



ugr | Universidad
de Granada

TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA

Aplicación de coaching deportivo para entrenamientos basados en potencia

Kratos Trainer

Autor
Ahisahar Pretel Rodríguez

Director
Miguel Damas Hermoso



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 24 de Noviembre de 2017



Kratos

Aplicación de coaching deportivo para entrenamientos basados en potencia

Kratos Trainer.

Autor

Ahisahar Pretel Rodríguez

Director

Miguel Damas Hermoso

Aplicación de coaching deportivo para entrenamientos basados en potencia: Kratos

Ahisahar Pretel Rodríguez

Palabras clave: pico_potencia, 1RM, IMU, smartwatch, bias, drift, Google Play, API, listener, wearable, Api Rest, NoSQL, Vpp, jerk, sensor_fusion

Resumen

Hacer deporte regularmente mejora la salud y la calidad de vida, haciendo que tareas cotidianas sean más fáciles de realizar. El entrenamiento basado en potencia es un tipo de entrenamiento que aporta muchos beneficios, dado que está dirigido a aumentar la potencia, que es el producto de fuerza por velocidad. Este proyecto busca facilitar y asistir al usuario a la hora de realizar un entrenamiento basado en potencia, visualizando en tiempo real la velocidad y potencia realizada, de una manera intuitiva y fácil. La portabilidad es una cualidad deseable pues los entrenamientos pueden ser desarrollados tanto en interior com en exterior, por lo que la aplicación se va a desarrollar en para un dispositivo móvil.

Se busca encontrar un punto máximo de velocidad y fuerza. Por lo que será necesario medir ambas magnitudes. Para realizar la medición se utilizará un dispositivo wearble (smartwatch), bajo el sistema operativo Android Wear. Dicho dispositivo será utilizado como un IMU, recogiendo los datos y enviandolos a la aplicación móvil. Los datos de los sensores, al recibirlos en la aplicación móvil, deberán ser procesados. Una vez los datos han sido tratados, se representarán los datos de una manera gráfica y numérica. Además, se utilizará un cloud backend utilizando Firebase, para la gestión de usuarios y para el almacenamiento de la información persistente.

An application for monitoring power training: Kratos Trainer

Ahisahar Pretel Rodríguez

Keywords: peak_power, 1RM, IMU, smartwatch, bias, drift, Google Play, API, listener, wearable, Api Rest, NoSQL, Vpp, jerk, sensor_fusion

Abstract

It is well known that doing sports improve your health and boosts the strength needed for daily tasks. A helpful training called power training is aimed at increasing power, which is the product of both strength and speed. To develop this training at its best, a measuring device is required. This work presents kratos, a noble training app which aims to measure the power of the workout using by using a wearable and mobile devices. Specifically, using a general purpose device, a wearable device (smartwatch). Which will register the movement using its sensors, sending that data to the mobile app. The mobile app will process that data received and will represent it graphically, so the user will be able to develop a more efficient training.

Agradecimientos

A mi familia, amigos y profesores. Porque ellos me han inspirado.

Índice general

1. Introducción	19
1.1. Motivación	19
1.1.1. El entrenamiento de fuerza	19
1.1.2. Entrenamiento de potencia	20
1.2. Objetivos	20
1.3. Estructura	21
2. Estado del arte	23
2.1. Estado del mercado	23
2.2. Tecnologías	24
2.2.1. Android	24
2.2.2. Android Wear	26
2.2.3. GoogleAPIClient	27
2.2.4. DataAPI	27
2.2.5. Firebase	28
2.2.6. Sensores	28
2.3. Sensores de movimiento en Android	29
2.3.1. Acelerómetro	29
2.3.2. Giroscopio	31
2.3.3. Magnetómetro	33
3. Especificación de requisitos	35
3.1. Requisitos funcionales	35
3.2. Requisitos no funcionales	36
3.3. Requisitos de información	37
4. Planificación	39
4.1. Planificación	39
4.1.1. Asistente entrenamiento potencia	40
4.1.2. Uso de un servidor	41
4.2. Coste del proyecto	41

5. Análisis de los datos	43
5.1. Introducción al problema	43
5.2. Calculo de la velocidad	43
5.3. Calibración de los sensores	45
5.3.1. Compensación gravitatoria utilizando Sensor Fusion	46
5.3.2. Filtro de paso bajo	47
5.3.3. Aceleración Lineal	47
5.4. Pruebas de calibración	48
5.4.1. Condición de parada	48
5.4.2. Condición de movimiento	51
5.5. Eliminación de ruido mecánico	54
6. Diseño	57
6.1. Diseño del sistema	57
6.2. Diagrama de clases aplicación móvil	58
6.3. Diagrama de clases aplicación wear	60
6.4. Casos de uso	61
6.5. Diseño de la base de datos	65
7. Implementación	67
7.1. Hito 1: Asistente entrenamiento potencia	68
7.2. Hito 2: Uso del servidor	69
7.3. Capturas	69
8. Pruebas	75
9. Conclusiones y trabajo futuro	79
Bibliografía	82
A. Manual de usuario	83
A.1. Instalación	83
A.2. Creación de usuario	85
A.3. Realización ejercicio	86
A.4. Consulta del histórico	90

Índice de figuras

1.1. Relación entre fuerza y velocidad, punto de máxima potencia 'Optimal Load' o 'Peak Power'	20
2.1. Arquitectura del SO Android	24
2.2. Pila de actividades de Android.	26
2.3. Estructura de la capa DataApi.	27
2.4. Acelerómetro piezoeléctrico	30
2.5. Sistema de coordenadas relativo a la API de los sensores.	30
2.6. Giroscopio de estructura vibrante.	32
2.7. Magnetómetro.	33
5.1. Señal del acelerómetro [1].	45
5.2. Compensación de la gravedad por medio del algoritmo de Madgwick (sin movimiento).	48
5.3. Velocidad calculada de la aceleración por medio del algoritmo de Madgwick (sin movimiento).	49
5.4. Compensación de la gravedad por medio de un filtro de paso bajo (sin movimiento).	49
5.5. Velocidad calculada de la aceleración por medio de un filtro de paso bajo (sin movimiento).	50
5.6. Compensación de la gravedad por medio de la aceleración lineal (API) (sin movimiento).	50
5.7. Velocidad calculada de la aceleración lineal (API) (sin movimiento).	51
5.8. Compensación de la gravedad por medio del algoritmo de Madgwick (con movimiento vertical).	52
5.9. Velocidad calculada de la aceleración por medio del algoritmo de Madgwick (con movimiento vertical).	52
5.10. Compensación de la gravedad por medio de un filtro de paso bajo (con movimiento vertical).	53
5.11. Velocidad calculada de la aceleración por medio de un filtro de paso bajo (con movimiento vertical).	53
5.12. Compensación de la gravedad por medio de la aceleración lineal (API) (con movimiento vertical).	54

5.13. Velocidad calculada de la aceleración lineal (API) (con movimiento vertical).	54
5.14. Ventana de discriminación [1].	55
6.1. Arquitectura del proyecto.	58
6.2. Diagrama de clases de la aplicación móvil.	59
6.3. Diagrama de clases de la aplicación wear.	60
6.4. Caso de uso 1 - Credenciales.	61
6.5. Caso de uso 2 - Realización del ejercicio.	62
6.6. Caso de uso 3 - Consulta del histórico.	64
7.1. Diagrama de clases de la aplicación móvil.	67
7.2. Pantalla principal Wear	70
7.3. Realizar medición Wear	70
7.4. Pantalla inicio sesión móvil	71
7.5. Pantalla principal móvil	71
7.6. Vista menú móvil	72
7.7. Vista selección ejercicio móvil	72
7.8. Vista descripción ejercicio móvil	73
7.9. Vista inicial ejercicio móvil	73
7.10. Vista realización ejercicio móvil	74
8.1. Realización del primer test de la aplicación.	75
A.1. Instalación desde origen desconocido.	84
A.2. Modo depuración Android Wear [2].	85
A.3. Registro 1.	85
A.4. Registro 2.	86
A.5. Pantalla principal móvil	87
A.6. Ejemplo selección ejercicio móvil	87
A.7. Ejemplo descripción ejercicio móvil	88
A.8. Ejemplo iniciar ejercicio móvil	88
A.9. Pantalla principal wear	89
A.10. Realizar medición wear	89
A.11. Ejemplo realización ejercicio móvil	89
A.12. Consulta del histórico	90

Índice de cuadros

4.1. Costes humanos del proyecto	42
4.2. Coste total del proyecto	42
6.1. Caso de uso - Crear cuenta	61
6.2. Caso de uso - Inicio sesión	62
6.3. Caso de uso - Recuperar contraseña	62
6.4. Caso de uso - Seleccionar ejercicio	63
6.5. Caso de uso - Inicio de la monitorización del ejercicio	63
6.6. Caso de uso - Visualización de los resultados	64
6.7. Caso de uso - Selección de una fecha	65
6.8. Caso de uso - Consulta de los datos del histórico	65
6.9. Estructura de la base de datos	66
8.1. Test 1	76
8.2. Test 2	76
8.3. Test 3	77

Capítulo 1

Introducción

Este capítulo está formado por tres secciones. La primera llamada Motivación 1.1 describe que es un entrenamiento basado en potencia y los beneficios que se obtienen de este tipo de entrenamiento. La segunda sección denominada Objetivos 1.2, contiene un resumen de los objetivos que se pretenden alcanzar con el desarrollo de este proyecto. La última sección 1.3 denominada Estructura, presenta la estructura de los capítulos del presente documento así como una breve descripción de sus contenidos.

1.1. Motivación

Desde hace décadas, entrenadores e investigadores buscan la forma mas óptima de mejorar el rendimiento deportivo de los deportistas. Existen dos métodos particularmente eficientes para conseguir dicho objetivo, el entrenamiento de resistencia y el entrenamiento de fuerza [3].

