# UNIVERSIDAD CARLOS III DE MADRID

## ESCUELA POLITECNICA SUPERIOR


INGENIERÍA EN INFORMÁTICA

# PROYECTO FIN DE CARRERA

# "STEPCALORIES" APLICACIÓN DEDICADA A UNA VIDA SALUDABLE PARA ANDROID

**Autor: Rafael Gálvez Sánchez**
**Tutor: Francisco Javier Ordóñez Morales**
**Octubre 2011**

# *AGRADECIMIENTOS*

Tras 6 años de duro trabajo y esfuerzo, y una vez que el fin de la etapa de aprendizaje de mi vida está cerca con la finalización de este proyecto, creo que es hora de echar la vista atrás y agradecer a todas esas personas que me han ayudado a ser lo que soy.

Empezar, como no, a mis padres Rafael y María Dolores, los cuales han sido un apoyo constante a lo largo de mis 24 años de mi vida, estando ahí en lo bueno y en lo malo, para que siempre diera lo mejor de mí. No sé si he hecho todo lo que esperabais, pero para mí vosotros lo habéis hecho con creces. GRACIAS.

Gracias a Tania, la persona que aunque solo lleva un año en mi vida es muy importante para mí, por estar siempre animándome a seguir cuando las cosas se torcían, por forzarme para que esto sea una realidad, por enseñarme a ser mejor persona, por ese maravilloso icono que destaca por encima de muchas cosas… En general, por estar a mi lado ahora y para siempre. Nana, GMDT!

Javi (grrr!), Maca, Javi, Coral, Rober, Elena, Manu, Edu, Makaay, Alicia, Belén, Sara… todos mis amigos en general, gracias por ayudarme a probar y desarrollar este proyecto, a enseñarme que la vida no solo es estudio y sacrificio, que también se aprende del día a día. Gracias por todo!!

Como olvidarme de esa magnífica familia de Viena que tengo desde hace un año, a mi hermana Sarita (Axoo, que he acabado!), a Laura, a Veve, a Carlitos, a Elif, a Caglar, a Alex, a Rubén, a Maribel, a Javi, a María, a Lucia, a Ferrán, a Xavi… gracias por haberme dado el mejor año de mi vida, que aunque os pensarais que no hacía nada, aquí tenéis el fruto de todo este año. Es una pena que nunca más se vuelva a repetir, pero siempre nos quedara el recuerdo del Ride, las clases de español, las preparties, prepreparties, postparties, el Interrail, Simmering!, el Donau… Dankeschön!!

A la gente que he conocido en esta universidad, ya sea en clase (José, Judith, Jorge, Patri, Lillo, Sergio, Luis, Isa, Hector…) o en la biblioteca (Juanlu, Rocio, Ro, Irenes, Estelen, Raquelen, Palomen, Patri, David, Laura, MJ…), gracias por ayudarme a pasar esta etapa de mi vida, porque esas horas de prácticas y estudio no se hicieran interminables… Os echare de menos amigos.

Por supuesto, darle las gracias a mi tutor, Francisco Javier Ordoñez, por la ayuda y el apoyo que me ha ofrecido en todo momento, haciendo que este largo camino pareciera algo sencillo.

A todos, por si no os habéis dado cuenta... LO HE CONSEGUIDO!!!

# RESUMEN

El nacimiento de la telefonía móvil hace varias décadas, generó un mercado que ha ido creciendo día a día, haciendo que se invierta cada vez más tiempo y dinero en la tecnología unida a este concepto. Actualmente, no se usan teléfonos móviles simples, sino que se han convertido en *smartphones*: verdaderas obras de ingeniería capaces de incluir sensores de movimiento o posición, conectividades Wi-Fi o Bluetooth, cámaras de foto y video, etc., todo un sinfín de posibilidades.

Este desarrollo hizo que fuera necesaria la creación de sistemas operativos avanzados, capaces de dar uso a estas características. Para ello, desde ese tiempo hasta ahora, se están desarrollando todo tipo de aplicaciones para estos aparatos, ya sean traductores, navegadores, videojuegos o herramientas para la salud del ser humano, que es el caso en el que este Proyecto Fin de Carrera se sitúa.

Este PFC consiste en la creación de una aplicación dedicada a la vida saludable del usuario, para el sistema operativo Android. La herramienta contabiliza el número de pasos, distancia, velocidad media y gasto calórico al realizar el usuario la actividad de andar en un tiempo determinado, utilizando para ello diversas fórmulas y algoritmos de aprendizaje automático, las cuales utilizan unos datos que se obtienen a partir de un sensor de movimiento y un receptor GPS, características que incluyen la gran mayoría de los *smartphones*. Todo ello se implementa en una interfaz de usuario sencilla y moderna, con el propósito de que la aplicación atraiga y sea usada por el mayor número de personas, independientemente de sus conocimientos tecnológicos.

# ABSTRACT

When the mobile phone was born, some decades ago, it generated a Market which has been growing day to day, getting people to invest more and more time and money in technology related to this concept. Nowadays, not only simple mobile phones are used, they have become into smartphones: really great engineering devices which incorporate movement and position sensors, Wi-Fi or Bluetooth connections, photo camera and videocamera, etc., anything you can imagine.

The development of the smartphones required the creation of advanced Operating Systems, which could make such features possible. For that purpose, since then, all kind of applications has been developed to that hardware, like translators, navigators, videogames or tools for human health (case in which this final thesis is focused).

This final thesis consists on creating an application for Android OS dedicated to the people's healthy life. This tool counts the steps, distance, speed and calories burned when the user walks or runs during a period of time. It is based on certain mathematic equations and machine-learning-algorithms, which use the data that has been obtained from the movement sensors and the GPS receptor included on smartphones. Moreover, this software is implemented in an easy and modern user interfaces for the purpose of this application to be used by the greatest number of people, regardless their technological knowledge.

# TABLA DE CONTENIDOS

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1. INTRODUCCIÓN

Cuando, en 1871, un joven americano llamado Antonio Meucci [6] creó un aparato llamado "teletrófono" para comunicarse con su mujer desde el despacho de la planta baja de su casa a la segunda planta de la misma, seguramente nunca pensó que ese aparato lo patentaría Alexander Graham Bell [7] cinco años más tarde como teléfono, y mucho menos que el teléfono, gracias a las ondas de radio, se convirtiera en móvil.

Desde la invención del teléfono móvil en la década de los 90, este aparato electrónico ha sufrido una increíble transformación, pasando de ser únicamente un aparato para poder comunicarte mediante voz y mensaje con otra persona con un aparato similar, a poder navegar por internet, consultar tu correo electrónico, escuchar archivos MP3, trabajar con documentos, utilizarlo como GPS o poder tener la cartografía mundial en él. Este dispositivo evolucionado del teléfono móvil es el denominado "*smartphone*" [8].

La característica principal que diferencia un *smartphone* de un teléfono móvil común es la capacidad que posee este tipo de aparatos de instalar diferentes programas desarrollados por cualquier empresa o particular.

Esta capacidad hace que el aparato necesite de un conjunto de programas que gestionen de forma eficiente los procesos básicos del aparato, de forma que las aplicaciones instaladas funcionen de forma correcta en el aparato. Este conjunto de programas es lo denominado Sistema Operativo (SO).

Esto no quiere decir que los primeros móviles no llevaran un sistema operativo tal y como lo conocemos en cualquier ordenador personal. Los primeros teléfonos móviles utilizaban un sistema operativo mucho más simple que los actuales, debido a que no poseían la gran cantidad de características que tienen los s*martphones* actuales (recordar que los primeros modelos de teléfonos móviles no poseían ni siquiera pantalla).

La característica que une a todos los móviles de primera generación, última generación y *smartphones* es el tipo de SO que usan, denominado Sistema Operativo Embebido [9]. Este tipo de SO posee, como característica principal, que se encuentra instalado dentro de la propia placa base del aparato, por lo que el dispositivo posee un SO exclusivo. Esto hace que, por ejemplo, un teléfono móvil como "HTC Magic" solo pueda poseer como sistema operativo Android y no Symbian. Eso no significa que no se pueda actualizar la versión del SO, ya que las actualizaciones de cualquier SO embebido contemplan el dispositivo donde se encuentra instalado.

Los *smartphones*, y más en general, el mundo de la tecnología (videoconsolas, netbooks, etc.), están atrayendo cada vez más el interés del ser humano actual. Pero, al igual que el mundo de las tecnologías, también existe cada vez más interés acerca de la salud, la apariencia física, etc. La gente acude en su tiempo libre a gimnasios, se observa cada vez más gente joven y adulta corriendo o dando paseos por zonas verdes y parques urbanos, existe una mayor preocupación por la apariencia y la moda (sobre todo en hombres). Según un estudio realizado por el Consejo Superior de Deportes (CSD) [10], el hábito deportivo ha aumentado más de un 10% entre la sociedad española desde 1975 al 2005, tal y como se puede observar en la Fig. 1.

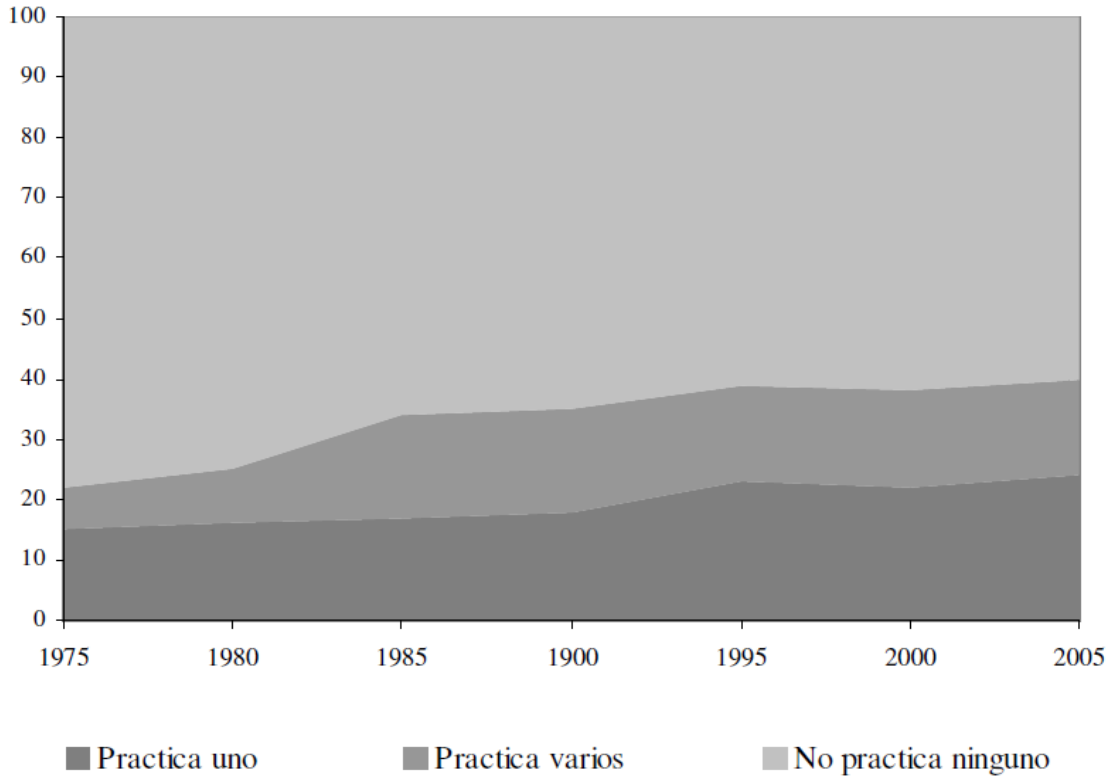



**Fig. 1. Estudio del hábito deportivo entre 1975-2005**

También el interés por la calidad nutricional ha aumentado respecto a épocas anteriores. Actualmente la sociedad vigila mucho más el consumo de comida sana, existen sitios especializados en el cuidado de la nutrición [12], y, en general, todo el mundo se empieza a preocupar por lo que come y como les afecta al organismo. Por ejemplo, en los sitios de comida rápida ahora aparece la tabla nutricional de cada plato, incluso en las web te permiten configurar tu plan nutricional en base a sus platos [11].

Como se ha comentado anteriormente, los actuales *smartphone* poseen, además de la capacidad de llamar y enviar mensajes, gran cantidad de características útiles para el ser humano, pero… ¿cómo puede el ser humano aprovecharse de estas características, combinándose con el estilo de vida actual?

La idea de este proyecto se basa en, debido a la preocupación actual que existe por la salud y al concepto de "vida sana" practicando deporte (como running, jogging, o, simplemente, pasear), utilizar algunas de las características que nos otorgan los *smartphones* (GPS, acelerómetro…) para calcular la cantidad de kilocalorías (Kcal) que se eliminan cuando se camina (esto se podría exportar a cualquier actividad física).

## *1.1 ESTRUCTURA DE LA MEMORIA*

Después de una breve introducción, a continuación se detalla la estructura del resto del documento:

1. *INTRODUCCIÓN*: Se realizará un pequeño resumen del universo de la aplicación, estructura de la memoria, objetivos primarios…

2. *GESTIÓN Y ORGANIZACIÓN DEL PROYECTO:* Se tratarán todos los temas relativos a la metodología que se va a seguir en la ejecución de este proyecto.

3. *ESTADO DEL ARTE:* Se analizarán las últimas novedades y recientes desarrollos en el mundo tecnológico actual que tengan algún tipo de relación con el uso de terminales móviles como soporte de un estilo de vida saludable.

4. *OBJETIVOS*: Se enumerarán los objetivos de este Proyecto Fin de Carrera, tanto el objetivo general como los específicos.

5. *DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN:* Se detallará el trabajo realizado ofreciendo una descripción de los requisitos, arquitectura y creación de la aplicación.

6. *PRUEBAS Y RESULTADOS* Se mostrarán los resultados obtenidos en la fase experimental de este proyecto, demostrando que se cumple con lo exigido en el diseño.

7. *CONCLUSIONES:* Se narrarán las conclusiones obtenidas tras la realización de este proyecto.

8. *LÍNEAS FUTURAS*: Por último, en este capítulo figurarán los trabajos futuros que se pueden llevar a cabo sobre la base del trabajo realizado.

# 2. GESTIÓN Y ORGANIZACIÓN DEL PROYECTO

Un proyecto tecnológico se considera exitoso cuando cumple con las especificaciones exigidas por el cliente, satisface a los usuarios, es mantenible (posibilidad de actualización de forma sencilla) y, sobre todo, está desarrollado dentro del presupuesto y del tiempo estimado para ello. Esto implica que por cualquier motivo, como falta de recursos, fallo en los objetivos, obviarse técnicas… el proyecto pueda considerarse como fracaso.

Por ello, todo proyecto necesita de una estructuración en la que se planifique tanto los pasos a realizar y cuánto tiempo se va a emplear en cada paso (planificación), como qué se va a tratar en cada paso (gestión u organización). En este apartado se va a tratar de describir esas diferentes etapas, es decir, cómo se va a gestionar y organizar cada una de las partes más importantes del proyecto.

Desde los años setenta, cuando se empezó a definir qué es considerado Ingeniería del Software [13], uno de los objetivos ha sido la búsqueda y creación de metodologías y estándares que consigan definir la creación del software como una cadena de trabajo continua, de forma que siguiendo esa metodología o estándar, se pueda crear un producto software más fácilmente. Actualmente existen varias metodologías y estándares que permiten crear software de una forma ordenada.

Este proyecto se basa, en concreto, en el estándar diseñado por la Agencia Espacial Europea (ESA) [14] en 1991, cuyo nombre es PSS-05-0 [15]. Este estándar tiene en cuenta la variación que un proyecto software puede tener en tamaño, complejidad, propósito y cantidad de recursos. Dependiendo de estos factores principales, además de otros secundarios como criticidad del software o riesgos y estabilidad de los requerimientos, el estándar distingue dos tipos distintos de proyectos, un proyecto *normal*, y un proyecto a *menor escala* (a este último tipo pertenece este proyecto).

La principal diferencia entre los dos tipos de proyectos es la duración, ya que el proyecto es pequeño si este es menor de dos años-hombre; sino el proyecto es un proyecto normal (también se considera un proyecto largo cuando es más de 20 años hombre, pero este proyecto no es realmente factible). Otras diferencias menos significativas entre los tipos de proyectos son:

- El equipo de trabajo (menor de 5 personas es un proyecto pequeño).
- El número de líneas de código (menor de 10000 líneas es un proyecto pequeño).

Una vez definido qué estándar se va a seguir durante el proyecto y qué tipo de proyecto se está realizando, se pasa a definir cada una de las características o pasos que se van a aplicar a partir del estándar descrito anteriormente.

## *2.1 CICLO DE VIDA DEL SOFTWARE*

Un producto software se inicia con la definición de los requisitos necesarios para el sistema y finaliza cuando el producto está en desuso y no se le realizan más actualizaciones o mantenimientos. Todo el desarrollo desde la fase inicial hasta la fase final, definiendo cada una de las sub-tareas incluidas en este proceso así como el enlace o comunicación entre las sub-tareas es lo denominado *ciclo de vida*.

Además de definir las tareas, también comentar que el ciclo de vida asegura que no se produzcan errores en el proceso, ya que su intención es que todos los productos se realicen siguiendo un ciclo de vida definido anteriormente, creando un proceso "automatizado". Para este proyecto, dentro de los diferentes ciclos de vida existentes, se ha elegido el *Modelo en Cascada,* en su versión *Retroalimentada*.

### 2.1.1 MODELO EN CASCADA RETROALIMENTADO

Este modelo está basado en el Modelo en Cascada, también conocido como "Modelo clásico" o "Modelo lineal tradicional", que fue diseñado entre 1966 y 1970 [16].

La idea básica de este modelo consiste en la definición completamente ordenada de cada una de las fases del ciclo de vida, de forma que una etapa no puede comenzar hasta que no ha finalizado la etapa anterior. La idea principal es muy lógica y útil, pero en el universo del desarrollo del software es conocido que es imposible realizar un proyecto sin tener que retocar cualquier cosa de una etapa anterior a la que se encuentra en ese momento el desarrollo.

A pesar de esta idea básica, existen diferentes variantes que desarrollan este modelo, ya que, debido a la simplicidad y eficacia del modelo, es uno de los más utilizados en la creación de software, sobre todo en los proyectos a pequeña y mediana escala. Entre estas variantes existe, por ejemplo, una basada en la refinación de etapas, o aquella que vamos a utilizar en este proyecto, y que está basada en la idea de la *retroalimentación*.

La retroalimentación consiste en una "ida y vuelta", es decir, que la salida de una etapa puede utilizarse como entrada de una etapa anterior. Por ejemplo, si ya se han definido y especificado los requisitos del sistema, se pasará a la etapa de diseño del sistema, pero lo más seguro es que en esa etapa el equipo de desarrollo se dé cuenta de algunos errores cometidos en la especificación de requisitos y deban volver atrás para realizar los ajustes necesarios.

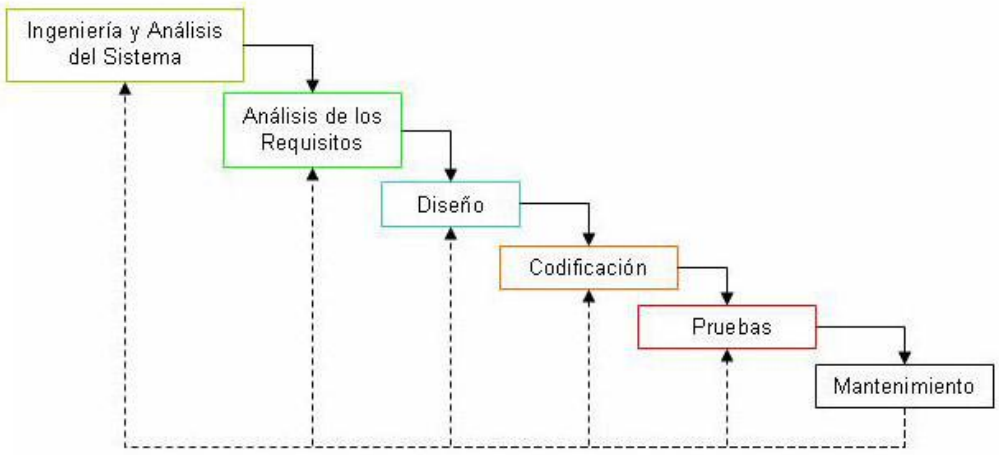



**Fig. 2. Etapas del Modelo en Cascada Retroalimentado**

Las distintas etapas con las que cuenta este modelo, y que estas definidas en el diseño que se encuentra en la Fig. 2, son las siguientes:

1. *INGENIERÍA Y ANÁLISIS DEL SISTEMA*

Debido a que el software casi siempre es parte de otro sistema mayor que el propio software, la primera etapa consiste en estudiar el sistema y establecer que requisitos son necesarios en el sistema, para conocer cuáles de esos requisitos pertenecen a nuestro software.

2. *ANÁLISIS DE REQUISITOS*

En la segunda etapa se extrae que es lo que el cliente demanda y se especifica cuáles son los requisitos a cumplir por el software. Todos estos requisitos serán consensuados sin posibilidad de cambio en el futuro.

El documento creado en esta etapa se denomina Documento de Especificación de Requisitos (SRD), que especifica lo que hará el sistema sin detalles de carácter interno.

3. *DISEÑO DEL SISTEMA*

En esta etapa se realiza la traducción de los requisitos especificados en el análisis de requisitos a un diseño del sistema, que se divide en dos: el diseño de alto nivel o diseño arquitectónico y el diseño detallado.

El diseño desarrollado se basa en cuatro pilares básicos: la arquitectura del software, la estructura de los datos, el detalle procedimental y la personalización de la interfaz.

El documento creado en esta etapa se denomina Documento de Diseño del Software (SDD), que especifica la estructura global del sistema y, a su vez, cada parte del software, describiendo también como se deben comunicar cada una de las partes.

*4. CODIFICACIÓN*

Es la fase en la que más se trabajará en este proyecto. Se realiza la traducción del diseño detallado a código fuente legible para el computador. Lo idóneo sería que este paso fuera totalmente automatizado y que, a partir del diseño detallado, se obtuviera el código fuente, pero esto es imposible de conseguir con las herramientas de programación actuales.

*5. PRUEBAS*

Una vez codificado y ensamblado el producto software, se deben realizar una serie de comprobaciones para chequear que el software es estable, se integra correctamente en el sistema y, por supuesto, cumple con lo especificado por el cliente en las primeras fases.

*6. MANTENIMIENTO*

Esta última fase se realiza una vez ha sido entregado el producto al cliente. Se suelen centrar gran cantidad de recursos en esta fase, ya que en ella se controla que, en caso de que el usuario final encuentre errores, corregirlos.

Además, en esta etapa se incluye que el software deba sufrir modificaciones debidos a cambios externos al sistema (por ejemplo, SO o algún periférico nuevo), o que el cliente quiera realizar cualquier tipo de actualización o ampliación del software.

- ***DESVENTAJAS***

Como cualquier otro, este modelo no es perfecto, y a pesar de ser uno de los más utilizados, también posee varias desventajas, como son:

- En casi todos los proyectos, aunque su idea principal sea seguir un desarrollo lineal, se acaba por realizar más de una iteración, lo que hace que el paradigma del método pierda coherencia.

- Normalmente, el cliente no sabe expresar de manera sencilla y concisa que es lo que realmente desea obtener, por lo que resulta muy difícil extraer todos los requisitos del sistema en las primeras fases y puede que a lo largo de la creación del software surjan incertidumbres difíciles de solucionar (por ello es importante el contrato firmado al final del análisis de requisitos).

- Cualquier error que sea descubierto en la fase de pruebas conlleva al rediseño y nueva codificación, lo que aumenta el coste del desarrollo.

## *2.2 REQUISITOS DE USUARIO Y DE SOFTWARE*

Como hemos visto en el apartado acerca del ciclo de vida del software, existen varias fases en las que se define qué es lo que el software debe hacer (en base al usuario) y cómo debe interactuar con el sistema (tratado por el ingeniero). Cada uno de los puntos que define alguna característica de qué o cómo debe actuar el producto, es lo que se considera en Ingeniería del Software *requisito*.

Según el estándar en el que este proyecto se basa (PSS-05-0), se distinguen dos tipos de requisitos:

- *Requisitos de Usuario:* Son aquellos puntos que el cliente demanda al software, aquello que el software debe ser capaz de hacer. Son definidos por el *usuario*, pero el ingeniero o analista debe ser capaz de, en base a entrevistas abiertas, reuniones en grupo con los usuarios finales, observación del entorno de trabajo, etc., extraer de la manera más exacta y concisa que es lo que el cliente quiere. En este proyecto se van a usar dos tipos de requisitos de usuario: *Capacidad* (que cosas debe hacer el software) y *Restricción* (limitaciones del software acerca de cómo debe construirse o debe operar).

- *Requisitos de Software*: Son aquellos requisitos que definen el comportamiento del sistema a desarrollar. Son definidos por el *ingeniero*, ya que en base a lo que ha aprendido del cliente con los Requisitos de Usuario, debe conocer que características del software ajenas a la funcionalidad debe poseer el producto.
  En este proyecto se van a usar los siguientes tipos de requisitos de software: *Funcionales* (aquellos que dicen que debe realizar el software), *Rendimiento* (especifican el valor de variables medibles como velocidad, frecuencia…), *Interfaz* (aquello que dicen los elementos del sistema con los que debe interactuar el software), *Operacionales* (especifican cómo el sistema se comunica con el ser humano y el resto del software), *Recursos* (definen los requisitos físicos de la maquina donde correrá el software) y *Seguridad* (precisan cual debe ser la seguridad del software respecto a integridad, disponibilidad…)

A la hora de especificar un requisito, los dos tipos de requisitos cuentan con los mismos atributos, que son los siguientes: identificador, nombre, necesidad, prioridad, estabilidad, fuente, verificabilidad y descripción.

Según esto, un ejemplo de requisito sería el mostrado en la Tabla 1:

| **ID: RU-C01** | | | |
|---|---|---|---|
| **Nombre:** | Idea genérica | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá contabilizar el número de pasos, distancia, velocidad media y gasto calórico al realizar el usuario la actividad de andar o correr en un tiempo determinado. | | |

**Tabla 1. Ejemplo de requisito de usuario**

Tal y como se asegura en el estándar seguido, todos los requisitos del documento deben ser verificables, es decir, que se debe usar algún mecanismo para demostrar que todos los requisitos que demando el cliente se están implementando a lo largo de todo el proyecto. Para ello, el mecanismo implementado con los requisitos de usuario y de software es la llamada Matriz de Trazabilidad [18], que chequea que cada uno de los requisitos de usuario esta implementado en, al menos, un requisito de software. Si en algún caso esto no se cumple, se deberían volver a chequear los requisitos, ya que algo en el proceso está fallando.

La definición de los requisitos se llevará a cabo en la sección 'REQUISITOS DE LA APLICACIÓN'.

## *2.3 DISEÑO ARQUITECTÓNICO Y DETALLADO*

Siguiendo el ciclo de vida utilizado en este proyecto, una vez extraídos los requisitos que el usuario y el ingeniero consideran necesarios, es el momento de pensar cómo queremos que el software esté construido. Haciendo un símil con la construcción de cualquier edificio, antes de empezar a construir se necesita definir la estructura, funcionamiento e interacción entre las distintas partes del software, lo que es llamado el diseño del software.

Todo diseño del sistema debe seguir los siguientes principios básicos:

- *Modularidad*: Cada sistema debe estar formado por una jerarquía de módulos. Los módulos de niveles inferiores son menores en alcance y tamaño comparados con los módulos de nivel superior y sirven para fragmentar procesos en funciones separadas.

- *Acoplamiento*: Los módulos de un sistema deben tener poca dependencia entre sí.

- *Cohesión*: Los módulos deben llevar a cabo sólo una función de procesamiento.

- *Extensión de Control*: Los módulos deben interactuar entre si y coordinar las funciones de un número limitado de módulos de nivel inferior.

- *Tamaño*: El número de instrucciones contenidas en un módulo debe ser limitado.

- *Uso compartido*: Las funciones no deben repetirse en módulos separados, sino establecerse en un único módulo que pueda utilizar cualquier otro cuando sea necesario.

A la hora de realizar el diseño del software, existen dos etapas claramente diferenciadas dependiendo del nivel de abstracción en el que se encuentre el diseño. En el *diseño arquitectónico* (también llamado 'diseño de alto nivel'), se intenta llegar a una buena comprensión del problema sin entrar en cómo va a ser la solución en cuanto a detalles de implementación. Una vez realizado esto y justo antes de realizar la codificación del software, se lleva a cabo la segunda etapa del diseño, el *diseño detallado*, en el que se describe la lógica del programa, el control jerárquico, las estructuras de datos, la unión de componentes, etc.

Al igual que ocurría en la etapa anterior, donde los requisitos de usuario debían ser verificados en los de software, en esta fase también se deberá realizar la verificación de que todo requisito de software debe ser satisfecho por, al menos, un componente de la arquitectura desarrollada, con el fin de asegurar la continuidad del proyecto hasta su etapa de codificación.

El diseño del software se abordará de forma más detallada en la sección 'ARQUITECTURA DE LA APLICACIÓN'.

## *2.4 IMPLEMENTACIÓN DEL SOFTWARE*

Una vez concluido el diseño del software, y con toda la información necesaria para llevar a cabo la realización del programa, se lleva a cabo la labor correspondiente a esta etapa, la codificación de la aplicación.

A pesar de que esta es la etapa que cuenta con menor documentación escrita, es, a su vez, la más importante y larga del proyecto, ya que todos los pasos anteriores están dirigidos a la correcta creación del software, que es el trabajo a realizar en este punto del proyecto. Además, en este paso se crea aquello que realmente se va a entregar al cliente, ya que toda la documentación escrita no es entregada al cliente porque es útil solo para el equipo de desarrollo.

Todo el contenido referente a la implementación de la aplicación se encuentra en el apartado 'IMPLEMENTACIÓN DE LA APLICACIÓN'.

## *2.5 EVALUACIÓN DE LOS RESULTADOS*

En este último paso del ciclo de vida de nuestro proyecto, se realizará una evaluación del software, comprobando que funciona correctamente y que los objetivos definidos al principio del proyecto han sido introducidos en el código final.

Para realizar esta evaluación, se utilizará una batería de pruebas del sistema. Esta batería consiste en un conjunto de pequeñas ejecuciones de la aplicación, explorando todas sus posibilidades en busca de errores o inconsistencia de datos, además de chequear que aquellas funcionalidades definidas cuando se creó la arquitectura, realmente han sido implementadas.

Toda la información correspondiente a la evaluación y pruebas del sistema se encuentra en el capítulo 'PRUEBAS Y RESULTADOS'.

# 3. ESTADO DEL ARTE

En este apartado se ofrece una visión general del área en el que se enmarca este Proyecto Fin de Carrera. De forma introductoria en el primer apartado se describe el sistema operativo sobre el que se basa la aplicación, así como las distintas versiones existentes de este SO. Posteriormente, en la sección 3.2 se realiza un resumen acerca de otras herramientas relacionadas con la salud y que son similares a la que se va a realizar en este proyecto, como por ejemplo la extendida herramienta Nike+. En la siguiente sección se explica teóricamente cómo se realiza el cálculo del gasto calórico. A continuación, en la sección 3.4, se habla sobre todo lo relacionado con el GPS y su funcionamiento. Por último, en la sección 3.5, se incluye la descripción de las herramientas que se han utilizado para llevar a cabo este Proyecto Fin de Carrera.

## *3.1 ANDROID*

En el mundo tecnológico en el que vivimos actualmente, con una gran cantidad de diferentes aparatos electrónicos pero azotado por la crisis económica mundial, el desarrollo de software con estándares libres y en el que tenga cabida toda la comunidad informática, es una parte verdaderamente importante. Por esta dirección es por la que apuesta el nuevo sistema operativo de Google, Android.

Android es el sistema operativo desarrollado por la *Open Handset Alliance* y Google, basado en Linux y cuyos principales destinatarios son los dispositivos móviles, como *smartphones* y *tablets*.

### 3.1.1 Historia

El sistema operativo comenzó su vida en el año 2003, cuando Andy Rubin [20] (co-fundador de Danger), Rich Miner [21] (co-fundador de Wildfire Communications, Inc.), Nick Sears (alguna vez VP en T-Mobile), y Chris White (quien encabezó el diseño y el desarrollo de la interfaz en WebTV) decidieron crear una empresa para desarrollar, según Rubin, "…pequeños dispositivos móviles que tengan en mayor consideración la localización y preferencias del propietario". Esta empresa descrita fue llamada *Android Inc*., y fundada en Palo Alto, California.

Pero cuando realmente este proyecto empezó a fraguarse fue a mediados del año 2005 (concretamente en Junio), momento en el cual Google compró esta pequeña compañía proveniente de Palo Alto.

Google seleccionó esta pequeña empresa porque el grupo de fundadores tenía gran experiencia en plataformas web, telecomunicaciones y aplicaciones móviles. Tras esta compra, Rubin, Miner y White pasaron a formar parte de la empresa Google después de esta adquisición. A pesar de que en ese momento no se tenía mucha información acerca de esta empresa, Google, con este movimiento, dejó más o menos claro que iba a entrar en el mundo de los dispositivos móviles.

Una vez en Google, el equipo liderado por Rubin desarrolló una plataforma para dispositivos móviles basada en el kernel de Linux, cuya premisa principal era la de tener un sistema flexible y actualizable. Además, Google entro en conversaciones con una serie de fabricantes de hardware y software, señalándoles que estaba abierto a diversos grados de cooperación por su parte.

Todas las especulaciones acerca de la entrada o no de Google al mercado móvil aumentaron en Diciembre de 2006, ya que diversas informaciones provenientes de la 'BBC' y '*The Wall Street Journal*' anunciaban que Google quería obtener sus buscadores y aplicaciones para los teléfonos móviles.

En Septiembre de 2007, '*InformationWeek*' difundió un estudio de '*Evalueserve'* en el que se demostraba que Google había solicitado diversas patentes en el área de la telefonía móvil, dando por asegurado la entrada de la compañía en el mercado (lo que todavía no estaba claro es si sería con un dispositivo, un SO…).

Varios meses después, el 5 de Noviembre de 2007, se crea la '*Open Handset Alliance'* (OHA) [22], un consorcio de varias compañías lidero por Google con otras 34 empresas (entre las que están Texas Instruments, Broadcom Corporation, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, Intel, LG, Marvell Technology Group, Motorola, HTC o T-Mobile) con el fin de desarrollar estándares abiertos para dispositivos móviles.

Al mismo tiempo que la formación del consorcio, la OHA estrenó su primer producto, *Android*, una versión beta de una plataforma para dispositivos móviles construida sobre la versión 2.6 del kernel de Linux. Con este importante anuncio, se cerraban todo tipo de especulaciones acerca de la entrada de Google a la industria móvil.

El 9 de diciembre de 2008 se anunció la unión al proyecto Android de 14 nuevos miembros, entre los que destacan Asustek, Garmin, Softbank, Sony Ericsson, Toshiba o Vodafone. Desde ese momento, poco a poco más empresas han ido entrando en el consorcio hasta Noviembre de 2010, formando el consorcio un total de 80 empresas.

Finalmente, el 23 de Septiembre de 2008, se lanzó el teléfono móvil 'HTC Dream' (G1), el primer teléfono móvil con sistema operativo Android, en su versión 1.0.

### 3.1.2 Características

Android fue diseñado para contener las siguientes características:

- *Diseño de dispositivo*: La plataforma es adaptable a pantallas más grandes, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 [23] y, en general, al diseño de *smartphone* tradicionales.

- *Multitarea*: Esta disponible la ejecución multitarea real de aplicaciones.

- *Conectividad*: Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE y WiMAX.

- *Entorno de desarrollo*: Incluye un emulador de dispositivos, herramientas para la depuración de memoria y análisis del rendimiento del software. El entorno de desarrollo integrado es Eclipse, usando el *plug-in* de Herramientas de Desarrollo de Android (SDK).

- *Almacenamiento*: El sistema operativo usa SQLite [23], una base de datos liviana, la cual es usada para propósitos de almacenamiento de datos.

- *Multi-táctil*: Android incluye soporte nativo para pantallas multi-táctiles, que inicialmente hicieron su aparición en dispositivos como el 'HTC Hero'. La funcionalidad fue originalmente desactivada a nivel de kernel (posiblemente para evitar infringir una patente de Apple relacionada con tecnología de pantallas táctiles), pero más tarde Google publicó una actualización para el Nexus One' y el 'Motorola Droid' que activaba el soporte para pantallas multi-táctiles de forma nativa.

- *Mensajería*: SMS y MMS son formas de mensajería, incluyendo mensajería de texto continua, y ahora la '*Android Cloud to Device Messaging Framework'* (C2DM) es también parte del servicio de '*Push Messaging de Android*'.

- *Navegador web*: El navegador web incluido en Android está basado en el motor de renderizado de código abierto 'WebKit', emparejado con el motor JavaScript V8 de 'Google Chrome'. El navegador obtiene una puntuación de 93/100 en el test Acid3 [25].

- *Soporte multimedia*: Android soporta los siguientes formatos multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en un contenedor 3GP o MP4), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.

- *Soporte de Java*: Aunque las aplicaciones son escritas en Java, no hay una Máquina Virtual de Java en la plataforma, por lo que el código Java no es ejecutado. Para solucionar esto, el código Java corre en la 'Máquina Virtual Dalvik' [26], con el código escrito en ejecutables Dalvik. El soporte para J2ME puede ser agregado mediante aplicaciones de terceros.

