosto de 2016

Título: SensorTag user guide
Autor: Texas Instruments
Enlace: http://processors.wiki.ti.com/index.php/SensorTag_User_Guide
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Tutorial de layouts en Android
Autor: Revelo, James
Enlace: http://www.hermosaprogramacion.com/2015/08/tutorial-layouts-en-android/
Creado el: 2 de agosto de 2015 **Última consulta:** 25 de agosto de 2016

Título: ¿Pisada de talón o con la punta del pie para prevenir lesiones?
Autor: Antonio Jesús
Enlace: https://entrenar.me/blog/running/pisada-talon-o-punta-pie-prevenir-lesiones/
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: The story of Fitbit: how a wooden box became a $4 billion company
Autor: Marshall, Gary
Enlace: http://www.wareable.com/fitbit/youre-fitbit-and-you-know-it-how-a-wooden-box-became-a-dollar-4-billion-company
Creado el: 30 de diciembre de 2015 **Última consulta:** 25 de agosto de 2016

Título: KXTJ9-AG product specifications
Autor: Kionix
Enlace: http://kionixfs.kionix.com/en/datasheet/KXTJ9-1007%20Specifications%20Rev%204.pdf
Creado el: última revisión de diciembre de 2012 **Última consulta:** 25 de agosto de 2016

Título: 2.4-GHz Bluetooth™ low energy and Proprietary System-on-Chip
Autor: Texas Instruments
Enlace: http://www.ti.com/lit/ds/symlink/cc2541.pdf
Creado el: última revisión de junio de 2013 **Última consulta:** 25 de agosto de 2016

Título: Bluetooth low energy
Autor: anónimo
Enlace: https://en.wikipedia.org/wiki/Bluetooth_low_energy
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Bluetooth low energy
Autor: Android Inc.
Enlace: https://developer.android.com/guide/topics/connectivity/bluetooth-le.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Activity
Autor: Android Inc.
Enlace: https://developer.android.com/reference/android/app/Activity.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: ieee830
Autor: G. Mendez
Enlace: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf
Creado el: 22 de octubre de 2008 **Última consulta:** 25 de agosto de 2016

Título: Tutorial MPU6050, acelerómetro y giroscopio
Autor: anónimo
Enlace: http://www.naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Steps to distance converter
Autor: anónimo
Enlace: http://www.exrx.net/Calculators/StepDistance.html
Creado el: sin registrar **Última consulta:** 25 de agosto de 2016

Título: Create your own simple pedometer application using Android 4.4 and Nexus 5
Autor: Bawa, Harimohan
Enlace: http://blog.bawa.com/2013/11/create-your-own-simple-pedometer.html
Creado el: 10 de noviembre de 2013 **Última consulta:** 25 de agosto de 2016

Anexo

Introducción

A continuación, se presenta el funcionamiento de la aplicación a nivel usuario, creando un guía de usuario mediante imágenes y texto junto a una breve descripción. También se detalla en que consiste la aplicación y como ha de instalarse.

Software

La aplicación Pro Run ha sido diseñada como un sistema de medición de rendimiento para practicantes del *running*. Con objetivo de garantizar esta medición, se requiere establecer una conexión inalámbrica con un dispositivo externo proporcionado por el usuario.

Si el usuario ya ha establecido dicha conexión, accede a un sistema de control de actividades y de registro de las mismas. Todo está organizado por una base de datos interna que guarda la información del usuario, incluida las diferentes actividades realizadas.

Para fomentar el uso de esta aplicación se le proporciona al usuario una serie de retos para medir sus capacidades físicas. Si bien, estos retos no proporcionan nada más que simple diversión al usuario, pueden ser mejorados en versiones futuras de la aplicación para incluir un sistema de recompensas.