Se ha observado que dichas cualidades (fuerza y velocidad) se encuentran simultáneamente en numerosos deportes [4]. Ambos entrenamientos suelen ser realizados de manera alterna, ya que si se realizan el mismo día y sin un período de recuperación adecuado producen interferencias y los resultados obtenidos no son los esperados, por lo que a priori puede parecer contraproducente [5].

1.1.1. El entrenamiento de fuerza

El entrenamiento de fuerza consiste en el uso de la resistencia para inducir una contracción muscular, la cual construye la fuerza y resistencia. Desarrolla la sincronización a nivel neural intra e intermuscular. Lo cual produce mayor velocidad de contracción de las fibras musculares y el aumento del área de sección transversal y, consecuentemente, del músculo [6]. La forma más común de realizar un entrenamiento de fuerza es utilizando autocargas o sobrecargas.

1.1.2. Entrenamiento de potencia

Una de las manifestaciones de la fuerza, es la potencia mecánica, la cual viene dada por:

$$\text{Potencia} = \text{Fuerza} * \text{Velocidad}$$

Generalmente, para realizar un entrenamiento de potencia, se busca añadir la variable velocidad a un entrenamiento de fuerza. Realizando el ejercicio de fuerza a mayor velocidad.

La potencia muscular es un factor determinante a la hora de mejorar el rendimiento deportivo, especialmente en aquellos deportes o actividades en la que se requiera realizar un levantamiento, acelerar, decelerar, saltar, o realizar cambios bruscos de dirección.

A la hora de realizar un entrenamiento basado en potencia utilizando cargas, se debe tener en consideración que la máxima ganancia se localiza en un punto donde la relación entre fuerza y velocidad son máximas. Este punto es conocido como '*Optimal Load*' [7].

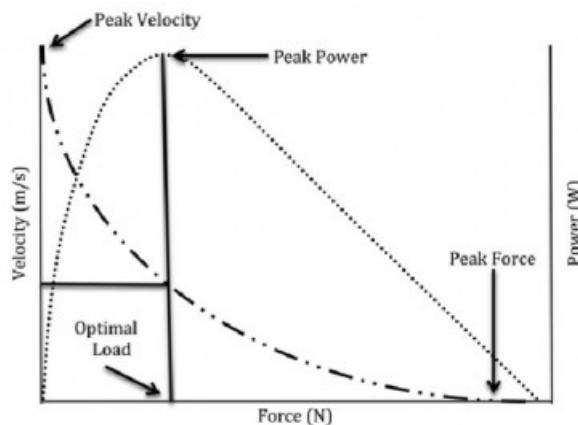


Figura 1.1: Relación entre fuerza y velocidad, punto de máxima potencia '*Optimal Load*' o '*Peak Power*'.

La carga seleccionada para el desarrollo óptimo de la potencia suele ser entre el 30-45 % del 1RM, aunque este porcentaje puede variar [8].

1.2. Objetivos

Los objetivos perseguidos en el desarrollo del proyecto son los siguientes:

1. **La creación de una aplicación que facilite la labor de un entrenador personal a la hora de realizar, medir y valorar los progresos de un usuario a lo largo del tiempo.**

Probar la viabilidad del desarrollo de una aplicación de salud y deporte utilizando dispositivos de propósito general con sensores, en lugar de un IMU específico.

2. **Investigación de métodos de posicionamiento y rastreo en un escenario real y 3D.**

Realizando la calibración de sensores de bajo coste con el fin de reducir el ruido y el error innato de los sensores.

3. **Aplicar los conocimientos adquiridos sobre ingeniería del software.**

Se ha perseguido la aplicación de los conocimientos adquiridos durante el grado en el desarrollo de software en un proyecto real.

4. **Elaboración de un prototipo funcional del proyecto.**

Se persigue la implementación de una aplicación funcional, una aplicación que será instalada en el smartwatch, que provea de los datos y mediciones de la realización del ejercicio. Y otra aplicación instalada en el smartphone que se encargará de recopilar los datos, procesarlos y mostrarlos.

1.3. Estructura

Este documento se encuentra dividido en ocho capítulos. Los cuales desarrollan:

1. **Capítulo 1: Introducción.** Es el capítulo en el que se encuentra esta sección, está formado por tres secciones. La primera llamada Motivación 1.1 describe que es un entrenamiento basado en potencia y los beneficios que se obtiene de este tipo de entrenamiento. La segunda sección denominada Objetivos 1.2, contiene un resumen de los objetivos que se pretenden alcanzar con el desarrollo de este proyecto. La última sección denominada Estructura 1.3, presenta la estructura de los capítulos del presente documento así como una breve descripción de sus contenidos.
2. **Capítulo 2: Estado del arte** Este capítulo se organiza en tres apartados. El primero, llamado Estado del mercado 2.1 define el estado actual del mercado de tecnologías que permiten realizar mediciones de entrenamientos basados en potencia. El segundo capítulo denominado Tecnologías 2.2 contiene una breve descripción de las tecnologías que nos van a permitir el desarrollo de este proyecto. Por último la sección llamada Sensores de movimiento en Android 2.3 trata sobre los principales tipos de sensores relacionados con el movimiento que se encuentran disponibles en Android.
3. **Capítulo 3: Especificación de requisitos** En este capítulo encontramos los distintos tipos de requisitos que debe cumplir el sistema. Se encuentra estructurado en tres secciones. Requisitos funcionales 3.1, los cuales definen funciones del sistema.

En la segunda sección encontramos los Requisitos no funcionales 3.2, los cuales contienen restricciones del sistema relacionadas con el diseño o la implementación. Finalmente, encontramos la sección Requisitos de información 3.3, los cuales hacen referencia a información que debe ser almacenada en el sistema.

4. **Capítulo 4: Planificación** Este capítulo se organiza en dos apartados. El primero llamado Planificación 4.1, contiene la planificación general del proyecto, haciendo uso de metodologías ágiles. El segundo apartado denominado Coste del proyecto 4.2, contiene un escueto presupuesto sobre el desarrollo del proyecto.
5. **Capítulo 5: Análisis de los datos** Este capítulo está formado por 5 secciones. La primera denominada Introducción al problema 5.1, nos habla sobre porque es importante tratar los datos de los sensores en el desarrollo de este proyecto. La segunda sección llamada Cálculo de la velocidad 5.2, describe como se obtiene la velocidad a partir de la aceleración. La tercera sección denominada Calibración de los sensores 5.3, comenta tres técnicas diferentes para conseguir eliminar la componente gravitacional de la aceleración. La sección Pruebas de calibración 5.4, pone las técnicas descritas en el apartado anterior a prueba. Finalmente, la sección Eliminación de ruido mecánico 5.5, describe un método para eliminar parcialmente el ruido mecánico de los sensores.
6. **Capítulo 6: Diseño** En este capítulo se encuentran 5 secciones. La primera, Diseño del sistema 6.1, describe la arquitectura general del sistema. La segunda, Diagrama de clases de la aplicación móvil 6.2, describe atributos, métodos y relaciones de la aplicación móvil. La tercera, Diagrama de clases de la aplicación wear 6.3, describe atributos, métodos y relaciones de la aplicación wear. La cuarta sección denominada Casos de uso 6.4, contiene los distintos casos de uso de los usuarios en el sistema. Finalmente, la sección Diseño de la base de datos 6.5, describe la estructura de la base de datos.
7. **Capítulo 7: Implementación** Formado por 3 secciones. La primera llamada Hito 1: Asistente entrenamiento potencia 7.1, describe los pasos realizados para implementar el asistente de entrenamiento. La segunda, denominada Hito 2: Uso del servidor, describe los pasos realizados para implementar la conexión con el servidor. Finalmente, la sección Capturas 7.3, contiene capturas sobre la aplicación.
8. **Capítulo 8: Conclusiones y trabajo futuro** Realiza un análisis sobre los objetivos buscados, conseguidos y los resultados del proyecto. Además de exponer cual será el trabajo futuro.

Capítulo 2

Estado del arte

2.1. Estado del mercado

Existen tres herramientas principales a la hora de monitorizar un entrenamiento basado en potencia:

Encoder Lineal

Un encoder lineal es un dinamómetro que, aplicado al deporte, se utiliza para realizar una medición directa y continua del espacio recorrido y el tiempo de movimiento de una carga conocida. Pudiendo utilizarse así para calcular la potencia, fuerza o velocidad [9].

Son muy fiables y precisos, pero presentan también algunos inconvenientes. El principal inconveniente es su precio, que puede ser realmente elevado. Por lo su venta no va dirigida a usuarios particulares, si no a profesionales. Además requiere de personal cualificado para su uso e interpretación.

Push Band

Push Band es un dispositivo wearable IMU, que conectado a una aplicación permite la monitorización del entrenamiento. Además, dispone de una interfaz web, accesible desde cualquier plataforma[10].

Si bien su coste no es tan elevado, también presenta inconvenientes. El principal de ellos es que está disponible solo para iOS.

Beast Sensor

Beast Sensor es también un dispositivo wearable IMU, que se conecta a través de una aplicación móvil (disponible en iOS y Android) y es usada para monitorizar el ejercicio[11]. Además también presenta una interfaz web donde se puede consultar la información de una cuenta.

Las diferencias entre Beast Sensor y Push Band son mínimas, su principal diferencia es que Beast funciona en Android mientras que Push no. Se debe destacar que, tanto

estas dos herramientas como los encoders lineales hacen uso de dispositivos de propósito específico, que el usuario debe adquirir expresamente si desea monitorizar su entrenamiento.

Por ello surge este proyecto, con el fin de poder realizar la monitorización del entrenamiento con un dispositivo de propósito general. Como son los smartwatches con Android Wear.