- *Soporte para streaming*: Android es capaz de reproducir streaming RTP/RTSP (3GPP PSS, ISMA) y soporta descarga progresiva de HTML (HTML5 <video> tag). El formato Adobe Flash Streaming (RTMP) y Adobe Flash HTTP Dynamic Streaming son soportados mediante el Adobe Flash Player. Apple HTTP Live Streaming es soportado por la versión móvil de RealPlayer, aunque se planea en la versión 3.0 que el sistema operativo sea capaz de reproducirlo. Por último, también se planea el soporte de Microsoft Smooth Streaming con el *plug-in* de Silverlight para Android.

- *Soporte para hardware adicional:* Android soporta cámaras de fotos, de vídeo, pantallas táctiles, receptores GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, termómetros, aceleradoras de bit 2D (con escaladores o conversión a formato pixel) y aceleradoras graficas 3D.

- *Bluetooth:* El soporte para A2DF y AVRCP fue agregado en la versión 1.5. El envío de archivos (OPP) y la exploración del directorio telefónico (PBAP) fueron agregados en la versión 2.0. El marcado por voz, junto con el envío de contactos entre teléfonos, fueron añadidos en la versión 2.2. Teclado, ratón y joystick (HID) están disponibles a través de modificaciones del sistema y aplicaciones de terceros.

- *Videollamada*: La versión principal de Android no soporta videollamada. Sin embargo, algunos dispositivos podrían tener una versión (personalizada para un operador determinado) del sistema operativo que lo soporta, ya sea por la red del operador (vía UMTS) o sobre IP. Para la versión 3.0 está planeado incluir videollamada a través de Google Talk [27].

- *Market*: El Android Market es un catálogo de aplicaciones que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.

- *Características basadas en voz*: La búsqueda en Google a través de voz está disponible desde la versión inicial del sistema. La posibilidad de llamar, escribir, navegar por internet… a través de voz, está disponible desde la versión 2.2.

- *Tethering*: Android soporta '*tethering'*, el cual permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico. Todos los teléfonos con una versión anterior a la versión 2.2., solo tenían esta característica a partir de aplicaciones de terceros disponibles en el Android Market, por ejemplo PdaNet [23]. A partir de la versión 2.2., se encuentra incluida dentro del sistema operativo.

### 3.1.3 Arquitectura

En la Fig. 3 se muestran los componentes principales del SO Android. Dentro de esta arquitectura se pueden diferenciar cinco grandes bloques:

- *Android Runtime*: Android incluye un conjunto de bibliotecas base, que son las encargadas de ofrecer la mayoría de las funciones disponibles en las bibliotecas del lenguaje de programación 'Java'. En este SO, cada aplicación se ejecuta sobre su propio proceso, con su propia instancia de la máquina virtual Dalvik, que se ha escrito de modo que un dispositivo puede ejecutar múltiples máquinas virtuales de manera eficiente. El formato ejecutado por la máquina virtual Dalvik es el formato *Dalvik Executable* (.dex), el cual está optimizado para ocupar la menor memoria posible. Dalvik, además, está basada en registros y posee una herramienta llamada 'DX', que ejecuta las clases compiladas por el compilador de Java que han sido transformadas al formato .dex.

- *Librerías*: Android incluye un conjunto de librerías de C y C++, que son utilizadas en diversos componentes del sistema. La capacidad de las librerías son expuestas a los desarrolladores a través del 'Framework de aplicaciones' de la capa superior. Algunas de las principales librerías incluidas son: '*System C library*' (implementación biblioteca C estándar); Medios de comunicación de librerías '*lde'* basadas en 'OpenCORE'; *PacketVideo* (reproducen y grabar en diferentes formatos de audio y vídeo, así como archivos de imagen, MPEG4, H.264, MP3, AAC, AMR, JPG y PNG); Superficie Manager, (gestiona el acceso a la pantalla y a los subsistemas compuestos 2D y gráficos 3D a partir de capas múltiples aplicaciones); *LibWebCore* (un moderno navegador web); SGL (el motor de gráficos 2D subyacente); Librerías 3D basada en '*OpenGL ES 1.0 API*'; *FreeType* (mapa de bits y vectores de la renderización de fuentes); *SQLite* (un ligero y potente motor de bases de datos relacionales).

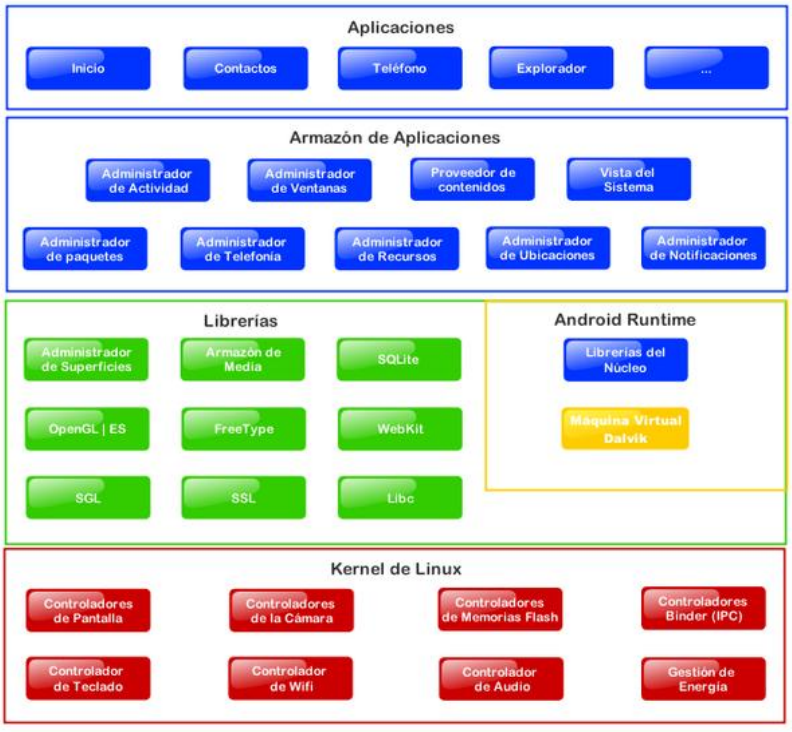



**Fig. 3. Arquitectura de Android**

- *Kernel de Linux*: Android está basado en la versión 2.6 del núcleo de Linux, que es utilizada para crear la capa de abstracción entre el hardware y el resto de elementos software del sistema, además de ser la capa que ofrece los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores.

- *Armazón de Aplicaciones (Application Framework)*: Los desarrolladores tienen pleno acceso a las mismas aplicaciones de este marco que las aplicaciones base. La arquitectura de aplicaciones se diseñó para simplificar la reutilización de componentes, de modo que cualquier aplicación puede publicar sus capacidades y, en otro momento, cualquier otra de las aplicaciones podrá hacer uso de esas capacidades (siempre que respete las limitaciones de seguridad impuestas por el marco). Este mismo mecanismo permite que los componentes sean sustituidos por el usuario. Detrás de todas las aplicaciones existen un conjunto de servicios y sistemas que incluyen una rica y extensible conjunto de vistas que se pueden utilizar para construir una solicitud (listas, botones, etc.).

- *Aplicaciones*: El sistema operativo Android, en su versión más básica, se suministra con un conjunto de aplicaciones base, que incluyen: un cliente de correo electrónico, programa de envío de mensajes SMS, calendario, mapas, navegador, contactos etc. Todas estas aplicaciones están escritas en lenguaje Java, que serán ejecutadas en Dalvik. Aparte de estas aplicaciones base, muchas más se pueden obtener desde Android Market, el catálogo de aplicaciones exclusivo de Android.

### 3.1.4 Versiones

Desde la primera versión estable lanzada al mercado en Septiembre de 2008, se han ido lanzando un gran número de nuevas versiones. Estas versiones normalmente corrigen algunos errores de la versión anterior y/o añaden nuevas funcionalidades. Google, para el nombrado de las versiones, ha usado nombres de diferentes postres en ingles ordenados alfabéticamente. Cada una de las versiones lanzadas son las siguientes:

- *1.0 (Android)*: Lanzada el 23 de Septiembre de 2008. Primera versión del sistema operativo, instalada en el 'HTC Dream', que cuenta con las siguientes características: soporta Wi-Fi y Bluetooth; Android Market; navegador; cámara (sin cambiar resolución, balance de blancos…); email; múltiples iconos en pantalla de inicio; Gmail sincronizado; Google Contacts; Google Calendar; Google Maps con *Street View* (usa también el GPS del teléfono); Google Sync; Google Search; Google Talk; mensajería instantánea; Media Player con soporte por Bluetooth; notificaciones en la barra de estado; fondo de pantalla detrás del menú inicio y Youtube, entre otras.

- *1.5 (Cupcake)*: Basada en la versión *kernel* de Linux 2.6.27, esta versión fue lanzada el 30 de Abril de 2009, incluyendo las siguientes nuevas funcionalidades: teclado virtual con predicción de texto y diccionario; *widgets* (mini aplicaciones que se incrustan en el menú inicio y que reciben actualizaciones periódicas); videocámara; galería con soporte para video; las características copia y pega se añaden al navegador; se puede añadir una foto a cada contacto y/o hacerlo favorito; transiciones animadas en los menús y posibilidad de subir videos y fotos a Youtube y Picassa.

- *1.6 (Donut)*: Basada en la versión *kernel* de Linux 2.6.29, esta versión fue lanzada el 15 de Septiembre de 2009, incluyendo las siguientes nuevas funcionalidades: búsqueda por voz; las aplicaciones pueden incluir su contenido en los resultados de las búsquedas; traductor de voz a texto en diferentes lenguas; mejoran las búsquedas en Android Market; se integran en un solo acceso galería, cámara y videocámara; selección múltiple de archivos en la galería; se mejora el SO con CDMA/EVDO, 802.1x y VPNs; soporte para pantallas WVGA; 'Navegador GPS Turn-by-turn' incluido.

- *2.0/2.1 (Eclair)*: Basada en la versión *kernel* de Linux 2.6.29, esta versión fue lanzada el 26 de Octubre de 2009, incluyendo las siguientes nuevas funcionalidades: múltiples cuentas pueden ser sincronizadas en un mismo móvil; combina la bandeja de entrada con el navegador para soportar múltiples cuentas de email en una sola página; Bluetooth 2.1; búsqueda entre todos los SMS y MMS; auto borrado de mensajes antiguos; la cámara soporta flash, zoom digital, modo escena, balance de blancos, y enfoque macro; mejora el teclado en cuanto a velocidad; los diccionarios son autodidactas e incluyen los contactos; navegador soporta HTML5; mejora de Google Calendar, pudiendo invitar a eventos; se optimiza la velocidad del hardware; mejor contraste de pantalla; la pantalla soporta ahora diferentes resoluciones; se actualiza a Google Maps 3.1.2; fondos de pantalla animados.

- *2.2 (Froyo)*: Basada en la versión *kernel* de Linux 2.6.32, esta versión fue lanzada el 20 de Mayo de 2010, incluyendo las siguientes nuevas funcionalidades: se optimiza memoria, velocidad y rendimiento; integración del motor JavaScript V8 de 'Google Chrome'; mejora del soporte a Microsoft Exchange; mejora de la ejecución de aplicaciones desde los accesos directos; USB *Tethering* y Wi-Fi *hotspot*; posibilidad de deshabilitar el acceso a datos a través de la red móvil; actualización de Android Market; atajos para el cambio de diccionarios del teclado; soporte de clave numéricas y alfanuméricas; posibilidad de instalar aplicaciones desde la tarjeta de memoria; Adobe Flash soportado; soporte para pantallas DPI y de hasta 4".

- *2.3 (Gingerbread)*: Basada en la versión *kernel* de Linux 2.6.35, esta versión fue lanzada el 6 de Diciembre de 2010, incluyendo las siguientes nuevas funcionalidades: videochat a través de Google Talk y Google Blog/video; mejora la interfaz de usuario para hacerla más rápida y sencilla; soporte para pantallas grandes (WXGA y superiores); llamadas a través de VoIP; teclado virtual más intuitivo y con mayor precisión; copia/pega disponible en todo el SO; nuevos efectos de audio como reverberación, ecualización…; mejora de la utilización de la batería; gestor de descargas para un fácil acceso a emails, datos del navegador y cualquier otro tipo; posibilidad de múltiples cámaras en el móvil, incluyendo una frontal para videollamada; soporta WebM/VP8 y el códec de audio AAC; soporte nativo a mas sensores (como giroscopios y barómetros); cambio del sistema de ficheros de YAFFS a ext4.

- *3.0 (Honeycomb)*: Basada en la versión *kernel* de Linux 2.6.36, esta versión fue lanzada el 22 de Febrero de 2011, incluyendo las siguientes nuevas funcionalidades: soporte para tablets optimizado con una nueva y "holográfica" interfaz de usuario; nuevo acceso rápido disponible en la parte superior de la pantalla para navegación, *widgets*, etc.; optimización de la multitarea, pudiendo saltar de una a otra fácilmente; rediseño del teclado virtual adaptado a tablets; copia/pega más intuitivo; modo "incognito" en el navegador; rápido acceso a los distintos modos de la camera; soporte para videochat a través de Google Talk; soporte para procesadores *multi-core*; visión de álbumes y videos en modo 'pantalla completa'; mejora del control de acceso a contactos; nueva interfaz de usuario para el manejo de emails.

- *3.x (Ice Cream Sandwich)*: Próxima versión del sistema operativo, que será lanzada en verano del año 2011.

Todas estas versiones están actualmente en el mercado, pero no todas se usan por igual. Según un estudio realizado a finales de Abril del año 2011, la cuota de mercado de cada versión se puede apreciar en la Fig. 4:

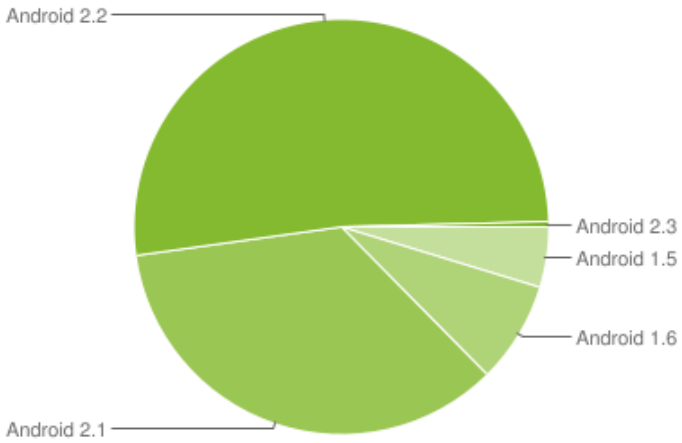



**Fig. 4. Cuota de versiones Android en el mercado.**

Analizando de forma rápida la Fig. 4, vemos que la versión más extendida es la 2.2. Esto es debido a que la mayoría de los móviles que están actualmente en el mercado son de gama media-alta, que son el tipo de *smartphone* que lleva como SO la versión 2.2. Las versiones anteriores están instaladas en muchos menos modelos de teléfono debido a la poca confianza de las compañías móviles a implantarlo, y las versiones más modernas del SO están sobre s*martphones* (y tablets) demasiado caros como para que extenderse en el mercado actual.

### 3.1.5 Comparativa con otros SO

Hasta ahora, se han demostrado las diferentes cualidades y características de este sistema operativo, pero… ¿Cómo es respecto al resto de sistemas operativos del mercado?

Actualmente existen en el mundo de los s*martphone* cuatro principales sistemas operativos: Android [29], Windows Phone 7 [30], Symbiam [31] e iOS 4.0 [32]. Se va a comparar diferentes características comunes en todos los SO con Android, y viendo porque se ha elegido este sistema operativo para el desarrollo de esta aplicación.

En cuanto a *requerimientos mínimos de hardware*, Apple y Microsoft garantizan que su SO corre de forma óptima en todos los s*martphone* donde se encuentran instalados, pero estos requerimientos son en ambos casos bastante elevados. En cambio, Symbian y Android han desarrollado SO capaces de ejecutarse en cualquier hardware, ya que poseen versiones con diferentes tipos de requerimientos (desde muy bajos a muy altos).

En cuanto a la *pantalla táctil*, Windows e iOS solo permiten pantallas de alta calidad capacitivas, mientras que Symbian y Android permiten tanto capacitivas como resistivas.

En cuanto al *sistema operativo*, de nuevo los SO de Microsoft y Apple son códigos cerrados, mientras que Nokia y Google permiten que su SO sea código abierto.

En cuanto a la *memoria externa*, todos los sistemas operativos permiten memorias externas, pero Windows Phone lo hace con una particularidad: la tarjeta de memoria que uses, será siempre la misma (se debe poner antes del primer arranque del móvil, y luego no se podrá quitar o cambiar nunca más). Esto es debido a que el SO hace uso de esta memoria como memoria interna, por lo que al iniciarse el s*martphone* por primera vez, pasa a pertenecer a la memoria global del teléfono.

En cuanto a si es *multitarea*, una vez más tanto Windows como iOS no son multitarea, cosa que Symbian y Android sí son.

En cuanto a *Tethering*, todos los sistemas operativos lo permiten menos Windows Phone.

La función *"cortar y pegar"*, tan básica en un sistema operativo, es algo complejo de desarrollar para un s*martphone*. Pese a ello, todos los SO la implementan de una forma u otra, salvo Windows Phone, que no la soporta (Microsoft ha asegurado que para la versión de finales de 2011, estará incluida).

Por último, en cuanto a *Adobe Flash*, solo Android y Symbian son capaces de soportar este *plug-in*. El resto no son capaces de soportarlo, aunque con aplicaciones de terceros son capaces de ejecutarlo.

Según vemos en estos datos, Android y Symbian son los sistemas operativos con mayor cantidad de características en el mercado actual. Entonces… ¿Qué nos ha hecho decantarnos por Android y no Symbian? Actualmente Nokia (compañía que desarrolla el SO Symbian) ha comenzado a instalar en sus nuevos s*martphone* el sistema operativo de Microsoft, lo que hace indicar que no confían mucho en el desarrollo de su SO.

Basándose en esto y en muchos más datos, la consultora Gartner [35] ha realizado en Abril de 2011 un estudio en el que predice cual será el número de SO vendidos en 2015. Los resultados son los siguientes:

**Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units)**

| OS | 2010 | 2011 | 2012 | 2015 |
|---|---|---|---|---|
| Symbian | 111,577 | 89,930 | 32,666 | 661 |
| Market Share (%) | 37.6 | 19.2 | 5.2 | 0.1 |
| Android | 67,225 | 179,873 | 310,088 | 539,318 |
| Market Share (%) | 22.7 | 38.5 | 49.2 | 48.8 |
| Research In Motion | 47,452 | 62,600 | 79,335 | 122,864 |
| Market Share (%) | 16.0 | 13.4 | 12.6 | 11.1 |
| iOS | 46,598 | 90,560 | 118,848 | 189,924 |
| Market Share (%) | 15.7 | 19.4 | 18.9 | 17.2 |
| Microsoft | 12,378 | 26,346 | 68,156 | 215,998 |
| Market Share (%) | 4.2 | 5.6 | 10.8 | 19.5 |
| Other Operating Systems | 11,417.4 | 18,392.3 | 21,383.7 | 36,133.9 |
| Market Share (%) | 3.8 | 3.9 | 3.4 | 3.3 |
| **Total Market** | **296,647** | **467,701** | **630,476** | **1,104,898** |

Source: Gartner (April 2011)

**Fig. 5. Previsión de SO en mercado realizado por la consultora Gartner.**

Como se aprecia en la Fig. 5, se estima que Android irá en una pendiente creciente durante los próximos años, hasta llegar a 2015 a dominar el mercado con una cuota de mercado del 48.8%. En cambio Symbian ira descendiendo de manera significativa hasta que en 2015 desaparecerá del mercado. Windows Phone crecerá debido a la entrada en teléfonos Nokia por Symbian, lo que contrasta con iOS, que seguirá con la misma cuota de mercado actual.

## *3.2 HERRAMIENTAS PARA UNA VIDA SALUDABLE*

En el mercado tecnológico actual existen una gran cantidad de herramientas (ya sea para iPhone, Android o totalmente independientes en un único aparato destinado para ello) con el fin de ayudar a controlar y mejorar diferentes apartados de la vida relacionados con la salud y el estilo de vida, ayudándose de las características propias de los dispositivos.

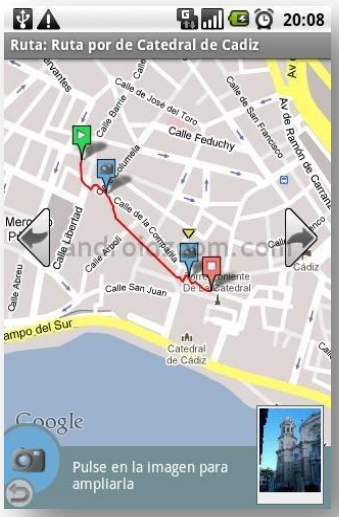
### 3.2.1 Aplicaciones para Android

En lo referente a Android, existen varias aplicaciones con características similares a la propuesta, pero sin llegar a combinar todas las que posee la aplicación a desarrollar en este proyecto. Algunas de ellas son:

- *CardioTrainer* [36]: Herramienta de pago exclusiva de Android que combina las ventajas y prestaciones de un entrenador personal en el móvil con el GPS que posee el propio s*martphone*. A través del receptor GPS (no incluye podómetro) registra los recorridos seguidos cuando practicamos deporte en Google Maps, con la velocidad que seguida, el tiempo tardado en realizarla, la morfología del terreno. Además, se puede elegir el deporte que se vaya a practicar (*running*, ciclismo, senderismo…) para realizar una estimación de las calorías consumidas en el recorrido realizado.


- *RunKeeper* [37]: Esta herramienta está disponible tanto para Android como para iOS (SO de Apple) y cuenta con dos versiones con diferentes precios. En su versión 'Normal' utiliza el GPS del aparato para registrar la actividad física realizada, incluyendo distancia recorrida, calorías quemadas, velocidad, etc. También permite realizar reportes a la web, la cual está asociada con *Facebook* de forma que puedes publicar tus progresos, recorridos, etc. En su versión 'Pro', además de lo anterior, posee varios incentivos extras como la posibilidad de descargar audios con rutinas de ejercicios, totalmente libres de publicidad comercial.

- *AndAndo* [38]: Aplicación española gratuita exclusiva de Android que gestiona las rutas que lleva a cabo el usuario. Mediante GPS registra las sucesivas posiciones del camino que se van siguiendo, aparte de recoger otra información como el tiempo o la velocidad empleada. Dispone de diferentes tipos de movilidad (andando, corriendo, en bicicleta, en coche…) y a lo largo de la ruta se pueden ir añadiendo diferentes marcas informativas como textos o fotografías. El sistema permite establecer la frecuencia de las marcas de posición, siendo posible que el s*martphone* suene o vibre cada vez que establece una marca. Como otra de las características principales de esta aplicación se encuentra la posibilidad de exportar las rutas en formato KML (formato usado por Google Earth), para compartir las rutas con gente que no posea la aplicación.




- *Moveo* [39]: Herramienta gratuita y exclusiva de Android basada en un podómetro común, dedicada a aquellas personas que practiquen senderismo, o que únicamente quieran saber la cantidad de pasos que han realizado en un espacio de tiempo determinado. Para ello utiliza el acelerómetro incluido en la mayoría de teléfonos, por lo que el cálculo es bastante exacto (se puede determinar la longitud de la zancada del usuario, para una mayor precisión). Además, es capaz de calcular las calorías consumidas en base al número de pasos y compararla con las cifras obtenidas en las últimas semanas o meses.


### 3.2.2 NIKE+

Nike+ [40] (también llamado Nikeplus) es la unión establecida en 2006 entre Nike y Apple para la creación de un sistema para deportistas que te permite llevar un registro de todos tus entrenamientos, distancias, calorías y velocidad.

El sistema consiste en dos partes: un chip, que cuenta con un acelerómetro y un transmisor que es introducido en la plantilla de la zapatilla de deporte (en un principio solo Nike vendía modelos de zapatilla con esta plantilla, pero actualmente las más famosas marcas de calzado poseen modelos con esta plantilla específica); y un receptor que se conecta a un iPod o a un iPhone. Nike, varios meses después de la puesta en marcha de este sistema, lanzo al mercado la venta del chip de forma individualizada, ya que no se le puede cambiar la batería al sensor, por lo que cada vez que la batería del sensor de agote, se debe comprar uno nuevo.

En el mercado, existen soluciones para aquellos usuarios que no quieran usar zapatillas específicas con este modelo de plantilla, como, por ejemplo, pequeñas bolsas o dispositivos que puede contener en su interior el *chip* y que son sujetos a los cordones de la zapatilla. No importa como el acelerómetro este integrado en la zapatilla del usuario, lo único que éste debe tener cuidado es de que el *chip* se encuentre firmemente sujeto y que no se produzcan sacudidas en el sensor, ya que podría hacer que el acelerómetro perdiera precisión.

El kit de deporte puede ser usado para realizar entrenamientos, que en este sistema son llamados "*workouts*". Un nuevo *workout* empieza cuando el receptor es unido al aparato de Apple utilizado (a partir de ahora se describirá el proceso con un iPod), haciendo que aparezca el nuevo menú de navegación del sistema en el aparato. El usuario elige el objetivo a conseguir con el entrenamiento, el cual puede ser una distancia específica, quemar un número de calorías o trabajar un tiempo determinado. Un *workout* también puede ser iniciado sin objetivo final, lo que es llamado "*Workout* Básico". Cuando el objetivo ya ha sido determinado, el receptor busca el chip emisor, posiblemente pidiendo al usuario andar hasta que lo detecte. Entonces, el usuario debe pulsar el botón central del iPod para iniciar el entrenamiento.

La aplicación posee un sistema de audio que anima al usuario a conseguir sus objetivos. El usuario puede elegir si prefiere una voz masculina o femenina, que también dependerá del entrenamiento elegido. Para entrenamientos con un objetivo impuesto, el audio responderá con diferentes mediciones restantes hasta la meta. Por ejemplo, en un entrenamiento de distancia, el sistema informará al usuario cada kilómetro que ha sido completado, así como la mitad del ejercicio, y te contará cada 100 metros de los últimos 400 metros del ejercicio a realizar.

Este kit también hace que el manejo del iPod cambie ligeramente durante el entrenamiento. El botón de "pausa" ahora no solo para la música, sino que también para el ejercicio. De igual manera, el botón de "menú" es usado para acceder a los controles para el finalizado del *workout*. Los botones "adelante" y "atrás" son cambiados, estableciendo las funciones de cambio de canción y repetición respectivamente. El botón central tiene ahora dos funciones: cuando es pulsado una vez, informa al usuario a través del sistema de audio acerca de la distancia o tiempo recorrido; mientras que si el botón se mantiene pulsado, el iPod salta a la "*PowerSong*" (canción poderosa), que es una canción elegida por el usuario, generalmente con la intención de motivarse.

Como un añadido al sistema de audio de este kit, existen varias felicitaciones pregrabadas por deportistas de elite como Lance Armstrong, Tiger Woods o Paula Radcliffe, que son lanzadas cuando consigues algún record personal (por ejemplo superar tu tiempo en 5km, 10km…) o si consigues alcanzar algunos objetivos muy complicados (como por ejemplo las 250 millas o recorrer 500 kilómetros). Estas "famosas felicitaciones" son oídas al final del menú estadístico del entrenamiento.

Mientras que el kit puede ser usado inmediatamente después de comprarlo, este reportará resultados con mayor precisión si es calibrado antes del primer uso y regularmente cada cierto tiempo después. Para calibrarlo, el usuario fijará una distancia que sea real de, al menos, 400 metros, y empezará el modo de calibración para andar o correr sobre esa distancia real. Cuando el ejercicio acabe, el dispositivo se calibrará y en los futuros reportes de los *workouts*, reflejará las estadísticas más al estilo de entrenamiento del usuario. En entrenamientos como andar o correr, la precisión del ritmo puede ser menor del 90%. Por ello, cada vez que se cambia de deporte, se debe calibrar el sensor.

Además de lo anterior, el sistema Nike+ se integra directamente con la página web de Nike. Todos los datos del entrenamiento son cargados automáticamente a la web durante la sincronización del iPod con iTunes o a través de otro programa similar. La mayoría de la información subida no identifica personalmente, pero también contiene algún tipo de información personal como el peso (todo esto es configurable). Los datos de cada *workout* son almacenados en archivos XML en el iPod, lo que permite a algunas webs y aplicaciones ofrecer reportes alternativos a los de la página oficial de Nike.

Nike, en su intento por aumentar la utilidad de este sistema de entrenamiento personal, ha realizado otros dos lanzamientos importantes:

- En Abril de 2008 Nike sacó al mercado el dispositivo "Nike+ Sportband" [42]. Este dispositivo de tan solo 23 gramos permite al usuario almacenar la información del entrenamiento sin necesidad de ningún aparato de Apple. Este dispositivo consta de dos partes: un reloj-muñequera y un receptor al que se ensambla el disco USB. El receptor muestra la información comparable al kit unido al dispositivo Apple. Después del entrenamiento, el receptor puede introducirse en el ordenador a través del puerto USB y mandar la información automáticamente a la web de Nike.

- En Junio de 2010, Polar y Nike lanzan al mercado una nueva correa con pulsímetro llamado "Polar WearLink+" [43], que trabaja con todo lo existente en el mercado relacionado con el sistema Nike+, además de usar un protocolo digital "dual-mode", por lo que también es compatible con la mayoría de ordenadores de entrenamiento de la compañía Polar.

## *3.3 GASTO CALÓRICO: RMR Y MET*

El ser humano, como cualquier otra entidad perteneciente a la Tierra, necesita de la energía para realizar cualquier actividad, ya sea andar, correr, o simplemente las actividades vitales como bombear sangre del corazón a todo el cuerpo (metabolismo basal).

El combustible que el ser humano necesita para obtener la energía proviene de los nutrientes aportados por los alimentos que ingieren. Esta energía obtenida, se calcula en forma de caloría, y cuya definición es "la cantidad de energía calorífica necesaria para elevar un grado Celsius la temperatura de un gramo de agua pura, desde 14,5 °C a 15,5 °C, a una presión normal de una atmósfera."

Para el cálculo estimado del gasto de calorías que el usuario gasta durante una actividad, se va a utilizar un método basado en la tasa metabólica en reposo del usuario y la tasa de actividad metabólica de cada ejercicio.

### 3.3.1 Tasa Metabólica en Reposo (RMR)

La tasa metabólica en reposo (RMR) [44] es el número de calorías que utiliza el cuerpo para su funcionamiento cuando está en reposo y representa la mayor cantidad de consumo de calorías de una persona. La tasa metabólica basal de un individuo depende de la respiración, digestión, ritmo cardíaco y función cerebral. La edad, el sexo, el peso y el tipo de actividad física también afectan la tasa metabólica basal, que aumenta según la cantidad de tejido muscular del individuo y se reduce con la edad.

Es complicado obtener una medida exacta de este valor, ya que son necesarias unas condiciones específicas, como son:

- Una habitación oscura.
- Haber dormido, al menos, 8 horas.
- 12 horas desde la última comida realizada.
- El sujeto debe encontrarse tumbado y relajado.
- Aparatos clínicos preparados para realizar el cálculo.

Como la mayoría de los usuarios no podían cumplir todas las condiciones comentadas anteriormente, en 1919 dos médicos llamados J.A Harris y F.G. Benedict publicaron un estudio biométrico del metabolismo basal del ser humano [45], sacando a la luz la formula Harris-Benedict, que les llevaría al éxito dentro del mundo especializado.

Esta fórmula predice el RMR a partir del sexo, edad, altura y peso del individuo. Las ecuaciones (dependiendo del sexo) son las mostradas en la Fig. 6:

*RMR para hombres (Kcal/día) = 66.5 + (13.75*peso) + (5.003*altura) - (6.775*edad)*

*RMR para mujeres (Kcal/día) = 665.1 + (9.563*peso) + (1.850*altura) - (4.676*edad)*

**Fig. 6. Fórmulas de Harris-Benedict para la estimación del RMR**

De acuerdo a estos cálculos, para un hombre adulto con características normales (30 años, 182cm de altura y 80kg de peso), su RMR será de 1873 Kcal/día; para una mujer adulta con características normales (30 años, 170cm de altura y 62kg de peso), su RMR será de 1432 Kcal/día. Estos valores obtenidos serían las calorías que un hombre o mujer debería obtener durante el día para mantenerse en su peso, si no realiza ningún tipo de actividad.

### 3.3.2 Tasa Metabólica Equivalente (MET)

La tasa metabólica equivalente [46] es un concepto ideado para expresar el coste energético de cualquier actividad física que realice el ser humano. Por equivalencias, 1 MET es igual a 3.5 ml $O_2 \cdot kg^{-1} \cdot min^{-1}$, o 1 $kcal \cdot kg^{-1} \cdot h^{-1}$, o 4.184 $kJ \cdot kg^{-1} \cdot h^{-1}$.

| Actividad Física | MET |
|---|---|
| Dormir | *0.9* |
| Ver la televisión | *1.0* |
| Leer, hablar por teléfono | *1.3* |
| Escribir, diseñar, pensar | *1.8* |
| Andar a menos de 3.2 km/h, en llano | *2.0* |
| Bajar escaleras | *2.5* |
| Bicicleta estática sin esfuerzo | *3.0* |
| Andar a menos de 4.0 km/h | *3.0* |
| Ejercicio en casa a nivel bajo o moderado | *3.5* |
| Bicicleta a menos de 16 km/h | *4.0* |
| Andar enérgicamente | *4.0* |
| Nadar despacio | *4.5* |
| Andar lo más rápido posible | *5.0* |
| Jogging despacio | *6.0* |
| Jogging a ritmo normal | *7.0* |
| Correr a menos de 10 km/h | *10.0* |
| Correr a menos de 13 km/h | *13.5* |
| Correr a menos de 16 km/h | *16* |
| Correr a alrededor de 23.5 km/h | *23* |

**Tabla 2. MET más frecuentes**

En cuanto a equivalencias con la actividad propiamente dicha, 1 MET es considerada la tasa metabólica obtenida cuando el ser humano se encuentra en reposo sentado. Los valores de las actividades físicas que el ser humano puede desarrollar se encuentran entre un rango de 0.9 (dormir) hasta 23 (correr a una velocidad de 22.5 km/h). Las actividades más comunes junto con sus valores MET, son expuestos en la Tabla 2 (para ver todas las actividades existentes y sus valores MET, consultar la página web oficial [47]).

La tabla que indica el número de MET por actividad física se creó en 1989, y poco a poco los valores han ido ajustándose ya que al principio se realizó la tabla con valores MET expresados en ml $O_2 \cdot kg^{-1} \cdot min^{-1}$, pero esta forma de cálculo no era del todo correcta. En diferentes estudios realizados durante los últimos años por N.M. Byrne [48] y S. Kozey [49], se descubrió que el MET tenía una subestimación del gasto energético de alrededor de un 20%, debido a que el cálculo se estaba realizando con las medidas anteriores que se basaban en el consumo de oxigeno del ser humano, cosa que varía en cada uno de los individuos.

Para solucionar esto, se utiliza el MET Corregido, que se basa en el peso, altura, edad y sexo de cada persona gracias a la fórmula del RMR obtenida por Harris-Benedict [50], ajustando el MET y haciendo que la estimación del gasto de cada actividad sea mucho más preciso. La fórmula se muestra en la Fig. 7:

$$\text{Corrected MET value} = \text{MET value (from Compendium code)} \times \frac{3.5\ ml^{-1} \cdot kg^{-1} \cdot min^{-1}}{\text{Harris-Benedict RMR } (ml^{-1} \cdot kg^{-1} \cdot min^{-1})}$$

**Fig. 7. Fórmula del MET Corregido.**

### 3.3.3 Calculo calórico

Una vez que se ha calculado el RMR del individuo y se ha ajustado el MET a partir de la fórmula del MET corregido, es posible conocer en un tiempo definido cuál es el gasto calórico que el usuario realiza en una actividad determinada con la fórmula de la Fig. 8:

$$\textit{Gasto calórico} = \textit{MET corregido } (kcal \cdot kg^{-1} \cdot h^{-1}) * \textit{Tiempo transcurrido (horas)} * \textit{Peso (kg)}$$

**Fig. 8. Fórmula final del Gasto Calórico**

Para un mejor entendimiento, vamos a realizar un ejemplo que demuestre el proceso de cálculo.

El supuesto usuario poseerá las siguientes características: hombre, 24 años, 187cm de altura, 93kg de peso. La actividad que realizará será andar durante 1 hora a una velocidad de entre 3.2 km/h y 4.0 km/h, por lo que el MET será de 3.0.

Lo primero que realizaremos será calcular el RMR del usuario en base a la fórmula de Harris-Benedict:

1. RMR = 66.5 + (13.75*93) + (5.003*187) - (6.775*24) = 2118.211 Kcal/día

Una vez hemos obtenido el RMR en kilocalorías por día, debemos transformarlo a kilocalorías por kilo y hora:

2. RMR ($kcal \cdot kg^{-1} \cdot h^{-1}$) = 2118.211 Kcal/día / (24*93) = 0.949 $kcal \cdot kg^{-1} \cdot h^{-1}$

Después de haber realizado esa transformación, debemos calcular el MET corregido, usando el valor anterior del RMR (recordar que en la fórmula original utilizan las unidades $kcal \cdot kg^{-1} \cdot h^{-1}$, pero aquí se utilizan las unidades $kcal \cdot kg^{-1} \cdot h^{-1}$, por lo que en vez de dividir entre 3.5, lo hacemos entre 1).

3. MET corregido = 3.0 * (1/0.949 $kcal \cdot kg^{-1} \cdot h^{-1}$) = 3.162 $kcal \cdot kg^{-1} \cdot h^{-1}$

Por último, una vez tenemos el MET corregido, solo debemos multiplicar por el número de horas y el peso del usuario, para obtener el gasto calórico del ejercicio realizado.