Hardware

Para el correcto funcionamiento de la aplicación se requiere un CC2541 SensorTag® o, en su defecto, un prototipo de dispositivo externo que contenga el mismo perfil genérico de atributos o GATT. Dicho dispositivo se encarga de la recopilación de datos mediante dos sensores diferentes, un acelerómetro y un giroscopio; esto permite localizar el pie del corredor en un mapa tridimensional.

La conexión inalámbrica establecida entre el dispositivo y la aplicación se realiza mediante el Bluetooth™ Low Energy system. Este tipo de conexión permite una rápida comunicación entre el dispositivo externo y el dispositivo móvil, haciendo que el envío de datos sea casi instantáneo. El rango de la conexión inalámbrica alcanza hasta los 50 metros, pero es recomendable que el usuario porte encima tanto el dispositivo externo como el dispositivo móvil.

Instalación

La instalación de la aplicación se realiza mediante el Play Store de google o con el archivo apk correspondiente. Se recomienda seguir los pasos de instalación del respectivo dispositivo móvil.

Manual de usuario

Al iniciar la aplicación, el usuario se encontrará con una pantalla vacía que constará de un solo botón (ver **Fig.48.**). En esta interfaz el usuario recibirá las alertas correspondientes a la activación del Bluetooth y de la geo localización.

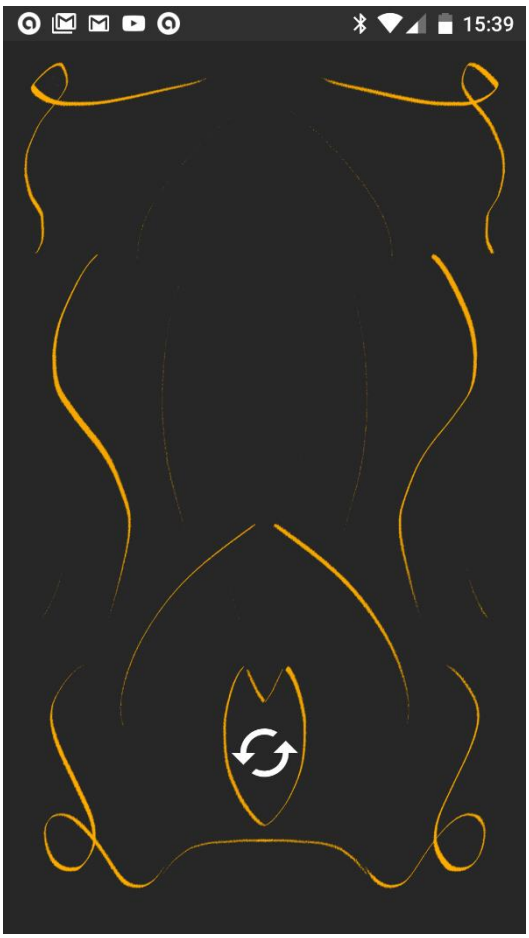

Fig.48. Pantalla de inicio de la aplicación Pro Run.

Para que el usuario pueda continuar interactuando con la aplicación, este debe pulsar el botón de escaneo para localizar los dispositivos BLE próximos. Este botón cambia de forma según si esta en reposo o haciendo la búsqueda (ver **Fig.49.**).

Fig.49. Botón en el estado de escaneo.

Una vez empezado el escaneo, se le muestra por pantalla al usuario los diferentes dispositivos encontrados en una lista (ver **Fig.50.**). Dicha lista tiene un formato que se compone de: un icono representando el dispositivo externo, el nombre dado por el usuario, el código de radio baliza del dispositivo y el nombre dado por el fabricante al dispositivo.

Cuando el usuario seleccione al dispositivo que se quiere conectar en la lista, la aplicación comprueba el nombre dado por el usuario al dispositivo. Si este es el nombre de fábrica, 'Nuevo dispositivo', se le redije a la interfaz de registro para nuevos dispositivos (ver **Fig.51.**). De lo contrario, si el dispositivo ya está registrado con un nombre diferente al de fábrica, se habilita la interfaz de usuario principal.