2.2. Tecnologías

2.2.1. Android

Android es un sistema operativo móvil desarrollado por Google y basado en el kernel de Linux, diseñado principalmente para dispositivos móviles y otros dispositivos de pantalla táctil, como smartphones, smartwatches y tabletas. Originalmente fue desarrollado por Android Inc., la cual fué posteriormente adquirida por Google. Android es además un sistema multiplataforma, lo cual lo ha hecho adquirir el gran éxito y popularidad de la que goza hoy en día.

Arquitectura de Android

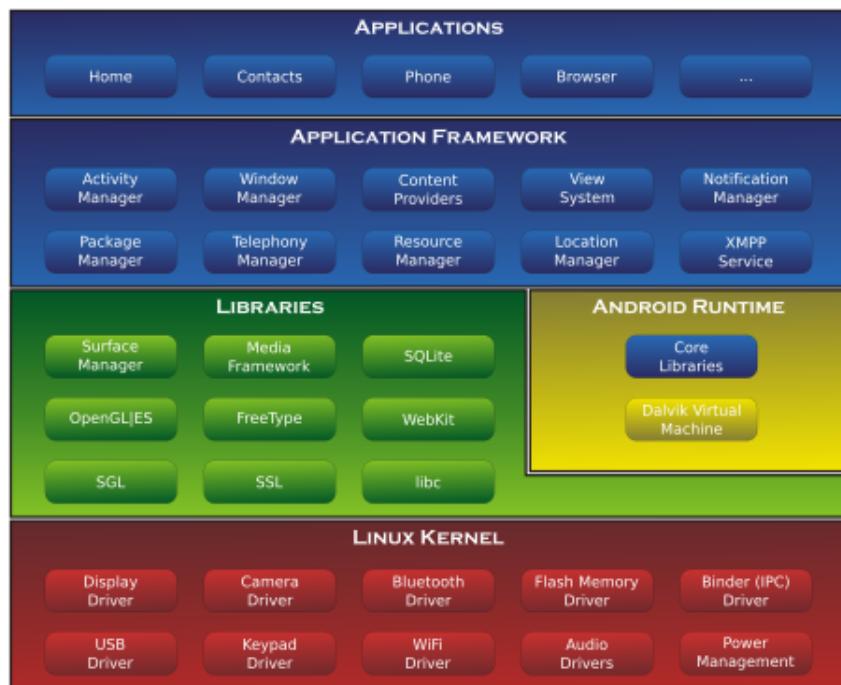


Figura 2.1: Arquitectura del SO Android

La arquitectura de Android se encuentra formada por capas. Como podemos apreciar en la figura superior, en la base de la pila se encuentra el kernel de Linux. Construido

en C principalmente, es el encargado de la gestión de la memoria, servicios de seguridad y de dar soporte a los controladores.

La capa inmediatamente superior, es la capa del framework de aplicación, la cual incluye librerías y la denominada 'Android Runtime'.

Las librerías son nativas, escritas en C y precompiladas. Entre ellas se destacan:

- SSL: Proporciona servicios de seguridad y encriptación.
- SQLite: Motor de bases de datos relacionales.
- WebKit: Ofrece soporte para el navegador web y su vista webview.

El entorno de ejecución o 'Android Runtime', basado en la máquina virtual de Java, recibe el nombre de Dalvik. La máquina virtual será la encargada de compilar y ejecutar de forma nativa fragmentos de código denominados tazas de código, cada vez que una aplicación es lanzada. Optimizando de esta manera los recursos del sistema y delegando al kernel el threading y el manejo de memoria a bajo nivel.

En la capa inmediatamente superior encontramos el marco de aplicación o 'Application Framework', el cual ofrece una plataforma a las aplicaciones, incluyendo librerías Java. Entre ellas, las más importantes son:

- Activity Manager. Manejador destinado al ciclo de vida de las actividades.
- Views. Conjunto de vistas.
- Location Manager. Proporciona servicios de localización a aplicaciones.
- Content Provider. Concede accesos a datos de otras aplicaciones.
- Notification Manager. Facilita el soporte para mostrar notificaciones a las distintas actividades.

En la capa superior de abstracción, encontramos la placa de aplicación. La cual engloba a todas las aplicaciones que se encuentren instaladas en el dispositivo.

Tareas y pila de actividades

Una aplicación generalmente está formada por una serie de actividades, las cuales son necesarias para el correcto funcionamiento de la aplicación. Por norma general, cada actividad debe estar diseñada en torno a una clase de acción que puede realizar el usuario y se deben poder comunicar con otras actividades, creándolas y enviando información entre ellas. Pudiendo incluso darse el caso de que una actividad inicie una actividad existente en otra aplicación del dispositivo, lo cual otorga una gran versatilidad.

Podemos definir una tarea como la colección de actividades cuyo objetivo o finalidad

es común. Dichas actividades son organizadas en la pila de actividades, la cual se va llenando en el orden en el que se abre cada actividad.

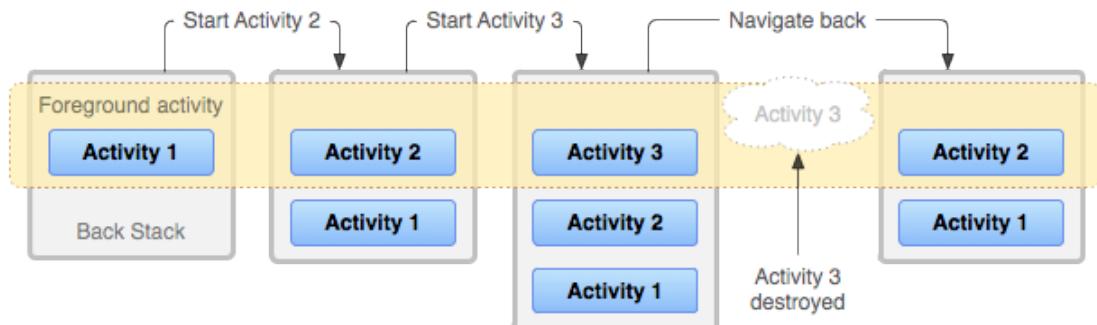


Figura 2.2: Pila de actividades de Android.

En la figura anterior, podemos ver el contenido de la pila en base a diferentes acciones. Cuando creamos una nueva actividad, la actividad anterior sigue mantenida en la pila de actividades, ocupando así la nueva actividad creada la cima de la pila. Al pulsar el botón hacia atrás o destruir una actividad, la actividad que ocupaba la segunda posición en la pila, queda en el tope de la pila y se vuelve a mostrar dicha actividad, liberando los recursos que estaba consumiendo la actividad anterior.

Cuando la pila de actividades se encuentra vacía, la tarea deja de existir y sus recursos son liberados totalmente. Además, si cambiamos a otra actividad de otra aplicación o volvemos a la pantalla de inicio, las actividades de la pila pasaran a segundo plano, deteniéndose, pero la pila de retroceso permanecerá intacta.

2.2.2. Android Wear

Android Wear es una versión del sistema operativo de Google, Android. Desarrollado para usarse como sistema operativo en smartwatches. El sistema operativo fue lanzado por Google en 2014 y puede ser conectado con dispositivos Android a partir de su versión 4.3 o en iPhone a partir de iOS 9. Siendo posible que la funcionalidad varíe en función de la plataforma y versión del sistema operativo.

Actualmente se encuentra en la versión 2.0 en la cual se da mayor independencia al dispositivo wear con respecto al smartphone. Otorgandole así mayor autonomía.

La mayor diferencia entre el ecosistema Android y Android Wear se aprecia en las actividades, tienen un conjunto diferente de eventos que se asocian con el sistema operativo, los más destacados son:

- Cuando cambia la hora.
- Cuando llega una notificación.

- Cuando la intensidad del brillo de la pantalla cambia.

2.2.3. GoogleAPIClient

La clase GoogleApiClient es el punto de enlace para los servicios de integración de Google Play (*Google Play Services*). Es decir, cualquier aplicación que quiera hacer uso de alguna API de los servicios de Google, deberá instanciar un cliente de dicha clase en la aplicación.

La clase dispone de una gran multitud de métodos. Algunos requieren que el cliente GoogleApiClient este conectado, otros se encolan cuando no hay una conexión establecida. Pero para poder ejecutar cualquier operación, debe existir una conexión[12].

2.2.4. DataAPI

Es una API que forma parte de los servicios de *Google Play Services*, los cuales proveen a las aplicaciones de un canal de comunicación adicional, en este caso, para un dispositivo wearable. La API Data Layer consiste en una serie de objetos que pueden ser utilizados para enviar y sincronizar datos, en conjunto con los '*listeners*', para establecer una comunicación. En nuestro caso vamos a utilizar la API para establecer un canal de comunicación entre el dispositivo Wearable y el móvil construido sobre el estándar Bluetooth [13].

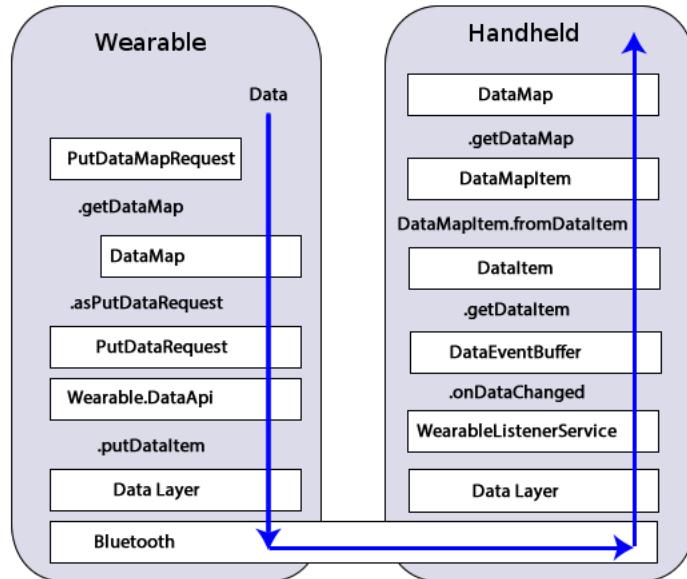


Figura 2.3: Estructura de la capa DataApi.