4. Gasto calórico = 3.162 $kcal \cdot kg^{-1} \cdot h^{-1}$ * 1 * 93 = 294.066 Kcal.

Como vemos, este usuario andando una hora a un ritmo de entre 3.2 km/h y 4 km/h, realizará un gasto calórico estimado de 294.066 kilocalorías. Esta estimación será la usada en la aplicación para el cálculo del gasto calórico.

## *3.4 GPS*

GPS son las siglas que identifican el sistema de posicionamiento global. El GPS consiste en el sistema de posicionamiento global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto con una precisión de varios metros de distancia. Está formado por una red de satélites que orbitan la Tierra en puntos fijos por encima del planeta y transmiten señales a cualquier receptor GPS en la Tierra. Estas señales llevan un código de tiempo y un punto de datos geográficos que permite al usuario identificar su posición exacta en cualquier parte del planeta.

### 3.4.1 Historia

En 1957, la antigua URSS lanzó el primer satélite realizado por el hombre: el Sputnik 1. Los científicos se dieron rápidamente cuenta nada más poner en órbita el satélite de dos cosas:

- Era posible calcular la órbita de un satélite sirviéndose del Doppler Effect [52].

- Si se le daba un uso totalmente contrario a este efecto, los satélites podían servir para calcular la posición de un receptor en tierra.

Tras esto, el ejército estadounidense estableció, a principios de los años 60, las bases del GPS moderno. La marina, la fuerza aérea y el ejército concibieron sus propios diseños e ideas para, en 1973, unir los resultados obtenidos durante años aprobando un diseño que incorporaba elementos de cada una de las propuestas. En ese momento nacía el NAVSTAR [53].

El primer satélite para el nuevo GPS del NAVSTAR se lanzó en el año 1974. Más tarde, entre los años 1978 y 1985, se lanzaron otros once satélites de prueba. Finalmente, en el año 1993, se lanzó el último satélite de prueba que completó la constelación de 24 satélites que componen NAVSTAR, que hoy en día permite gracias a su sistema de navegación disfrutar de cobertura GPS en todo el mundo.

Inicialmente, el GPS se creó para darle únicamente un uso militar. Pero entonces sucedió una tragedia [54]: el 1 de septiembre de 1983, el vuelo KAL007 de Korean Airlines de Anchorage a Seúl se salió de la ruta prevista, penetrando por dos veces en el espacio aéreo de la URSS, tras lo cual fue abatido por un avión de combate soviético Su-15, muriendo los 269 pasajeros y la tripulación. Dos semanas más tarde, el presidente estadounidense Ronald Reagan propuso que el GPS se pusiera a disposición de los civiles para evitar que errores de navegación como este provocasen catástrofes de tal calibre.

Tras gastar unos 12.000 millones de dólares para desarrollar el sistema de navegación más preciso del mundo, el gobierno de Estados Unidos introdujo en el NAVSTAR una opción llamada disponibilidad selectiva (DS), la cual limitaba la precisión ofrecida a los usuarios civiles para asegurarse de que ningún enemigo ni grupo terrorista pudiese utilizar el GPS para fabricar armas de precisión. Funcionaba introduciendo errores deliberados en los datos que transmitía a cada uno de los satélites. Los militares podían acceder al sistema exacto descifrando una segunda frecuencia protegida que se emitía al mismo tiempo.

Más tarde, durante la Guerra del Golfo (la guerra que popularizo el uso de este sistema de posicionamiento), el ejército estadounidense se dio cuenta de que necesitaba muchos más receptores GPS que los que tenía. La única solución que encontró al problema fue utilizar receptores GPS civiles, pero estos receptores contaban con la opción DS activada, por lo que para aumentar la precisión esos dispositivos se tuvo que deshabilitar temporalmente esta opción.

Finalmente, en el año 2000, el presidente estadounidense Bill Clinton anunció que la opción DS se deshabilitaría completamente debido a que las "evaluaciones de amenazas" llevadas a cabo por el gobierno estadounidense indicaban que eliminar la DS no afectaría prácticamente a la seguridad nacional. Sin embargo, en el mismo discurso, Clinton indicó que Estados Unidos podría "denegar selectivamente" las señales GPS en determinadas zonas cuando la seguridad nacional se viera amenazada.

### 3.4.2 Funcionamiento

El sistema GPS realiza su función en cinco pasos lógicos:

1. *Triangulación*

Nuestra posición se calcula en base a la medición de las distancias a los satélites. Matemáticamente se necesitan cuatro mediciones de distancia a los satélites para determinar la posición exacta pero en la práctica se resuelve la posición de un objeto con solo tres mediciones. La cuarta medición únicamente se utiliza por razones técnicas que se explican más adelante.

2. *Medición de distancia*

La distancia al satélite se determina midiendo el tiempo que tarda una señal de radio, emitida por el mismo, en alcanzar el receptor de GPS. Para efectuar dicha medición se asume que ambos, el receptor GPS y el satélite, están generando el mismo "Código Pseudo Aleatorio" [55] en exactamente el mismo momento. Comparando cuanto retardo existe entre la llegada del código aleatorio proveniente del satélite y la generación del código del receptor de GPS, se puede determinar cuánto tiempo le llevó a dicha señal llegar hasta el receptor. Finalmente, multiplicando dicho tiempo de viaje por la velocidad de la luz obtenemos la distancia al satélite.

3. *Timing perfecto*

Un timing muy preciso es clave para medir la distancia a los satélites. Los satélites son exactos porque llevan un reloj atómico a bordo; en cambio, los relojes de los receptores GPS no son exactos. Esto es debido a que los receptores no necesitan ser tan exactos porque la medición de un rango a un satélite adicional permite corregir los errores de medición. Esta medición a un satélite adicional es la cuarta medición comentada en la Triangulación.

4. *Posicionamiento*

Para utilizar los satélites como puntos de referencia se debe previamente conocer exactamente donde están en cada momento. Los satélites de GPS están ubicados a tal altura que sus órbitas son muy predecibles. A pesar de ello, el departamento de defensa de los Estados Unidos controla y mide variaciones menores en sus órbitas, enviando esta información sobre errores a los satélites para que estos, a su vez, retransmitan su posición corregida junto con sus señales de timing a los receptores GPS.

*5. Corrección*

La capas de la atmosfera existentes entre el satélite y el receptor GPS (ionosfera y troposfera) causan demoras en la señal de GPS, las cuales se traducen en errores de posicionamiento. Mediante modelación y fórmulas matemáticas estas correcciones son realizadas, obteniendo la ubicación del receptor en la Tierra.

### 3.4.3 Uso en Smartphones

Quien piense que los receptores GPS incluidos en los *smartphones* solo sirven para disponer de un simple navegador como TomTom [56] en el teléfono o hacer uso de utilidades como Google Maps, se equivoca estrepitosamente. Aparte de esta importante funcionalidad, actualmente existe otro concepto que usa las posibilidades de un receptor GPS y el *smartphone*, la geolocalización. Con ello, se puede geolocalizar todo: fotos, eventos, sitios de interés, etc.

En torno a este nuevo concepto se han creado aplicaciones que toman automáticamente la posición del usuario y la actualizan en sus sitios de la red como Twitter o Facebook. Los fines son comunes: compartir con los amigos sus movimientos y dar a conocer sitios interesantes a su alrededor.

Aparte de la geolocalización, hace poco han empezado a surgir nuevas aplicaciones que utilizan el GPS de forma muy parecida a los navegadores: calculan la distancia que recorre el objeto, pero en este caso es un deportista al practicar cualquier actividad, ya sea running, ciclismo, etc. En este proyecto vamos a darle al GPS un uso más dirigido a este último descrito.

## *3.5 HERRAMIENTAS*

En este apartado se da una breve introducción a las herramientas utilizadas para la realización de este Proyecto Fin de Carrera, además de resumir las características más importantes de cada una de ellas.

### 3.5.1 Android SDK

Android SDK es el kit de desarrollo necesario para crear aplicaciones para el sistema operativo Android. Este kit de desarrollo, que se puede descargar de manera gratuita desde la página oficial de Android, contiene los siguientes componentes:

- *Las herramientas SDK*: Es el paquete completo de herramientas para el SDK enfocadas al desarrollo, depuración y pruebas de las aplicaciones desarrolladas.

- *Los complementos SDK:* Proveen de un entorno de desarrollo para una librería externa especifica o personalizada de Android (pero totalmente compatible). Además, ofrece el complemento de todas las API de Google Maps, las cuales permiten que las aplicaciones accedan a una gran cantidad de posibilidades de esta librería.

- *Drivers USB para Windows:* Este kit contiene drivers para Windows que permiten ejecutar y depurar las aplicaciones desde tu terminal actual conectado por USB a la estación de desarrollo.

- *Ejemplos:* Otorga código de ejemplo y aplicaciones disponibles para cada plataforma de desarrollo Android. Estos ejemplos están pensados, sobre todo, para los desarrolladores que se inician en código para Android.

- *Documentación:* Posee una copia local de la última versión de la API existente para Android.

Para aquellos desarrolladores que buscan una experiencia de desarrollo sencilla, en este kit de desarrollo incluye, además, un *plug-in* para Eclipse que integra todas las características del SDK dentro del entorno de desarrollo de Eclipse. Por último, este kit cuenta con un sitio web con un blog [57] en el que se han incluido grupos de discusión con la idea de crear una comunidad de desarrollo de herramientas para Android, de modo que sea más fácil contribuir al desarrollo de la plataforma y compartir los conocimientos que cada uno vaya obteniendo.

Para comenzar con la construcción de aplicaciones para Android, los desarrolladores necesitan descargarse de la web el SDK de Android [58], alguno de los ejecutables disponibles para computadores x86. Entre los sistemas operativos disponibles se encuentran: Windows XP, Vista, 7 o superior; Mac OS 10.4.8 o superior; Linux Ubuntu Dapper Drake o superior (otras distribuciones de Linux también podrían ser aceptadas, pero no existe un soporte directo). Los desarrolladores que quieran, además, utilizar el *plug-in* de Eclipse, también necesitarán Eclipse en su versión 3.2 o superior con 'Java Development Kit' o Java con "Javac" en su versión 1.5 o 1.6, Apache Ant (un entorno de desarrollo integrado) y Python 2.2 o superior.

### 3.5.2 Eclipse

"*Una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular.*"

Proyecto Eclipse

Eclipse es un entorno integrado de desarrollo (IDE) multiplataforma para crear aplicaciones clientes de cualquier tipo. Es libre y fue creado originalmente por IBM, aunque ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que apoya una comunidad de código abierto.

La aplicación más importante que ha sido realizada con este entorno es el afamado entorno de desarrollo integrado Java, llamado *Java Development Toolkit* (JDT). Este es el entorno de desarrollo que se está convirtiendo en el estándar de facto para Java, de hecho otros IDE's comerciales como *JBuilder* han anunciado que su próxima versión se basará en Eclipse.

Eclipse no es tan sólo un IDE, sino que se trata de un *framework* ampliable mediante módulos (en inglés *plug-in*). Estos módulos o *plug-ins* proporcionan más funcionalidades, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Incluso hay *plug-ins* que permiten que el entorno de desarrollo soporte otros lenguajes además de Java, como por ejemplo PHP, Perl, etc.

Los componentes gráficos de Eclipse están basados en un juego de herramientas de tercera generación para Java de IBM llamado *SWT,* que mejora los de primera y segunda generación de Sun. La interfaz de usuario de Eclipse cuenta con una capa intermedia de interfaz gráfica (GUI) llamada *JFace*, lo que simplifica la creación de aplicaciones basadas en *SWT*.

### 3.5.3 Weka

El aprendizaje automático es una rama de la inteligencia artificial que se está utilizando mucho actualmente. Esta técnica permite a las computadoras aprender a partir de ejemplos con información no estructurada, dando como resultado fórmulas y algoritmos que permiten predecir nuevos resultados.

Weka [59] es el acrónimo de *Waikato Enviroment for Knowledge Analysis (*Entorno para Análisis del Conocimiento de la Universidad de Waikato). Weka es un paquete que implementa numerosos algoritmos de aprendizaje automático para tareas de minería de datos. Ha sido desarrollado en la Universidad de Waikato (Nueva Zelanda) usando el lenguaje Java, y pertenece al grupo de software libre distribuido bajo la licencia GNU-GPL.

Este paquete contiene diversas herramientas útiles para la minería de datos, como es el preprocesado, la clasificación, clustering, regresión, agrupamiento, selección y visualización de datos. Es también una herramienta adecuada para el desarrollo de nuevos esquemas de aprendizaje.

Sus técnicas se fundamentan en unos datos que se encuentran en un fichero plano o una relación, formados por registros, los cuales están a su vez formados por un número fijo de atributos numéricos o nominales. Trata con estos datos aplicándoles las herramientas descritas antes, y mostrando un resultado al usuario. Aparte de obtener los datos de un fichero plano, también es capaz de trabajar con bases de datos SQL a través del JDBC, devolviendo el resultado en una consulta.

Para proporcionar mayor flexibilidad, este paquete dispone de una API que permite a desarrolladores implementar nuevos algoritmos, así como modificar los algoritmos de aprendizaje que incluye, ya que esta implementado en lenguaje Java. Además, esto le añade la ventaja de ser un paquete totalmente portable, siendo capaz de ejecutarse en casi todas las máquinas.

Otro dato positivo a destacar de este paquete es su interfaz gráfica, cosa que hace que su manejo sea sencillo, incluso para usuarios no entrados en la materia. Además, posee interfaces distintas dependiendo del uso que se le quiera dar: tratamiento de datos con un algoritmo, comparativa de algoritmos o únicamente utilizar un aprendizaje incremental.

Como punto negativo al paquete de herramientas Weka, decir que éste no cubre el modelado de secuencias.

# 4. OBJETIVOS

En este capítulo se presenta el objetivo principal y los objetivos específicos de este Proyecto Fin de Carrera, enmarcado en el área de las herramientas saludables para Android.

El objetivo principal del proyecto es el desarrollo de una herramienta para el sistema operativo Android que calcule el número de pasos dados mediante diversos dispositivos y técnicas de aprendizaje automático, además de controlar el gasto calórico que se produce al desarrollar el usuario la actividad física de andar o caminar a una velocidad determinada.

Además del objetivo principal indicado, existen varios objetivos específicos:

- Estudio de las diferentes aplicaciones creadas para Android, relacionadas con el cálculo de la actividad realizada por el ser humano.

- Conocimiento en profundidad de todo lo relacionado con el consumo calórico al realizar actividades físicas.

- Aprendizaje del desarrollo de una herramienta para la plataforma Android en cualquiera de sus versiones.

- Análisis de las capacidades del dispositivo en el que se instalará la aplicación, como podría ser el GPS, el acelerómetro, la pantalla táctil, etc.

- Análisis y diseño de un método válido para la integración de datos de una actividad, obtenidos a través del podómetro y del GPS.

- Aprendizaje del uso de una herramienta de aprendizaje automático, gracias a la cual se desarrollan gran cantidad de nuevas fórmulas y algoritmos que tratan una gran cantidad de datos, facilitando la labor al ser humano.

- Análisis de los métodos existentes en el mundo tecnológico actual para el cálculo del gasto calórico en base a datos recogidos de diferentes aparatos (por ejemplo, acelerómetro o GPS).

- Comprobación del correcto funcionamiento de la aplicación desarrollada y exportación a un *smartphone* real.

Para llevar a cabo todos estos objetivos, este sistema se basará en tres pilares básicos, que han sido explicados más en profundidad en el apartado "ESTADO DEL ARTE", pero que son recordados brevemente aquí:

- *Podómetro*: Aparato concebido para calcular el número de pasos dados. Adicionalmente, en base a estos pasos se puede calcular la distancia y la velocidad con la que se avanza.

- *GPS*: Sistema global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto con una precisión de varios metros de distancia.

- *Calculo calórico*: Gracias a la gran cantidad de estudios realizados en los últimos 60 años, se han obtenido eficientes formulas estimadoras del gasto calórico que realiza un ser humano al practicar cualquier actividad durante un tiempo definido.

- *Aprendizaje automático*: Esta técnica de aprendizaje basada en la inducción, consiste en crear programas capaces de generalizar comportamientos a partir de una información no estructurada, suministrada a partir de una lista de ejemplos.

# 5. DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN

En este apartado se encuentra relatado todo lo referido al desarrollo real de la aplicación, desde los requerimientos impuestos por el usuario hasta la implementación final de la herramienta.

Al ser este un modelo en cascada, cada subapartado detallado en esta sección es origen del siguiente y continuación del anterior, siguiendo una lógica descendente. Esto quiere decir, que a pesar de ser el modelo retroalimentado, en esta memoria no se apreciará tal característica, a pesar de que sí se ha utilizado durante el desarrollo real de la herramienta.

## *5.1 REQUISITOS DE LA APLICACIÓN*

Los requisitos constituyen el punto de inicio de una aplicación, ya que son capaces de describir todo lo que el sistema será capaz de hacer (y no hacer incluso), una vez terminado.

Cada uno de los requisitos (ya sean de usuario o de software) están recogidos en tablas con los mismos campos (tal y como se definió en el apartado 2.2), con los que el requisito con tiene la información necesaria para su correcta comprensión, sin dar lugar a otros posibles alcances o soluciones.

Para cada tipo de requisito se ha usado una nomenclatura específica, que es la definida a continuación:

- **RU-Cxx:** Requisito de Usuario del tipo Capacidad.
- **RU-Rxx:** Requisito de Usuario del tipo Restricción.
- **RS-Fxx:** Requisito de Software del tipo Funcional.
- **RS-Ixx:** Requisito de Software del tipo Interacción.
- **RS-RExx:** Requisito de Software del tipo Requerimiento.
- **RS-RNxx:** Requisito de Software del tipo Rendimiento.

### 5.1.1 Requisitos de Usuario

Los requisitos de usuario definen aquellas características que el cliente demanda a la aplicación, y son los siguientes:

| **ID: RU-C01** | | | |
|---|---|---|---|
| **Nombre:** | Idea genérica | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá contabilizar el número de pasos, distancia, velocidad media y gasto calórico al realizar el usuario la actividad de andar o caminar en un tiempo determinado. | | |

**Tabla 3. RU-C01 / Idea genérica**

| **ID: RU-C02** | | | |
|---|---|---|---|
| **Nombre:** | Perfil de usuario | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá soportar un perfil de usuario para el cálculo del gasto calórico, en el que quede registrado el peso, la altura, la edad y tasa metabólica en reposo (RMR). | | |

**Tabla 4. RU-C02 / Perfil de usuario**

| **ID: RU-C03** | | | |
|---|---|---|---|
| **Nombre:** | Calculo del recorrido | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá ser capaz de calcular la distancia realizada por el usuario haciendo uso del GPS del *smartphone*. | | |

**Tabla 5. RU-C03 / Calculo del recorrido**

| **ID: RU-C04** | | | |
|---|---|---|---|
| **Nombre:** | Almacenamiento | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema tendrá que almacenar la información relevante registrada por la aplicación en una base de datos eficiente. | | |

**Tabla 6. RU-C04 / Almacenamiento**

| **ID: RU-C05** | | | |
|---|---|---|---|
| **Nombre:** | Ayuda al usuario | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá poseer un documento con las preguntas más frecuentes por el usuario con sus respuestas, así como un completo manual de usuario. | | |

**Tabla 7. RU-C05 / Ayuda al usuario**

| **ID: RU-C06** | | | |
|---|---|---|---|
| **Nombre:** | Interfaces | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá poseer una serie de interfaces navegables entre sí, donde se exploten todas las cualidades de la aplicación. | | |

**Tabla 8. RU-C06 / Interfaces**

| **ID: RU-C07** | | | |
|---|---|---|---|
| **Nombre:** | Multiusuario | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá ser multiusuario, es decir, que varios usuarios podrán ejecutar la aplicación en un mismo *smartphone*. | | |

**Tabla 9. RU-C07 / Multiusuario**

| **ID: RU-R01** | | | |
|---|---|---|---|
| **Nombre:** | Funcionamiento | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá ser lo suficientemente rápido y efectivo como para poder seguir en tiempo real el número de pasos, distancia, velocidad, gasto calórico y distancia realizada. | | |

**Tabla 10. RU-R01 / Funcionamiento**

| **ID: RU-R02** | | | |
|---|---|---|---|
| **Nombre:** | Versión Android | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema deberá indicar cuál será la versión mínima del sistema operativo que el usuario deberá tener instalada en su *smartphone* para poder utilizar esta aplicación. | | |

**Tabla 11. RU-R02 / Versión Android**

| **ID: RU-R03** | | | |
|---|---|---|---|
| **Nombre:** | Idioma de la aplicación | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | Usuario |
| **Descripción:** | El sistema tendrá que estar disponible en, al menos, castellano e inglés. Se debe poder extender a cualquier otro idioma en un futuro. | | |

**Tabla 12. RU-R03 / Idioma de la aplicación**

### 5.1.2 Requisitos de Software

Los requisitos de software definen cual es el comportamiento del sistema en base a lo extraído de los requisitos de usuario redactados anteriormente, y son los siguientes:

| **ID: RS-F01** | | | |
|---|---|---|---|
| **Nombre:** | Podómetro | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema contará con un podómetro, capaz de contar la cantidad de pasos dados al andar el usuario. | | |

**Tabla 13. RS-F01 / Podómetro**

| **ID: RS-F02** | | | |
|---|---|---|---|
| **Nombre:** | Cálculo del gasto calórico | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema deberá ser capaz de calcular el gasto calórico que ha realizado el usuario en base a los datos recogidos por el acelerómetro y/o el GPS. | | |

**Tabla 14. RS-F02 / Cálculo del gasto calórico**

| **ID: RS-F03** | | | |
|---|---|---|---|
| **Nombre:** | Seis diferentes velocidades | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema deberá reconocer seis diferentes velocidades a la hora de caminar, en base a los datos recibidos por el acelerómetro. | | |

**Tabla 15. RS-F03 / Seis diferentes velocidades**

| **ID: RS-F04** | | | |
|---|---|---|---|
| **Nombre:** | Cronómetro | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema poseerá un cronometro que controlará el tiempo de realización de la actividad física. | | |

**Tabla 16. RS-F04 / Cronómetro**

| **ID: RS-F05** | | | |
|---|---|---|---|
| **Nombre:** | Fecha | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01, RU-C04 |
| **Descripción:** | El sistema deberá conocer el día en el que se está ejecutando la aplicación (a través de la fecha interna del *smartphone*), para poseer un historial en base a la fecha en la que se realizó. | | |

**Tabla 17. RS-F05 / Fecha**

| **ID: RS-F06** | | | |
|---|---|---|---|
| **Nombre:** | Ejecución en segundo plano | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema será capaz de continuar ejecutándose mientras el usuario realiza cualquier otra actividad en el *smartphone*, como llamar o reproducir música. | | |

**Tabla 18. RS-F06 / Ejecución en segundo plano**

| **ID: RS-F07** | | | |
|---|---|---|---|
| **Nombre:** | Perfil de usuario (nombre) | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C02, RU-C04 |
| **Descripción:** | El sistema deberá recoger, dentro de un perfil, el nombre del usuario para identificar a un usuario de otro. | | |

**Tabla 19. RS-F07 / Perfil de usuario (nombre)**

| **ID: RS-F08** | | | |
|---|---|---|---|
| **Nombre:** | Perfil de usuario (peso) | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C02, RU-C04 |
| **Descripción:** | El sistema deberá recoger, dentro de un perfil, el peso del usuario para cálculos posteriores. | | |

**Tabla 20. RS-F08 / Perfil de usuario (peso)**

| **ID: RS-F09** | | | |
|---|---|---|---|
| **Nombre:** | Perfil de usuario (altura) | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C02, RU-C04 |
| **Descripción:** | El sistema deberá recoger, dentro de un perfil, la altura del usuario para cálculos posteriores. | | |

**Tabla 21. RS-F09 / Perfil de usuario (altura)**

| **ID: RS-F10** | | | |
|---|---|---|---|
| **Nombre:** | Perfil de usuario (edad) | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C02, RU-C04 |
| **Descripción:** | El sistema deberá recoger, dentro de un perfil, la edad del usuario para cálculos posteriores. | | |

**Tabla 22. RS-F10 / Perfil de usuario (edad)**

| **ID: RS-F11** | | | |
|---|---|---|---|
| **Nombre:** | Perfil de usuario (RMR) | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C02, RU-C04 |
| **Descripción:** | El sistema deberá calcular, dentro de un perfil, la tasa metabólica en reposo (RMR) en base a otros valores del usuario para cálculos posteriores. | | |

**Tabla 23. RS-F11 / Perfil de usuario (RMR)**

| **ID: RS-F12** | | | |
|---|---|---|---|
| **Nombre:** | Crear perfil | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04, RU-C07 |
| **Descripción:** | El sistema será capaz de crear un perfil de usuario, introduciendo su nombre, edad, peso, altura y sexo. | | |

**Tabla 24. RS-F12 / Crear perfil**

| **ID: RS-F13** | | | |
|---|---|---|---|
| **Nombre:** | Cargar perfil | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04, RU-C07 |
| **Descripción:** | El sistema será capaz de cargar un perfil de usuario creado previamente, utilizando el nombre de usuario. | | |

**Tabla 25. RS-F13 / Cargar perfil**

| **ID: RS-F14** | | | |
|---|---|---|---|
| **Nombre:** | Actualizar perfil | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04, RU-C07 |
| **Descripción:** | El sistema será capaz de actualizar un perfil de usuario creado previamente, utilizando el nombre de usuario. | | |

**Tabla 26. RS-F14 / Actualizar perfil**

| **ID: RS-F15** | | | |
|---|---|---|---|
| **Nombre:** | Longitud de paso | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema utilizará la longitud de paso introducida por el usuario (no estará asociada al usuario) para calcular la distancia recorrida en base al número de pasos. | | |

**Tabla 27. RS-F15 / Longitud de paso**

| **ID: RS-F16** | | | |
|---|---|---|---|
| **Nombre:** | Google Maps | | |
| **Necesidad:** | [ ]Esencial [ ]Deseable [X]Opcional | | |
| **Prioridad:** | [ ]Alta [ ]Media [X]Baja | **Estabilidad:** | Baja |
| **Verificabilidad:** | [ ]Alta [ ]Media [X]Baja | **Fuente:** | RU-C03 |
| **Descripción:** | El sistema obtendrá la distancia realizada por el usuario bajo la integración de la aplicación con Google Maps. | | |

**Tabla 28. RS-F16 / Google Maps**

| **ID: RS-F17** | | | |
|---|---|---|---|
| **Nombre:** | Sensación térmica | | |
| **Necesidad:** | [ ]Esencial [ ]Deseable [X]Opcional | | |
| **Prioridad:** | [ ]Alta [ ]Media [X]Baja | **Estabilidad:** | Baja |
| **Verificabilidad:** | [ ]Alta [ ]Media [X]Baja | **Fuente:** | RU-C02 |
| **Descripción:** | El sistema deberá obtener la sensación térmica en el momento de la realización de la actividad a través de internet, para un cálculo más exacto del gasto calórico. | | |

**Tabla 29. RS-F17 / Sensación térmica**

| **ID: RS-F18** | | | |
|---|---|---|---|
| **Nombre:** | Interfaz en castellano (predeterminado) | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R03 |
| **Descripción:** | El lenguaje predeterminado del sistema será el castellano. Toda la interfaz será presentada en este idioma. | | |

**Tabla 30. RS-F18 / Interfaz en castellano (predeterminado)**

| **ID: RS-F19** | | | |
|---|---|---|---|
| **Nombre:** | Interfaz en inglés | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R03 |
| **Descripción:** | El sistema tendrá, como segundo idioma, inglés. Toda la interfaz podrá ser presentada en esta lengua. | | |

**Tabla 31. RS-F19 / Interfaz en inglés**

| **ID: RS-F20** | | | |
|---|---|---|---|
| **Nombre:** | Historial de guardado | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04 |
| **Descripción:** | El sistema deberá tener un historial con la actividad realizada por el usuario y el gasto calórico de la misma. | | |

**Tabla 32. RS-F20 / Historial de guardado**

| **ID: RS-F21** | | | |
|---|---|---|---|
| **Nombre:** | FAQ | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C05 |
| **Descripción:** | El sistema tendrá un documento con las preguntas más frecuentes realizadas por los usuarios, además de sus correspondientes respuestas. | | |

**Tabla 33. RS-F21 / FAQ**

| **ID: RS-F22** | | | |
|---|---|---|---|
| **Nombre:** | Manual de usuario | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [ ]Alta [X]Media [ ]Baja | **Estabilidad:** | Media |
| **Verificabilidad:** | [ ]Alta [ ]Media [X]Baja | **Fuente:** | RU-C05 |
| **Descripción:** | El sistema otorgará al usuario un manual en formato electrónico en el que se especificará el uso de cada una de las funcionalidades del mismo. | | |

**Tabla 34. RS-F22 / Manual de usuario**

| **ID: RS-F23** | | | |
|---|---|---|---|
| **Nombre:** | Unidades de medida | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema dará la posibilidad al usuario de elegir si desea utilizar el sistema métrico (kg/km) o el sistema imperial (lb/mi). | | |

**Tabla 35. RS-F23 / Unidades de medida**

| **ID: RS-F24** | | | |
|---|---|---|---|
| **Nombre:** | Borrado de base de datos | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04 |
| **Descripción:** | El sistema permitirá al usuario el borrado de toda la base de datos, o únicamente los historiales de un determinado usuario. | | |

**Tabla 36. RS-F24 / Borrado de base de datos**

| **ID: RS-I01** | | | |
|---|---|---|---|
| **Nombre:** | Acelerómetro | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema utilizará el acelerómetro perteneciente al *smartphone* para calcular cuando el usuario da un paso, contando con cinco niveles distintos de sensibilidad. | | |

**Tabla 37. RS-I01 / Acelerómetro**

| **ID: RS-I02** | | | |
|---|---|---|---|
| **Nombre:** | GPS | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema utilizará el receptor GPS perteneciente al *smartphone* para el posicionamiento del dispositivo. | | |

**Tabla 38. RS-I02 / GPS**

| **ID: RS-I03** | | | |
|---|---|---|---|
| **Nombre:** | Pantalla táctil | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R01 |
| **Descripción:** | El sistema utilizará la pantalla táctil del dispositivo para que el usuario interactúe con la aplicación. | | |

**Tabla 39. RS-I03 / Pantalla táctil**

| **ID: RS-I04** | | | |
|---|---|---|---|
| **Nombre:** | Notificación al usuario | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema notificará en todo momento al usuario ante cada cambio que se produzca en la ejecución mediante una serie de mensajes en pantalla. | | |

**Tabla 40. RS-I04 / Notificación al usuario**

| **ID: RS-RE01** | | | |
|---|---|---|---|
| **Nombre:** | Versión Android | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R02 |
| **Descripción:** | El sistema funcionará con todas las versiones a partir de la actualización 1.6 del sistema operativo. | | |

**Tabla 41. RS-RE01 / Versión Android**

| **ID: RS-RE02** | | | |
|---|---|---|---|
| **Nombre:** | Base de datos SQLite | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04 |
| **Descripción:** | El sistema almacenará todos los datos necesarios (historial, perfil del usuario) dentro de una base de datos SQLite. | | |

**Tabla 42. RS-RE02 / Base de datos SQLite**

| **ID: RS-RN01** | | | |
|---|---|---|---|
| **Nombre:** | Tiempo de respuesta | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R01 |
| **Descripción:** | El sistema deberá funcionar en tiempo real, con unos valores de respuesta en la interacción con el usuario menores de un segundo. | | |

**Tabla 43. RS-RN01 / Tiempo de respuesta**

| **ID: RS-RN02** | | | |
|---|---|---|---|
| **Nombre:** | Validación de campos | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R01 |
| **Descripción:** | Todos los campos y valores introducidos en la aplicación deberán ser validados previamente, para evitar errores de formato. | | |

**Tabla 44. RS-RN02 / Validación de campos**

| **ID: RS-RN03** | | | |
|---|---|---|---|
| **Nombre:** | Requerimientos mínimos | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-R01 |
| **Descripción:** | El dispositivo deberá poseer, al menos, una versión del SO Android superior a la 1.6, GPS, conexión a internet, acelerómetro y pantalla táctil. | | |

**Tabla 45. RS-RN03 / Requerimientos mínimos**

| **ID: RS-OP01** | | | |
|---|---|---|---|
| **Nombre:** | Acceso al sistema | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01, RU-C06 |
| **Descripción:** | Se accederá al sistema a través de una pantalla de inicio, desde la cual se podrá ir a cualquiera de las pantallas del sistema. | | |

**Tabla 46. RS-OP01 / Acceso al sistema**

| **ID: RS-OP02** | | | |
|---|---|---|---|
| **Nombre:** | Interfaces del sistema | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01, RU-C06 |
| **Descripción:** | El sistema contará con una serie de pantallas en las que se encuentren todas las funcionalidades ofrecidas. Las interfaces son: Perfil, Ejecución, Historial, FAQ y Opciones. | | |

**Tabla 47. RS-OP02 / Interfaces del sistema**

| **ID: RS-OP03** | | | |
|---|---|---|---|
| **Nombre:** | Formato en base de datos | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04 |
| **Descripción:** | El sistema definirá un formato único de información dentro de la base de datos, de forma que el almacenamiento sea optimó en espacio y velocidad. | | |

**Tabla 48. RS-OP03 / Formato en base de datos**

| **ID: RS-OP04** | | | |
|---|---|---|---|
| **Nombre:** | Transformación de datos | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C04 |
| **Descripción:** | El sistema será capaz de transformar la información obtenida del dispositivo (acelerómetro, GPS) en un formato legible para el sistema y el usuario. | | |

**Tabla 49. RS-OP04 / Transformación de datos**

| **ID: RS-OP05** | | | |
|---|---|---|---|
| **Nombre:** | Integración acelerómetro-GPS | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema dispondrá de un algoritmo o fórmula que permita ajustar la estimación de pasos del sistema mediante la integración de los datos obtenidos por el acelerómetro y el receptor GPS. | | |

**Tabla 50. RS-OP05 / Integración acelerómetro-GPS**

| **ID: RS-OP06** | | | |
|---|---|---|---|
| **Nombre:** | Formulas calóricas | | |
| **Necesidad:** | [X]Esencial [ ]Deseable [ ]Opcional | | |
| **Prioridad:** | [X]Alta [ ]Media [ ]Baja | **Estabilidad:** | Alta |
| **Verificabilidad:** | [X]Alta [ ]Media [ ]Baja | **Fuente:** | RU-C01 |
| **Descripción:** | El sistema contará con varias fórmulas específicas para realizar el cálculo del gasto calórico. | | |

**Tabla 51. RS-OP06 / Formulas calóricas**

### 5.1.3 Trazabilidad

Una vez que se han obtenido todos los requisitos, se debe comprobar que existe una continuidad entre lo que ha demandado el usuario, y las características que realmente se han concluido que tendrá la herramienta.

Para ello es necesario elaborar una matriz de trazabilidad, en la que se unen requisitos de usuario y de software, de forma que un requisito de software debe estar relacionado con, al menos, uno o más requisitos de usuario.

La matriz de trazabilidad que se encuentra en la Tabla 52 ha comprobado lo dicho en el párrafo anterior, obteniéndose un resultado satisfactorio. Esto quiere decir que no se deben retocar los requisitos por problemas de cohesión en el desarrollo, lo que no implica que no se pueda modificar o añadir algún requisito en el futuro por otras causas.

| | RU-C01 | RU-C02 | RU-C03 | RU-C04 | RU-C05 | RU-C06 | RU-C07 | RU-R01 | RU-R02 | RU-R03 |
|---|---|---|---|---|---|---|---|---|---|---|
| **RS-F01** | X | | | | | | | | | |
| **RS-F02** | X | | | | | | | | | |
| **RS-F03** | X | | | | | | | | | |
| **RS-F04** | X | | | | | | | | | |
| **RS-F05** | X | | | X | | | | | | |
| **RS-F06** | X | | | | | | | | | |
| **RS-F07** | | X | | X | | | | | | |
| **RS-F08** | | X | | X | | | | | | |
| **RS-F09** | | X | | X | | | | | | |
| **RS-F10** | | X | | X | | | | | | |
| **RS-F11** | | X | | X | | | | | | |
| **RS-F12** | | | | X | | | X | | | |
| **RS-F13** | | | | X | | | X | | | |
| **RS-F14** | | | | X | | | X | | | |
| **RS-F15** | X | | | | | | | | | |
| **RS-F16** | | | X | | | | | | | |
| **RS-F17** | | X | | | | | | | | |
| **RS-F18** | | | | | | | | | | X |
| **RS-F19** | | | | | | | | | | X |
| **RS-F20** | | | | X | | | | | | |
| **RS-F21** | | | | | X | | | | | |
| **RS-F22** | | | | | X | | | | | |
| **RS-F23** | X | | | | | | | | | |
| **RS-F24** | | | | X | | | | | | |
| **RS-I01** | X | | | | | | | | | |

| | RU-C01 | RU-C02 | RU-C03 | RU-C04 | RU-C05 | RU-C06 | RU-C07 | RU-R01 | RU-R02 | RU-R03 |
|---|---|---|---|---|---|---|---|---|---|---|
| **RS-I02** | X | | | | | | | | | |
| **RS-I03** | | | | | | | | X | | |
| **RS-I04** | X | | | | | | | | | |
| **RS-RE01** | | | | | | | | | X | |
| **RS-RE02** | | | | X | | | | | | |
| **RS-RN01** | | | | | | | | X | | |
| **RS-RN02** | | | | | | | | X | | |
| **RS-RN03** | | | | | | | | X | | |
| **RS-OP01** | X | | | | | X | | | | |
| **RS-OP02** | X | | | | | X | | | | |
| **RS-OP03** | | | | X | | | | | | |
| **RS-OP04** | | | | X | | | | | | |
| **RS-OP05** | X | | | | | | | | | |
| **RS-OP06** | X | | | | | | | | | |

**Tabla 52. Matriz de trazabilidad RU/RS**

## *5.2 ESPECIFICACIÓN DE LOS CASOS DE USO*

Un caso de uso es una secuencia de interacciones que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

En este apartado se van a describir las secuencias de uso más comunes que se ejecutarán en el sistema, ya que a estas alturas del desarrollo no se pueden contemplar todas y cada una de ellas. Se dividirá este apartado en dos, en una de ellas se detallará de forma textual el caso de uso, y en otra de ellas se explicará de forma gráfica.