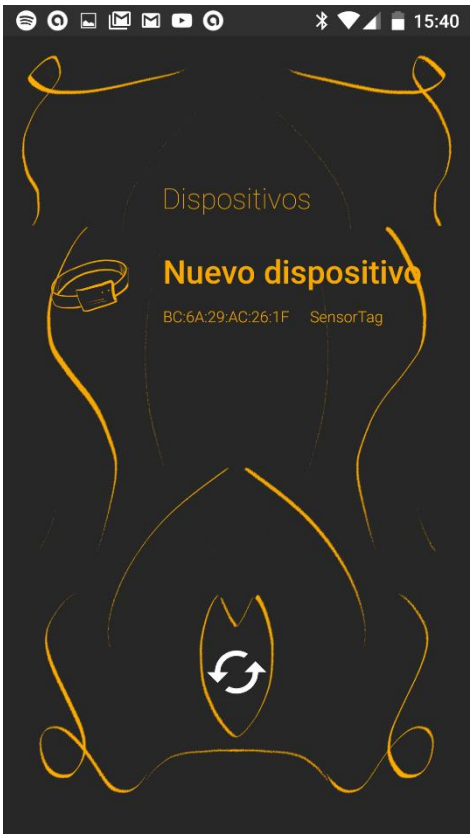

Fig.50. Lista de dispositivos encontrados en el escaneo

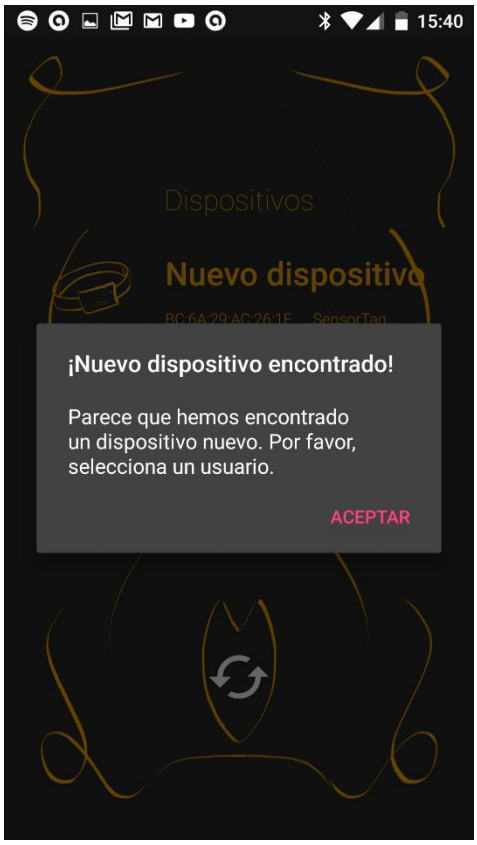

Fig.51. Alerta que indica al usuario que va ser redirigido al registro de dispositivos.

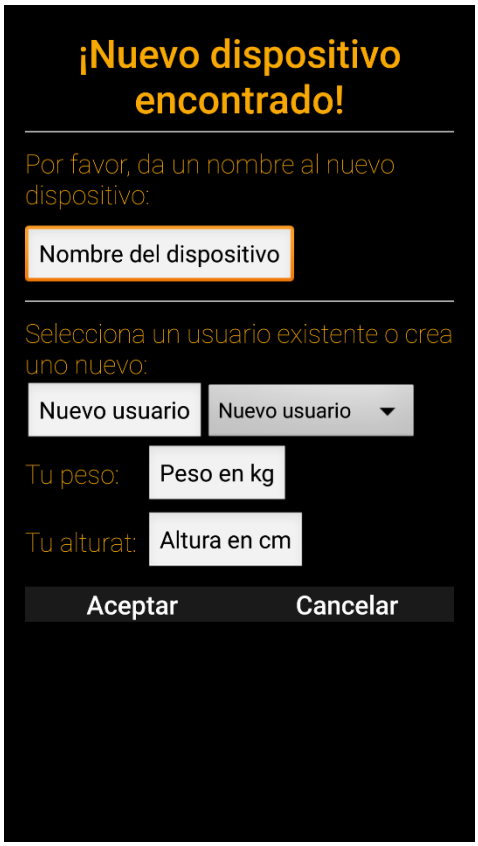

Fig.52. Cuestionario para el registro de nuevos dispositivos en la aplicación.