Para emplear esta API, se debe disponer de una versión de Android igual o superior a la 4.3 (API 18) en el dispositivo wear. La primera versión disponible de Android Wear

está basado en Android 4.4.2 KitKat (API 19), por lo que es compatible con todos los dispositivos Android Wear. Además, cabe destacar que se debe de utilizar la última versión de Google Play services [14].

2.2.5. Firebase

Firebase [15] es una aplicación web y móvil del tipo BaaS (Backend-as-a-Service), es decir ofrece funcionalidades propias de un servidor por medio de una interfaz. Permitiendo a las aplicaciones conectarse y sincronizarse con un servidor de una manera rápida y sencilla sin necesidad de implementar una API Rest.

En su esencia, es principalmente una base de datos en tiempo real, la cual hace uso del protocolo WebSockets en lugar de HTTP. Y su principal ventaja es que todos los clientes de la aplicación en tiempo real sincronizan sus datos, de manera que todos los clientes de la base de datos los reciben de manera instantánea.

Utiliza un modelo de base de datos NoSQL. Los datos se almacenan en formato JSON. Y como tal ofrece diferentes funcionalidades en comparación con una base de datos relacional, siendo la velocidad su mayor ventaja.

Además de brindar una base de datos, provee una amplia lista de funcionalidades , sus principales son:

- Almacenamiento de archivos. Permite almacenar archivos en '*Google Cloud Storage*' de forma directa desde el cliente.
- Autenticación. Ofrece un sistema de autenticación por medio de email/contraseña, además soporta autenticación en base a '*OAuth2*' con multitud de servicios, como Google, Facebook, Twitter y Github.
- Servicio de alojamiento. Permite almacenar y servir archivos estáticos utilizando el protocolo HTTP/2.

2.2.6. Sensores

Se define como sensor a un elemento o dispositivo electrónico, que detecta eventos o cambios en el entorno en el que se encuentran y envían la información de dicho evento a otro dispositivo electrónico.

La sensibilidad es definida como el ratio entre la señal de salida y la magnitud de la propiedad medida. La mayoría de sensores tienen una función de transferencia lineal.

Desviaciones de los sensores

Los sensores no pueden replicar una función de transferencia ideal. Presentando las diferentes fluctuaciones:

- El rango de la señal de salida esta limitado, soliendo obedecer normalmente a un

Vpp. Por lo que la salida del sensor alcanzará eventualmente un mínimo y un máximo.

- La sensibilidad práctica difiere del valor especificado. Denominado como error de sensibilidad.
- Si la señal de salida difiere del valor correcto por una constante y, se dice que el sensor posee un error de *offset* (desplazamiento) o error bias.
- La no linearidad es la desviación de la función de transferencia de un sensor.
- La desviación causada por cambios rápidos en la propiedad que se está midiendo se conoce como error dinámico.
- Si la señal de salida cambia de manera muy lenta, independientemente de la variación de la propiedad medida, dicho fenómeno es conocido como *drift* (desvío).
- El ruido es una desviación aleatoria de la señal que varía con respecto al tiempo.
- Los sensores con una señal de salida digital, la salida es esencialmente una aproximación de la propiedad medida. Dicho fenómeno es llamado error de quantización.
- Cuando la señal es monitorizada digitalmente, la frecuencia de muestreado puede causar errores dinámicos, o si la entrada de datos cambia periódicamente a otra frecuencia, puede producir errores de *antialiasing* (acoplo).

2.3. Sensores de movimiento en Android

La API de Android provee de varios sensores con el fin de poder monitorizar el movimiento del dispositivo.

Android provee de sensores basados en software o hardware. Los sensores software como la gravedad, vector de rotación, contador de pasos etc, han de ser implementados usando sensores hardware (acelerómetros, giroscopios y magnetómetros).

Actualmente, la gran mayoría de dispositivos Android tradicionales contienen acelerómetros, giroscopios y magnetómetros. Sin embargo no todos los dispositivos Android Wear disponen de estos tres sensores, siendo el caso más común que dispongan de acelerómetro y giroscopio únicamente [16].

2.3.1. Acelerómetro

Un acelerómetro es un dispositivo que mide vibración o aceleración propia de una estructura. Siendo la aceleración el ratio del cambio de velocidad con respecto al tiempo. La fuerza causada por las vibraciones de un cambio en el movimiento (aceleración) causa que la masa mueva el material piezoelectrónico generando una carga que es proporcional a la fuerza que experimenta (Fuerza de la gravedad incluida) .

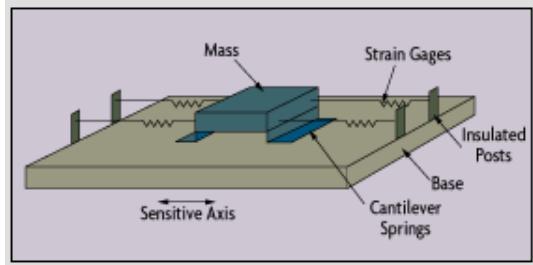


Figura 2.4: Acelerómetro piezoelectrónico

Los acelerómetros *multi-axis* (varios ejes) son capaces de detectar magnitudes y direcciones de la propia aceleración como un vector cuantitativo [17].

Acelerómetro en Android

El acelerómetro mide la aceleración aplicada al dispositivo móvil, incluyendo la fuerza de la gravedad. Siguiendo el estándar de coordenadas que podemos apreciar en la siguiente figura:

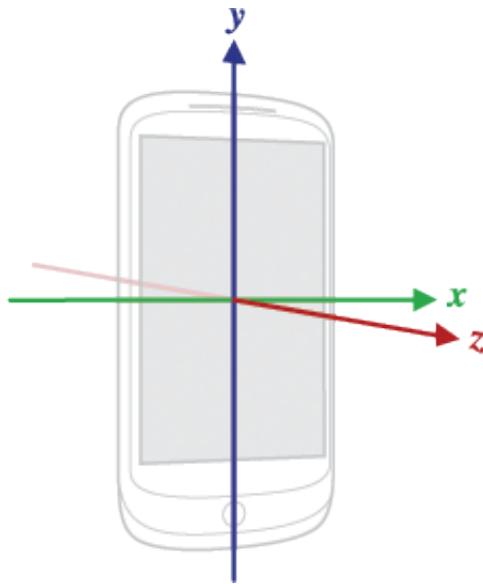


Figura 2.5: Sistema de coordenadas relativo a la API de los sensores.

Es el sensor principal que se debe monitorizar para realizar un seguimiento de movimiento. Puesto que esta presente en todos los dispositivos móviles comerciales y tiene un consumo de energía de hasta 10 veces menor que el resto de sensores. La API provee de una serie diferente de posibilidades para el uso del acelerómetro [18]:

Sensor	Estructura de Datos	Descripción	Unidades
TYPE_ACCELEROMETER (API 3)	Vector[3]	Fuerza de aceleración a lo largo de los 3 ejes. Incluyendo la gravedad.	m/s ²
TYPE_ACCELEROMETER_UNCALIBRATED (API 26)	Vector[3]	La aceleración medida a lo largo de los 3 ejes sin ningún tipo de desviación bias.	m/s ²
TYPE_GRAVITY (API 9)	Vector[3]	La aceleración producida por la fuerza de la gravedad.	m/s ²
TYPE_LINEAR_ACCELERATION (API 9)	Vector[3]	La aceleración a lo largo de los 3 ejes. Excluyendo la gravedad	m/s ²

2.3.2. Giroscopio

Un giroscopio es un dispositivo usado para medir el grado de rotación en radianes por segundo a lo largo de un eje de coordenadas. Es decir, son dispositivos que miden la velocidad angular. En los últimos años, ha aumentado el empleo de giroscopios de estructura vibrante en todo tipo de industrias, como móviles, vehículos, fotografía, etcétera.

Giroscopios de estructura vibrante

Los giroscopios de estructura vibrante miden la velocidad angular de la fuerza causada por el efecto Coriolis aplicada a un elemento en vibración. Por lo que la precisión de la velocidad angular medida difiere dependiendo del material. En la siguiente figura se puede apreciar la estructura:

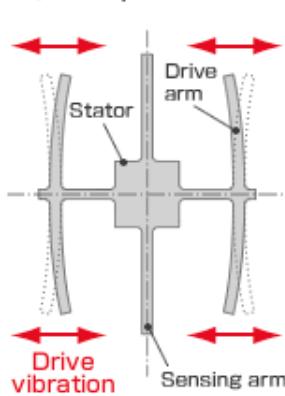


Figura 2.6: Giroscopio de estructura vibrante.

Giroscopios en Android

El giroscopio mide el grado de rotación a lo largo de los 3 ejes de coordenadas del dispositivo, siguiendo el mismo sistema de coordenadas de la API. La rotación es positiva en el sentido antihorario.