### 5.2.1 Descripción Textual

En la descripción textual de cada uno de los casos de uso, se especificarán los campos que se muestran a continuación:

- *Identificador (ID)*: Determinará de forma unívoca cada uno de los casos de uso que se detallarán a continuación. El formato a seguir será: CU-XX siendo las X números comprendidos entre 0 y 9. Se comenzará por el 01 y se irá incrementando una unidad por cada caso de uso nuevo.

- *Título*: Nombre descriptivo acerca del caso de uso.

- *Actor*: Siempre hará referencia a un único tipo de usuario, ya que en el sistema no existen distintos perfiles de usuario.

- *Descripción*: En este campo se realizará una breve descripción del caso de uso a comentar.

- *Precondiciones*: Condiciones previas que se deberán cumplir para poder ejecutar el caso de uso.

- *Postcondiciones*: Condiciones que se producirán tras la ejecución del caso de uso.

- *Escenario principal*: Trata de la secuencia común de interacciones ordenadas, especificando la interacción del usuario con el sistema.

- *Escenario alternativo*: Describirá la ejecución del caso de uso con condiciones de error o caminos de decisión distintos al principal.

Los casos de uso textuales son los siguientes:

| **ID: CU-01** | | | |
|---|---|---|---|
| **Título:** | Crear usuario | **Actor:** | Usuario |
| **Descripción:** | Dar de alta a un usuario en el sistema. | | |
| **Precondiciones:** | No exista anteriormente. | | |
| **Postcondiciones:** | Ya no podrán crearse más usuarios con ese nombre. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Perfil".<br>3. Rellena correctamente cada uno de los campos (nombre, altura, peso, edad, sexo).<br>4. Hace clic en el botón "Guardar Perfil".<br>5. Se crea el usuario y se vuelve al menú principal, notificando al usuario la correcta creación. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>3a. En el caso de que no se rellene correctamente algún campo, se notificará al usuario acerca de que campo es el erróneo.<br>5a. Si el usuario ya existe, se actualizará en la base de datos con los nuevos valores. | | |

**Tabla 53. CU-01 / Crear usuario**

| **ID: CU-02** | | | |
|---|---|---|---|
| **Título:** | Cargar usuario | **Actor:** | Usuario |
| **Descripción:** | Se carga a un usuario ya existente en el sistema. | | |
| **Precondiciones:** | El usuario exista. | | |
| **Postcondiciones:** | Ninguna. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Perfil".<br>3. Hace clic en el botón "Cargar Perfil".<br>4. Aparecerá un campo donde pondrá el nombre del usuario a cargar, dándole al finalizar al botón "Aceptar".<br>5. Se cargará el usuario y se vuelve al menú principal, notificando al usuario la correcta carga. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>4a. En el caso de que no exista el usuario, se notificará al usuario acerca de ello. | | |

**Tabla 54. CU-02 / Cargar usuario**

| **ID: CU-03** | | | |
|---|---|---|---|
| **Título:** | Iniciar Stepping | **Actor:** | Usuario |
| **Descripción:** | El usuario comienza un nuevo entrenamiento. | | |
| **Precondiciones:** | Exista al menos un usuario creado anteriormente. | | |
| **Postcondiciones:** | El sistema ejecutara el cálculo del entrenamiento. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Stepping".<br>3. Hace clic en el botón "Empezar Stepping".<br>4. El entrenamiento comienza, notificando al usuario el inicio del mismo. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>3a. En el caso de que no exista un usuario creado anteriormente, al entrenar no se contabilizarán las calorías. | | |

**Tabla 55. CU-03 / Iniciar Stepping**

| **ID: CU-04** | | | |
|---|---|---|---|
| **Título:** | Finalizar Stepping | **Actor:** | Usuario |
| **Descripción:** | El usuario finaliza el entrenamiento que se está ejecutando. | | |
| **Precondiciones:** | El entrenamiento este ejecutándose. | | |
| **Postcondiciones:** | El sistema parará de ejecutar el entrenamiento. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Stepping".<br>3. Hace clic en el botón "Finalizar Stepping".<br>4. El entrenamiento finaliza, notificando al usuario el fin del mismo. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>2a. El usuario puede ya encontrarse en la interfaz de entrenamiento, ejecutándose el entrenamiento.<br>2b. El usuario puede ya encontrarse en la interfaz de entrenamiento, con el entrenamiento en pausa. | | |

**Tabla 56. CU-04 / Finalizar Stepping**

| **ID: CU-05** | | | |
|---|---|---|---|
| **Título:** | Pausar Stepping | **Actor:** | Usuario |
| **Descripción:** | El usuario pausa el entrenamiento que se está ejecutando. | | |
| **Precondiciones:** | El entrenamiento este ejecutándose. | | |
| **Postcondiciones:** | El sistema parará de ejecutar el entrenamiento. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Stepping".<br>3. Pulsa el botón "Menú" del *smartphone*.<br>4. Aparece el menú de opciones de la interfaz, incluyendo el botón "Pausa".<br>5. Hace clic en el botón "Pausa".<br>6. El entrenamiento se detiene, notificando al usuario el fin del mismo. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>2a. El usuario puede ya encontrarse en la interfaz de entrenamiento, ejecutándose el entrenamiento. | | |

**Tabla 57. CU-05 / Pausar Stepping**

| **ID: CU-06** | | | |
|---|---|---|---|
| **Título:** | Continuar Stepping | **Actor:** | Usuario |
| **Descripción:** | El usuario continúa el entrenamiento que se está ejecutando. | | |
| **Precondiciones:** | El entrenamiento no esté ejecutándose. | | |
| **Postcondiciones:** | El sistema continuará ejecutando el entrenamiento. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Stepping".<br>3. Pulsa el botón "Menú" del *smartphone*.<br>4. Aparece el menú de opciones de la interfaz, incluyendo el botón "Continuar".<br>5. Hace clic en el botón "Continuar". El entrenamiento continuará, notificando al usuario el hecho. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>2a. El usuario puede ya encontrarse en la interfaz de entrenamiento, con el entrenamiento pausado. | | |

**Tabla 58. CU-06 / Reiniciar Stepping**

| **ID: CU-07** | | | |
|---|---|---|---|
| **Título:** | Ver historial | **Actor:** | Usuario |
| **Descripción:** | El usuario accede a un historial de entrenamiento determinado. | | |
| **Precondiciones:** | El usuario este cargado en ese momento y el historial que busca exista. | | |
| **Postcondiciones:** | Ninguna. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Historiales".<br>3. Aparecen todos los historiales de ese usuario ordenados por fecha.<br>4. Hace clic en el historial que desea acceder.<br>5. El historial se carga con éxito, apareciendo el resumen en la pantalla. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación. | | |

**Tabla 59. CU-07 / Ver historial**

| **ID: CU-08** | | | |
|---|---|---|---|
| **Título:** | Borrar historiales | **Actor:** | Usuario |
| **Descripción:** | El usuario borra todos sus historiales de entrenamiento. | | |
| **Precondiciones:** | El usuario este cargado en ese momento. | | |
| **Postcondiciones:** | No existirán historiales de ese usuario en el sistema. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Historiales".<br>3. Pulsa el botón "Menú" del *smartphone*.<br>4. Aparece el menú de opciones de la interfaz, incluyendo el botón "Borrar Historiales".<br>5. Hace clic en el botón "Borrar Historiales". Se borran todos los historiales del usuario, notificando que no existen historiales al usuario y volviendo al menú principal. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>6a. Puede que el usuario no contenga historiales, por lo que se notificará al usuario de que no existen historiales y se volverá al menú principal. | | |

**Tabla 60. CU-08 / Borrar historiales**

| **ID: CU-09** | | | |
|---|---|---|---|
| **Título:** | Leer preguntas frecuentes | **Actor:** | Usuario |
| **Descripción:** | El usuario accede a las preguntas frecuentes porque posee alguna duda. | | |
| **Precondiciones:** | Ninguna. | | |
| **Postcondiciones:** | Ninguna. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Preguntas Frecuentes".<br>3. El usuario lee las preguntas y respuestas, solucionándose el problema. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación. | | |

**Tabla 61. CU-09 / Leer preguntas frecuentes**

| **ID: CU-10** | | | |
|---|---|---|---|
| **Título:** | Cambiar unidad de medida | **Actor:** | Usuario |
| **Descripción:** | El usuario cambia la unidad de medida usada en la aplicación. | | |
| **Precondiciones:** | Tener una unidad de medida seleccionada. | | |
| **Postcondiciones:** | Ninguna. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Menú" del *smartphone*.<br>3. Aparece el menú de opciones de la interfaz, incluyendo el botón "Borrar Opciones".<br>4. Hace clic en el botón "Opciones", cargándose la interfaz.<br>5. Hace clic en la opción "Unidades de medida", cargándose el menú donde seleccionar la unidad a elegir, marcada en verde la opción cargada hasta ese momento.<br>6. El usuario selecciona el tipo de unidad que desea. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación. | | |

**Tabla 62. CU-10 / Cambiar unidad de medida**

| **ID: CU-11** | | | |
|---|---|---|---|
| **Título:** | Cambiar lenguaje | **Actor:** | Usuario |
| **Descripción:** | El usuario cambia el lenguaje usado en la aplicación. | | |
| **Precondiciones:** | Tener un lenguaje seleccionado. | | |
| **Postcondiciones:** | El sistema tendrá el lenguaje seleccionado. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Menú" del *smartphone*.<br>3. Aparece el menú de opciones de la interfaz, incluyendo el botón "Borrar Opciones".<br>4. Hace clic en el botón "Opciones", cargándose la interfaz.<br>5. Hace clic en la opción "Lenguaje", cargándose el menú donde seleccionar la unidad a elegir, marcada en verde la opción cargada hasta ese momento.<br>6. El usuario selecciona el lenguaje que desea, cargándose en ese momento toda la interfaz en el idioma seleccionado. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación. | | |

**Tabla 63. CU-11 / Cambiar lenguaje**

| **ID: CU-12** | | | |
|---|---|---|---|
| **Título:** | Cambiar longitud del paso | **Actor:** | Usuario |
| **Descripción:** | El usuario cambia la longitud de paso usada en la aplicación. | | |
| **Precondiciones:** | Ninguna. | | |
| **Postcondiciones:** | El sistema tendrá la longitud de paso seleccionada. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Menú" del *smartphone*.<br>3. Aparece el menú de opciones de la interfaz, incluyendo el botón "Borrar Opciones".<br>4. Hace clic en el botón "Opciones", cargándose la interfaz.<br>5. Hace clic en la opción "Longitud de paso", cargándose un menú con un campo donde introducir la longitud de paso.<br>6. El usuario introduce la longitud de paso y pulsa el botón "Aceptar", guardándose la longitud de paso. | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>6a. En el caso de que no se introduzca correctamente la longitud de paso, se notificará al usuario acerca de ello. | | |

**Tabla 64. CU-12 / Cambiar longitud de paso**

| **ID: CU-13** | | | |
|---|---|---|---|
| **Título:** | Borrar base de datos | **Actor:** | Usuario |
| **Descripción:** | El usuario borra toda la base de datos de la aplicación. | | |
| **Precondiciones:** | Ninguna. | | |
| **Postcondiciones:** | El sistema no tendrá ningún usuario ni historial almacenado. | | |
| **Escenario Principal:** | 1. El usuario entra en la aplicación.<br>2. Hace clic en el botón "Menú" del *smartphone*.<br>3. Aparece el menú de opciones de la interfaz, incluyendo el botón "Borrar Opciones".<br>4. Hace clic en el botón "Opciones", cargándose la interfaz.<br>5. Hace clic en la opción "Borrar base de datos", cargándose un menú donde se debe confirmar si se desea borrar o no.<br>6. El usuario confirma el borrado haciendo clic en "Si", y se borra la base de datos, notificando de ello al usuario | | |
| **Escenario Alternativo:** | 1a. El usuario puede ya encontrarse en la propia aplicación.<br>6a. El usuario prefiere no borrarla, por lo que pulsa "No", y se vuelve a la interfaz de opciones de la aplicación. | | |

**Tabla 65. CU-13 / Borrar base de datos**

### 5.2.2 Descripción Gráfica

Una vez descritos los casos de uso típicos de esta aplicación con sus escenarios alternativos, se van a describir de forma gráfica estos casos de uso junto con alguno más no incluido en el punto anterior, ya que algunas de ellas son muy similares entre si y no varían mucho respecto a alguna de las desarrolladas de forma textual.

A la hora de realizar los gráficos de los casos de uso, estos se van a dividir dependiendo a que interfaz de la aplicación estén destinados. Por todo ello, se van a presentar cuatro gráficos (uno por interfaz): Perfil, Stepping, Historiales y Preguntas Frecuentes. Se ha omitido la interfaz del menú principal porque es obvio que desde él se puede acceder a todos los casos de uso del sistema.

Además, como se verá en las figuras, desde cualquier interfaz se podrá acceder al menú de opciones, una gran ventaja también para el fácil manejo del sistema.

Los diagramas de uso se pueden observar en las Fig. 9, Fig. 10, Fig. 11 y Fig. 12:

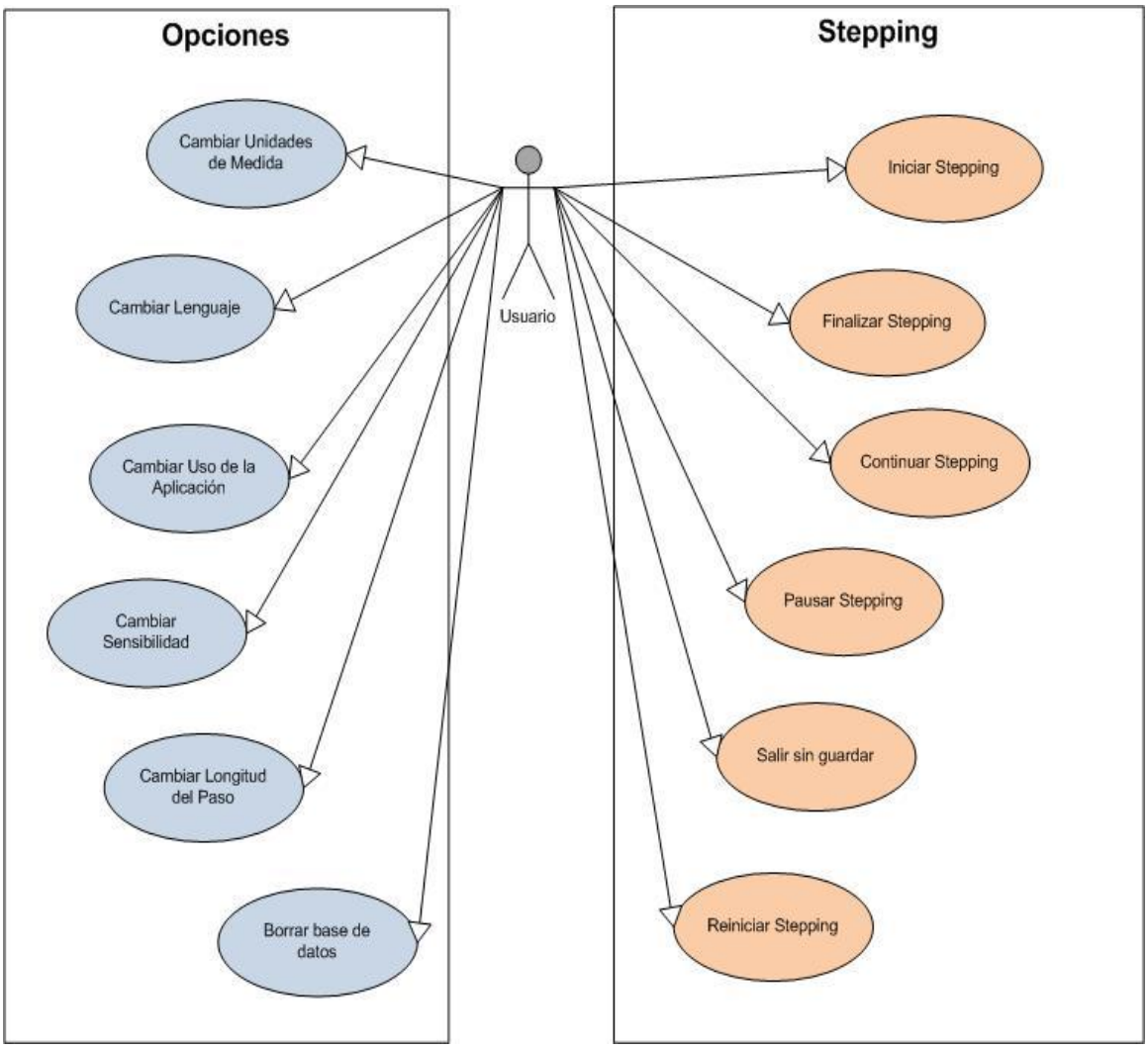



**Fig. 9. Caso de Uso "Stepping"**

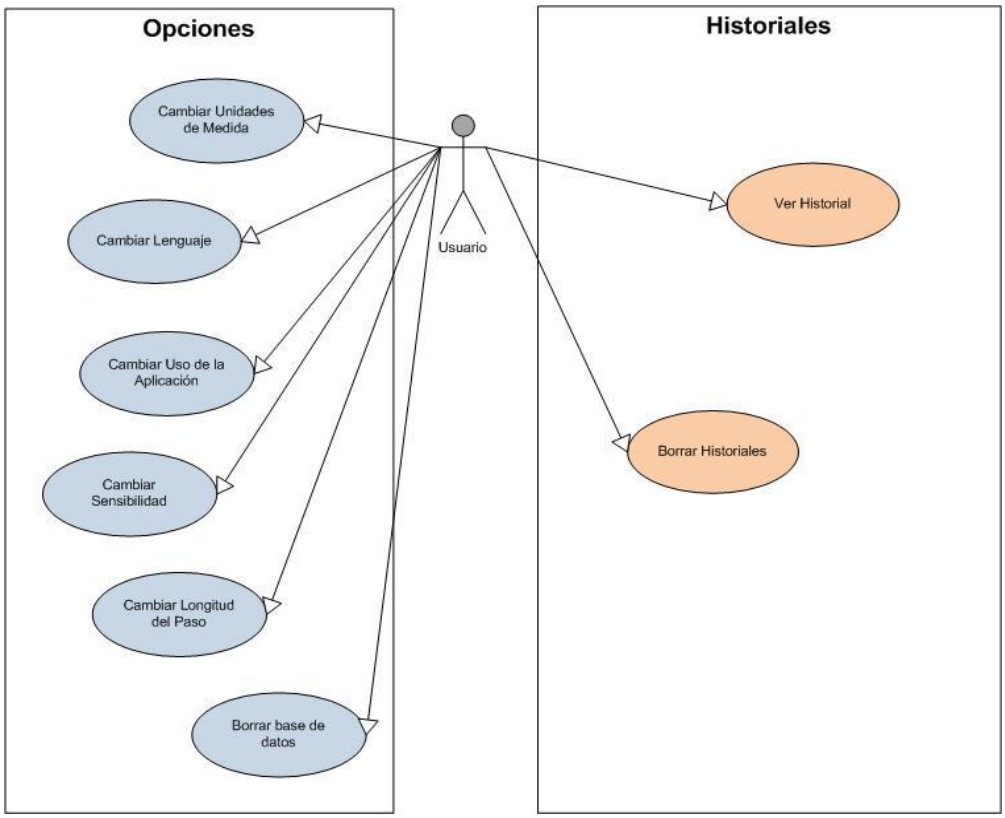



**Fig. 10. Caso de Uso "Historiales"**

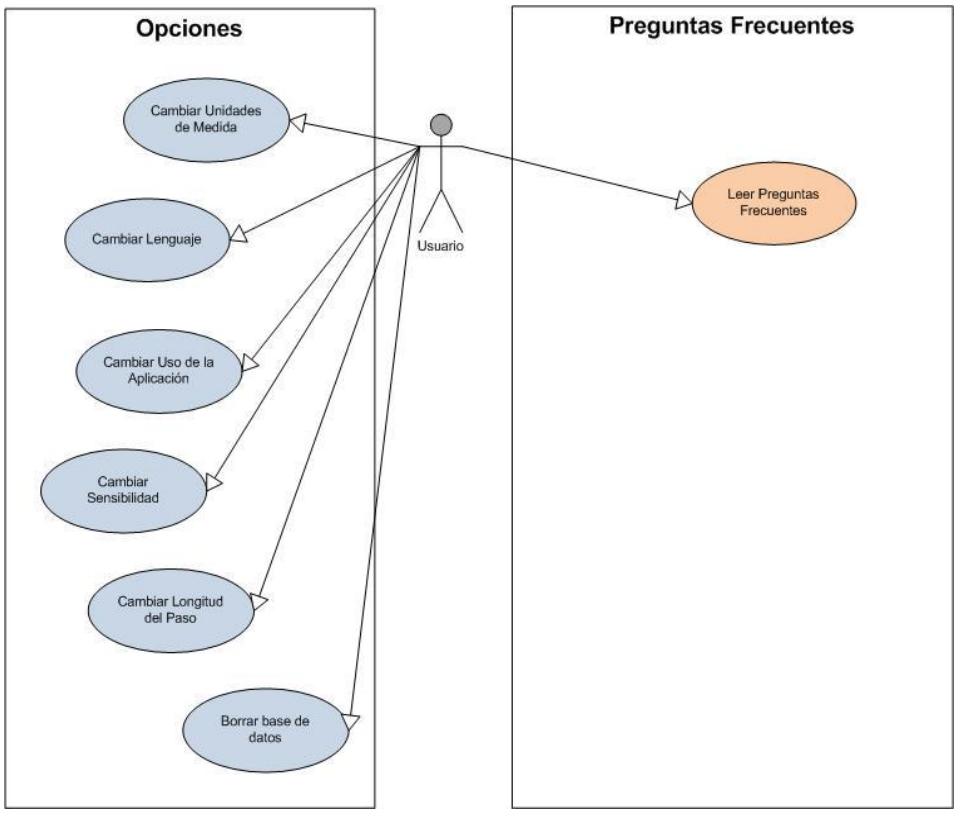



**Fig. 11. Caso de Uso "Preguntas Frecuentes"**

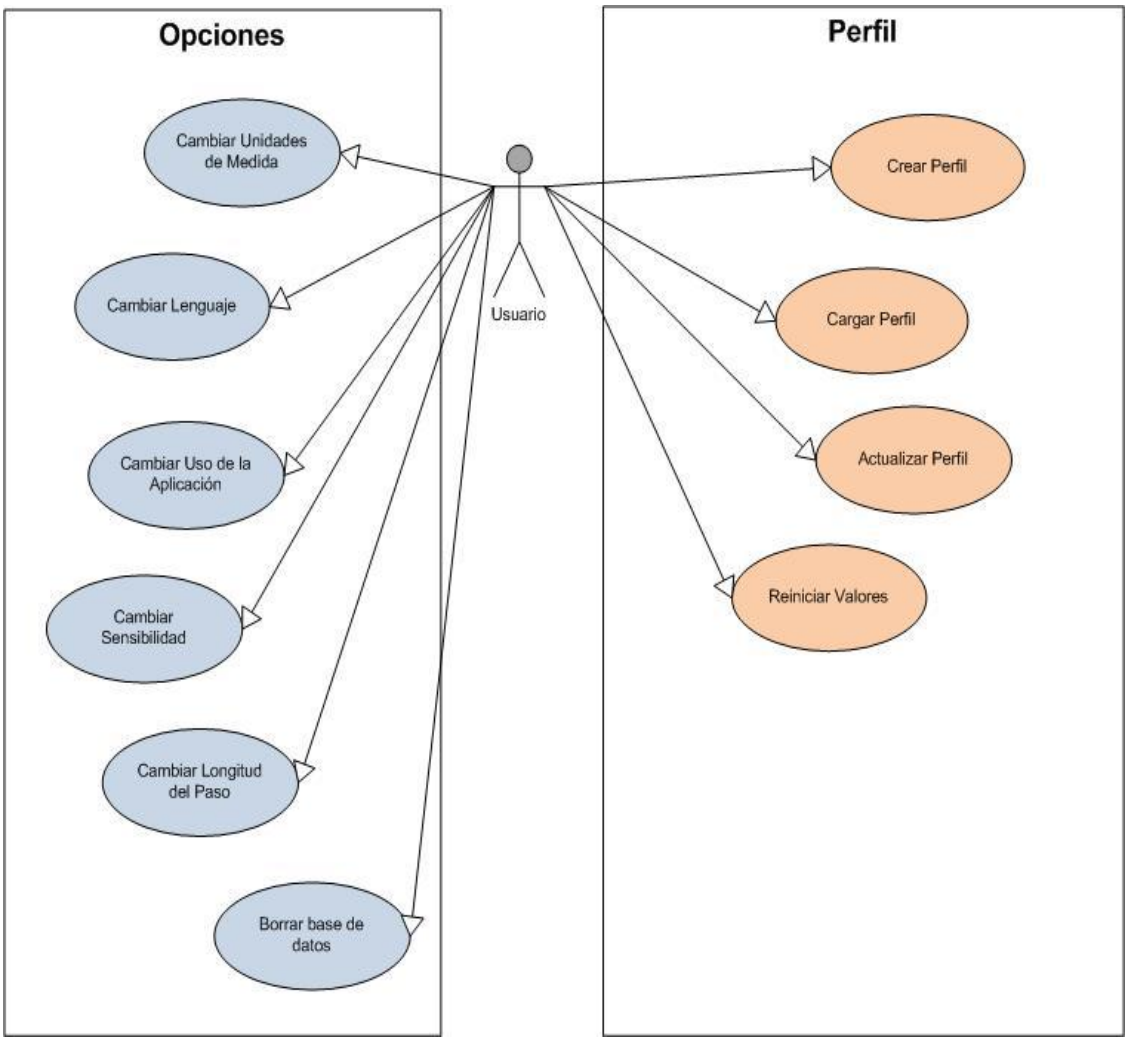



**Fig. 12. Caso de uso "Perfil"**

## *5.3 ARQUITECTURA DE LA APLICACIÓN*

Después de haber definido que es lo que el usuario desea obtener de esta aplicación y haber visto ejemplos prácticos de uso de la futura aplicación, es el momento de que se comience la creación del sistema, realizando un diseño de la aplicación a alto nivel.

Este sistema, en base a lo observado en las peticiones del cliente y a las características del sistema operativo en el que se va a desarrollar, será un sistema que contendrá diferentes interfaces con las que el usuario interactuará y una parte dedicada solo a la lógica del sistema y el tratamiento de la aplicación con el SO, por lo que para la interacción entre las interfaces y la lógica, este sistema necesitará otra pequeña parte dedicada a la comunicación entre las interfaces y la lógica, haciendo de puente entre ellas dos.

Visto todo esto, y una vez estudiados los patrones más conocidos y utilizados en la ingeniería del software, se comprueba que el patrón más acorde a este sistema sería un patrón "Modelo-Vista-Controlador (MVC)". Este patrón está formado por tres partes diferenciadas:

- *Modelo*: Es el encargado de tratar toda la lógica del negocio, representando específicamente toda la información con la que el sistema va a trabajar y notificando a la vista los cambios que se producen en el sistema. Es el "motor" de la aplicación.

- *Vista*: Es la parte del sistema con la que interacciona el usuario, recibiendo y mostrando datos en pantalla. Normalmente se cuenta con más de una interfaz, por lo que también se encarga de navegar entre las distintas interfaces. Es la "cara" de la aplicación.

- *Controlador*: Procesa las peticiones que realiza el usuario y las envía a la lógica del sistema para que las trate y devuelva un resultado la información adecuada para mostrársela al usuario. Es el "transportador" de la aplicación.

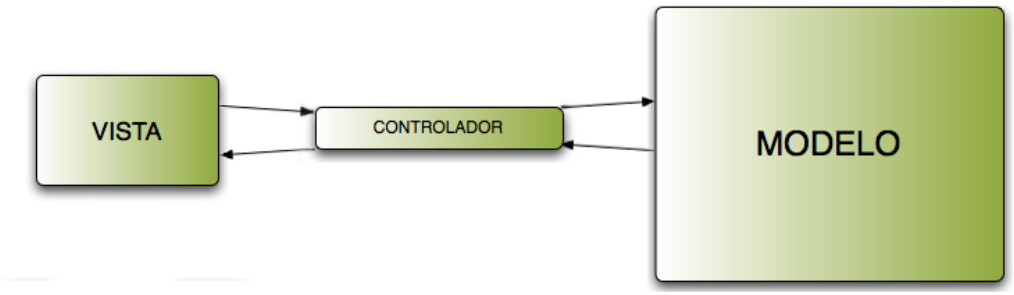



**Fig. 13. Diagrama MVC**

Para que el entendimiento del funcionamiento de la arquitectura del sistema sea más sencillo, se muestra el siguiente ejemplo genérico:

1. El usuario desea ejecutar una actividad cualquiera de la aplicación, por lo que realiza la acción determinada en la pantalla del *smartphone*.

2. La vista que se esté ejecutando en ese momento identifica el evento que ha ocurrido y se lo notifica al controlador, mandándole además los datos que necesite.

3. El controlador se comunica con el modelo, pasándole los datos que ha recibido de la vista.

4. El modelo, con la información recibida, ejecuta todos los comandos y acciones que sean necesarias, para, a continuación, enviar una respuesta al controlador.

5. El controlador recibe la respuesta enviada por el modelo y se la da a la vista en formato legible para el usuario.

6. La vista recibe la respuesta dada y se la muestra al usuario, cambiando incluso de interfaz si fuera necesario.

### 5.3.1 Trazabilidad

Esta arquitectura, como se comentó en su momento, debe cumplir con la premisa de que todo requisito de software debe encontrarse en, al menos, un componente de la arquitectura, por lo que es necesario desarrollar un mecanismo de comprobación. La matriz de trazabilidad de la Tabla 66 confirma esto.

| | MODELO | VISTA | CONTROLADOR |
|---|---|---|---|
| **RS-F01** | X | | |
| **RS-F02** | X | | |
| **RS-F03** | X | | |
| **RS-F04** | X | | |
| **RS-F05** | X | | |
| **RS-F06** | | X | |
| **RS-F07** | X | X | |
| **RS-F08** | X | X | |
| **RS-F09** | X | X | |
| **RS-F10** | X | X | |
| **RS-F11** | X | X | |
| **RS-F12** | X | | |
| **RS-F13** | X | | |
| **RS-F14** | X | | |
| **RS-F15** | X | X | |
| **RS-F16** | X | | |
| **RS-F17** | X | | |
| **RS-F18** | | X | |
| **RS-F19** | | X | |
| **RS-F20** | X | | |
| **RS-F21** | | X | |
| **RS-F22** | X | | |
| **RS-F23** | | X | |
| **RS-F24** | X | X | |
| **RS-I01** | X | | |
| **RS-I02** | X | | |
| **RS-I03** | | X | |
| **RS-I04** | | X | |
| **RS-RE01** | X | X | |
| **RS-RE02** | X | | |

| | MODELO | VISTA | CONTROLADOR |
|---|---|---|---|
| **RS-RN01** | | X | |
| **RS-RN02** | | X | |
| **RS-RN03** | X | | |
| **RS-OP01** | | X | |
| **RS-OP02** | | X | |
| **RS-OP03** | | | X |
| **RS-OP04** | | | X |
| **RS-OP05** | X | | |
| **RS-OP06** | X | | |

**Tabla 66. Matriz de trazabilidad RS/Componentes**

Una vez comprobado que la construcción del sistema continua siendo estable, es el momento de empezar a desarrollar la aplicación a más bajo nivel, incluyendo el diagrama de clases, atributos y métodos y, por supuesto, la codificación del sistema.

## *5.4 IMPLEMENTACIÓN DE LA APLICACIÓN*

En este apartado se tratará todo el proceso de codificación real del sistema, describiendo el lenguaje de programación usado y el entorno de programación en el que se ha desarrollado la aplicación, mostrando el diagrama de clases y explicando las características fundamentales de cada una y, por último, explicando de forma detallada cada una de las decisiones importantes tomadas a la hora de la implementación en código.

### 5.4.1 Lenguaje de Programación y Entorno de Desarrollo

Android, que es el sistema operativo donde se va a desarrollar esta aplicación, utiliza como lenguaje de programación el conocido lenguaje Java. Este es un lenguaje de alto nivel, y a pesar de que la mayoría de aplicaciones para Android son realizadas en Java (por la facilidad de desarrollar en alto nivel frente al bajo nivel), conviene saber que los archivos y librerías de Android se encuentran codificadas en lenguaje C/C++, las cuales son compiladas en código nativo ARM y ejecutadas por el SO.

Para el desarrollo de aplicaciones para Android, Google (creador del sistema operativo) ofrece a todos los programadores un kit de desarrollo, el Android SDK (ya que Android es un sistema operativo de código abierto). Este kit, contiene (tal y como se ha descrito en el punto 3.5.1): las herramientas para el desarrollo del código; complementos como la librería para trabajar con Google Maps; drivers para Windows que permiten la ejecución del código en un dispositivo mediante USB; ejemplos de código para programadores principiantes; documentación con la última API de Android.

Google, como se acaba de comentar, también otorga a los desarrolladores una API [61] muy moderna y completa (incluso más que la propia API de Java), donde se puede encontrar desde todos los atributos y métodos disponibles de Android, hasta gran cantidad de ejemplos con explicaciones de cómo crear clases, interfaces, servicios…

El entorno de desarrollo elegido para la creación de este sistema fue Eclipse (en su versión 3.5.2), debido principalmente a dos razones:

- El lenguaje de programación iba a ser Java. Bajo el punto de vista del creador (y de gran parte de la comunidad desarrolladora), para el desarrollo de código en lenguaje Java existen dos principales entornos de desarrollo, NetBeans y Eclipse. Cualquiera de estos dos hubiera sido útil, pero fue elegido Eclipse por la otra gran razón descrita a continuación.

- El kit de desarrollo SDK contiene un plug-in para Eclipse. Esta fue la principal razón para elegir Eclipse en vez de NetBeans. Este plug-in crea una configuración de proyecto de tal forma que sólo hay que preocuparse por la codificación del sistema, ya que se encarga de preparar el entorno con una configuración de proyecto que permite compilar y ejecutar de forma similar a la que se haría si fuera un proyecto Java únicamente.

Toda aplicación de Android posee, al menos, cuatro archivos que depende unos de otros y se encuentran relacionados entre sí, y que, sin alguno de ellos, la aplicación no funcionaría. Estos archivos son:

- **.xml:* Este tipo de archivos desarrollan la interfaz gráfica de la aplicación, de modo que cada interfaz del sistema debe ser un archivo XML distinto. Toda aplicación debe contener, al menos, una interfaz, es decir, un archivo XML.

- *AndroidManifest:* Archivo XML imprescindible para cualquier aplicación. Solo puede haber uno por sistema, y en él se encuentra la información del sistema necesaria antes de ejecutar cualquier parte del código, como el nombre del paquete, permisos, nivel mínimo de API, etc.

- **.class:* Archivos desarrollados en Java que poseen todo el código de ejecución de la aplicación, la lógica del sistema. Toda aplicación debe contener, al menos, un fichero .class (que se encuentran asociados a un archivo .java, de modo uno a uno).

- *R.java:* Se genera automáticamente en cada proyecto de Android, pudiendo solo haber uno por aplicación. Se utiliza para abreviar cualquier recurso incluido en los ficheros XML nombrados anteriormente, de forma que sea más sencillo de utilizarlos en los ficheros .class.

Java, por su parte, facilita la codificación de la aplicación con el uso de "paquetes" (ver Fig. 14), que permiten una codificación más o menos organizada. Esto hace que los archivos XML se encuentren en un paquete, los archivos .class en otro, el fichero R.java en otro, las librerías en otro paquete aparte…

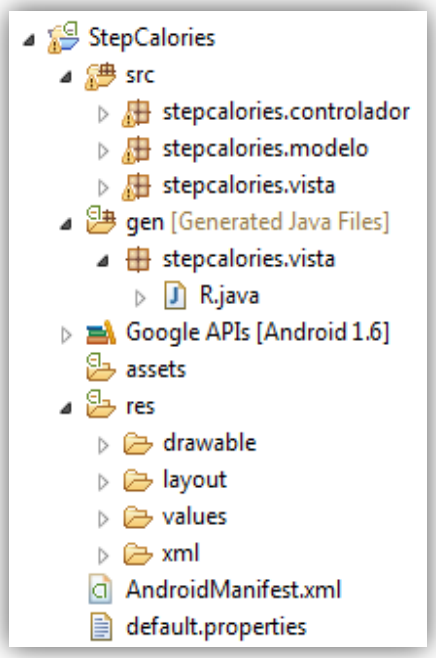



**Fig. 14. Estructura del proyecto**

Además, otorga sus librerías para añadirlas a las propias de Android, de forma que es mucho más sencillo el manejo de fechas, las salidas por pantalla, el manejo de excepciones, etc.