La actividad de registro de nuevo dispositivo, consta de un simple formulario donde el usuario debe indicar ciertos datos (ver **Fig.52.**). El nombre del dispositivo no tiene que dejarse en blanco y no admite el nombre de fábrica 'nuevo dispositivo'. El campo para el nombre de usuario no puede dejarse en blanco, se admite cualquier longitud de nombre de usuario. El peso del usuario debe introducirse en kilogramos, otras unidades no están implementadas en el dispositivo; lo mismo ocurre con la altura, sólo se admite en centímetros, otras unidades no están implementadas en el dispositivo.

Si el usuario presiona el botón de guardar, los datos son registrados en la base de datos. En caso contrario, si el usuario presiona el botón cancelar, el guardado de datos no es efectivo. Una vez guardados los cambios, o no, el usuario vuelve a la primera pantalla, teniendo que volver a buscar el dispositivo.

Habiendo registrado el dispositivo de forma exitosa, ahora aparece con el nombre dado por el usuario y podrá acceder a la interfaz de usuario principal después de seleccionar un usuario (ver **Fig.54.**). También se le asigna un reto diario a completar, este es mostrado mediante una alerta de diálogo al usuario (ver **Fig.53.**).

Hay implementados tres retos diarios diferentes en la aplicación: un reto de distancia, que consiste en correr 10 km; un reto de quema de calorías, este pide al usuario que queme 500 calorías a lo largo del día y por último, un reto cronometrado que exige al usuario que recorra una distancia de 3 km en 10 minutos.

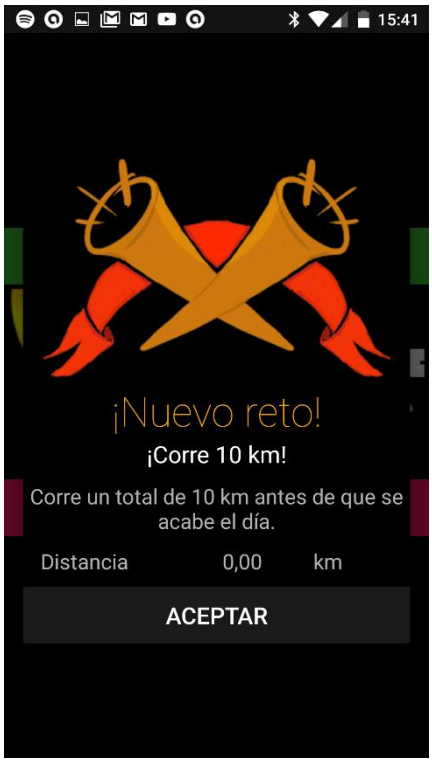

Fig.53. Alerta de nuevo reto diario.

Fig.54. Interfaz de usuario principal.

Una vez se ha accedido a la interfaz principal, el usuario puede interactuar con la aplicación mediante el inicio de actividades y su detención. También puede clicar el icono del trofeo para saber que reto diario debe completar. El icono del corredor sobre la pista le permite al usuario conocer sus estadísticas globales. Los engranajes permiten el inicio del dialogo de configuración y el icono de la papelera permite el borrado de datos por parte del usuario.

Cada vez que el usuario termine una actividad mediante el botón para detenerla, esta será registrada en la base de datos con el siguiente formato (ver **Fig.55.**):

Fig.55. Diálogo de alerta que muestra al usuario una actividad finalizada.