Sensor	Estructura de Datos	Descripción	Unidades
TYPE_GYROSCOPE (API 3)	Vector[3]	Ratio de rotación 3-axis.	rad/s
TYPE_GYROSCOPE_UNCALIBRATED (API 18)	Vector[3]	Ratio de rotación 3-axis.Sin compensación drift.	rad/s
TYPE_ROTATION_VECTOR (API 9)	Vector[3]	Rotación vector componente a lo largo de los ejes (eje * Sin(O/2))	Sin unidades

Por defecto los giroscopios también proveen de datos sin ningún tipo de filtrado, por lo que la varianza 'bias' y el ruido inherente están presentes en los datos obtenidos. El tipo de sensor **TYPE_GYROSCOPE** provee de datos con compensación 'bias'. El tipo de sensor **TYPE_GYROSCOPE_UNCALIBRATED** no provee ningún tipo de calibración, obedeciendo a la siguiente relación para cada eje.

$$calibrated_x = uncalibrated_x - bias_{estimate_x}$$

2.3.3. Magnetómetro

Un magnetómetro es un instrumento cuya finalidad es medir la intensidad de un campo magnético. Pudiendo dividirse en dos tipos:

- Magnetómetros escalares. Los cuales miden la fuerza total de los campos magnéticos bajo los que se encuentran. Es decir, miden la sumatoria de la fuerza magnética en todas las direcciones.
- Magnetómetros vectoriales. Tienen la capacidad de medir la intensidad del campo magnético bajo los que se encuentran en una dirección en concreto.



Figura 2.7: Magnetómetro.

Magnetómetro en Android

El magnetómetro en Android mide cambios en la intensidad del campo magnético terrestre .

Sensor	Estructura de Datos	Descripción	Unidades
TYPE_MAGNETIC_FIELD (API 3)	Vector[3]	Campo geomagnético 3-axis.	μT
TYPE_MAGNETIC_FIELD_UNCALIBRATED (API 18)	Vector[6]	Campo geomagnético 3-axis sin calibración de 'Iron' Estimación bias de 'Iron' 3-axis	μT

El tipo **TYPE_MAGNETIC_FIELD** es similar es calibrado siguiendo la siguiente regla:

$$\text{calibrated}_x = \text{uncalibrated}_x - \text{bias}_{\text{estimate}}_x$$

Se debe tener en cuenta que los sensores no calibrados pueden contener desviación bias, pero contienen menos saltos en la frecuencia de muestreo, por lo que dichos sensores proveen datos más frecuentemente y consecuentemente, más fiables en cuanto a frecuencia.

Capítulo 3

Especificación de requisitos

Todo software es creado con el fin de cumplir un servicio o una necesidad. Y como en todo proyecto de software, los requisitos deben ser especificados en la fase previa al desarrollo con el fin de dar cobertura a todas las necesidades del usuario.

Se entiende como requisito una funcionalidad o requerimiento que un usuario necesita para poder cumplir una necesidad en concreto o resolver un problema.

En este capítulo se analizan los distintos requisitos del sistema que vamos a detallar en este documento. Se encuentra estructurado en tres secciones. Requisitos funcionales 3.1, los cuales definen funciones del sistema. En la segunda sección encontramos los Requisitos no funcionales 3.2, los cuales contienen restricciones del sistema relacionadas con el diseño o la implementación. Finalmente, encontramos la sección Requisitos de información 3.3, los cuales hacen referencia a información que debe ser almacenada en el sistema.

3.1. Requisitos funcionales

Los requisitos funcionales son los encargados de definir una función del sistema. De manera que se cubran todas las necesidades de los usuarios.

El sistema cuenta con los siguientes requisitos funcionales:

RF-1. Autenticación: Control de usuarios.

- **RF-1.1** El usuario debe poder darse de alta en la aplicación.
- **RF-1.2** El usuario debe poder iniciar sesión en la aplicación.
- **RF-1.3** El usuario debe poder recuperar sus credenciales.

RF-2. Control: Control del ejercicio.

- **RF-2.1** Se debe poder controlar el inicio del ejercicio/repetición.
- **RF-2.2** Se debe poder controlar el final del ejercicio/repetición.
- **RF-2.3** El usuario puede seleccionar el tipo de ejercicio.
- **RF-2.4** El usuario debe ser capaz de realizar una nueva serie.
- **RF-2.5** El usuario debe ser capaz de introducir el peso con el que esta realizando el ejercicio.

RF-3. Visualización: Representación de la información.

- **RF-3.1** El usuario podrá consultar la potencia realizada.
- **RF-3.2** Visualizar el contenido de un dia en concreto.

3.2. Requisitos no funcionales

Los requisitos no funcionales describen restricciones del sistema que se relacionan con el diseño o la implementación del proyecto.

A continuación se describen los requisitos no funcionales de este proyecto:

- **RNF-1.** Ambas aplicaciones han de ser implementadas en Java.
- **RNF-2.** El envío de datos debe ser en tiempo real.
- **RNF-3.** La visualización debe ser creada en tiempo real, durante la realización del ejercicio.
- **RNF-4.** El sistema deberá enviar la información al servidor, con el fin de no ocupar almacenamiento local en el dispositivo.
- **RNF-5.** El sistema deberá almacenar la información en local si no dispone de conexión a Internet.
- **RNF-6.** La frecuencia de muestreo debe ser ajustada para usar la menor cantidad de memoria posible.
- **RNF-7.** Los datos deben ser tratados para conseguir la máxima precisión posible.

3.3. Requisitos de información

Los requisitos de información se refieren a la información que es imprescindible almacenar en el sistema.

A continuación vamos a describir los requisitos de información del sistema:

- **RI-1** Almacenamiento de credenciales: correo electrónico y contraseña.
- **RI-2** Historial de entrenamiento del usuario.

Capítulo 4

Planificación

En este capítulo se pueden encontrar dos secciones, en la primera se aborda la Planificación 4.1, para el desarrollo del proyecto y en la segunda el Coste del proyecto 4.2, la cual contiene un presupuesto sobre el coste del desarrollo de este proyecto.

4.1. Planificación

Se ha realizado la planificación del proyecto en base a una metodología ágil. Obteniendo así una mayor flexibilidad en el desarrollo, para ello se han realizado historias de usuario, definiendo en ellas la funcionalidad mínima para el correcto funcionamiento del sistema, iterando desde las tareas de máxima prioridad hasta las de prioridad más baja. Se han desarrollado dos grandes hitos:

- **1. Asistente entrenamiento potencia**
- **2. Uso de un servidor**

Se han conseguido desarrollar los dos hitos, a falta de mejorar la interfaz de cara a una mejor experiencia del usuario.

A continuación se van a detallar los hitos:

4.1.1. Asistente entrenamiento potencia

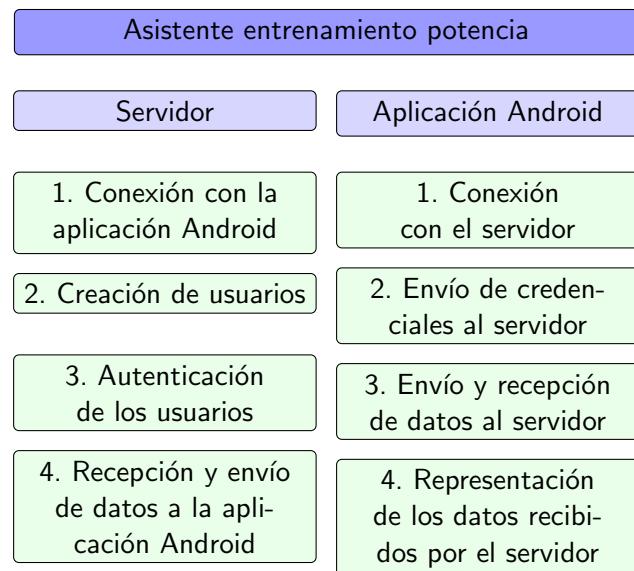


Una vez finalizado este hito, se ha conseguido una arquitectura totalmente operativa para realizar y monitorizar un entrenamiento basado en potencia en tiempo real. Por lo que el objetivo principal del proyecto se podría dar por conseguido.

En el siguiente hito dotaremos al proyecto de funcionalidades adicionales, que mejoren el funcionamiento del mismo y amplíen sus capacidades.

Para ello dotaremos a nuestro sistema de conexión a un servidor, el cual ampliará las capacidades de la aplicación de cara al usuario y también el propio sistema.

4.1.2. Uso de un servidor



Una vez finalizado este hito se ha completado las competencias de este proyecto y su funcionalidad es completa, a falta de una reconstrucción de la interfaz que mejore la experiencia de los usuarios del sistema.

4.2. Coste del proyecto

En esta sección se va a realizar una estimación de los costes materiales y humanos del proyecto tratado en esta memoria.

Costes materiales

El equipo utilizado para este proyecto consta de:

Ordenador utilizado para el desarrollo, junto al siguiente software:

- Sistema Operativo: Ubuntu 16.04
- Editor de texto: ShareLatex (Online)
- IDE: Android Studio
- Diseño de programas UML: Visual Paradigm CE
- Gestión de versiones: Github

Además se ha utilizado un móvil Android (Honor 8 lite) y dos smartwatches (Moto 360 y Sony Smartwatch Sport 3).

Por lo tanto los costes materiales son de :

- Ordenador. Coste total 900. Con una vida útil estimada de 5 años [19], tiene un coste anual de 180 euros/año, 15 euros por mes.
- Móvil Android. Coste total 200. Con una vida útil media de 2 años [20], supone un coste de 100 euros/año, 8,3 euros/mes.
- Dispositivos wearable. Coste total 150 euros y 120 euros respectivamente. Con un ciclo de vida media de dos años [21], suponen un coste de 75 euros/año y 60 euros/año y, consecuentemente, un coste de 6,25 euros/mes y 5 euros/mes.

Costes humanos

Se ha realizado un seguimiento de horas empleadas en cada tarea para estimar los costes humanos del proyecto. Para estimar el coste, se ha teniendo en el salario medio de un programador junior en España [22], siendo de 19.787 euros/año, el coste por hora es de 8,83 euros.