### 5.4.2 Diagrama de Clases

En un apartado anterior se mostró la arquitectura del sistema, lo que daba lugar a un primer esbozo a muy alto nivel de cuál sería la estructura del sistema: tres grandes bloques, cada uno con una función bien definida. Este modelo lógico a alto nivel, se ha ido detallando cada vez más, hasta llegar a un diseño a nivel de clases, utilizando para ello el lenguaje UML [62].

El diagrama de clases UML muestra las clases del sistema con sus atributos y métodos, así como sus relaciones estructurales y de herencia existentes entre ellas. En la siguiente figura (Fig. 15), se mostrará el diseño de clases completo, sin mostrar atributos ni métodos, para, a continuación, mostrar las figuras (Fig. 16, Fig. 17 y Fig. 18) que corresponden a cada paquete del sistema con sus clases y relaciones, incluyendo, además, los atributos y métodos de cada clase.

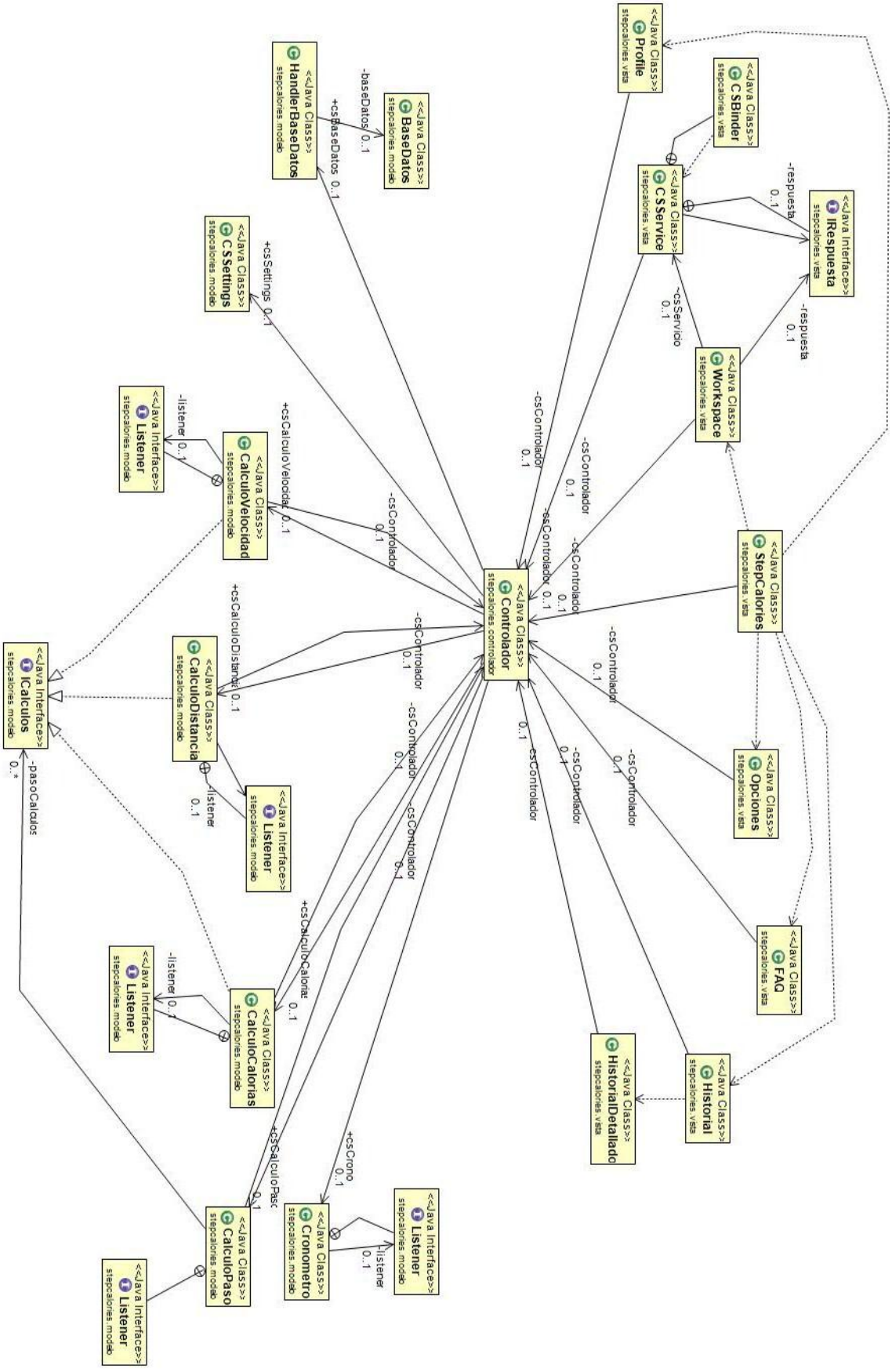



**Fig. 15. Diagrama UML completo**

En el diagrama UML completo de la Fig. 15, se puede observar todas las clases Java que pertenecen al desarrollo de la aplicación, pero sin incluir aquellas clases que pertenecen al desarrollo propio de Android, como serían los documentos XML para el desarrollo de las interfaces, el *AndroidDocument* o la clase *R.java*.

Continuando en el punto del desarrollo donde se encontraba el proyecto, se concluyó que el diseño arquitectónico seguiría el patrón MVC, con tres partes claramente diferenciadas: las interfaces, la lógica y la comunicación entre ambas partes. En el diseño de clases se ha seguido este patrón al pie de la letra, desarrollándose todos los ficheros Java en tres paquetes diferenciados, usando las facilidades que nos permite este lenguaje de programación con la creación de paquetes. Estos paquetes son Modelo, Vista y Controlador (se analizarán a continuación), cada uno perfectamente diferenciado del otro.

Si prestamos atención al diagrama completo, vemos que, efectivamente, se pueden diferenciar estos tres paquetes claramente:

- En la parte superior del diagrama se encuentran las clases correspondientes al paquete "Vista", que trata con únicamente con la clase Controlador.

- En la parte intermedia del diagrama observamos una única clase, que se corresponde con el paquete "Controlador", y que posee relaciones con las clases de los otros dos paquetes.

- En la parte inferior del diagrama se encuentran todas las clases que forman el paquete "Modelo", y que únicamente interaccionan con la clase Controlador.

Todo esto indica, de forma clara, que no existe ningún tipo de comunicación real entre el paquete "Modelo" y el paquete "Vista", respetando una de las ideas principales de este patrón. A pesar de ello, y aunque no aparece representado, sí que existe alguna referencia de una clase del paquete "Vista" al paquete "Modelo", pero no existe comunicación real de datos en esas referencias.

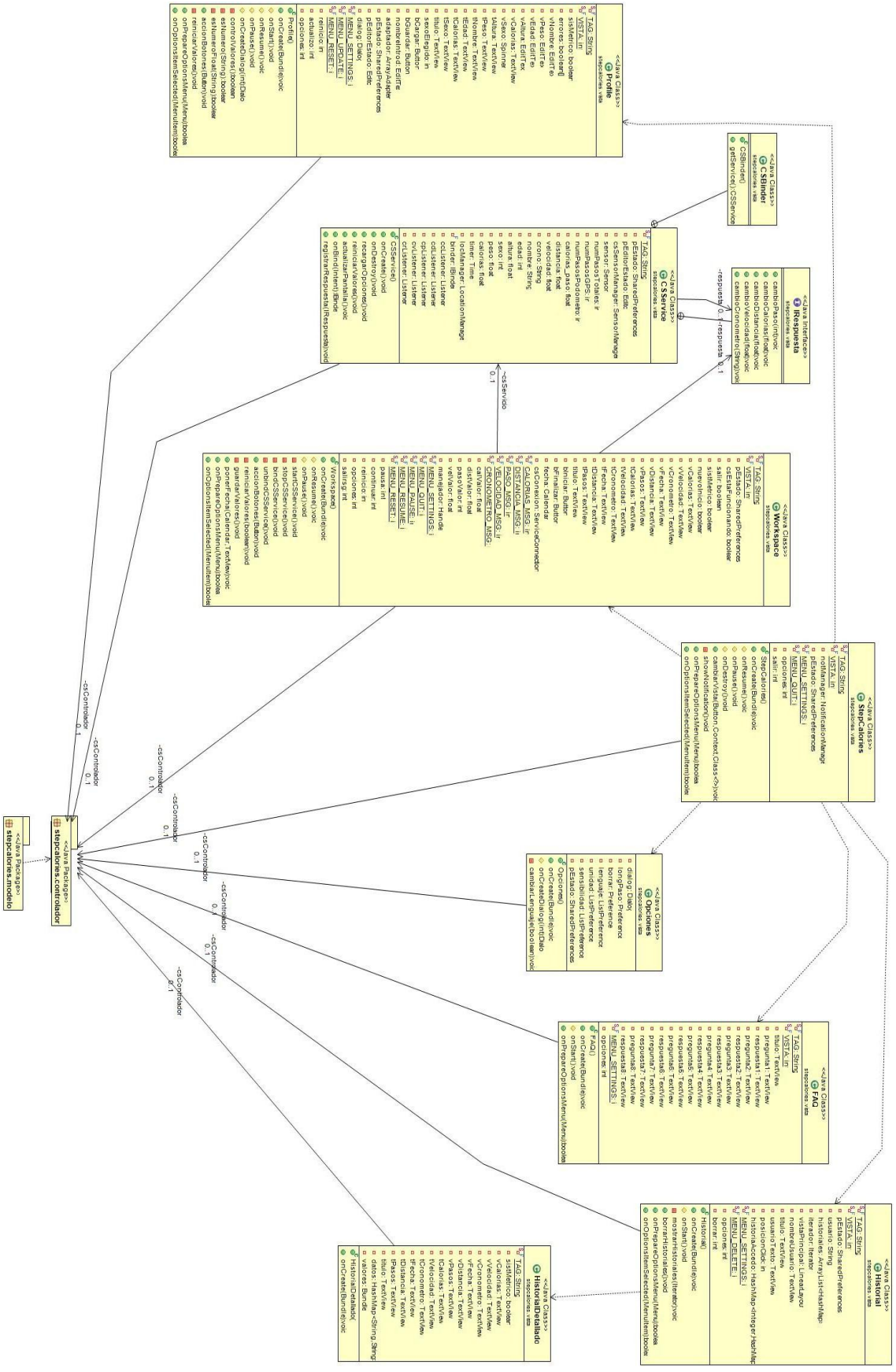


**Fig. 16. Diagrama UML Vista**

En el diagrama UML de la Fig. 16, se puede observar de forma más específica (con sus atributos y métodos), cada una de las clases que forman el paquete "Vista", correspondiente a la parte del patrón arquitectónico que posee el mismo nombre.

La "Vista" del sistema es la encargada de desarrollar la interfaz del sistema, aquello con lo que trata el usuario. Para ello, se ha desarrollado una clase por interfaz, ya que, tal y como demanda Android, debe existir un archivo XML por interfaz, por lo que a cada archivo XML le corresponde una clase Java donde se desarrolla toda esta interfaz. De esta forma, existen 7 interfaces distintas: StepCalories (interfaz inicial), Workspace (encargada del entrenamiento), Profile (tratamiento del perfil de usuario), FAQ (ayuda al usuario, preguntas frecuentes), Historial (historiales del usuario), HistorialDetallado (un solo historial en detalle), Opciones (interfaz de selección de diversas opciones del sistema).

Mención aparte merecen la clase CSService y sus subclases IRespuesta y CSBinder. En Android, aparte de las interfaces (llamadas Activity), existen otro tipo de clases capaces de ejecutar código como si fueran una interfaz, y que son las llamadas clases Service. La característica principal de un Service es que no se ejecuta de cara al usuario, sino que funciona en un segundo plano, por lo que la clase Servicio ejecuta en segundo plano lo que se ejecutaría en primer plano en la clase Workspace (esto se explicará detalladamente en la sección Ejecución en Segundo Plano, del apartado 5.4.3). Las clases IRespuesta y CSBinder se encargan de la comunicación entre la clase Workspace y la clase CSService mediante el mecanismo de paso de mensajes existente en Android.

Aparte de lo mencionado anteriormente, también comentar que existen partes del código comunes a todas las interfaces, ya que son necesarias en todas y cada una de ellas. Estas partes son: el código destinado al menú contextual de la interfaz, que se crea de la misma forma en todas las interfaces; el código destinado al cambio de una interfaz a otra, necesario en todas las interfaces; atributos útiles para la depuración del código.

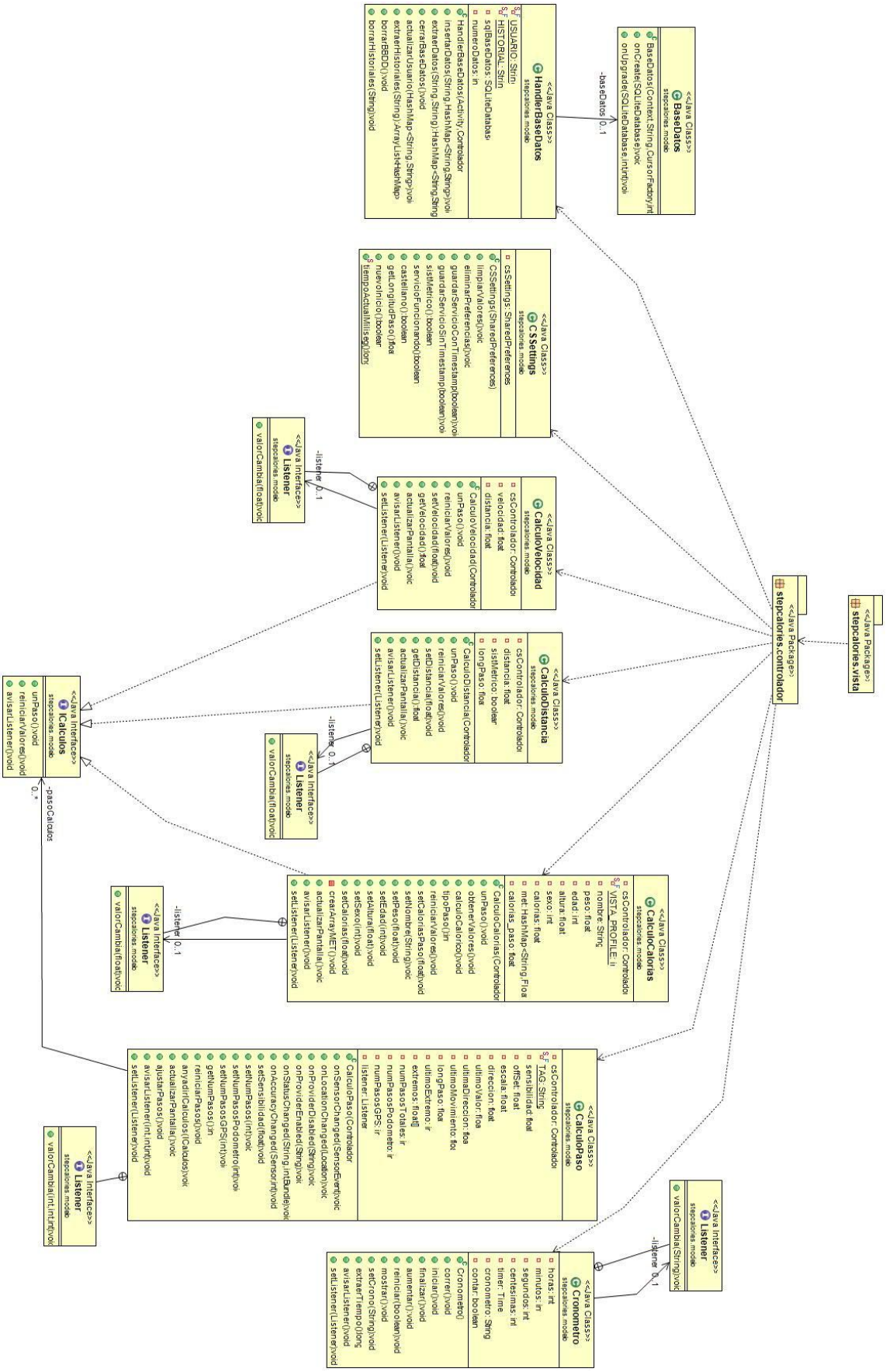


**Fig. 17. Diagrama UML Modelo**

En el diagrama UML de la Fig. 17, se puede observar de forma más específica (con sus atributos y métodos), cada una de las clases que forman el paquete "Modelo", correspondiente a la parte del patrón arquitectónico que posee el mismo nombre.

El "Modelo" del sistema es el encargado de la lógica del sistema, aquella parte de la aplicación que el usuario no ve, pero que es tanto o más importante que la interfaz. Para realizar el cálculo de cada paso (CalculoPaso), cada caloría (CalculoCalorias), la velocidad (CalculoVelocidad) o la distancia (CalculoDistancia), se ha creado una clase encargada de ello, todas ellas descendiente de una interfaz creada en otra clase llamada ICalculos. Además, cada una de estas clases dedicadas al cálculo poseen otra subclase llamada Listener, que es igual para todas las clases, y que se explicara su función en el punto siguiente (básicamente tratan con las interfaces, avisándolas cuando se produce un cambio de valor).

Otra clase es la llamada Cronometro, y que como su propio nombre indica, desarrolla el cronómetro que se usa en la aplicación para calcular el tiempo del entrenamiento. También posee una subclase Listener, exactamente igual a las que existen en las clases Calculos (incluso en la función).

Las clases HandlerBaseDatos y BaseDatos se encargan de todo el tratamiento de la base de datos del sistema, que contiene los perfiles de usuario y los historiales de cada usuario. Para el tratamiento de las opciones del sistema, ya sea para almacenarlas o para acceder a ellas y utilizarlas en otras clases, se ha desarrollado la clase CSSettings.

En este paquete también se encuentra una clase que ni siquiera se ha introducido en el diagrama porque no contiene relación ninguna con ninguna clase, pero es necesaria en el desarrollo de la aplicación y por lo tanto merece mencionarse aquí. Esta clase se llama EditarTextoPreferencias, y se usa para, dependiendo de la unidad de medida utilizada en el sistema y el lenguaje seleccionado, mostrar, en la opción donde se introduce la longitud de paso, el texto adecuado.

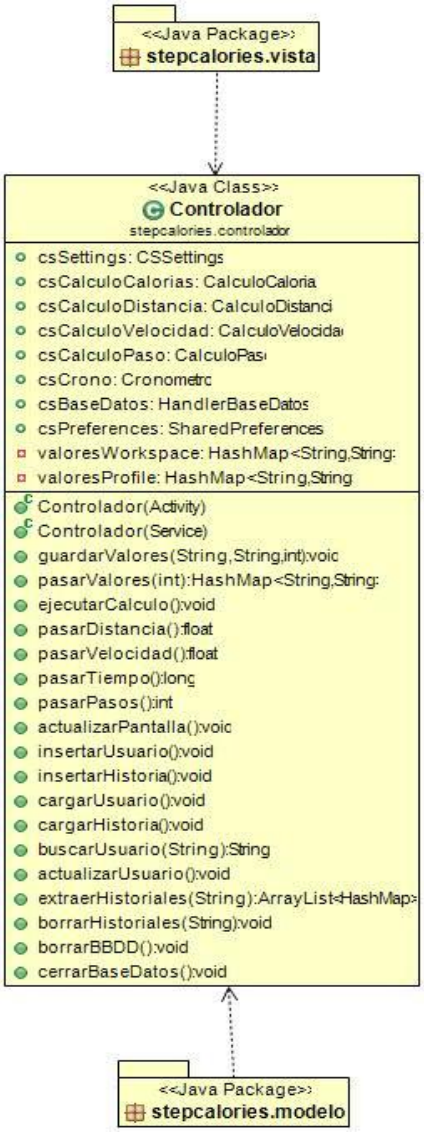



**Fig. 18. Diagrama UML Controlador**

En el diagrama UML de la Fig. 18, se puede observar de forma más específica (con sus atributos y métodos), la clase que forma el paquete "Controlador", correspondiente a la parte del patrón arquitectónico que posee el mismo nombre.

El "Controlador" del sistema es el encargado de la comunicación entre las clases de los otros dos paquetes que conforman el sistema. Este paquete sólo posee una única clase, llamada también Controlador, ya que las comunicaciones no se pueden dividir en más de una clase.

Otra particularidad de esta clase, como vemos en el diagrama, es que posee dos constructores. Esto es debido a que, dependiendo si la comunicación es con una interfaz o con un servicio, se crearán los objetos de diferentes clases pertenecientes al Modelo.

### 5.4.3 Decisiones de Implementación

Durante el desarrollo de la aplicación normalmente surgen incógnitas que no habían aparecido antes o que simplemente sólo pertenecen a este punto del desarrollo, y que son solventadas tomando decisiones que únicamente entiende el equipo de desarrollo. Estas decisiones deben ser, por tanto, explicadas para que todo aquel que estudie el código o simplemente hago uso de la aplicación, entienda el por qué.

En este apartado se van a explicar las decisiones más importantes tomadas durante la implementación, y que se han dividido en base a las características principales que posee el sistema.

*1. Interfaces*

Cuando se inicia el desarrollo de una aplicación, una de las cosas que hay que tener en cuenta es el número de interfaces y cómo van a interactuar entre ellas.

En este proyecto se mostraban dos claras tendencias a la hora del diseño de las interfaces: una tendencia, que se pensó al principio del proyecto cuando no se tenía muy en cuenta lo que el usuario demandaba y no parecía tan extensa la aplicación, pretendía mostrar una única interfaz que mostraría el entrenamiento del usuario, con un botón donde se pudiera acceder al menú de opciones; la otra tendencia, que se pensó una vez se vieron todos los requisitos y que es la que se usa finalmente, consiste en varias interfaces navegables entre ellas, con acceso directo desde todas las interfaces al menú de opciones.

En el sistema que se está desarrollando, finalmente existen 6 interfaces distintas (además de la interfaz de opciones, que es aparte) distinguidas en tres niveles de acceso dependiendo de cómo el usuario accede a ellas, tal y como se puede ver la en la Fig. 19.

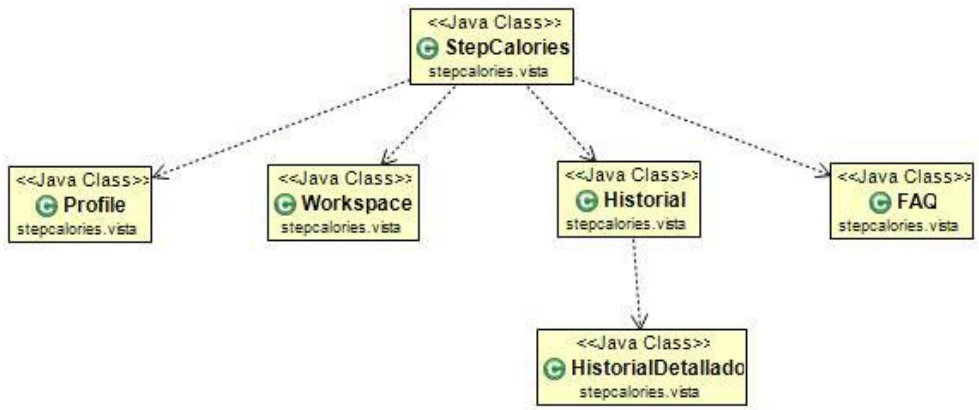



**Fig. 19. Estructuración en niveles de las interfaces**

Android, a las interfaces de una aplicación se las denomina *Activity*. Cada *activity* se corresponde con una vista del usuario, que en código se corresponde a dos ficheros, un fichero XML y un fichero Java. En el fichero XML se desarrolla toda la parte de diseño, colocando los elementos en la posición que se desee, indicando márgenes, tipos de letra, colores, etc. En el fichero Java se desarrolla todo el código de la aplicación, los textos a insertar en los campos, el tratamiento de los valores, el cambio entre *activities*, los menús contextuales… Para explicar las interfaces vamos a hacer una "unión" entre estos dos tipos de ficheros, explicando tanto el diseño como la lógica de las interfaces.

Las interfaces existentes en la aplicación son las siguientes:

- *StepCalories*: Interfaz perteneciente a la pantalla principal de la aplicación. Es la primera interfaz del sistema con la que trata el usuario, a partir de ella se accede al resto de interfaces. Cuenta con un botón para poder acceder a cada una de las interfaces de segundo nivel y un menú contextual donde se muestran dos botones: uno para salir de la aplicación y otro para acceder al menú de opciones.


**Fig. 20. Pantalla "StepCalories"**

- *Profile*: Interfaz perteneciente a la pantalla dedicada al tratamiento del perfil de usuario. Destacar de esta interfaz la gran cantidad de campos que contiene, los cuales son el nombre del usuario, la edad, la altura, el peso y el sexo. En base a estos valores, la aplicación estima las calorías por día que debe consumir el usuario sin realizar ningún tipo de ejercicio, mostrándolas también por pantalla (esto nos servirá también para el gasto calórico). En esta pantalla también se comprueba que los valores introducidos sean correctos (en la edad solo números, el peso no contenga letras…), antes de guardar o actualizar el perfil.

El trato que se otorga a los perfiles es muy amplio, permitiendo mediante los botones de la pantalla o mediante el menú contextual cargar un perfil de usuario, actualizarlo con nuevos valores o guardar uno nuevo. Además en esta pantalla se da la posibilidad de reiniciar los valores y acceder a las opciones del sistema.


**Fig. 21. Pantalla "Perfil"**

- *Stepping*: Interfaz correspondiente a la pantalla dedicada al propio *stepping (*que es una denominación ideada que se otorga en esta aplicación al ejercicio que se recoge en el sistema) o entrenamiento. Es la parte más importante del sistema, ya que la aplicación se basa en lo realizado en esta interfaz, sin ella el resto de la aplicación no sería útil. Tiene relación directa con la clase CSService y la clase IRespuesta, que se explicarán más adelante ya que no son interfaces propiamente dichas.
  Esta interfaz muestra al usuario la fecha del día que se está realizando el entrenamiento, el número de pasos dados, la distancia recorrida, la velocidad media del entrenamiento, el número de calorías consumidas durante el ejercicio y el tiempo consumido.
  En el menú contextual existen cuatro botones: uno para la pausa o continuación del ejercicio, otro para reiniciar los valores, otro para acceder a las opciones, y el último permite al usuario acabar del ejercicio sin guardar, en caso de que desee descartarlo.
  Para el uso de los botones existieron dudas, debido a que en un principio la aplicación se hizo de tal forma que, cuando en el menú principal pulsaras el botón de *stepping*, este se iniciara directamente, pero finalmente se cambió esto de forma que ahora, cuando pulsas en *stepping* en el menú principal se carga la interfaz, de forma que para iniciar el entrenamiento debes pulsar en el botón que te lo indica. Aparte de este botón existe otro para finalizar el entrenamiento.


**Fig. 22. Pantalla "Stepping"**

- *FAQ*: Interfaz correspondiente a la pantalla dedicada a la ayuda al usuario, que incluye las preguntas frecuentes que podría realizar cualquier usuario. A la hora de realizar esta interfaz, surgió la duda de si incluir las preguntas y respuestas en un único campo de texto o dividir cada pregunta y respuesta en un campo de texto único. Al final, y de cara al futuro (poder añadir más preguntas, cambiar el diseño…), se eligió la segunda opción.
  Al ser una interfaz de consulta no existen botones, pero sí que existe el menú contextual existente en todas las interfaces, y que esta vez solo incluye el acceso a las opciones.

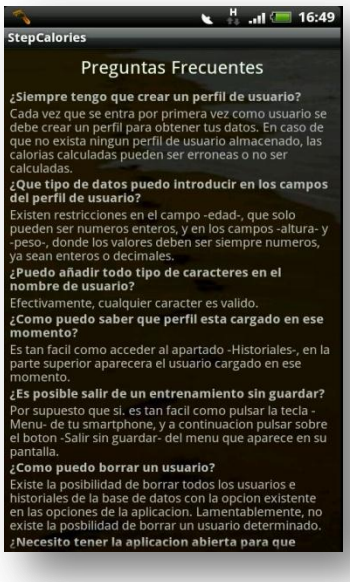



**Fig. 23.Pantalla "Preguntas Frecuentes"**

- *Historial*: Interfaz correspondiente a la pantalla dedicada a mostrar los historiales del usuario que este cargado en ese momento. Esta pantalla dio gran cantidad de problemas debido a que cada usuario en cada momento poseerá una cantidad distinta de historiales. Debido a esto, la única opción existente es que la interfaz se creará en tiempo de ejecución. Por ello el archivo XML correspondiente a esta interfaz no posee prácticamente objetos, sino que se crean directamente en la clase Java.
  En esta pantalla se muestran los historiales de forma resumida, es decir, solo muestran la fecha de ejecución (clave para identificar el ejercicio), el tiempo realizado, los kilómetros recorridos y las calorías quemadas. Una vez que haces clic en cualquier historial, se accede a otra interfaz que se explica a continuación. Cuenta con el menú contextual que permite acceder a las opciones del sistema, además de borrar los historiales de ese usuario.


**Fig. 24. Pantalla "Historial"**

- *HistorialDetallado*: Interfaz correspondiente a la pantalla que muestra el historial detallado seleccionado desde la interfaz de historiales. Esta interfaz, que es la única existente en el tercer nivel de estructuración de clases visto anteriormente, es exactamente igual a la interfaz *Stepping*, pero eliminándole los botones de acción, ya que esta interfaz es de consulta, solo muestra datos. Por lo tanto, desde esta interfaz no se puede ni siquiera acceder a las opciones, ya que siguiendo la política de creación de la aplicación, desde un nivel inferior al nivel de opciones no se puede acceder a este menú.


**Fig. 25. Pantalla "Historial Detallado"**

Mención aparte merece el menú de Opciones. Esta interfaz, que se corresponde con la pantalla dedicada a otorgar al usuario las distintas opciones del sistema, no es una interfaz de la aplicación creada exclusivamente para ella, sino que hereda de la pantalla de opciones clásica de las aplicaciones de Android. Por ello no se ha considerado una interfaz, sino más bien un menú. En esta pantalla tenemos una serie de opciones de la aplicación que se describirán en el punto "Implementación de las Opciones".

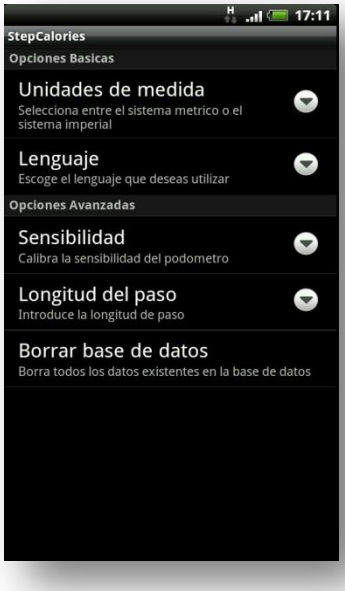



**Fig. 26. Pantalla "Opciones"**

### *2. Ejecución en Segundo Plano*

En la aplicación que se está desarrollando, sería muy útil que el usuario pudiera rellenar o cargar su perfil, iniciar su entrenamiento y después de iniciarlo salir de la aplicación y continuar realizando otras actividades como escuchar música o realizar llamadas, mientras el sistema continúa recogiendo los datos del entrenamiento del usuario. Esto, que podría parecer una utopía hace unos años, o incluso actualmente, no está nada lejos de la realidad.

Android, como sistema operativo móvil, ofrece una cualidad que hace que destaque por delante de otros sistemas operativos, y esta es la capacidad de continuar ejecutando aplicaciones en un segundo plano. Para ello, Android posee un tipo de fichero que lo ha llamado *services* (Servicios). Estos servicios, al trabajar en un segundo plano, el usuario no interactúa con ellos, es más, ni siquiera conoce de su existencia.

Gracias a esto, aquello que parecía tan lejano se convierte en realidad. Esta aplicación permite al usuario seguir registrando sus valores del entrenamiento mientras realiza cualquier otra actividad como llamar por teléfono o enviar SMS.

Todo esto, en el código de la aplicación, está desarrollado por las clases CSServices (y sus subclases CSBinder e IRespuesta) y Workspace, como se puede apreciar en la Fig. 27.

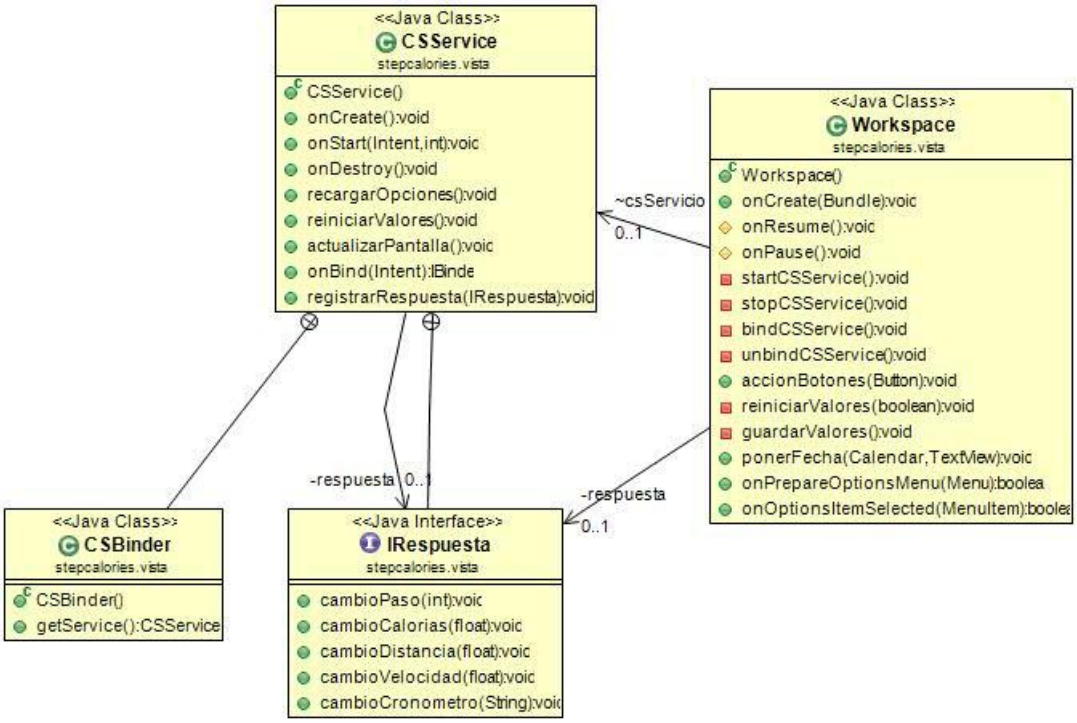



**Fig. 27. Diagrama de ejecución en segundo plano**

La forma de realizar todo lo contado anteriormente es de la siguiente forma:

1. Workspace es la clase principal, con la que el usuario trata. Cuando el usuario entra en esta interfaz por primera vez, se le muestran todos los valores iniciales e interiormente la clase carga todos los valores existentes en las opciones, activa el botón de iniciar y desactiva el botón de finalizar en los métodos onCreate() y onStart(). Una vez hecho todo esto, la interfaz espera que el usuario interactué con ella, pulsando el botón de inicio.

2. Cuando el usuario pulsa el botón, automáticamente la clase Workspace crea un objeto de la clase CSService y la inicia con el método startCSService(), a continuación le pasa el control con el método bindCSService(), para, finalmente, acabar su trabajo pasando a un estado de espera con el método onStop().

3. La clase CSService, por su parte, una vez que ha sido creada y tiene el control, comienza a ejecutar su código, que consiste en permanecer a la espera de que se produzca un paso o reciba la señal GPS del cambio de posición, para actualizar los valores. Aquí es donde se encuentra la parte importante de la ejecución en segundo plano:

   - Si el usuario se encuentra en la interfaz que muestra el *stepping*, verá cómo se actualizan los valores del sistema a través de la subclase IRespuesta, que envía un mensaje interno desde la clase CSService a la clase Workspace (que ha sido de nuevo activada llamando al método onStart() de su clase) con el valor que debe actualizar, el cual recibirá un objeto de tipo Handler dentro de esta clase que actualizará el valor con el método setText() (ver diagrama de secuencia de la Fig. 28).

   - Si el usuario no se encuentra en esta interfaz, los valores se siguen actualizando, pero la aplicación no necesita mostrarlos por pantalla, por lo que no necesita activar la clase Workspace.

4. Aparte de mostrar el valor cada vez que se da un paso o cambia la posición, el cronometro se actualiza con un Timer creado únicamente para ello, que llama al método actualizarPantalla() cada décima de segundo.

5. Cuando el usuario desee finalizar la ejecución (guardando los valores o no), la clase Workspace llamara al método unbindCSService(), para a continuación llamar al método stopCSService que hará que en la clase CSService se llame al método onDestroy() que hace que muera el objeto de esa clase que se estaba ejecutando.

6. Finalmente, en la clase Workspace se llamara también al método finish(), que acabará con esta vista y pasará de nuevo al menú principal del sistema.

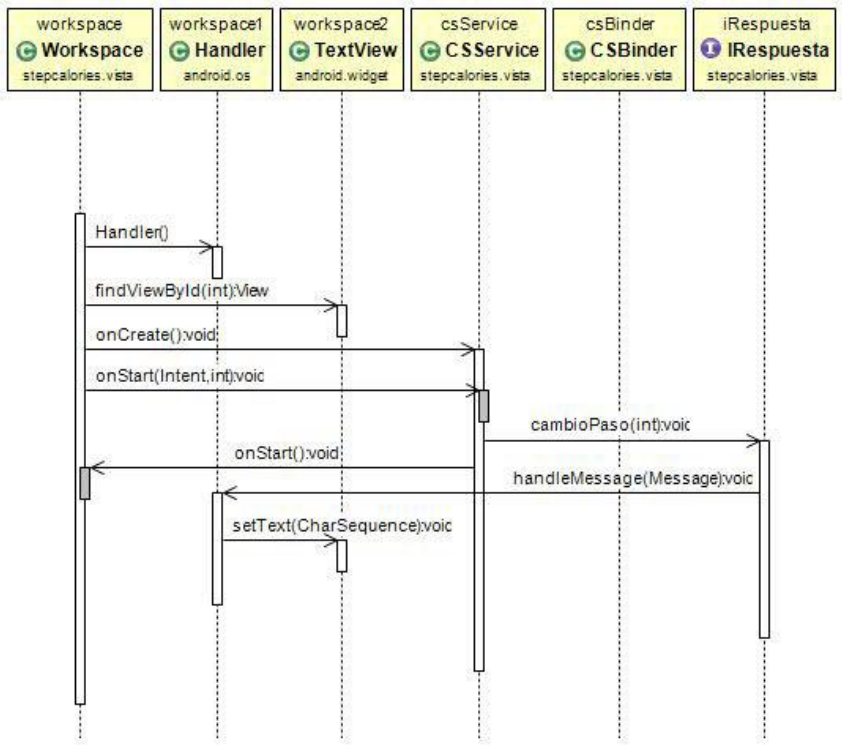



**Fig. 28. Diagrama de secuencia de la comunicación en el paquete Vista al dar un paso**

### *3. Calculo del Paso*

El ser humano se destaca del resto de mamíferos y otras especies por su capacidad para caminar erguido, desarrollada durante miles de años. Ese movimiento bípedo genera un consumo calórico. Cada vez que el ser humano da un paso, realiza un movimiento de todo su cuerpo (principalmente las extremidades inferiores), que puede ser detectado por algún tipo de aparato, como es el *smartphone*.

En los últimos años, la mayoría, por no decir todos, los *smartphone* llevan incluido en sus características un acelerómetro y un receptor GPS. En este proyecto, desde su primer boceto, se pretendía aunar estas dos vertientes, uniéndolas en su sola y creando una aplicación que fuera capaz de hacer una estimación de los pasos dados por el usuario utilizando estas dos características. Comentar que en un primer momento del desarrollo también se pensó en la posibilidad de utilización de Google Maps para el cálculo de la distancia obtenida desde el GPS (tal y como se puede comprobar en la Tabla 28), pero finalmente no ha sido incluida esta idea, ya que como se comprueba al leer la descripción de la implementación desarrollada, no es necesario ningún valor o método de la API otorgada por Google Maps.

Esta unión se ha conseguido, pero antes de explicar la unión de estas dos partes en una sola, se va a explicar cómo funciona cada una por separado, para, a continuación, explicar cómo se integran.

- *Acelerómetro*

Un acelerómetro, como su propio nombre indica, mide la propia aceleración, que es la aceleración física experimentada por un objeto, ya sea lanzado el objeto por una persona u objeto, o que el objeto caiga por si solo en caída libre. Este tipo de aceleración se mide normalmente en términos de G-force (o fuerzas G).

La aceleración está relacionada con la gravedad terrestre. Por ejemplo, un acelerómetro que este sobre una superficie plana con una orientación vertical, devolverá una aceleración de -9.81 m/s, es decir, un G, Pero esto no representa una aceleración real porque el elemento no se ha movido, y sólo representa una fuerza.

Para detectar el movimiento en un entorno tridimensional, necesitaríamos tres sensores, cada uno de ellos orientados a cada uno de los ejes ortogonales (X, Y. Z). El acelerómetro contiene esos tres sensores, de forma que por cada movimiento que se realice, el acelerómetro proporcionará las fuerzas ejercidas en las tres coordenadas del espacio, como se muestra en la Fig. 29. Por ejemplo, si el sensor tiene un movimiento lateral con una aceleración de 1g en el eje X positivo, el resultado seria -9.81 m/s (-1g) en el eje X, y 0 en los ejes Y, Z.

Usando el acelerómetro del *smartphone*, la aplicación es capaz de detectar el movimiento que genera el usuario en cada paso, calculando cuando se realiza uno de ellos.


**Fig. 29. Fuerzas detectadas por un acelerómetro**

En el ejemplo de la Fig. 30, se puede observar como el acelerómetro del *smartphone* detecta la aceleración que produce el ser humano en un paso. El primer pico se corresponde con la pisada del individuo, y los siguientes picos se producen por el movimiento del *smartphone* en el bolsillo, balanceándose en cada paso que dé el humano, lo que produce la detección del paso. En el segundo paso el pico es menor debido a que la aceleración es realizada hacia el lado opuesto del dispositivo, además de que la aceleración es menor y más armonizada con el movimiento del cuerpo entero que en el primero. Por último, en el tercer paso los picos son similares debido a que el movimiento es constante con el segundo paso e igual de armónico respecto al resto del cuerpo.

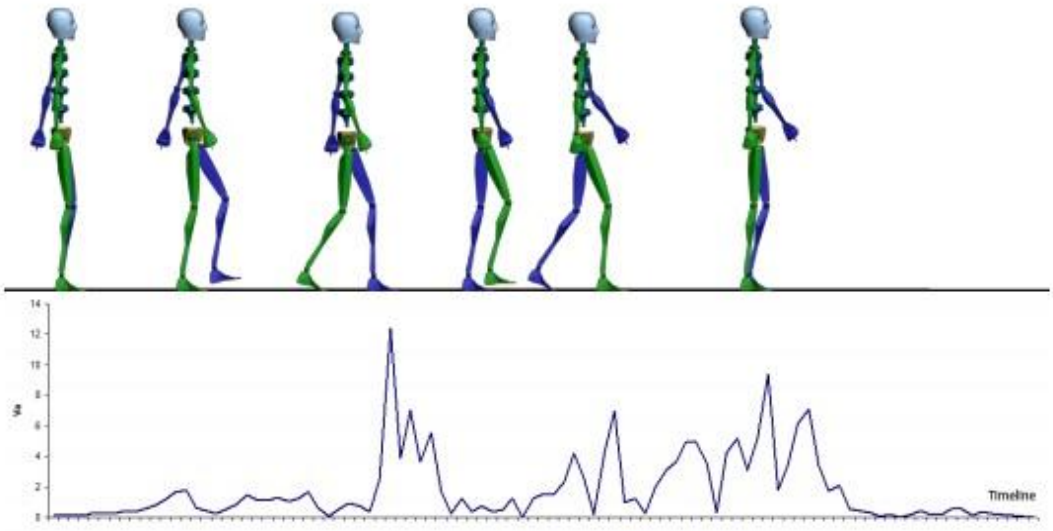



**Fig. 30. Representación del movimiento realizado al caminar por un ser humano**

Toda esta teoría ha sido aplicada en el sistema, desarrollando un algoritmo de cálculo del paso incluido en el método onSensorChanged() de la clase CalculoPaso, la cual hereda el método de la clase SensorEventListener incluida en las librerías de Java. El algoritmo es el siguiente:

1. Se escoge el acelerómetro de todos los sensores disponibles en el *smartphone*, y se suman todos los valores de los tres ejes con los que cuenta (X, Y, Z).

2. Se divide la suma entre los tres ejes y se compara con el último valor registrado por el sensor, obteniendo la dirección del movimiento del sensor.

3. Analizamos la dirección para continuar con la posibilidad de que exista paso:

    a. Si la dirección cambia, guardamos la dirección para el posible próximo paso y obtenemos el valor absoluto de la aceleración.

b. Si no cambia la dirección, se descarta como paso pero se guarda la dirección y la aceleración para el posible próximo paso.

4. Si la dirección ha cambiado, dependiendo de la sensibilidad y otros factores se contabilizará o no como paso:

   a. Si el movimiento es mayor que la sensibilidad del sensor puesta por el usuario en las opciones del sistema (se explica a continuación), la aceleración es mayor que dos tercios de la aceleración del anterior movimiento, la última aceleración es mayor que un tercio de la aceleración actual y el último extremo es distinto que el extremo actual, entonces es que existe paso.

   b. En caso de que el movimiento sea menor que la sensibilidad, o la aceleración es menor que dos tercios de la aceleración del anterior movimiento, o la última aceleración es menor que un tercio de la aceleración actual, o el último extremo es igual que el extremo actual, entonces no existe paso y se guarda la aceleración para el siguiente posible paso.

6. Si existe un paso, se aumenta el número de pasos contabilizados por el podómetro y se llama al método ajustarPasos(), que es el método que une los pasos calculados aquí con los del receptor GPS.

Una vez que se ha llamado a este método, se debería actualizar la interfaz con todos los valores excepto el número de pasos, que se hará una vez se haya extraído el número de pasos conjunto. Para ello contamos con otra de las ventajas que nos otorga Java, las interfaces.

Se ha creado en la parte "Modelo" del sistema una interfaz llamada ICalculos, que contiene tres métodos: unPaso(), reiniciarValores() y avisarListener(). A esta interfaz pertenecen las clases CalculoCalorias, CalculoDistancia y CalculoVelocidad. En la clase CalculoPaso existe un ArrayList con un objeto de cada clase anterior, de forma que cada vez que el algoritmo anterior detecta un paso, con un iterador se recorre todo el ArrayList, haciendo que cada objeto realice el método unPaso(). Este método, en cada clase hará que el método avisarListener() avise al listener para que le diga a la interfaz correspondiente del cambio de valor (tal y como se vio en el punto anterior, Ejecución en Segundo Plano).

Un último apunte es la inclusión de la sensibilidad. Si se observa de nuevo la Fig. 30, se aprecia como el acelerómetro va realizando una gráfica de fuerzas G a lo largo del tiempo. Para saber si se está realizando un paso o no, no se puede tomar cada pico como un paso, sino que se debe colocar un filtro que determine hasta qué punto del movimiento del sensor se considera un paso. En el sistema desarrollado, se le da al usuario la posibilidad de ajustar la detección del paso a la cantidad de movimiento del *smartphone*, pudiendo elegir entre cinco niveles de sensibilidad.

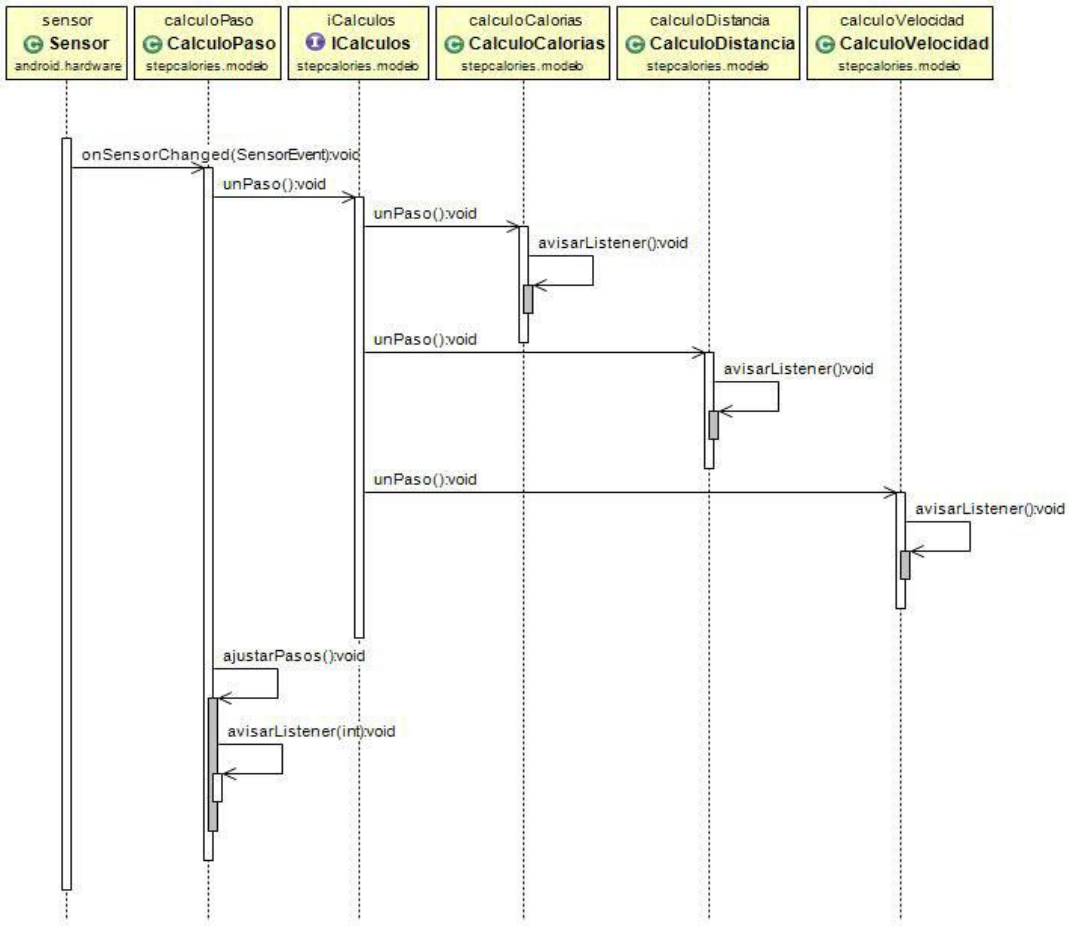



**Fig. 31. Diagrama de secuencia de detección de un paso por acelerómetro**

- *Receptor GPS*

El receptor GPS también es una parte importante para Android, por lo que también contiene sus librerías propias, que son capaces de darle una gran cantidad de usos. El desarrollo de la parte del código del GPS hace uso de estas librerías, de las cuales las parte usada es una interfaz denominada LocationManager, que nos permite conocer la posición del *smartphone* en todo momento. Esta posición no es exacta, sino que devuelve la posición con un porcentaje de error. Esta clase posee varios métodos que nos permitirán realizar aquello que el usuario demandó en los requisitos.

Las clases implicadas en este desarrollo son la clase CSService, la clase CalculoPaso y la clase CalculoDistancia. Obviamos comentar que al participar clases de los paquetes Vista y Modelo participa la clase Controlador del paquete del mismo nombre, pero esta no se incluye ya que solo trata las comunicaciones entre ellos, algo que no es importante en el desarrollo de este apartado. La implementación que se ha llevado a cabo, es la siguiente:

1. Se ha determinado que la clase CalculoPaso implemente la interfaz LocationManager comentada, por lo que esta clase implemente todos los métodos de la interfaz. Pero el único que realmente interesa para el desarrollo y que es codificado es el método onLocationChanged().

2. En la clase CSService se crea un objeto de la clase LocationManager, para a continuación llamar al método requestLocationUpdates(), que posee varios parámetros, entre los que se encuentra un tiempo y una distancia. Este método lo que produce es una llamada al método onLocationChanged() de la clase CalculoPaso, y los parámetros indican el tiempo que debe pasar entre la llamada anterior y la actual o cada cuanta distancia entre la señal anterior del GPS y la actual se debe realizar la llamada, respectivamente. En nuestro código hemos establecido cero el tiempo entre llamadas, y diez metros la distancia que debe realizar el usuario para que se produzca una llamada al método.

3. Cuando se produce la llamada al método onLocationChanged(), se ejecuta su código, que consiste en que se extrae la longitud de paso introducida por el usuario en las opciones para dividir los diez metros entre la longitud de paso y sumárselos al número de pasos contabilizados por el GPS, de forma que nos salga el número de pasos actual (en valor entero, obviamos decimales de la división). Una vez que tenemos los pasos, llamamos al método ajustarPasos() para unir estos pasos con los del acelerómetro y por último, al igual que cuando detectaba un paso el acelerómetro, se llama a todas y cada una de las clases pertenecientes a la interfaz ICalculos para que actualicen sus valores, tal y como se explicó en el apartado del acelerómetro.

- *Unión del acelerómetro y el GPS*

Una vez que se ha obtenido un paso a partir del acelerómetro o del GPS… ¿Cómo unir esto? Pues esta labor se realiza en el método ajustarPasos(), de la siguiente forma:

1. Se comprueba si existe número de pasos del GPS:

   a) Si no existen pasos del GPS quiere decir que a este método se le ha llamado por detección de un paso del acelerómetro, por lo que se asigna el número de pasos del podómetro como el número de pasos dados hasta ese momento por el usuario.

   b) Si existen pasos del GPS quiere decir que por lo menos una vez se ha llamado a este método desde método de detección del GPS, por lo que se ejecuta la formula obtenida a partir del aprendizaje automático, la cual une los pasos del acelerómetro con los del receptor GPS, devolviendo una estimación del número de pasos dados por el usuario, y que es la expresada en la Fig. 32.

*Pasos estimados = (0.658 * Pasos acelerómetro) + (0.5202 * Pasos GPS) + 3.4468*

**Fig. 32. Fórmula para la estimación de los pasos**

2. Se llama al método avisarListener() para que se actualice el número de pasos dados en la interfaz.

En la Fig. 31 se puede ver el diagrama de secuencia de la detección de un paso realizada por el acelerómetro, mientras que en la Fig. 33 se observa la detección de un paso por parte del GPS. En ambas se ha omitido la parte de la notificación a la interfaz, ya que se encuentra incluida en la Fig. 28.

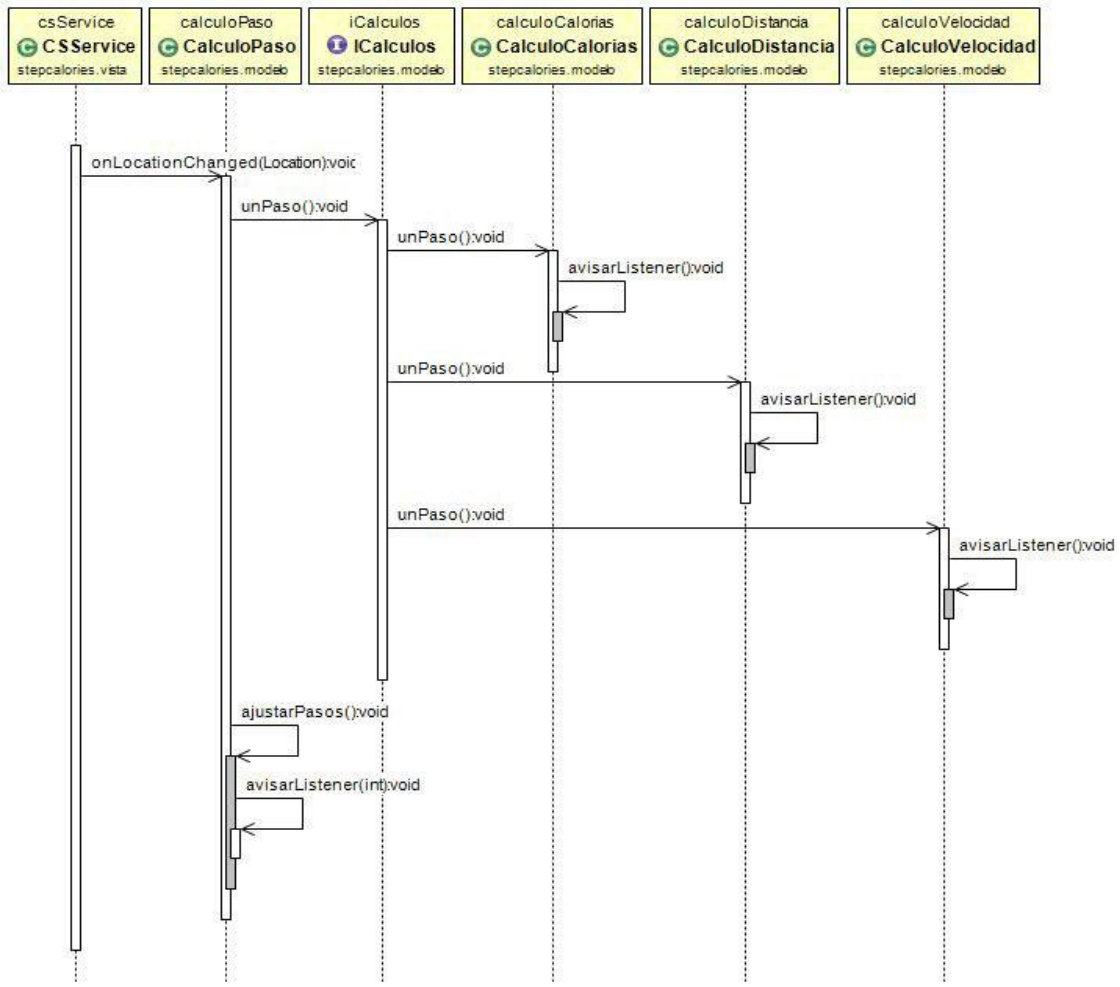



**Fig. 33. Diagrama de secuencia de detección de un paso a partir del GPS**

De esta forma, cada vez que se detecta movimiento del *smartphone* (ya sea a través de la aceleración o del desplazamiento) se comprobara si es un paso, y en caso de que si, se actualizará la interfaz.

*4. Aprendizaje automático*

Se ha hablado en el punto anterior acerca de una expresión que une los pasos del acelerómetro con los pasos del GPS y devuelve la estimación de pasos dados por el usuario, y que se corresponde con la Fig. 32. Esta expresión esta obtenida a partir de un algoritmo de aprendizaje automático.

Para realizar el aprendizaje automático se ha seguido la técnica de regresión lineal. Esta técnica consiste la existencia de un número de variables independientes, las cuales, cuando son tomadas a la vez en una formula, producen un resultado: una variable dependiente. Si se observa esta descripción, se puede comprobar que este tipo de aprendizaje concuerda con los datos que se encuentran en el sistema: una variable dependiente (número de pasos estimado), la cual depende de dos variables independientes entre sí (pasos del acelerómetro y pasos del GPS).

Pero antes de elegir este sistema de aprendizaje, se analizaron distintos tipos de algoritmos de aprendizaje automático a través de la herramienta Weka, descrita en el punto 3.5.3. Esta herramienta permite comprobar de forma rápida y sencilla que algoritmo es el que mejor puede estimar un valor en base a distintos ratios.

El estudio que se ha realizado consiste en una base de datos de pruebas de cincuenta valores, en los que se incluía por cada registro el número de pasos del acelerómetro, el número de pasos del GPS y el número de pasos reales, contados por el propio usuario. A esta base de datos de pruebas, se le aplicaron los siguientes algoritmos:

- *M5 Rules* [64]: Genera una lista de decisión para problemas de regresión usando separación y conquista, en cada iteración genera un árbol M5 y elige la mejor hoja como regla.

- Decision Table *[65]:* Crea una tabla de decisión, clasificando las condiciones obtenidas de los valores.

- *M5P* [66] [67]: Lleva a cabo una regresión por tramos, con cada tramo determinado a partir de un árbol de regresión.

- *Rep Tree* [68]: Árbol que aprende mediante decisión rápida. Construye un árbol de decisión/regresión usando la información acerca de la reducción de la varianza y poda usando la técnica de podado por reducción de error. Solo funciona con atributos con valores numéricos.

- *SVMreg* [69]: Implementa una maquina vectorial para realizar la regresión. Los parámetros de este clasificador pueden aprender a partir de varios algoritmos.

- *Linear Regression* [70]: Implementa una regresión lineal para calcular los coeficientes de la función por predicción. Usa el criterio Akaike para seleccionar el modelo y es capaz de tratar con instancias que poseen pesos determinados.

- *MultiLayer Perceptron* [70]: Se trata de una red neuronal, la cual usa propagación hacia atrás para entrenar. Esta red puede ser construida a mano, por un algoritmo, o por ambas formas a la vez. La red puede también ser monitorizada y modificada durante el tiempo de entrenamiento.

Los resultados obtenidos a través de Weka se pueden observar en la tabla comparativa de la Tabla 67:

| | RAIZ CUADRADA DEL ERROR CUADRATICO MEDIO | DESVIACION ESTANDAR |
|---|---|---|
| **M5Rules** | 11.50 | 4.29 |
| **Decision Table** | 16.59 | 5.12 |
| **M5P** | 11.50 | 4.29 |
| **REP Tree** | 15.57 | 6.74 |
| **SVMReg** | 11.22 | 4.63 |
| **Linear Regression** | 11.50 | 4.29 |
| **Multilayer Perceptron** | 13.87 | 4.73 |

**Tabla 67. Comparativa de algoritmos de aprendizaje en Weka**

En esta tabla se puede observar que los valores obtenidos por los clasificadores que utilizan algoritmos de regresión (como el algoritmo M5Rules, M5P o Linear Regression) son mejores que los obtenidos por el resto de técnicas, ya que la desviación estándar se encuentra entre los 6.74 registros erróneos del estimador más impreciso (Rep Tree), a los 4.29 registros erróneos de los estimadores más precisos, como son el M5Rules, M5P o Linear Regression.

Estos valores se han tomado usando como estimador la Raíz Cuadrada del Error Cuadrático Medio (RMSE), la cual es utilizada para obtener la diferencia entre el valor predicho de un modelo o estimador y los valores reales de aquello que fue estimado o modelizado, y cuya fórmula es la que aparece en la Fig. 34. Para realizar las comparativas que se muestran en la Tabla 67 se llevó a cabo un *t-test*, ejecutando una validación cruzada de 10 *folders* repetidas veces y con un nivel de significación del 5%.

$$\mathrm{RMSE}(\theta_1,\theta_2) = \sqrt{\mathrm{MSE}(\theta_1,\theta_2)} = \sqrt{\mathrm{E}((\theta_1-\theta_2)^2)} = \sqrt{\frac{\sum_{i=1}^{n}(x_{1,i}-x_{2,i})^2}{n}}.$$

**Fig. 34. Fórmula de obtención de la raíz cuadrada del error cuadrático medio**

Finalmente se ha escogido el M5Rules, debido a que nos otorgaba de forma directa una formula, la cual se podía exportar al sistema desarrollado de una forma bastante sencilla. La fórmula se puede observar en la Fig. 32, y como ya se ha dicho, nos otorga una estimación media por debajo de doce pasos de error, respecto a los obtenidos por el acelerómetro y el GPS.

### *5. Gasto Calórico*

Otra de las características que destacan de este sistema es la estimación del gasto calórico durante el ejercicio que realiza el usuario.

En el apartado 3.3 se describieron los pasos a seguir y las fórmulas correctas para obtener una estimación precisa de las calorías quemadas al realizar un ejercicio. En el sistema se han aplicado estos pasos, tal y como se muestran en aquel apartado. Para ello, han intervenido las clases CalculoCalorias, CalculoVelocidad y Cronometro (obviamos la clase Controlador, que también se encuentra en cada interacción entre Modelo y Vista).

Las calorías se calculan en cada paso o cada vez que el GPS indique una nueva posición, debido a que en cada paso o nueva distancia se actualizan los factores que intervienen en las fórmulas utilizadas para la estimación (la distancia, la velocidad y el tiempo). La secuencia lógica de cálculo de calorías quemadas se puede observar en la Fig. 35:

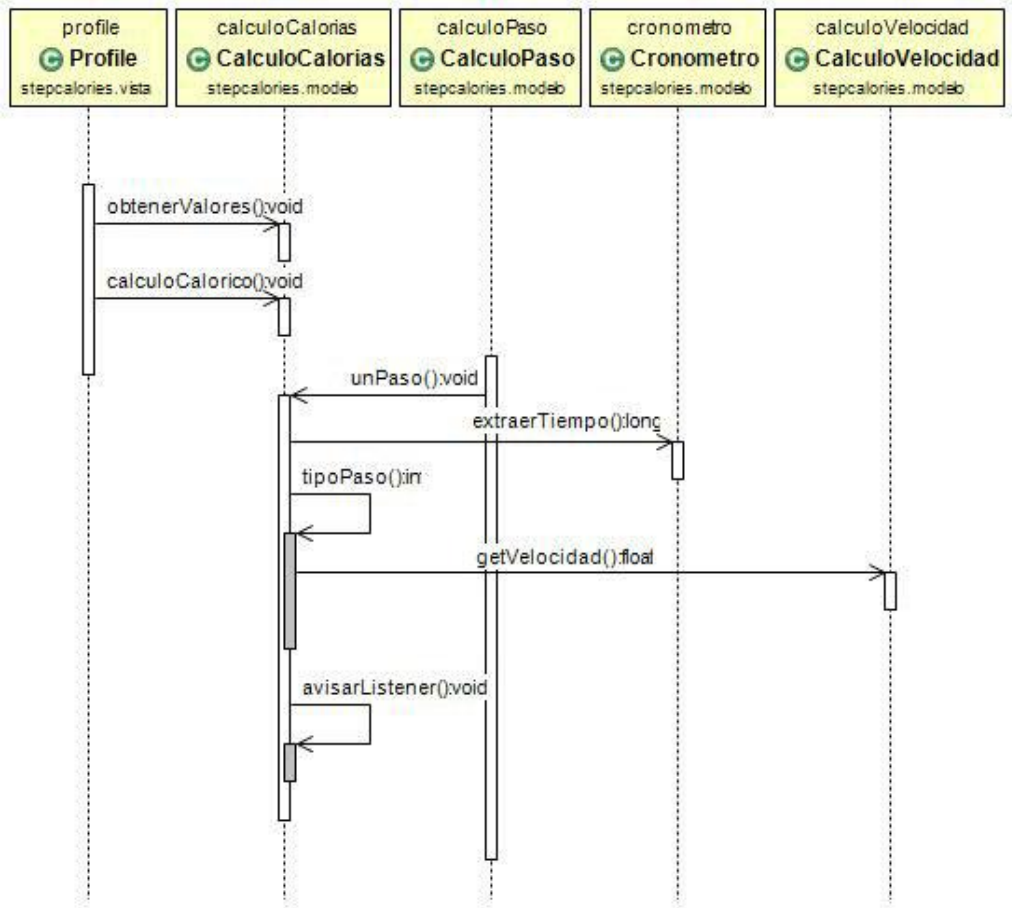



**Fig. 35. Diagrama de secuencia del cálculo calórico**

El usuario, en la interfaz de perfil, introduce sus valores: nombre, edad, peso, altura y sexo. Estos valores, cuando el usuario pulsa el botón "Guardar", se almacenan mediante el método obtenerValores() en un HashMap en la clase CalculoCalorias, para a continuación, con esos valores y utilizando la fórmula de Harris-Benedict que aparece en la Fig. 6, calcular el BMR del usuario, devolviéndolo con el método calculoCalorico() y mostrándoselo por pantalla en la interfaz de perfil, justo antes de volver al menú principal.

Una vez en la interfaz de entrenamiento e iniciado el *stepping*, cada vez que se reconozca que se ha dado un nuevo paso o el GPS actualice la distancia, desde CalculoPaso se llamará al método unPaso() de la clase CalculoCalorias (ver punto anterior). En este método, lo primero que se hace es obtener el tiempo actual llamando al método extraerTiempo() de la clase Cronometro, para a continuación llamar al método tipoPaso(), que en función de la velocidad a la que se está realizando el ejercicio (y que se obtiene mediante el método getVelocidad() de la clase CalculoVelocidad), devolverá un valor entre uno y seis, que indicará que tipo de ejercicio estamos realizando y devolverá el valor MET correspondiente.

Una vez tenemos el tiempo y el valor MET, se pasa a calcular primero el MET corregido con su fórmula correspondiente (ver Fig. 7), y después el gasto calórico en el tiempo que lleve de actividad, con la fórmula final de calculo que se muestra en la Fig. 8. Cuando se haya calculado el gasto calórico, se avisara a la interfaz de ello y se actualizará la pantalla.

Conviene comentar que en el momento en que la clase CalculoPaso llama al método unPaso(), el orden en el que lo hace es importante, ya que antes de avisar a CalculoCalorias, debe hacerlo a CalculoVelocidad, porque si no se podría hacer el cálculo del gasto calórico con un MET erróneo.

### *6. Base de Datos (Perfiles e Historiales)*

El usuario, en sus requisitos, describió como requisito el almacenamiento de la información relevante del sistema en una base de datos eficiente, por lo que en la aplicación se ha desarrollado una base de datos para el almacenamiento de perfiles de usuarios e historiales.

En un primer momento se pensó que la mejor opción a la hora de realizar una base de datos sería guardar los valores en ficheros editados internamente por la aplicación con una codificación propia, pero estudiando a fondo el sistema operativo se descubrió que en todas sus versiones incluye el sistema de gestión de bases de datos SQLite, por lo que se tomó la decisión final de implementar la base de datos del sistema en SQLite, aprovechando otro de los recursos que nos otorga el sistema operativo.

Esta base de datos está formada por los datos que maneja el sistema, y que son los perfiles de usuarios y los historiales. Estos dos tipos de datos se almacenan en dos tablas de datos distintas, pero están relacionados de forma que cada historial está relacionado con un usuario, y un usuario posee uno o más historiales, actuando el identificador del usuario como clave externa de un historial, tal y como se puede comprobar en el modelo relacional de la Fig. 36.

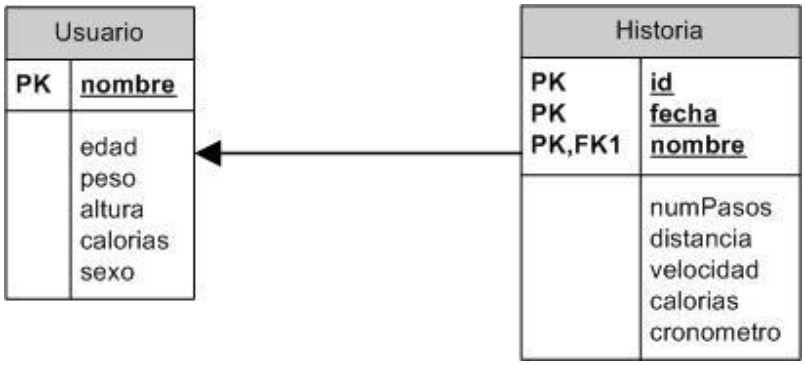



**Fig. 36. Modelo relacional de la base de datos**

El usuario, a pesar de que sabe que existe la BBDD, no la percibe durante el uso del sistema. En la aplicación, dependiendo de la tabla a la que se acceda, se utiliza en los siguientes casos:

- En el caso de los *perfiles*, cuando un usuario entra en la interfaz "Perfil" y crea, carga u actualiza un perfil de usuario, se accede a la base de datos para insertar un nuevo registro de usuario, cargar un registro ya creado o actualizar el registro deseado.

- En el caso de los *historiales*, cada vez que el usuario ejecuta un entrenamiento y, en la interfaz "Stepping", hace clic en el botón para finalizar un entrenamiento, directamente en la base de datos se guarda ese historial, el cual podrá ser revisado en un futuro desde la interfaz "Historiales".

En lo referente a la parte del código, la base de datos es tratada en las clases BaseDatos y HandlerBaseDatos. Además, las clases Profile y CSService a través de la clase Controlador hacen las llamadas comentadas anteriormente, ya sea para guardar, cargar o actualizar los registros de las tablas.

La clase BaseDatos se refiere a la propia base de datos del sistema. Hereda de la clase perteneciente a Android SQLiteOpenHelper, la cual posee un método onCreate() que se ejecuta cada vez que se abre la base de datos, y que es donde se encuentra la creación de las tablas. Para ello, se usa el método execSQL(), que sirve para ejecutar código SQL en Android, cuyo código SQL debe estar escrito en un dato de tipo String. Las tablas se han creado con las siguientes sentencias:

- ***Usuarios*:** `"CREATE TABLE Usuario(nombre TEXT NOT NULL, edad INTEGER NOT NULL, peso FLOAT NOT NULL, altura FLOAT NOT NULL, calorias FLOAT NOT NULL, sexo INTEGER NOT NULL, PRIMARY KEY (nombre))".`

- ***Historial*:** `"CREATE TABLE Historia(id INTEGER NOT NULL, fecha DATE NOT NULL, numPasos INTEGER NOT NULL, distancia FLOAT NOT NULL, velocidad FLOAT NOT NULL, calorias FLOAT NOT NULL, cronometro TEXT NOT NULL, usuario TEXT NOT NULL, FOREIGN KEY(usuario) REFERENCES Usuario(nombre), PRIMARY KEY (id, fecha, usuario))".`

La clase HandlerBaseDatos es la única clase que realiza todos los accesos directos a la base de datos. Para ello, lo primero que se hace es crear un objeto de la clase BaseDatos y abrir la BBDD en modo escritura, añadiendo además con otra sentencia que las claves ajenas sean permitidas en esta BBDD. En el resto de la clase existen métodos que realizan las inserciones, actualizaciones y recuperaciones de datos, de la siguiente forma:

- Para realizar una *inserción*, se crea un objeto ContentValues, en el cual se va introduciendo los atributos del registro que queremos insertar, y a continuación con el método insert() de la API de SQLite para Android, se inserta el registro en la tabla correspondiente.

- Para *recuperar* un registro, se realiza una consulta a la base de datos, que variara dependiendo de lo que se desee recuperar. Para ejecutar la consulta, primero se deben crear dos arrays donde se insertan por un lado las columnas a recuperar y por otro los valores de los campos que se vayan a utilizar para la consulta. Después se realiza la consulta con el método query() de la API de SQLite para Android, donde se introduce la tabla a consultar y los dos arrays creados anteriormente, obteniendo un objeto de la clase Cursor (también de la API) con el resultado de la consulta. Para obtener los registros, se debe recorrer ese objeto Cursor, hasta que no existan más registros.

- Para realizar la *actualización* de un registro, se deben conjugar estas dos acciones descritas anteriormente, realizando primero la consulta, para, a continuación, realizar con el método update() de la API de SQLite para Android, introducir un objeto de la clase ContentValues, al igual que se describió para insertar un registro nuevo.