Cuadro 4.1: Costes humanos del proyecto

Nombre de la tarea	Número de horas	Coste por hora	Total (euros)
Estudio previo	120	8,83 euros/hora	1059,6
Estudio de los requisitos	8	8,83 euros/hora	70,64
Estudio del mercado	8	8,83 euros/hora	70,64
Planificación	8	8,83 euros/hora	70,64
Desarrollo del Hito 1	240	8,83 euros/hora	2119,2
Desarrollo del Hito 2	40	8,83 euros/hora	353,2
Elaboración de las pruebas	6	8,83 euros/hora	52,98
Elaboración de la documentación	50	8,83 euros/hora	441,5
Total	480	8,83 euros/hora	4240 euros

Se añaden los costes materiales a los humanos.

Cuadro 4.2: Coste total del proyecto

Descripción	Tipo Recurso	Coste por hora	Coste mensual	Tiempo	Total
Ordenador	Material	-	15 euros	3 meses	45 euros
Móvil	Material	-	8,3 euros	3 meses	24,9 euros
Moto 360	Material	-	6,25 euros	3 meses	18,75 euros
Sony Smartwatch	Material	-	5 euros	3 meses	15 euros
Desarrollo	Humano	8,83 euros	-	3 meses	4238,5 euros
Total					4342,05 euros

Y por lo tanto, el coste total del proyecto será de 4342,05 euros.

Capítulo 5

Análisis de los datos

Este capítulo está destinado al estudio y tratamiento de los datos. Se encuentra dividido en 5 secciones. La primera denominada Introducción al problema 5.1, nos habla sobre porque es importante tratar los datos de los sensores en el desarrollo de este proyecto. La segunda sección llamada Cálculo de la velocidad 5.2, describe como se obtiene la velocidad a partir de la aceleración. La tercera sección denominada Calibración de los sensores 5.3, comenta tres técnicas diferentes para conseguir eliminar la componente gravitacional de la aceleración. La sección Pruebas de calibración 5.4, pone las técnicas descritas en el apartado anterior a prueba. Finalmente, la sección Eliminación de ruido mecánico 5.5, describe un método para eliminar ruido de los sensores.

5.1. Introducción al problema

Se supone el siguiente escenario. El usuario llevará un dispositivo wearable atado a la muñeca y se debe calcular la potencia desarrollada por el usuario en base a los datos que puede aportar el wearable. Como se ha presentado en el capítulo 1 de esta memoria, se deberá calcular la velocidad y la fuerza con el fin de obtener la potencia.

Para calcular la velocidad se procederá a integrar la aceleración y para calcular la fuerza tambien se deberá utilizar la aceleración. Por lo que para el correcto funcionamiento de la aplicación y para que sea lo más precisa posible, se debe calcular con la máxima fiabilidad la aceleración.

5.2. Calculo de la velocidad

En esta sección se va a proceder a calcular la velocidad utilizando la integración matemática. Por lo que previamente, se va a definir la aceleración.

La aceleración se define como la variación de la velocidad por unidad de tiempo. Viene representada por:

$$a = \frac{dv}{dt}$$

Cuando la aceleración no es constante aparece una nueva magnitud denominada '*jerk*'. La constante jerk se define como la variación de aceleración respecto al tiempo. Viene representada por:

$$j = \frac{da}{dt}$$

Por lo que se deduce que jerk es la derivada de la aceleración. Para calcular la aceleración solo se debe integrar.

$$da = j * dt$$

$$\int_{a_0}^a da = \int_0^{\partial t} j * dt$$

$$a - a_0 = j * \partial t$$

$$a = a_0 + j * \partial t$$

Por otro lado se tiene que la aceleración es la derivada de la velocidad, por lo que:

$$a = \frac{dv}{dt}$$

$$dv = a * dt$$

$$dv = a_0 + j * \partial t dt$$

$$\int_{v_0}^v dv = \int_0^{\partial t} (a_0 + jt) dt$$

$$v - v_0 = a_0 * \partial t + \frac{1}{2} j \partial t^2$$

Teniendo así finalmente la ecuación de la velocidad:

$$v = v_0 + a_0 * \partial t + \frac{1}{2} j \partial t^2$$

Una manera de calcular la velocidad es la integral debajo de la curva, donde la integración es la suma de las áreas de longitud mínima.

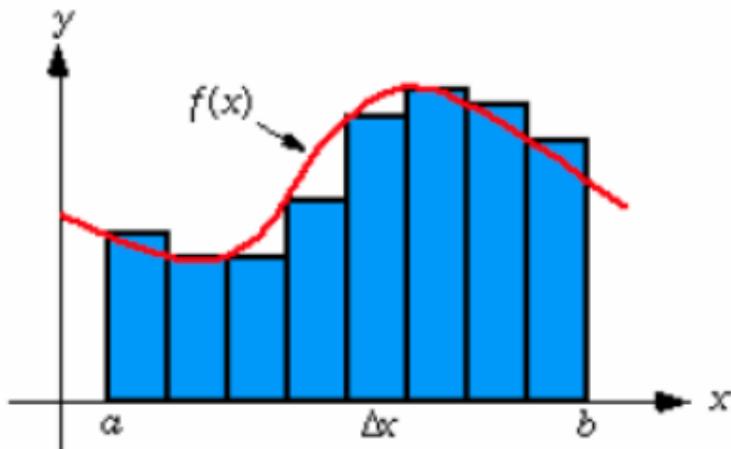


Figura 5.1: Señal del acelerómetro [1].

Donde:

$$\int_a^b f(x)dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(x_i)\partial x$$

$$\partial x = \frac{b-a}{n}$$

$$Area_n = Muestra_n + \frac{|Muestra_n - Muestra_{n-1}|}{2} * T$$

5.3. Calibración de los sensores

Los sensores aportan datos con diferentes ruidos y fluctuaciones como se pudo ver en la sección 2.2 de esta memoria. En este apartado se va a tratar de compensar y calibrar

el acelerómetro.

La salida del acelerómetro por norma general varía entre 0V y Vdd, y dicha salida es interpretada por un comparador (A/D). Por lo tanto el valor 0 de la aceleración deberá corresponder con Vdd/2.

Un acelerómetro por defecto esta sometido a la fuerza de la gravedad, así que por defecto la gravedad sobre el eje 'Y' será (situando el sistema de coordenadas sobre el eje de la tierra):

$$g = 9,81 \text{ m/s}^2$$

Por lo que será necesario compensar la aceleración con el fin de eliminar este valor, para ello se van a comparar y aplicar tres técnicas con el fin de averiguar cuál nos ofrece unos datos más precisos.

5.3.1. Compensación gravitatoria utilizando Sensor Fusion

Sensor Fusion se define como el uso de varios sensores con el fin de incrementar la precisión de las mediciones.

En este caso, se va a utilizar el algoritmo de Madgwick, el cual hace uso del acelerómetro, giroscopio y magnetómetro para calcular de manera computacionalmente eficiente la orientación del dispositivo IMU[23].

El algoritmo produce una representación de la orientación en forma de quaternion[24]. Para ello calcula la rotación que alinea la rotación del dispositivo con la de la tierra utilizando un algoritmo de gradiente descendiente. Se puede encontrar el algoritmo en el siguiente repositorio, bajo licencia GNU [25].

La baja carga computacional de este algoritmo y su capacidad de operar a frecuencias de muestreo bajas, lo hacen ideal para medir el movimiento en dispositivos wearable.

Una vez se ha obtenido la orientación del dispositivo se puede eliminar la componente gravitatoria de cada eje de coordenadas de una manera mucho más precisa, de la siguiente manera:

$$\text{quaternion} = q0, q1, q2, q3$$

$$\text{aceleracion}_x = \text{raw_acceleracion}_x - 2 * (q1 * q3 - q0 * q2);$$

$$\text{aceleracion}_y = \text{raw_acceleracion}_y - 2 * (q0 * q1 + q2 * q3);$$

$$\text{aceleracion}_z = \text{raw_acceleracion}_z - q0 * q0 - q1 * q1 - q2 * q2 + q3 * q3;$$

5.3.2. Filtro de paso bajo

Un filtro de paso bajo es un filtro electrónico que deja pasar señales de frecuencia baja y reduce la amplitud de las señales con frecuencias más altas que la frecuencia de corte.

Utilizando un filtro de paso bajo se puede aislar la fuerza de la gravedad de la aceleración y por lo tanto corregir el valor real de la aceleración.

Antes de aplicar el filtro, se debe definir una constante alfa como:

$$\alpha = t/(t + dT)$$

Siendo t la constante de tiempo del filtro y dT la tasa de llegada de datos.

$$gravity_0 = \alpha * gravity_0 + (1 - \alpha) * acceleration_x;$$

$$gravity_1 = \alpha * gravity_1 + (1 - \alpha) * acceleration_y;$$

$$gravity_2 = \alpha * gravity[2] + (1 - \alpha) * acceleration_z;$$

Eliminando así la componente gravitatoria de cada eje:

$$acceleration_x = raw_acceleration_x - gravity_0$$

$$acceleration_y = raw_acceleration_y - gravity_1$$

$$acceleration_z = raw_acceleration_z - gravity_2$$

5.3.3. Aceleración Lineal

La API SensorManager provee de un tipo de sensor denominado TYPE_LINEAR_ACCELERATION comentado en la sección del 2.3. Conceptualmente el sensor proporciona la aceleración siguiendo la siguiente norma:

$$acceleration_lineal = acceleration - gravedad$$

Se debe tener en cuenta que este sensor siempre presenta un offset(compensación), por lo que una rutina de calibración deberá ser necesaria, leyendo y eliminando el valor offset de los datos del sensor.

5.4. Pruebas de calibración

Se va a mostrar como se comportan los diferentes sensores en condiciones de parada y de movimiento (todos bajo las mismas circunstancias), durante un intervalo de 10 y 30 segundos respectivamente. Y se podrá observar el calculo de la velocidad obtenida en base a los distintos datos resultantes.