También existe la posibilidad de borrado, pudiendo borrar solamente los historiales de un usuario o toda la base de datos. Estos dos borrados se ejecutan así:

- Para borrar los *historiales*, el usuario solo debe pulsar el botón de borrar en la interfaz "Historiales", lo que hará que se ejecute el método borrarHistoriales() de la clase HandlerBaseDatos, donde se realiza la llamada al método delete() de la API de SQLite para Android, indicando la tabla Historial, con la condición de que el campo "usuario" sea el del usuario que actualmente está usando la aplicación.

- Para borrar *toda la base de datos*, el usuario deberá pulsar la opción de borrar toda la base de datos en el menú de opciones de la aplicación, lo que hará que se ejecute el método borrarBBDD() de la clase HandlerBaseDatos, donde se realiza una doble llamada al método delete() de la API de SQLite para Android, una indicando la tabla Usuario y otra indicando la tabla Historial, sin condición alguna.

### *7. Implementación de las Opciones*

Este sistema, como cualquier otra aplicación de este calibre, debe de otorgar al usuario una serie de características que permitan configurar el sistema tal y como desea, ampliando así el rango de usuarios capaces de utilizar el sistema. El usuario solicitó en los requisitos una serie de opciones de personalización, que han sido ampliadas para otorgar al sistema de mayor posibilidad de configuración.

Las opciones que han sido llevadas a cabo en el sistema y que se encuentran divididas entre opciones básicas y opciones avanzadas dependiendo de si influyen en el comportamiento del entrenamiento o sólo en las interfaces, son las siguientes:

- *Unidades de medida*: Con esta opción básica se permite al usuario elegir entre el sistema métrico (basado en kilómetros, kilogramos…) o el sistema imperial (aquel que se basa en millas, libras…). Por defecto, el sistema tiene definido el sistema métrico.


**Fig. 37. Pantalla opción "Unidades de medida"**

- *Lenguaje*: Con esta opción básica se permite al usuario elegir entre el castellano o el inglés. Estos son los lenguajes que se encuentran de momento, pero en un futuro se podrían ampliar debido a la facilidad de su desarrollo (ver siguiente punto, Creación de los lenguajes). Por defecto, el sistema tiene definido el lenguaje castellano.


**Fig. 38. Pantalla opción "Lenguaje"**

- *Sensibilidad*: Con esta opción básica se permite al usuario elegir entre los cinco niveles de sensibilidad que posee el sistema, y que son: Muy Bajo, Bajo, Medio, Alto y Muy Alto. Por defecto, el sistema tiene definida la sensibilidad con un nivel medio.


**Fig. 39. Pantalla opción "Sensibilidad"**

- *Longitud de paso*: Con esta opción básica se permite al usuario introducir la longitud de su paso. Este campo, al ser sólo utilizado cuando se realiza el cálculo de la distancia en pasos, se bloquea cuando se elige la opción de uso de la aplicación en el exterior, no dando al usuario la opción de introducir o cambiar su zancada. Por defecto, el sistema tiene definida la longitud de paso a 80 centímetros, ya que es la zancada media de una persona adulta, obtenida de varios estudios realizados, descritos en [63].


**Fig. 40. Pantalla opción "Longitud de paso"**

- *Borrar base de datos*: Con esta opción avanzada se permite al usuario borrar toda la base de datos del sistema. Esta opción, una vez que haces clic en ella, muestra al usuario un mensaje de confirmación, para que, en caso de que el usuario haya marcado la opción sin querer, pueda mantener su base de datos intacta, aumentando la seguridad del sistema.

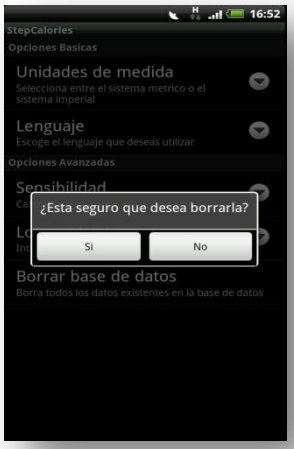



**Fig. 41. Pantalla opción "Borrar base de datos"**

En cuanto al desarrollo de estas opciones, Android recomienda que estas se definan en un fichero XML. Este fichero usa el menú de opciones que posee Android por defecto, de forma que casi toda la interfaz se encuentra ya desarrollada, y solo es necesario incluir las opciones deseadas. La aplicación utiliza este fichero XML para el desarrollo de la mayoría de estas opciones, pero no todo lo que muestra el menú de opciones de la aplicación se encuentra en este fichero XML.

Para ello se ha creado una clase llamada Opciones dentro del paquete "Vista", que se encuentra relacionada con el fichero XML, al igual que se encuentran las interfaces del sistema con el resto de clases de este paquete. Esta clase desarrolla la opción de borrar la base de datos, ya que es una opción que no se encuentra dentro de las opciones predeterminadas del SDK de Android. Para ello, crea un *Dialog*, que consiste en una ventana emergente que aparece cuando pulsas un botón u opción, donde muestra al usuario la decisión de si desea o no borrar la base de datos. En caso de que pulse "Sí", a través de la clase Controlador llamará al método borrarBBDD() de la clase HandlerBaseDatos, tal y como se explicó en el punto anterior, borrando la base de datos.

### *8. Creación de los lenguajes*

Las aplicaciones se crean para atraer al mayor número de usuarios, de forma que realizar una aplicación en un solo idioma hace que el universo de posibles usuarios disminuya notablemente. Por ello, este sistema se encuentra desarrollado en dos idiomas: el idioma oficial del país de desarrollo, el castellano, y el idioma más hablado en el mundo, el anglosajón.

Android, al tener un pensamiento de desarrollo similar al descrito en el párrafo anterior, otorga al desarrollador mecanismos para facilitar el cambio de lenguaje en las aplicaciones. Este sistema operativo da la posibilidad de escribir cada campo de texto del sistema en un dato de tipo *String* dentro de un fichero XML. De esta forma, si introduces todos los textos de un idioma en un fichero XML, cambiando cada texto del fichero a otro idioma y guardándolo en otro fichero XML, ya tendrás otro idioma, y solo hace falta que a la hora de mostrarlo, compruebes que lenguaje ha seleccionado el usuario en las opciones, y mostrarlo en ese idioma. Y eso es justo lo que se ha hecho.

El sistema contiene dos ficheros, uno con los textos en castellano y otro con los textos en inglés. En todas las interfaces, antes de cargar cualquier texto, se consulta que lenguaje eligió el usuario en las opciones, y en base a ello y con un sencillo *if-else*, el sistema es capaz de mostrar toda la interfaz en el idioma que haya seleccionado el usuario.

Además, esto hace que si en un futuro la aplicación tiene éxito y se desea ampliar el número de idiomas, únicamente será necesario crear un nuevo fichero XML con los textos en el nuevo idioma, ampliar en las opciones los lenguajes disponibles y añadir una condición a los *if-else*.

# 6. PRUEBAS Y RESULTADOS

En este capítulo se van a presentar las pruebas realizadas al sistema, con las cuales aseguraremos que los requisitos definidos al principio del proyecto se han cumplido. En un primer punto se describirá el entorno de pruebas donde se han realizado, para continuar en los siguientes puntos con las pruebas llevadas a cabo. En el último punto de este apartado se hará un pequeño resumen de los resultados obtenidos, destacando los puntos más importantes de las pruebas.

## *6.1 DESCRIPCIÓN DEL ENTORNO DE PRUEBAS*

Para la realización de las pruebas, se han utilizado dos *smartphones* de última generación con Android OS instalado. Las características de ambos son las siguientes:

- *HTC SENSATION:*
  - Plataforma: Android 2.3.4 (Gingerbread)
  - Procesador: 1,2GHz Dual Core
  - Tamaño: 126 x 65 x 11
  - Peso: 148 Gr.
  - Pantalla: Táctil Capacitiva, 4,3" (540x960)
  - Acelerómetro: Si
  - GPS: Si

- *HTC DESIRE:*
  - Plataforma: Android 2.2 (Froyo)
  - Procesador: Qualcomm Snapdragon 1GHz
  - Tamaño: 119 x 60 x 12
  - Peso: 135 Gr.
  - Pantalla: Táctil Capacitiva, 3,7" (480x800)
  - Acelerómetro: Si
  - GPS: Si

## *6.2 PRUEBAS REALIZADAS*

En este apartado se explican cada una de las pruebas realizadas al sistema, para comprobar el correcto funcionamiento de la aplicación. Se realizaron cinco tipos de pruebas, de forma que se pudiera abarcar la mayor posibilidad de entornos distintos, ya que cada una está orientada a un objetivo de comprobación distinto.

### 6.2.1 Validación de la estimación de pasos

Con esta prueba se comprueba la estimación del número de pasos que realiza el sistema, demostrando que la estimación es mejor que los datos que pueden devolver el acelerómetro y el GPS por separado. Esto es importante, ya que uno de los objetivos principales (y, por tanto, la novedad de este sistema respecto a otros), es la mejora en la estimación de los pasos.

Para realizar esta prueba, se ha recurrido a los datos que se utilizaron como base de datos de entrenamiento del aprendizaje automático, así como a nuevos valores. Se han realizado 50 registros, de los que 25 son procedentes de los datos de entrenamiento, y los otros 25 son totalmente nuevos. En cada registro se incluye el número de pasos del acelerómetro, el número de pasos del GPS, el número de pasos reales y el número de pasos estimados.

El *smartphone* utilizado para esta prueba ha sido únicamente el HTC Sensation, ya que no es necesaria la participación del otro dispositivo en esta prueba.

En la Tabla 68 se muestran los resultados de calcular, a partir de los 50 registros comentados anteriormente, el error cuadrático medio y la desviación estándar del número de pasos otorgados por el podómetro, los otorgados por el GPS y los estimados aplicando la formula obtenida a través del aprendizaje automático.

| | RAIZ CUADRADA DEL ERROR CUADRATICO MEDIO | DESVIACION ESTANDAR |
|---|---|---|
| **Pasos podómetro** | 18.89 | 11.70 |
| **Pasos GPS** | 18.59 | 11.50 |
| **Pasos Estimados** | 11.50 | 4.29 |

**Tabla 68. Resultados de la prueba de aprendizaje automático**

Como se aprecia, la raíz cuadrada del error cuadrático y la desviación de los pasos obtenidos a través del podómetro y el GPS son bastantes más altos que los estimados, ya que existe una clara diferencia entre 18 pasos de error de media y los 11 que nos da la estimación. Esto demuestra que la decisión de trabajar en el aprendizaje automático para obtener unos datos más exactos ha sido la correcta.

En la Fig. 42 podemos apreciar una gráfica con los 50 registros de esta prueba, donde aquí sí que se observa de forma clara como la línea determinada por los valores estimados permanece muy pegada a la línea determinada por el número de pasos reales, mientras que las otras dos líneas (podómetro y GPS) fluctúan de una manera mucho más irregular alrededor de la real.

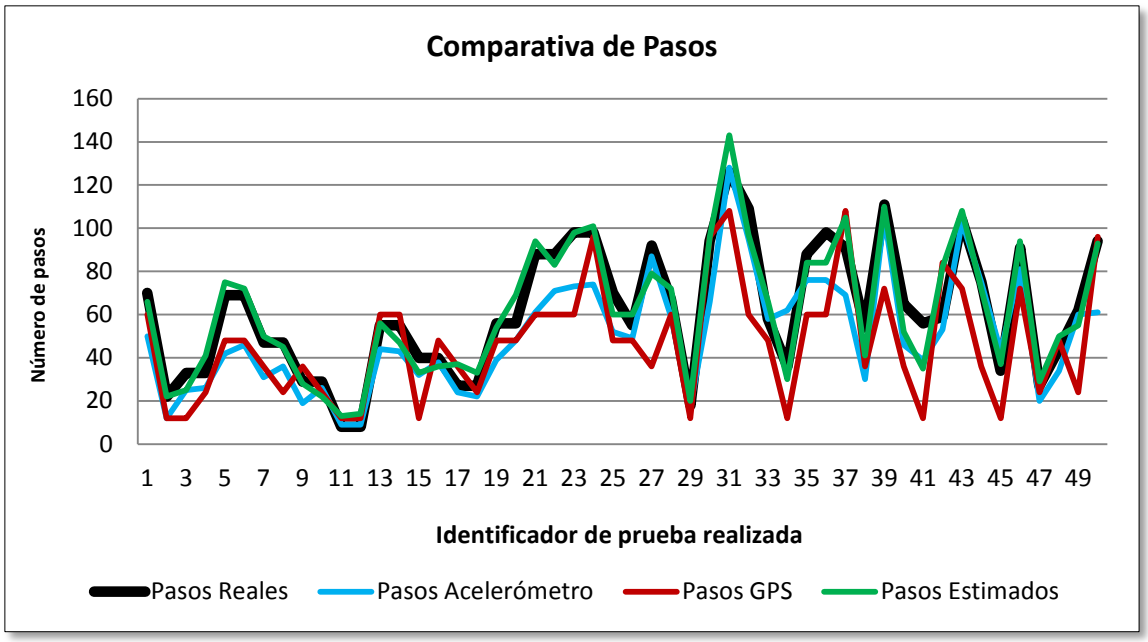



**Fig. 42. Gráfica comparativa de pasos**

### 6.2.2 Prueba de largo recorrido

Una vez comprobada la efectividad del sistema respecto al número de pasos reales en distancias más o menos cortas, se ha realizado una prueba de largo recorrido. Esta prueba, que se ha realizado sobre el HTC Sensation, ha consistido en recorrer 1000 pasos de forma continua, sin detención, variando la velocidad y el ejercicio entre correr y andar. Los resultados obtenidos de esta prueba se muestran en la Tabla 69.

| | HTC SENSATION |
|---|---|
| **Número de pasos reales:** | 1000 |
| **Número de pasos estimados:** | 1045 |
| **Tiempo:** | 00:09:34:7 |
| **Distancia:** | 0.836 km |
| **Velocidad:** | 5.24594 km/h |
| **Calorías:** | 171.7841 kcal |

**Tabla 69. Prueba de largo recorrido**

Como se aprecia en los resultados de esta prueba, estos continúan bastante ajustados respecto a la realidad, ya que el error es de un 4,5%. Esto indica que la formula no posee desviaciones con valores altos, tal y como era de esperar.

### 6.2.3 Comparativa con otra aplicación similar: CardioTrainer

En esta prueba se pretende demostrar el correcto funcionamiento del sistema respecto a otras aplicaciones. Esta aplicación, aparte de la estimación de los pasos, muestra al usuario la distancia, velocidad media, calorías quemadas y tiempo empleado, datos que podrían ser mostrados erróneamente al usuario. Por ello, se ha utilizado otra aplicación de reconocido prestigio como es CardioTrainer, para comprobar si los datos coinciden o no, y que tanto por ciento de desviación existe.

Se han realizado cuatro pruebas distintas, dependiendo de la distancia y del número de *smartphones* utilizados:

1. *Distancia larga en un mismo smartphone a la vez*

Esta prueba se ha realizado únicamente en el HTC Sensation ejecutando a la vez las dos aplicaciones, y ha consistido en recorrer durante alrededor de veinte minutos una distancia determinada, variando la velocidad y el ejercicio entre correr y andar. La Tabla 70 muestra una comparativa entre las dos aplicaciones:

| | STEPCALORIES | CARDIOTRAINER |
|---|---|---|
| **Numero de pasos reales:** | 550 | |
| **Número de pasos estimados:** | 906 | 537 |
| **Tiempo:** | 00:20:03:7 | 00:19:45:0 |
| **Distancia:** | 0.7248 km | 0.43 km |
| **Velocidad:** | 8.57751 km/h | 1.888 km/h |
| **Calorías:** | 91.0872 kcal | 29 kcal |

**Tabla 70. Distancia larga en un mismo smartphone a la vez**

Como se aprecia en los datos, existe una diferencia importante de casi el doble de pasos, lo que marca como erróneos todos los datos dependientes de este, como es la distancia, velocidad y calorías. Una de las posibles respuestas a esta diferencia podría ser que CardioTrainer se detuvo durante algún tiempo (esto se puede comprobar en la diferencia de tiempo), pero este tiempo no es tan destacable como la diferencia de pasos. Puede que viendo el resto de pruebas comparativas, se obtenga una solución más exacta acerca de esto.

2. *Distancia corta en un mismo smartphone a la vez*

Esta prueba se ha realizado en un único *smartphone*, el HTC Sensation, y ejecutando a la vez de nuevo, pero esta vez varia la distancia, para comprobar si los resultados de la prueba anterior son debidos a un fallo casual, o realmente poseen algo de lógica. Al igual que antes, también se ha variado la velocidad y el ejercicio entre correr y andar. La Tabla 71 muestra una comparativa entre las dos aplicaciones:

| | STEPCALORIES | CARDIOTRAINER |
|---|---|---|
| **Numero de pasos reales:** | 300 | |
| **Número de pasos estimados:** | 445 | 288 |
| **Tiempo:** | 00:02:23:1 | 00:02:19:0 |
| **Distancia:** | 0.35599 km | 0.23 km |
| **Velocidad:** | 8.96700 km/h | 6.000 km/h |
| **Calorías:** | 42.09411 kcal | 16 kcal |

**Tabla 71. Distancia corta en un mismo smartphone a la vez**

De nuevo, al igual que en la prueba anterior, el número de pasos vuelve a ser más o menos el doble en StepCalories que en CardioTrainer. Esto es realmente curioso, ya que no es una variación aleatoria, sino que siempre es de forma más o menos exacta, el número de pasos en las dos pruebas realizadas hasta ahora. Podría ser algo relacionado con la ejecución en el mismo momento, por lo que se realizarán dos pruebas más, las cuales están orientadas a comprobar esto.

*3. Distancia corta en el mismo smartphone en distinto momento*

Esta prueba vuelve a ejecutar las dos aplicaciones en el HTC Sensation, pero en vez de ejecutarlas a la vez, en este caso ha sido una a continuación de la otra, para comprobar lo comentado en el punto anterior. Se han realizado exactamente quinientos cincuenta pasos (alrededor de cinco minutos), variando la velocidad y el ejercicio entre correr y andar. La Tabla 72 muestra una comparativa entre las dos aplicaciones:

| | STEPCALORIES | CARDIOTRAINER |
|---|---|---|
| **Número de pasos reales:** | 550 | |
| **Número de pasos estimados:** | 561 | 587 |
| **Tiempo:** | 00:04:58:4 | 00:05:00:0 |
| **Distancia:** | 0.4488 km | 0.47 km |
| **Velocidad:** | 5.45101 km/h | 5.642 km/h |
| **Calorías:** | 88.75163 kcal | 38 kcal |

**Tabla 72. Distancia corta en el mismo smartphone en distinto momento**

A partir de los datos de la tabla, se aprecia que lo que se suponía era cierto: al ejecutar a la vez las dos aplicaciones, el receptor GPS debe enviar la señal a StepCalories cada vez que detecta movimiento en CardioTrainer y en StepCalories, lo que hace que el número de pasos ejecutando a la vez sea de aproximadamente el doble.

Resulta la duda, si se observan los datos de ambas aplicaciones respecto a los datos reales, vemos que el número de pasos en StepCalories (561) es bastante más cercano que en CardioTrainer (587), respecto al número de pasos reales (550). Además las calorías son mucho más reales en StepCalories que en CardioTrainer, debido a las formulas usadas para este cálculo, y que han sido explicadas en apartados anteriores.

### 6.2.4 Ejecución en distintos smartphones

Esta prueba se ha realizado ejecutando a la vez la aplicación *StepCalories* en el HTC Desire y en el HTC Sensation. Esta prueba es importante para comprobar cómo se desenvuelve la aplicación en distintos *smartphones*. Ha consistido en recorrer durante siete minutos una distancia determinada, variando la velocidad y el ejercicio entre correr y andar. La Tabla 73 muestra una comparativa entre los dos *smartphones*:

| | HTC SENSATION | HTC DESIRE |
|---|---|---|
| **Numero de pasos reales:** | 600 | |
| **Número de pasos estimados:** | 609 | 348 |
| **Tiempo:** | 00:07:06:5 | 00:07:02:6 |
| **Distancia:** | 0.4872 km | 0.2784 km |
| **Velocidad:** | 4.13953 km/h | 2.38118 km/h |
| **Calorías:** | 128.77641 kcal | 127.9254 kcal |

**Tabla 73. Ejecución en distintos smartphones con la misma sensibilidad**

Los resultados muestran datos algo desiguales entre los dos *smartphone*. Mientras que en el HTC Sensation los datos son más cercanos a los reales, en el HTC Desire están algo más alejados a los reales, con unos valores realmente menores. Si nos fijamos en el tamaño y peso de ambos *smartphone*, vemos que la HTC Desire es más gruesa, lo que hace que la detección del paso sea algo más difícil si el dispositivo va introducido en un bolsillo.

Esta diferencia es salvable con una de las características que incluye el sistema, y que precisamente estaba diseñada para estos casos: la sensibilidad. Si se varía la sensibilidad, la detección del paso será mayor, y por lo tanto equiparable. Por ello, se ha vuelto a realizar la prueba, pero esta vez el tiempo es menor y la sensibilidad del HTC Desire se encuentra en valor "Muy Alto". Los resultados se muestran en la Tabla 74.

| | HTC SENSATION | HTC DESIRE |
|---|---|---|
| **Numero de pasos reales:** | 500 | |
| **Número de pasos estimados:** | 485 | 562 |
| **Tiempo:** | 00:04:49:9 | 00:04:48:1 |
| **Distancia:** | 0.388 km | 0.4496 km |
| **Velocidad:** | 4.85505 km/h | 5.62195 km/h |
| **Calorías:** | 87.44153 kcal | 85.50231 kcal |

**Tabla 74. Ejecución en distintos smartphones con distinta sensibilidad**

Con estos últimos datos, se comprueba que la variación de la sensibilidad afecta de forma notable a la estimación. En este caso habría hecho una sobrevaloración, por lo que es lógico pensar que, entonces, su sensibilidad óptima sería en el valor "Alto". Por ello, se aconseja a todos los usuarios una primera prueba antes de comenzar a usar el sistema, con el fin de establecer la sensibilidad óptima.

### 6.2.5 Prueba con usuarios finales

Esta última prueba consiste en evaluar la experiencia del usuario. Para ello, se ha escogido a cinco personas de distintas edades y conocimientos para que utilicen la aplicación en uno de los *smartphone* que se dispone para las pruebas, durante un tiempo que variará entre diez y veinte minutos. En ese tiempo, podrán realizar con el sistema lo que deseen, ya sea utilizarlo para realizar *stepping*, manejar la interfaz, cambiar opciones, etc.

A estos usuarios se les ha entregado previamente el manual de usuario para que conozcan el funcionamiento de la aplicación. En caso de duda deberán solventarlo por ellos mismos sin precisar cualquier tipo de atención del equipo de desarrollo, para probar que el manual de usuario y las preguntas frecuentes incluidas son material suficiente para el manejo de este sistema.

Una vez finalizada la prueba, se le entrega a cada uno un cuestionario para que realicen una evaluación del sistema, cuyos cuestionarios se encuentran en la Tabla 75, Tabla 76, Tabla 77, Tabla 78 y Tabla 79. Cuando los cuestionarios han sido finalizados y entregados, finaliza esta prueba.

| **CUESTIONARIO DE PRUEBA STEPCALORIES** | |
|---|---|
| **Nombre** | María Dolores Sanchez Galán |
| **Smartphone utilizado** | HTC Desire |
| **Duración de la prueba** | 15 minutos |
| **Calidad de la Interfaz** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Navegabilidad** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Facilidad de uso** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Utilidad del sistema** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Puntos positivos** | - Facilidad de uso.<br>- Interfaz muy cuidada, teniendo en cuenta todos los detalles. |
| **Puntos negativos** | Ninguno. |
| **Cosas a añadir** | - Que se pudieran subir los datos a Facebook.<br>- Menú de ayuda en el propio sistema como un PDF o similar. |

**Tabla 75. Cuestionario María Dolores Sanchez**

| CUESTIONARIO DE PRUEBA STEPCALORIES | |
|---|---|
| **Nombre** | Tania Marques Pérez |
| **Smartphone utilizado** | HTC Sensation |
| **Duración de la prueba** | 20 minutos |
| **Calidad de la Interfaz** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Navegabilidad** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Facilidad de uso** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Utilidad del sistema** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Puntos positivos** | - Fácil acceso a la actividad, permitiendo incluso pararla y reiniciarla.<br>- Aporta una herramienta para la salud en un dispositivo de uso común y constante. |
| **Puntos negativos** | - Posible imprecisión debido a impactos ajenos a un paso. |
| **Cosas a añadir** | - Visualización de la evolución a partir de los resultados guardados en el historial.<br>- Añadir voz al programa, para que según se está ejecutando un stepping, vaya diciendo al usuario las calorías quemadas, distancia recorrida, número de pasos, etc.<br>- Crear un modo a pantalla completa donde se muestre el cronometro con el número de pasos en grande, a modo de fondo de pantalla cuando se bloquea el dispositivo. |

**Tabla 76. Cuestionario Tania Marques**

| CUESTIONARIO DE PRUEBA STEPCALORIES | |
|---|---|
| **Nombre** | Rafael Gálvez Zúñiga |
| **Smartphone utilizado** | HTC Desire |
| **Duración de la prueba** | 20 minutos |
| **Calidad de la Interfaz** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Navegabilidad** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Facilidad de uso** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Utilidad del sistema** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Puntos positivos** | - La diferenciación entre mayúsculas y minúsculas en la pantalla de perfil.<br>- El funcionamiento general de la aplicación, rápido y sencillo. |
| **Puntos negativos** | - No poder eliminar un perfil de usuario individualmente.<br>- No poder eliminar un único historial de usuario, sino que se deben borrar todos. |
| **Cosas a añadir** | - La coordinación entre la ejecución del stepping y los pasos para medir la duración del ejercicio. Se podría añadir una opción en la que el inicio, parada, reinicio y fin del stepping ocurriera automáticamente a partir de los datos recibidos por el podómetro. |

**Tabla 77. Cuestionario Rafael Gálvez**

| CUESTIONARIO DE PRUEBA STEPCALORIES | |
|---|---|
| **Nombre** | Javier Martín Cabezuelo |
| **Smartphone utilizado** | HTC Sensation |
| **Duración de la prueba** | 20 minutos |
| **Calidad de la Interfaz** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Navegabilidad** | [ ]Muy buena [ ]Buena [X]Normal [ ]Mala [ ] Muy mala |
| **Facilidad de uso** | [ ]Muy buena [ ]Buena [X]Normal [ ]Mala [ ] Muy mala |
| **Utilidad del sistema** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Puntos positivos** | - Selección correcta de los nombres de los menús, tanto el principal como los demás.<br>- Inclusión del idioma en inglés para usuarios que no entiendan castellano. |
| **Puntos negativos** | - Al final no se ha desarrollado un mapa donde se muestre el avance del usuario. |
| **Cosas a añadir** | - La inclusión del mapa comentado en el apartado de puntos negativos. |

**Tabla 78. Cuestionario Javier Martín**

| CUESTIONARIO DE PRUEBA STEPCALORIES | |
|---|---|
| **Nombre** | Sara Baños Lozano |
| **Smartphone utilizado** | HTC Sensation |
| **Duración de la prueba** | 10 minutos |
| **Calidad de la Interfaz** | [X]Muy buena [ ]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Navegabilidad** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Facilidad de uso** | [ ]Muy buena [ ]Buena [X]Normal [ ]Mala [ ] Muy mala |
| **Utilidad del sistema** | [ ]Muy buena [X]Buena [ ]Normal [ ]Mala [ ] Muy mala |
| **Puntos positivos** | - Posibilidad de saber las calorías que quemas a lo largo del día.<br>- La aplicación es para Android |
| **Puntos negativos** | - Que se pudiera borrar un solo historial y no todos a la vez. |
| **Cosas a añadir** | Ninguna. |

**Tabla 79. Cuestionario Sara Baños**

## *6.3 RESUMEN DE EVALUACIÓN*

Una vez realizadas las pruebas, se ha llegado a una serie de conclusiones, entre las que se encuentran las siguientes:

- La aplicación realiza una estimación de los pasos realizados por el usuario bastante ajustado. La media de errores de 11 pasos en la primera prueba y de 45 pasos (4,5% de error) en la segunda prueba reafirman esta sentencia.

- No existe desviación de la fórmula de aprendizaje automático en valores altos de pasos, tal y como se ha comprobado en la segunda prueba.

- En comparación a otros sistemas parecidos ya existentes en el mercado, éste realiza una estimación algo mejor respecto al número de pasos, y bastante mejor respecto al gasto calórico, como demuestran las pruebas de la tercera sección.

- La ejecución de esta aplicación en el mismo momento que se está ejecutando otra con características similares, parece que hace que se duplique el número de pasos contabilizados, pero es algo que no se puede asegurar al cien por cien.

- La opción de variar la sensibilidad, aparecida en este sistema en un principio como algo que no podría tener demasiado uso ha demostrado todo lo contrario en la cuarta prueba. Dependiendo de las características del dispositivo, se necesitará mayor o menor sensibilidad a la hora de ejecutar esta aplicación adecuadamente.

- La opinión de los usuarios finales que han probado ya esta versión es bastante buena, sobre todo en cuanto a la calidad de la interfaz y su uso, tal y como reflejan los resultados de las encuestas mostradas en la última sección de pruebas.

- Aquellas características que, bajo su opinión, podrían haberse incluido ya en este sistema, se tomarán en cuenta para una futura versión de esta aplicación.

# 7. CONCLUSIONES

En este Proyecto Fin de Carrera se ha creado para el sistema operativo Android una aplicación dedicada a la vida saludable, la cual consiste en un estimador de pasos gracias a una formula obtenida a partir de un algoritmo de regresión basado en aprendizaje automático, que usa valores que provienen de los sensores de movimiento y GPS con los que cuenta cualquier *smartphone* del mercado. A partir de los pasos estimados, el sistema es capaz de determinar la distancia recorrida por el usuario, la velocidad media y las calorías quemadas, que se calculan a partir de diversas ecuaciones basadas en el metabolismo basal del ser humano, dando unos valores específicos al usuario dependiendo de sus características básicas, como son la altura, peso, edad o sexo.

La aplicación, desde el principio de su desarrollo, se determinó que usaría el patrón de arquitectura Modelo Vista Controlador, debido a que éste era el patrón más adecuado tras chequear aquello que nos pedía el usuario. Esta arquitectura nos ha permitido separar la lógica del sistema de las interfaces, algo que era necesario debido a la obligatoria utilización de varias pantallas para la correcta creación de la aplicación, ya que Android trabaja a partir de interfaces.

Todo el desarrollo se ha realizado bajo el lenguaje de programación Java, que es el lenguaje de alto nivel en el que se deben crear las aplicaciones para Android. De este lenguaje se ha intentado explotar todas y cada una de las características que nos otorga: el desarrollo por paquetes nos permitió continuar con la arquitectura diseñada anteriormente a la creación; se han usado las interfaces para notificar al resto de clases de la existencia de un paso; los listeners creados han permitido avisar a la interfaz correspondiente en caso de que hubiera un paso.

Se ha estudiado el modo de desarrollar aplicaciones para Android, de forma que se pudiera sacar todo el partido a las API que otorga el sistema operativo. Por ello, para la creación de esta aplicación se han utilizado *activities* para la creación de interfaces, *services* para la ejecución en segundo plano, *intents* para la navegación, *managers* para obtener los valores que devuelven los sensores existentes en el dispositivo, SQLite para crear la completa base de datos que se incluye en el sistema, etc.

También se ha utilizado Weka, ya que esta aplicación, junto a sus clasificadores y algoritmos de regresión basados en aprendizaje automático, ha permitido realizar la integración entre los valores obtenidos por el receptor GPS y el acelerómetro del *smartphone* a través de una formula.

A partir de las tablas creadas por una comunidad dedicada a contabilizar el número de calorías quemadas dependiendo del tipo e intensidad del ejercicio, y de trabajos e investigaciones realizadas durante años por científicos y médicos tales como J.A. Harris o F.G. Benedict, se han empleado varias fórmulas dirigidas a calcular el número de calorías quemadas por cada usuario, dependiendo de su velocidad a la hora de caminar.

Gracias a las pruebas realizadas, se ha comprobado que todo el trabajo realizado hasta ese momento funcionaba correctamente, ya que los errores y desviaciones obtenidos en todas y cada una de ellas es mínimo, llegando a ser incluso menor que otros sistemas existentes actualmente en el mercado. También se ha demostrado que la opción de poder variar la sensibilidad es realmente útil, ya que dependiendo del dispositivo, se debe variar el valor de la misma. Además, se ha visto que los usuarios finales a los que se les dejo probar el sistema, están realmente satisfechos con el resultado.

Por último, y en relación con el objetivo principal del proyecto, remarcar que este trabajo aporta como novedad dentro del ámbito de aplicaciones en el que se encuentra (herramientas dedicadas a una vida saludable del usuario), la utilización del aprendizaje automático y la regresión para la estimación de pasos, además de realizar un cálculo de las calorías quemadas por velocidad de paso mucho más personalizado para el usuario.

# 8. LÍNEAS FUTURAS

En este capítulo se planten líneas de desarrollo que pueden ser estudiadas y llevadas a cabo en un futuro. Entre estas futuras líneas, se proponen las siguientes:

- *Aumento del número de actividades*: El sistema está preparado para el cálculo de calorías quemadas siempre y cuando el usuario use esta aplicación a la hora de andar o correr. Sin embargo, y debido a que toma los datos de la distancia en el exterior a partir del GPS, esta aplicación se podría usar para realizar otras actividades como ciclismo, patinaje, ski, etc. Además habría una manera muy sencilla de realizarlo, ya que todos los datos de las actividades se encuentran juntos en una única tabla, por lo que solo sería añadirle valores y pedirle al usuario que seleccione un tipo de ejercicio.

- *Mayor nivel de idiomas*: Como hemos comentado en el punto 5.4.3, en la aplicación se han desarrollado únicamente dos idiomas, castellano por ser la lengua del país de desarrollo, e inglés por ser hoy en día el idiomas más hablado. Sin embargo, tal y como se comentó, el número de idiomas podría aumentarse debido a la facilidad para introducir un nuevo idioma. Solo habría que insertar un nuevo archivo XML con los textos en ese idioma e introducir en el código una nueva condición por este idioma.

- *Mejorar la localización*: El sistema toma la localización a partir de los datos que nos otorga el GPS, que como se ha dicho anteriormente, contiene una variación de entre 20 y 30 metros, por lo que puede dar valores más o menos erróneos (aunque el error que otorga actualmente no es muy grande). Por ello, una de las posibilidades sería la de estudiar una mejora en el posicionamiento, investigando nuevas formas como podría ser el obtener los datos a partir del acceso a internet del *smartphone*.

- *Expansión a internet y unión a las redes sociales*: Algo que se echa en falta en este sistema es algún tipo de conexión con internet. Cualquier aplicación actual contiene algún tipo de interacción con la web, y en este caso no debería ser menos. Un ejemplo de esto podría ser la inclusión del sistema a las redes sociales tales como Facebook o Twitter. Para este sistema sería un gran adelanto que una de las cosas a compartir en estas redes sociales fueran cada uno de los Stepping realizados por el usuario, de forma que al finalizar el entrenamiento, este fuera compartido con el resto de usuarios. Otra opción podría ser una web donde cada usuario dejara sus resultados y los pudiera comparar con otros realizados por el mismo anteriormente, o incluso compartir con otros usuarios, además de crear rankings, etc.

- *Recalibración del sistema*: Actualmente, el sistema utiliza fórmulas de aprendizaje automático para realizar una estimación del número de pasos. Para un futuro, se podría aumentar la base de conocimiento para un mejor ajuste, o incluso realizar un ajuste o calibración óptimo para cada usuario, a partir de sus propios datos.

- *Mayor número de datos recogidos*: Los datos recogidos por la aplicación incluyen pasos, distancia, velocidad, calorías y tiempo. En una posible futura mejora, se podrían incluir valores tales como la media de tiempo por kilómetro (paso por kilómetro), la sensación térmica para una mayor precisión del gasto calórico (se encontraba como requisito opcional en el sistema), asociarse con algún pulsómetro, etc. Esto daría al usuario más datos acerca del ejercicio, algo que mejoraría la calidad del sistema.

# GLOSARIO DE TÉRMINOS

- ***Stepping***: Concepto creado para este sistema que consiste en el entrenamiento o ejercicio realizado por el usuario, el cual queda recogido y almacenado siempre que lo desee. Puede detenerse y reiniciarse, y entre sus atributos cuenta con la velocidad, el número de pasos, la distancia, el número de calorías y el tiempo empleado.

- ***Usuario***: Individuo que utiliza el dispositivo y realiza las múltiples operaciones del sistema con distintos propósitos.

- ***Hardware***: Conjunto de los componentes que conforman la parte material (física) de una computadora, incluyendo los componentes físicos internos y los periféricos.

- ***Software***: Soporte lógico e inmaterial que permite que la computadora pueda desempeñar tareas inteligentes, dirigiendo a los componentes físicos o hardware con instrucciones y datos a través de diferentes tipos de programas.

- ***Sistema operativo***: Se denomina así al conjunto de programas informáticos que permite la administración eficaz de los recursos de una computadora, gestionando el hardware desde los niveles más básicos y permitiendo además la interacción con el usuario.

- ***Smartphone***: Pequeño dispositivo que integra las funcionalidades de un teléfono móvil con las funcionalidades más comunes de una PDA, como almacenar información, enviar y recibir mensajes e E-mail o instalar programas.

- ***GPS***: Del inglés "Global Positioning System", se traduce al castellano como el Sistema Global de Posicionamiento, el cual nos permite fijar a escala mundial la posición de un objeto, una persona, un vehículo o una nave a través del trabajo conjunto de varios satélites.

- ***Acelerómetro***: Dispositivo insertado en el dispositivo que mide la aceleración y las fuerzas inducidas por la gravedad en cada una de las tres dimensiones espaciales.

- ***Notificación***: Alerta que emite el sistema operativo y ciertos programas o servicios para advertir de algo al usuario.

- ***Interfaz***: Conjunto de protocolos y técnicas para lograr interactividad e intercambio de información entre un usuario y una aplicación computacional.

- ***Paquete***: En Java se describe este concepto como una forma de organizar grupos de clases. Un paquete contiene un conjunto de clases relacionadas bien por finalidad, por ámbito o por herencia.

- ***UML***: Del inglés "Unified Modeling Language", se traduce al castellano como el Lenguaje Unificado de Modelado, y sirve para construir, documentar, visualizar y especificar un sistema de software.

- ***Activity***: En Android es el concepto más básico, y se detalla como aquello que muestra al usuario una interfaz gráfica.

- ***Service***: En Android se describe como las clases que se ejecutan en segundo plano que son manejadas por las aplicaciones en vez de por los usuarios.

- ***XML***: Del inglés "Extensible Markup Language", se traduce al castellano como el Lenguaje de Etiquetado Extensible, y se trata de un metalenguaje extensible de etiquetas que fue desarrollado por el "Word Wide Web Consortium" (W3C), un consorcio internacional que elabora recomendaciones para la "World Wide Web" (WWW).

- ***Timer***: Componente de Android que realiza tareas cada cierto tiempo, determinado en el código de la aplicación.

- ***Estimador***: Cualquier variable aleatoria que se defina a partir de la sucesión de variables aleatorias, que integran una muestra de tamaño "n" extraída al azar de una población, es decir, toma un valor para cada "n" observaciones o datos.

- ***Array***: Colección ordenada de elementos de un mismo tipo de datos, agrupados de forma consecutiva en memoria. Cada elemento del array tiene asociado un índice, que no es más que un número natural que lo identifica inequívocamente y permite al programador acceder a él.

- ***API***: Del inglés "Application Programming Interface", se traduce al castellano como la Interfaz de Programación de Aplicaciones, y está formada por el conjunto de procedimientos que ofrece un determinado componente para poder ser utilizado como una capa de abstracción.

- ***Plug-in***: Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande, del cual depende (no puede existir este módulo sin el sistema superior).

# BIBLIOGRAFÍA

[1] Vogel, L. Android Development Tutorial – Gingerbread. http://www.vogella.de/articles/Android/article.html#first_project. Accedido en Agosto 2011.

[2] Introducción a Android – Entorno Eclipse. http://www.demalagana.es/tecnologia/tutoriales/introduccion-a-android-entorno-eclipse. Accedido en Enero 2009.

[3] Android Developer. http://developer.android.com/index.html. Accedido en Agosto 2011.

[4] Bagilevi. Pedometer – Android App. https://github.com/bagilevi/android-pedometer. Accedido en Diciembre 2010.

[5] Wei-Meng, L. Using Google Maps in Android. http://www.devx.com/wireless/Article/39145/0/page/1. Accedido en Septiembre 2008.

[6] Antonio Meucci. http://es.wikipedia.org/wiki/Antonio_Meucci. Accedido en Agosto 2011.

[7] Alexander Grahan Bell. http://es.wikipedia.org/wiki/Alexander_Graham_Bell. Accedido en Agosto 2011.

[8] Smartphone. http://es.wikipedia.org/wiki/Smartphone. Accedido en Septiembre 2011.

[9] Sistema Embebido. http://es.wikipedia.org/wiki/Sistema_embebido. Accedido en Septiembre 2011.

[10] Encuesta Sobre Habitos Deportivos de los Españoles. http://www.kirolan.org/El%20sector%20del%20empleo%20deportivo/encuesta%20habitos%20deportivos.pdf. Accedido en Abril 2005.

[11] Configurador Nutricional Burger King. http://www.burgerking.es/home.htm#nutricion/configurador. Accedido en 2010.

[12] Natur House.
http://www.naturhouse.com/. Accedido en Septiembre 2011.

[13] Ingeniería del Software.
http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software. Accedido en Septiembre 2011.

[14] Agencia Espacial Internacional.
http://www.esa.int/esaCP/index.html. Accedido en Septiembre 2011.

[15] Estándar de Ingeniería del Software de ESA.
ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/PSS050.pdf. Accedido en Febrero 1991.

[16] Winston W. Royce (1970). Managing the development of large software systems.
http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf. Accedido en Agosto 1970.

[17] Modelo de Proceso o Ciclo de Vida.
http://es.wikipedia.org/wiki/Software#Modelos_de_proceso_o_ciclo_de_vida . Accedido en Septiembre 2011.

[18] Matriz de Trazabilidad.
http://www.softqatest.net/?p=77. Accedido en Junio 2010.

[19] Historia de Android.
http://kronox.org/2009/06/09/historia-de-android/. Accedido en Junio 2009.

[20] Andy Rubin.
http://es.wikipedia.org/wiki/Andy_Rubin. Accedido en Agosto 2011.

[21] Rich Miner.
http://en.wikipedia.org/wiki/Rich_Miner. Accedido en Enero 2011.

[22] Open Handset Alliance.
http://en.wikipedia.org/wiki/Open_Handset_Alliance. Accedido en Septiembre 2011.

[23] OpenGL ES SDK.
http://www.khronos.org/opengles/sdk/. Accedido en 2010.

[24] SQLite.
http://es.wikipedia.org/wiki/SQLite. Accedido en Agosto 2011.

[25] Acid3.
http://en.wikipedia.org/wiki/Acid3. Accedido en Agosto 2011.

[26] Dalvik (Maquina Virtual).
http://en.wikipedia.org/wiki/Dalvik_virtual_machine. Accedido en Septiembre 2011.

[27] PDANet para Android.
http://junefabrics.com/android/index.php. Accedido en Septiembre 2011.

[28] Google Talk.
http://es.wikipedia.org/wiki/Google_Talk. Accedido en Agosto 2011.

[29] Android.
http://www.android.com/. Accedido en Septiembre 2011.

[30] Windows Phone.
http://www.microsoft.com/windowsphone/es-es/default.aspx?cmpid=MSCOM_ES_HP_Nav_Todos. Accedido en Septiembre 2011.

[31] Symbian.
http://symbian.nokia.com/. Accedido en Septiembre 2011.

[32] iOS 4.
http://www.apple.com/es/iphone/ios4/. Accedido en Septiembre 2011.

[33] Comparativa: Windows7 vs iOS, Android, Symbian y Maemo.
http://moviltoday.com/comparativa-windows-phone-7-vs-ios-android-symbian-y-maemo/. Accedido en Noviembre 2010.

[34] Android, Windows Phone 7, IOS y Blackberry en 2015.
http://www.configurarequipos.com/actualidad-informatica/3555/android-windows-phone-7-ios-y-blackberry-en-2015. Accedido en Abril 2011.

[35] Gartner Consulting.
http://www.gartner.com/technology/consulting/. Accedido en Septiembre 2011.

[36] Cardio Trainer.
http://www.worksmartlabs.com/cardiotrainer/. Accedido en Septiembre 2011.

[37] RunKeeper.
http://runkeeper.com/. Accedido en Septiembre 2011.

[38] AndAndo.
http://andando.javielinux.com/. Accedido en Septiembre 2011.

[39] Moveo Pedometer.
http://www.androidfreeware.net/download-moveo-pedometer.html.
Accedido en Junio 2010.

[40] Nike+.
http://nikerunning.nike.com/nikeos/p/nikeplus/es_ES/plus/#//dashboard/.
Accedido en Septiembre 2011.

[41] Nike+iPod.
http://en.wikipedia.org/wiki/Nike%2BiPod. Accedido en Julio 2011.

[42] Nike+ Sportsband.
http://nikerunning.nike.com/nikeos/p/nikeplus/es_AR/products/sportband.
Accedido en Septiembre 2011.

[43] Transmisor Polar Wearlink+.
http://www.polariberica.es/es/productos/accesorios/transmisor_wearlinkplus. Accedido en Septiembre 2011.

[44] Tasa Metabólica en Reposo.
http://en.wikipedia.org/wiki/Basal_metabolic_rate. Accedido en Agosto 2011.

[45] Harris, JA., Benedict, FG. (1919). Biometric studies of basal metabolism in man: Washington DC. Carnegie Institute of Washington, publication 279.

[46] Metabolismo Equivalente.
http://en.wikipedia.org/wiki/Metabolic_equivalent. Accedido en Agosto 2011.

[47] Compendio de Actividades Físicas.
https://sites.google.com/site/compendiumofphysicalactivities/Activity-Categories. Accedido en 2011.

[48] Byrne NM, Hills AP, Hunter GR, Weinsier RL, Schutz Y (2005). Metabolic equivalent: one size does not fit all.

[49] Kozey S, Lyden K, Staudenmayer J, Freedson P (2010). Errors in MET estimates of physical activities using 3.5 ml $O_2 \cdot kg^{-1} \cdot min^{-1}$ as the baseline oxygen consumption.

[50] Ecuación de Harris-Benedict.
http://es.wikipedia.org/wiki/Ecuaci%C3%B3n_de_Harris-Benedict. Accedido en Septiembre 2011.

[51] Historia del GPS.
http://homepages.mty.itesm.mx/al584299/mypaper.htm. Accedido en 2002.

[52] Efecto Doppler.
http://en.wikipedia.org/wiki/Doppler_effect. Accedido en Septiembre 2011.

[53] NAVSTAR GPS.
http://www.spaceandtech.com/spacedata/constellations/navstar-gps_consum.shtml. Accedido en 2001.

[54] La Tragedia del Vuelo 007.
http://ochentas.com.mx/2009/06/02/la-tragedia-del-vuelo-007/. Accedido en Junio 2009.

[55] GPS y el Codigo Aleatorio.
http://www.gps-data.com.ar/gps-y-codigo-aleatorio.php. Accedido en 2009.

[56] TomTom GPS Mobile.
http://www.tomtom.com/es_es/products/mobile-navigation/. Accedido en 2011.

[57] Android Community.
http://source.android.com/community/index.html. Accedido en 2011.

[58] Android SDK.
http://developer.android.com/sdk/index.html. Accedido en 2011.

[59] Frank, E., Witten, I. (2000). Data mining: practical machine learning tools and techniques with Java implementations.
http://www.cs.waikato.ac.nz/~eibe/pubs/99IHW-EF-LT-MH-GH-SJC-Tools-Java.pdf. Accedido en 2000.

[60] Modelo Vista Controlador.
http://www.neleste.com/modelo-vista-controlador/. Accedido en Julio 2008.

[61] API Android SDK.
http://developer.android.com/reference/packages.html. Accedido en 2011.

[62] UML.
http://www.uml.org/. Accedido en 2011.

[63] Biel, E. (2002). La marcha humana: biomecánica, exploraciones, normas y alteraciones.
http://books.google.es/books?id=XBDYQoJwAaAC&pg. Accedido en 2002.

[64] M. Hall, G. Holmes, E. Frank (1999). "Generating Rule Sets from Model Trees". Twelfth Australian Joint Conference on Artificial Intelligence, Sydney, Australia. Springer-Verlag, pp. 1-12.

[65] Kohavi R. (1995). *The Power of Decision Tables.* In Proc European Conference on Machine Learning.

[66] Quinlan J. R. (1992). Learning with continuous classes. Proceedings of the Australian Joint Conference on Artificial Intelligence. 343--348. World Scientific, Singapore.

[67] Wang, Y and Witten, I. H. (1997). Induction of model trees for predicting continuous classes. Proceedings of the poster papers of the European Conference on Machine Learning. University of Economics, Faculty of Informatics and Statistics, Prague.

[68] Ian H. Witten and Eibe Frank, (2005), "Data Mining: Practical Machine Learning Tools and Techniques", 2nd Edition, Morgan Kaufmann, San Francisco.

[69] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy (1999). Improvements to the SMO Algorithm for SVM Regression.

[70] Malcolm Ware (2000). WEKA Documentation. University of Waikoto.

# ANEXOS

## *ANEXO A: PLANIFICACIÓN (DIAGRAMA DE GANTT)*

En este primer anexo del proyecto se muestra la planificación que se ha ido realizando durante el desarrollo de la aplicación. Para ello se utiliza el Diagrama de Gantt, el cual muestra el tiempo de dedicación previsto y real para las diferentes tareas o actividades a lo largo del tiempo de desarrollo.

Para analizar la planificación seguida, se van a mostrar tres diagramas diferentes:

- El diagrama de la Fig. 43 muestra la planificación ideal cuando se inició el proyecto:

- El diagrama de la Fig. 44 muestra el tiempo real de dedicación en el desarrollo del sistema.

- El diagrama de la Fig. 45 muestra una comparativa entre la planificación inicial y el tiempo real empleado en el desarrollo, mostrándose la desviación temporal existente.

Tal y como se ve en los diagramas de Gantt, ha habido una pequeña desviación tanto en tiempo como en el orden de realización de cada una de las etapas. Esto es debido a que la planificación inicial estaba basada en un modelo ideal de planificación, pero esta idea inicial no fue posible, ya que se descubrió que el desarrollo sería más sencillo si se iniciaba primero el apartado de la memoria, debido a que se empezaría a tomar contacto con el lenguaje de programación y el entorno de desarrollo del proyecto.

El retraso también se ha visto influido positivamente por la capacidad de perfeccionismo del equipo de desarrollo, ya que han trabajado duramente gastando algo más del tiempo de desarrollo para obtener una interfaz atractiva y moderna para el usuario.

Por último, comentar que este desfase en el desarrollo no ha influido en el trabajo final, ya que lo que desde un principio se deseaba era finalizarlo antes del fin del año lectivo, como finalmente ha sido.

| | Nombre de tarea | Duración |
|---|---|---|
| 1 | Investigación | 14 días |
| 2 | **Análisis** | **18 días** |
| 3 | Establecimiento de requisitos | 15 días |
| 4 | Diseño de interfaces | 3 días |
| 5 | **Diseño** | **21 días** |
| 6 | Arquitectura del sistema | 5 días |
| 7 | Casos de uso | 7 días |
| 8 | Diseño de clases | 7 días |
| 9 | Verificación del sistema | 2 días |
| 10 | Codificación | 60 días |
| 11 | Pruebas | 7 días |
| 12 | **Redacción Memoria** | **129 días** |
| 13 | Diseño | 3 días |
| 14 | Introducción | 3 días |
| 15 | Estado del arte | 15 días |
| 16 | Organización del proyecto | 15 días |
| 17 | Objetivos | 5 días |
| 18 | Diseño e implementación | 50 días |
| 19 | Pruebas y resultados | 5 días |
| 20 | Conclusiones | 2 días |
| 21 | Líneas futuras | 2 días |
| 22 | Manual de usuario | 7 días |
| 23 | Glosario | 7 días |
| 24 | Anexos | 15 días |



**Fig. 43. Planificación inicial**

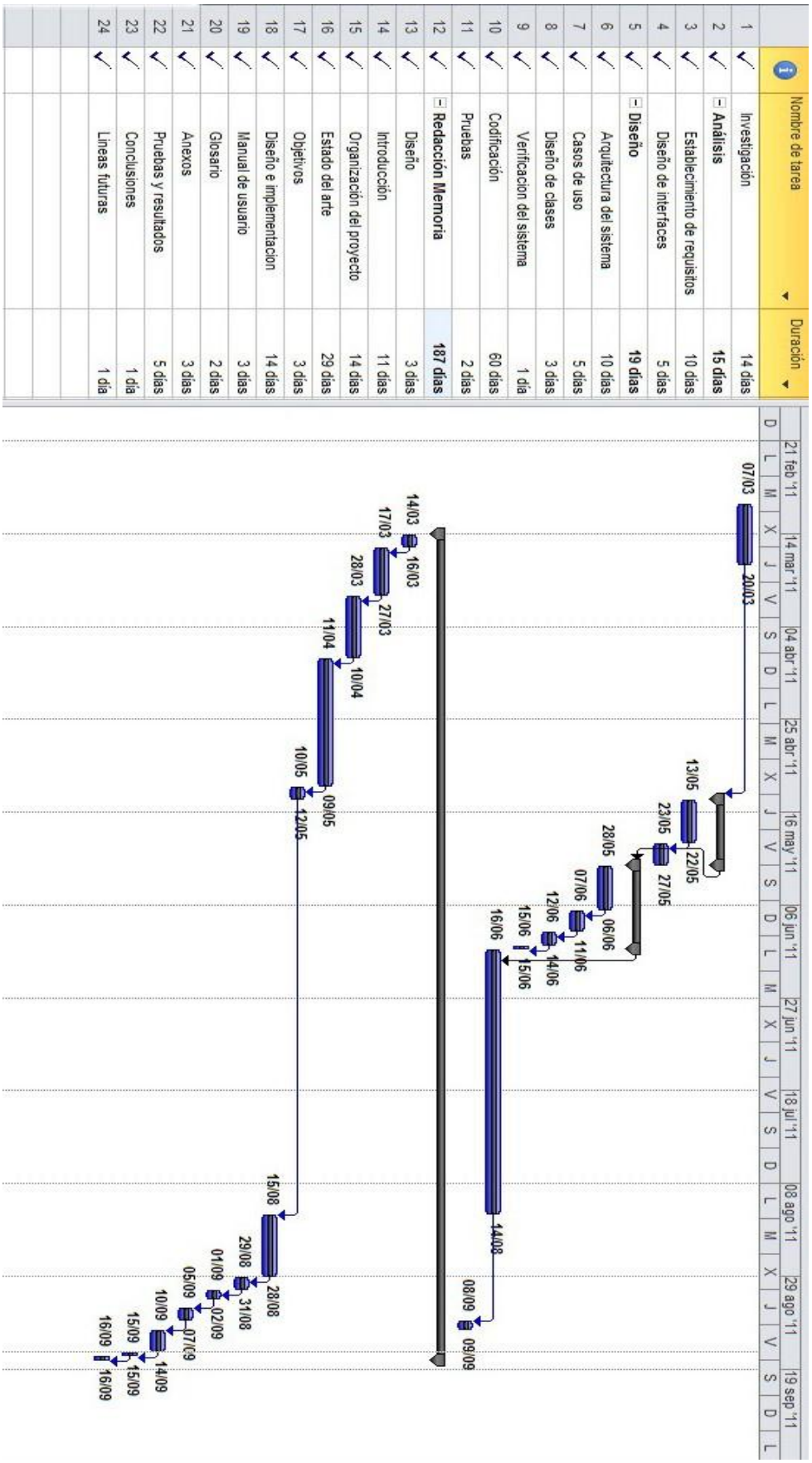
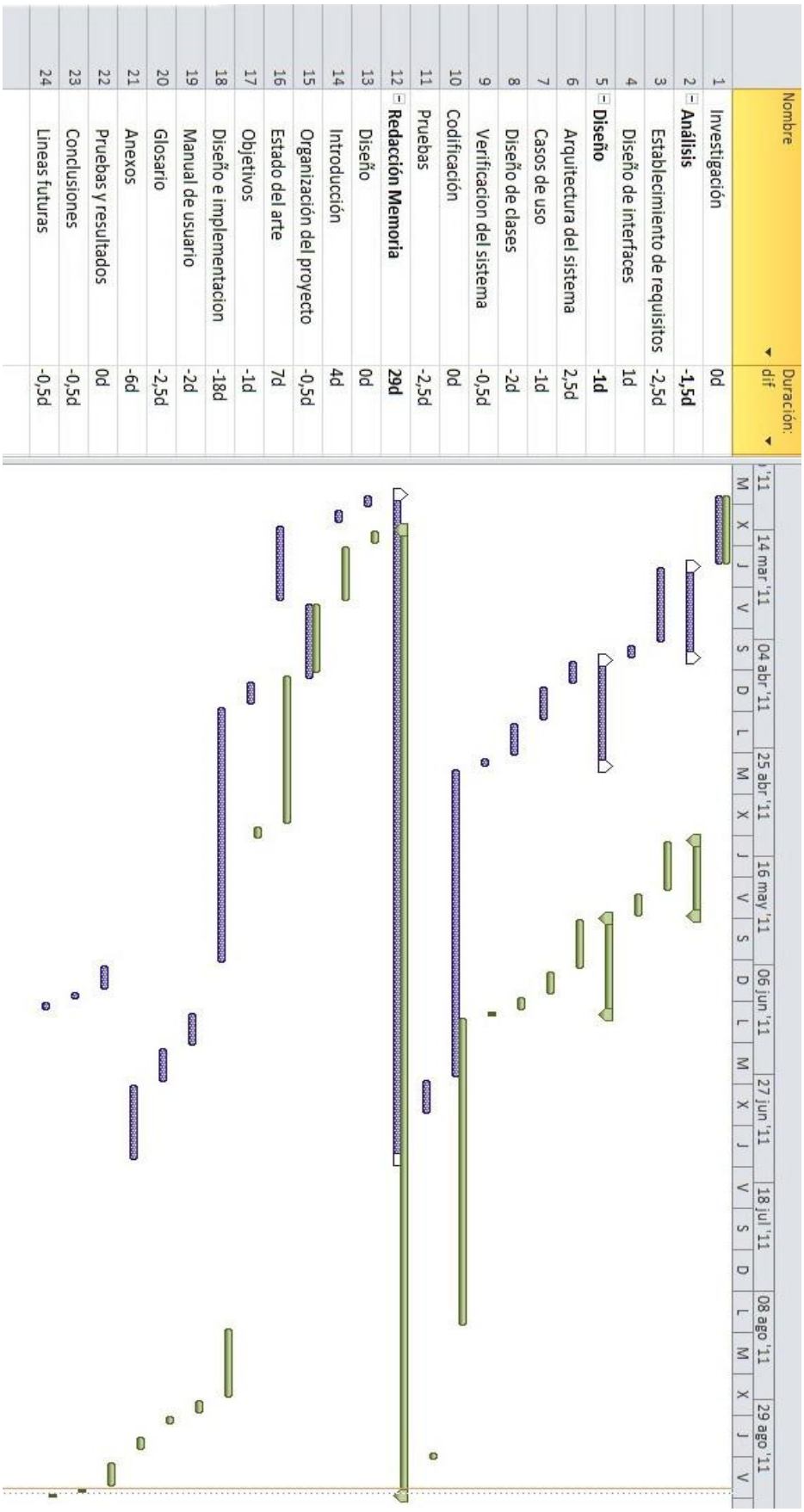



**Fig. 44. Tiempo real de desarrollo**



| | Nombre | Duración: dif |
|---|---|---|
| 1 | Investigación | 0d |
| 2 | **Análisis** | **-1,5d** |
| 3 | Establecimiento de requisitos | -2,5d |
| 4 | Diseño de interfaces | 1d |
| 5 | **Diseño** | **-1d** |
| 6 | Arquitectura del sistema | 2,5d |
| 7 | Casos de uso | -1d |
| 8 | Diseño de clases | -2d |
| 9 | Verificacion del sistema | -0,5d |
| 10 | Codificación | 0d |
| 11 | Pruebas | -2,5d |
| 12 | **Redacción Memoria** | **29d** |
| 13 | Diseño | 0d |
| 14 | Introducción | 4d |
| 15 | Organización del proyecto | -0,5d |
| 16 | Estado del arte | 7d |
| 17 | Objetivos | -1d |
| 18 | Diseño e implementacion | -18d |
| 19 | Manual de usuario | -2d |
| 20 | Glosario | -2,5d |
| 21 | Anexos | -6d |
| 22 | Pruebas y resultados | 0d |
| 23 | Conclusiones | -0,5d |
| 24 | Lineas futuras | -0,5d |



**Fig. 45. Comparativa entre planificación y tiempo real.**

# *ANEXO B: MANUAL DE USUARIO*

En este anexo se presenta un breve tutorial sobre el manejo de la aplicación, en el que se incluyen instrucciones de cómo ejecutarla y configurar los parámetros de la misma.

## 1. Requisitos mínimos

Los requisitos mínimos establecidos para el correcto funcionamiento de la aplicación en cualquier *smartphone*, son los siguientes:

- Sistema operativo Android (superior a 1.6).
- Receptor GPS.
- Acelerómetro.

## 2. Instalación

Para instalar esta aplicación en el *smartphone*, se puede realizar de dos formas:

- Descargándola de la tienda de aplicaciones de Android (Android Market). Se debe buscar la aplicación "StepCalories" en la tienda, y una vez encontrada solo se debe pulsar el botón "Descargar", y la aplicación se descargará e instalará automáticamente.

- Si posees el fichero .apk, lo que se debe hacer es introducirlo en el *smartphone* a través de la conexión a PC, y una vez este el fichero en la memoria del aparato, solo se debe buscar el fichero con un explorador de archivos del *smartphone* y hacer clic en él, instalándose automáticamente. Para instalarlo de esta forma, recordar que se debe tener activa la opción de permitir instalar aplicaciones distintas del Market en las opciones del sistema operativo.

## 3. Perfiles

Las acciones que se pueden realizar con los perfiles de usuario son las siguientes:

### *1. Crear un usuario*

Es posible crear tantos usuarios como se desee, ya que es una aplicación multiusuario. Un usuario se crea de la siguiente forma:

1. Desde el menú principal, se hace clic en el botón "Perfil".
2. Se rellena correctamente cada uno de los campos (nombre, altura, peso, edad, sexo). En caso de que el campo no sea válido, se mostrará con un fondo rojo.
3. Se hace clic en el botón "Guardar Perfil".
4. La aplicación crea el usuario y vuelve al menú principal.

### *2. Cargar un usuario*

La aplicación permite cargar perfiles de usuarios creados anteriormente en el sistema. Para realizar esto, son necesarios los siguientes pasos:

1. Desde el menú principal, se hace clic en el botón "Perfil".
2. Se hace clic en el botón "Cargar Perfil".
3. Aparece una ventana, donde se escribe el nombre del usuario a cargar.
4. Se hace clic en el botón "Aceptar".
5. La aplicación carga el perfil y vuelve al menú principal. Si el usuario no existe, se notificará al usuario para que lo introduzca de nuevo.

### *3. Actualizar un usuario*

En caso de que se desee variar algún dato de un usuario existente, es posible actualizar el perfil de la siguiente forma:

1. Desde el menú principal, se hace clic en el botón "Perfil".
2. Se pulsa el botón "Menú" del *smartphone*, apareciendo el menú de opciones de la interfaz, incluyendo el botón "Actualizar".
3. Se hace clic en el botón "Actualizar".
4. Aparece una ventana, donde se escribe el nombre del usuario a cargar.
5. Se hace clic en el botón "Aceptar".
6. La aplicación carga el perfil en la interfaz para que pueda ser editado. Si el usuario no existe, se notificará al usuario para que lo introduzca de nuevo.
7. Una vez actualizado el perfil, se hace clic en el botón "Guardar Perfil", y se guarda el perfil actualizado.

### *4. Reiniciar los valores*

Si los valores que ha introducido han sido erróneos, es posible limpiar la interfaz, para empezar de nuevo la creación o actualización de un usuario. Esto se realiza haciendo clic en el botón "Reiniciar" del menú de opciones de la interfaz, que aparece pulsando el botón "Menú" del *smartphone*.

# 4. Stepping

Stepping es definido como la actividad o entrenamiento que puede desarrollar el usuario en la aplicación. Las acciones que se pueden realizar relacionadas con este concepto son las siguientes:

### *1. Iniciar stepping*

Para iniciar un stepping, es necesario lo siguiente:

1. Desde el menú principal, se hace clic en el botón "Stepping".
2. Se hace clic en el botón "Empezar Stepping".
3. Después de esto, el entrenamiento comienza.

### *2. Finalizar stepping*

Cuando se desee finalizar el stepping, se debe realizar lo siguiente:

1. Desde el menú principal, se hace clic en el botón "Stepping".
2. Se hace clic en el botón "Finalizar Stepping".
3. El entrenamiento finaliza, volviendo al menú principal.

### *3. Pausar stepping*

En el caso de que se necesite detener el stepping (por ejemplo un semáforo), pero luego continuar con ello, existe la opción de pausar la actividad. Se realiza de la siguiente forma:

1. Desde el menú principal, se hace clic en el botón "Stepping".
2. Se pulsa el botón "Menú" del *smartphone*, apareciendo el menú de opciones de la interfaz, incluyendo el botón "Pausa".
3. Se hace clic en el botón "Pausa", deteniéndose el entrenamiento.

### *4. Continuar stepping*

Una vez que se ha pausado, es posible continuar con el stepping realizando los siguientes pasos:

1. Desde el menú principal, se hace clic en el botón "Stepping".

2. Se pulsa el botón "Menú" del *smartphone*, apareciendo el menú de opciones de la interfaz, incluyendo el botón "Continuar".
3. Se hace clic en el botón "Continuar", reanudándose el entrenamiento.

*5. Reiniciar stepping*

También es posible reiniciar la actividad, sin importar que se esté ejecutando en ese momento o esté detenida. En el caso de querer reiniciar un stepping, se debe realizar lo siguiente:

1. Desde el menú principal, se hace clic en el botón "Stepping".
2. Se pulsa el botón "Menú" del *smartphone*, apareciendo el menú de opciones de la interfaz, incluyendo el botón "Reiniciar".
3. Se hace clic en el botón "Reiniciar", reiniciándose el entrenamiento.

*6. Salir sin guardar*

Otra de las posibilidades de la aplicación consiste en salir del stepping sin almacenarlo en el historial. Para ello, se debe realizar lo siguiente:

1. Desde el menú principal, se hace clic en el botón "Stepping".
2. Se pulsa el botón "Menú" del *smartphone*, apareciendo el menú de opciones de la interfaz, incluyendo el botón "Salir sin guardar".
3. Se hace clic en el botón "Salir sin guardar", finalizando automáticamente el entrenamiento y volviendo al menú principal.

## 5. Historiales

Cada usuario tendrá sus historiales almacenados en la base de datos de la aplicación. Las acciones a realizar relacionadas con los historiales son las siguientes:

*1. Ver historiales*

Para ver todos sus historiales, únicamente hace falta hacer clic en el botón "Historiales" del menú principal de la aplicación. Aparecerá un resumen todos los stepping realizados por usted, ordenados por la fecha de realización.

*2. Ver un historial*

Si se desea consultar de forma detallada un historial de la lista, únicamente debe que hacer lo siguiente:

1. Desde el menú principal, se hace clic en el botón "Historiales".
2. Se hace clic en el historial que desea acceder.
3. La aplicación carga el historial deseado, apareciendo en pantalla la misma información que cuando se ejecutó esa actividad.

*3. Borrar los historiales*

Si se desea borrar todos los historiales realizados hasta ese momento, existe esa posibilidad. Se ejecuta siguiente estos pasos:

1. Desde el menú principal, se hace clic en el botón "Historiales".
2. Se pulsa el botón "Menú" del *smartphone*, apareciendo el menú de opciones de la interfaz, incluyendo el botón "Borrar historiales".
3. Se hace clic en el botón "Borrar Historiales", borrándose todos los historiales y volviendo al menú principal.

## 6. Opciones

La aplicación le permite configurar algunos parámetros del sistema, de forma que pueda personalizarla a su medida. Para configurar cualquiera de estas opciones, solo es necesario hacer clic en el botón "Opciones" del menú de opciones de cualquiera de las interfaces, que aparecen al pulsar el botón "Menú" del *smartphone*. Las distintas opciones de configuración de la aplicación son las siguientes:

*1. Unidades de medida*

Con esta opción básica es posible elegir entre el sistema métrico (basado en kilómetros, kilogramos…) o el sistema imperial (aquel que se basa en millas, libras…). Por defecto, el sistema tiene definido el sistema métrico.

*2. Lenguaje de la aplicación*

Con esta opción básica es posible elegir entre el castellano o el inglés. Por defecto, el sistema tiene definido el lenguaje castellano.

*3. Uso de la aplicación*

Con esta opción avanzada es posible elegir si la aplicación se va a utilizar en el exterior o en el interior. Por defecto, el sistema tiene definido su uso en el exterior.

4. *Sensibilidad del acelerómetro*

Con esta opción básica es posible elegir entre los cinco niveles de sensibilidad que posee el sistema, y que son: Muy Bajo, Bajo, Medio, Alto y Muy Alto. Estos niveles hacen que la detección del acelerómetro varíe, yendo desde una detección muy sensible (Muy Alto) a una detección menos sensible (Muy Bajo). Por defecto, el sistema tiene definida la sensibilidad con un nivel medio.

5. *Longitud de paso*

Con esta opción básica es posible introducir la longitud de paso. Por defecto, el sistema tiene definida la longitud de paso a 80 centímetros.

6. *Borrar base de datos*

Con esta opción avanzada es posible borrar toda la base de datos del sistema. Esta opción muestra al usuario un mensaje de confirmación para que, en caso de que se haya pulsado la opción sin querer, se pueda mantener la base de datos intacta.

# 7. Preguntas frecuentes

1. *¿Siempre tengo que crear un perfil de usuario?*
   Cada vez que se entra por primera vez como usuario se debe crear un perfil para obtener tus datos. En caso de que no exista ningún perfil de usuario almacenado, las calorías calculadas pueden ser erróneas o no ser calculadas.

2. *¿Qué tipo de datos puedo introducir en los campos del perfil de usuario?*
   Existen restricciones en el campo -edad-, que solo pueden ser números enteros, y en los campos -altura- y -peso-, donde los valores deben ser siempre números, ya sean enteros o decimales

3. *¿Puedo añadir todo tipo de caracteres en el nombre de usuario?*
   Efectivamente, cualquier carácter es válido.

4. *¿Cómo puedo saber qué perfil está cargado en ese momento?*
   Es tan fácil como acceder al apartado -Historiales-, en la parte superior aparecerá el usuario cargado en ese momento.

5. *¿Es posible salir de un entrenamiento sin guardar?*
   Por supuesto que sí. Es tan fácil como pulsar la tecla -Menú- de tu smartphone, y a continuación pulsar sobre el botón -Salir sin guardar- del menú que aparece en su pantalla.

6. *¿Cómo puedo borrar un usuario?*
   Existe la posibilidad de borrar todos los usuarios e historiales de la base de datos con la opción existente en las opciones de la aplicación. Lamentablemente, no existe la posibilidad de borrar un usuario determinado.

7. *¿Necesito tener la aplicación abierta para que funcione?*
   No es necesario, la aplicación seguirá funcionando aunque no se encuentre en primer plano.

8. *Si me llaman por teléfono... ¿sigue funcionando mi aplicación?*
   Como se ha comentado en la pregunta anterior, no hace falta que la aplicación este en primer plano, por lo que podrá hablar por teléfono sin necesidad de detener la aplicación.

# 8. Otros

Se recomienda encarecidamente al usuario realizar una calibración de la sensibilidad antes del primer uso real del sistema, para garantizar una correcta utilización de la aplicación.

# *ANEXO C: PRESUPUESTO DEL PROYECTO*

En este anexo del proyecto se muestra el presupuesto para el desarrollo del sistema, en el que se hace una descripción detallada de los gastos del proyecto, incluyendo el personal que ha desarrollado el sistema y los equipos utilizados para ello. Además, se ha hecho un estudio en el que se analiza el número de ventas mínimo para amortizar el desarrollo de este proyecto. El presupuesto detallado se encuentra en la Fig. 46.

**UNIVERSIDAD CARLOS III DE MADRID**
**Escuela Politécnica Superior**

**PRESUPUESTO DE PROYECTO**

1.- Autor:
Rafael Galvez Sanchez

2.- Departamento:
CAOS: Control, Aprendizaje y ó de Sistemas

3.- Descripción del Proyecto:
- Titulo: ***"StepCalories": Aplicación dedicada a una vida saludable para Android***
- Duración (meses): ***7***

4.- Presupuesto total del Proyecto (valores en Euros):
80.000,00 Euros

5.- Desglose presupuestario (costes directos)

**PERSONAL**

| *Categoría* | *Dedicación (hombres mes)* [a)] | *Coste hombre hora* | *Coste hombre mes* | *Coste (Euro)* |
|---|---|---|---|---|
| Jefe de Proyecto | 130 | 30 € | 3.900,00 | 27.300,00 |
| Analista | 120 | 25 € | 3.000,00 | 21.000,00 |
| Programador | 140 | 15 € | 2.100,00 | 14.700,00 |
| Encargado de las pruebas | 100 | 15 € | 1.500,00 | 10.500,00 |
| | | | **Total** | **73.500,00** |

[a)] Máximo mensual de dedicación de 1 hombres mes (140 horas)

**EQUIPOS**

| *Descripción* | *Coste (Euro)* | *% Uso dedicado proyecto* | *Dedicación (meses)* | *Periodo de depreciación* | *Coste imputable* [d)] |
|---|---|---|---|---|---|
| LG R510 | 800,00 | 100 | 7 | 60 | 93,33 |
| HTC SENSATION | 600,00 | 50 | 7 | 60 | 35,00 |
| HTC DESIRE | 400,00 | 50 | 7 | 60 | 23,33 |
| | | | | **Total** | **151,67** |

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

**A** = nº de meses desde la fecha de facturación en que el equipo es utilizado
**B** = periodo de depreciación (60 meses)
**C** = coste del equipo (sin IVA)
**D** = % del uso que se dedica al proyecto (habitualmente 100%)

6.- Resumen de costes

| | *Presupuesto Costes Totales* |
|---|---|
| Personal | 73.500,00 € |
| Amortización | 151,67 € |
| **Total** | **73.651,67 €** |

7.- Amortización del producto

| *Descripción* | *Coste total proyecto* | *Coste inscripcion Android* | *Precio Venta* | *Porcentaje Android* | *Beneficio por venta* | ***Total Ventas*** |
|---|---|---|---|---|---|---|
| StepCalories | 73.651,67 € | 18 € | 6,99 € | 30% | 4,89 € | **15.056** |

**Fig. 46. Presupuesto total del proyecto**