5.4.1. Condición de parada

Con una circunstancia de movimiento 0 del wearable. (Por lo tanto la aceleración ideal deberá ser 0). Se toman muestras para cada método:

Validación compensación gravitatoria utilizando Sensor Fusion

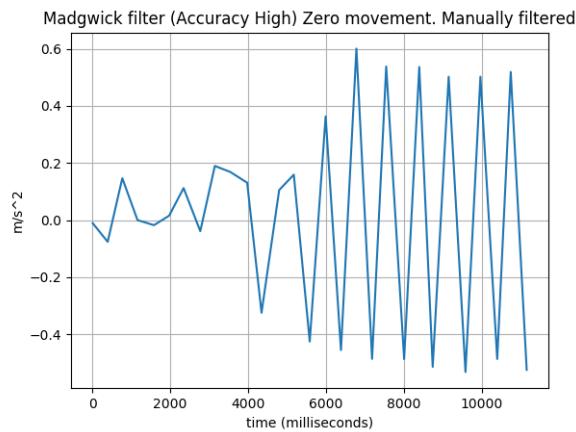


Figura 5.2: Compensación de la gravedad por medio del algoritmo de Madgwick (sin movimiento).

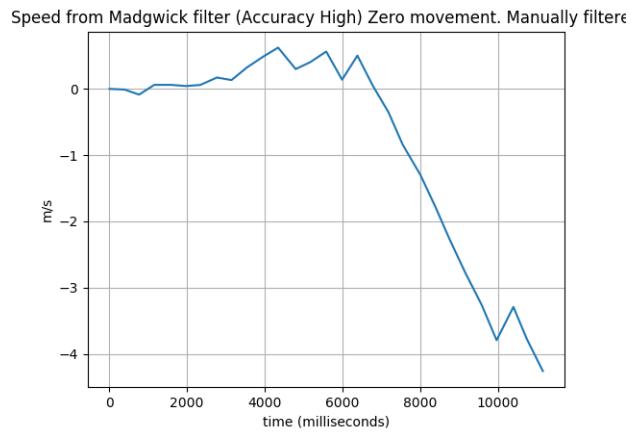


Figura 5.3: Velocidad calculada de la aceleración por medio del algoritmo de Madgwick (sin movimiento).

Como se puede ver, la aceleración varía rápidamente y su amplitud va aumentando segundo a segundo, lejos de ser estable y precisa.

Validación filtro de paso bajo

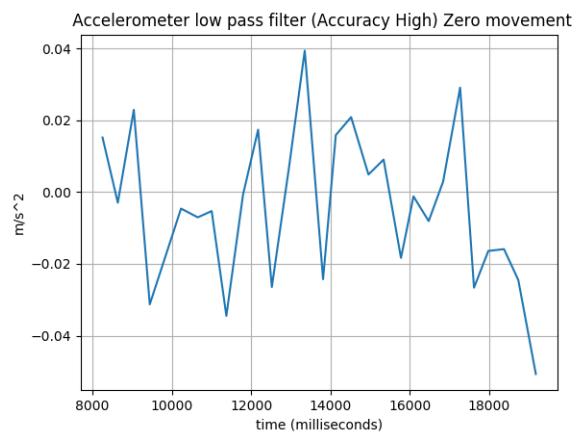


Figura 5.4: Compensación de la gravedad por medio de un filtro de paso bajo (sin movimiento).

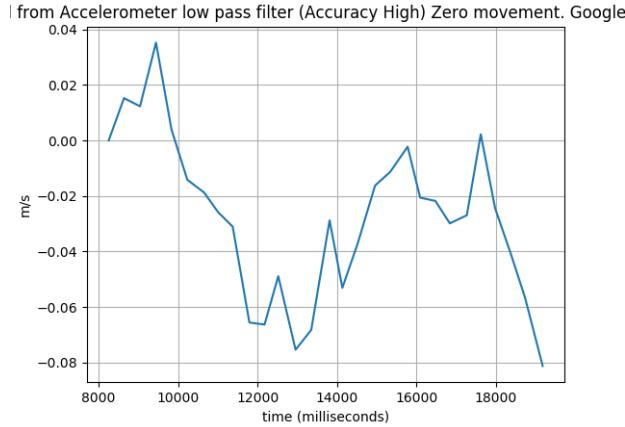


Figura 5.5: Velocidad calculada de la aceleración por medio de un filtro de paso bajo (sin movimiento).

Se puede observar como la aceleración varía de considerablemente menos que en el método anterior, teniendo una reducción de un -93,33 % en el valor máximo. Donde la diferencia es más notoria es en el cálculo de la velocidad, habiendo una diferencia a los 11000 ms de 4 m/s^2 a $-0,02 \text{ m/s}^2$.

Validación aceleración lineal

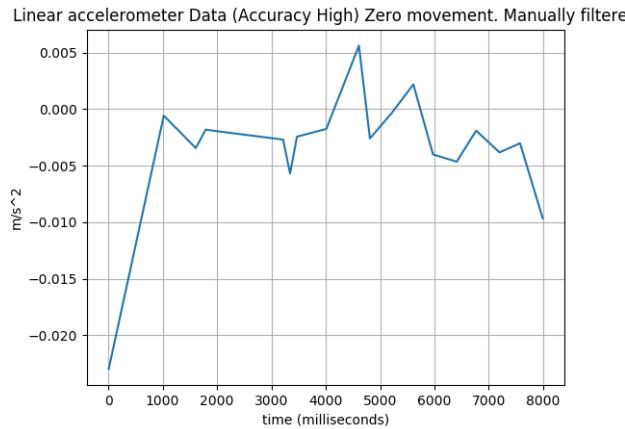


Figura 5.6: Compensación de la gravedad por medio de la aceleración lineal (API) (sin movimiento).

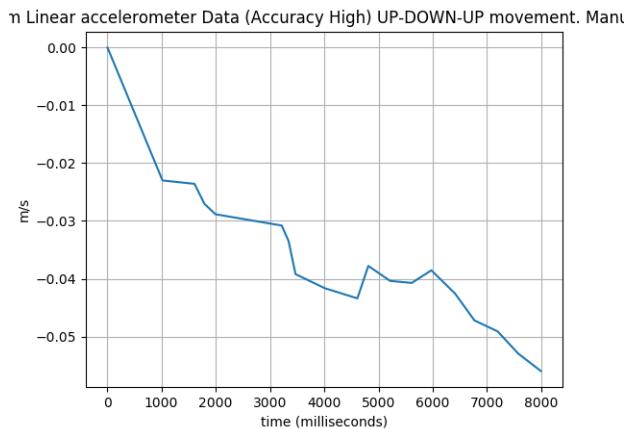


Figura 5.7: Velocidad calculada de la aceleración lineal (API) (sin movimiento).

En este caso, se aprecia como la aceleración máxima varía en un -82,61 % con respecto al filtro de paso bajo.

Su funcionamiento en líneas generales es muy parecido al del filtro de paso bajo.

5.4.2. Condición de movimiento

Esta vez se realizarán las pruebas en base al siguiente escenario. Circunstancia de movimiento de 60 cm en total a lo largo del eje Y (+30 cm y -30cm) del wearable y 30 segundos de tiempo de muestreo. Quedando el wearable apoyado sobre una superficie plana al final de la medición. Se debe tener en cuenta que la aceleración máxima producida es de $0,15\text{m/s}^2$.

Tomamos muestras para cada método:

Validación compensación gravitatoria utilizando Sensor Fusion

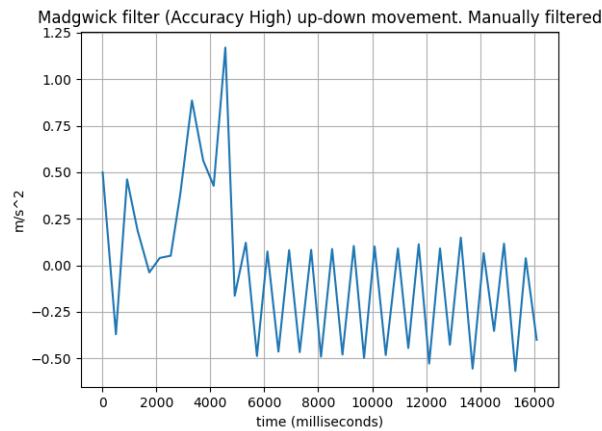


Figura 5.8: Compensación de la gravedad por medio del algoritmo de Madgwick (con movimiento vertical).

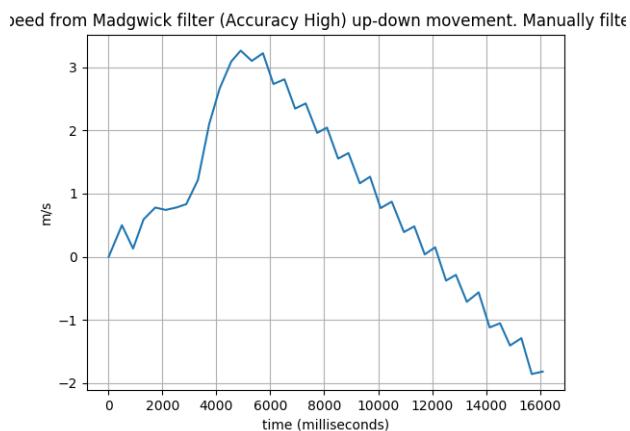


Figura 5.9: Velocidad calculada de la aceleración por medio del algoritmo de Madgwick (con movimiento vertical).

Como se puede apreciar, la aceleración no es nada estable, sobre todo cuando se encuentra en condición de parada. Aumentando la amplitud de la señal en función del tiempo.

Validación filtro de paso bajo

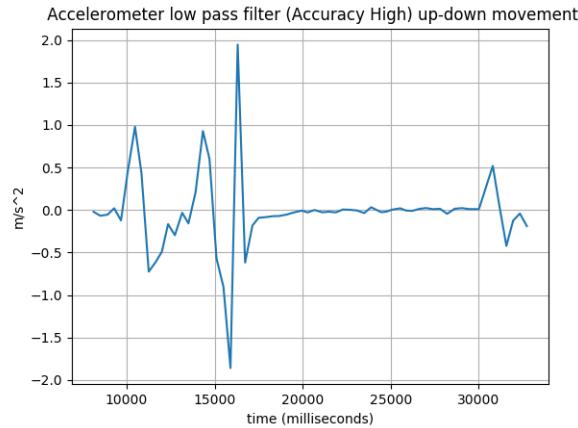


Figura 5.10: Compensación de la gravedad por medio de un filtro de paso bajo (con movimiento vertical).

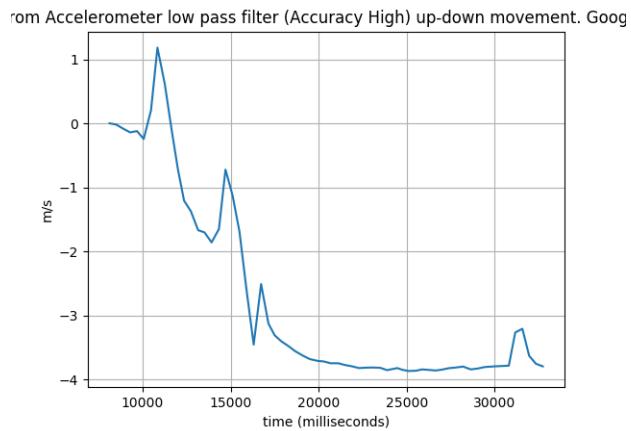


Figura 5.11: Velocidad calculada de la aceleración por medio de un filtro de paso bajo (con movimiento vertical).

Se puede ver como la aceleración esta vez no fluctúa cuando se encuentra en condición de parada, monitorizando correctamente cuando se encuentra en movimiento el dispositivo. Sin embargo, se aprecia como la aceleración en movimiento alcanza valores sustancialmente elevados y altos, no ajustados a la aceleración real.

Validación aceleración lineal

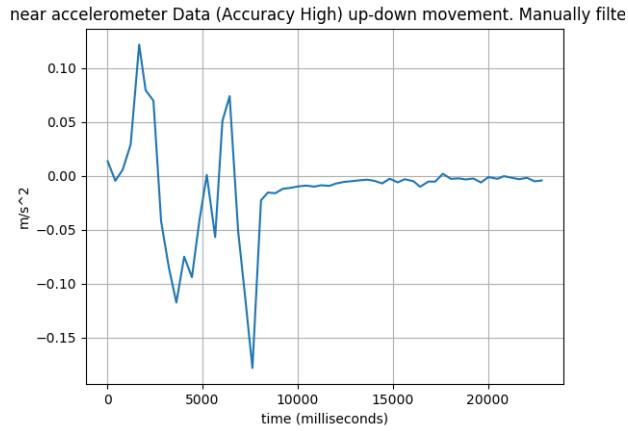


Figura 5.12: Compensación de la gravedad por medio de la aceleración lineal (API) (con movimiento vertical).

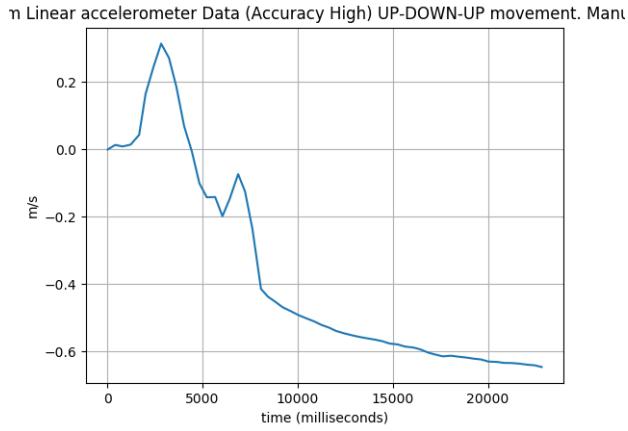


Figura 5.13: Velocidad calculada de la aceleración lineal (API) (con movimiento vertical).

En este caso, la aceleración sí varía entre unos baremos reales, ajustándose más a los valores reales que el resto de métodos. Por lo que a priori parece el método más adecuado para utilizar en el sistema.

5.5. Eliminación de ruido mecánico

Una vez se ha eliminado la componente gravitatoria de los datos, aún cuando el wearable se encuentra en condición de parada, el acelerómetro produce valores inestables.

Por lo que es necesario construir una ventana de discriminación con el fin de considerar los valores que son ruido y los valores que representan aceleración real. De esta manera se mejora la precisión de la aceleración y consecuentemente de la velocidad calculada.

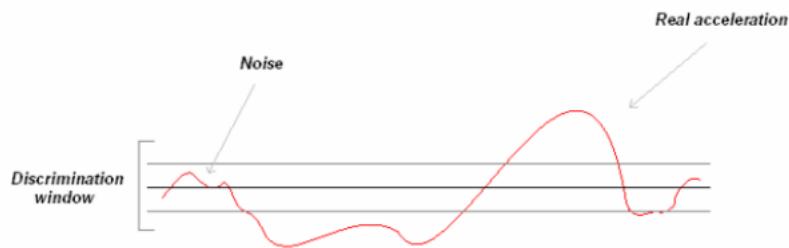


Figura 5.14: Ventana de discriminación [1].

Capítulo 6

Diseño

En este capítulo se va a tratar el diseño del sistema. Encontrando 5 secciones. La primera, Diseño del sistema 6.1, describe la arquitectura general del sistema. La segunda, Diagrama de clases de la aplicación móvil 6.2, describe atributos, los principales métodos y relaciones de la aplicación móvil. La tercera, Diagrama de clases de la aplicación wear 6.3, describe atributos, métodos y relaciones de la aplicación wear. La cuarta sección denominada Casos de uso 6.4, contiene los distintos casos de uso de los usuarios en el sistema. Finalmente, la sección Diseño de la base de datos 6.5, describe la estructura de la base de datos.

6.1. Diseño del sistema

Se pueden diferenciar tres partes principales que interactúan entre ellas. La primera una aplicación Android para el dispositivo wearable que será la encargada de recopilar información y enviarla, la segunda, una aplicación Android móvil la cual recibirá los datos, los procesará y los representará visualmente. La tercera parte formada por un servidor(Firebase), será la encargada de almacenar los datos del usuario y los datos que cree.

Así que la estructura del proyecto del cual trata este trabajo, seguirá el siguiente diseño:

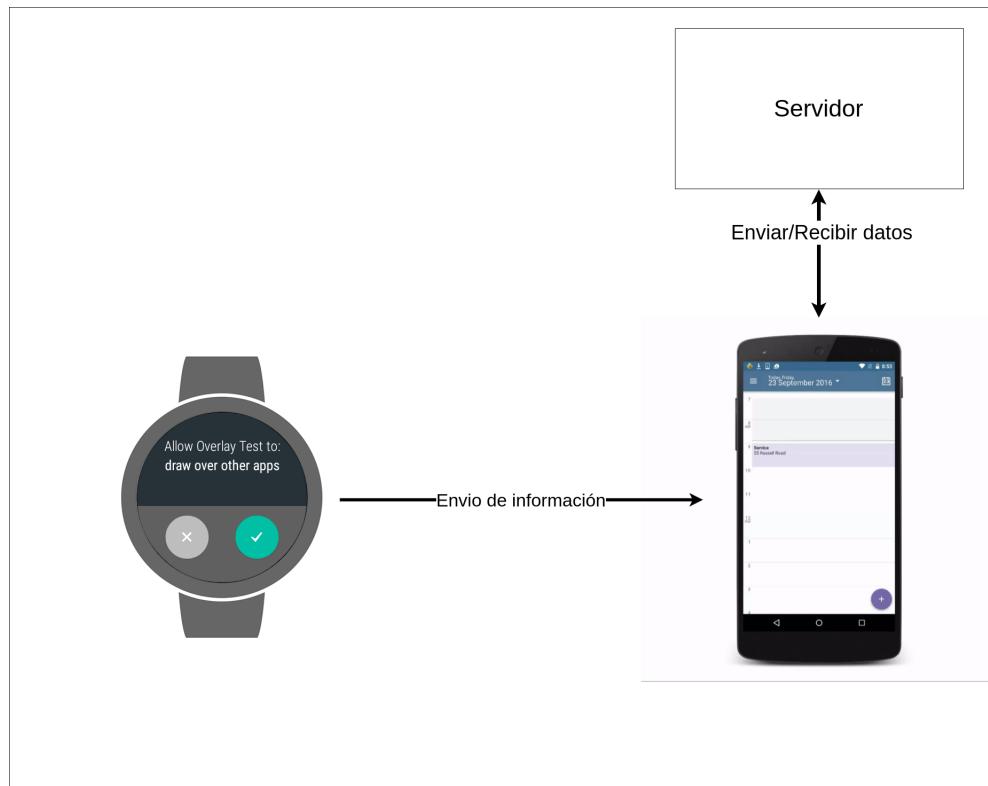


Figura 6.1: Arquitectura del proyecto.

En las siguientes secciones de este capítulo se tratará el diseño de cada uno de los componentes de la figura anterior.

6.2. Diagrama de clases aplicación móvil

Esta herramienta describe de manera gráfica utilizando el lenguaje UML las clases y sus relaciones dentro de la aplicación. Otorgando al sistema de todas sus funcionalidades.

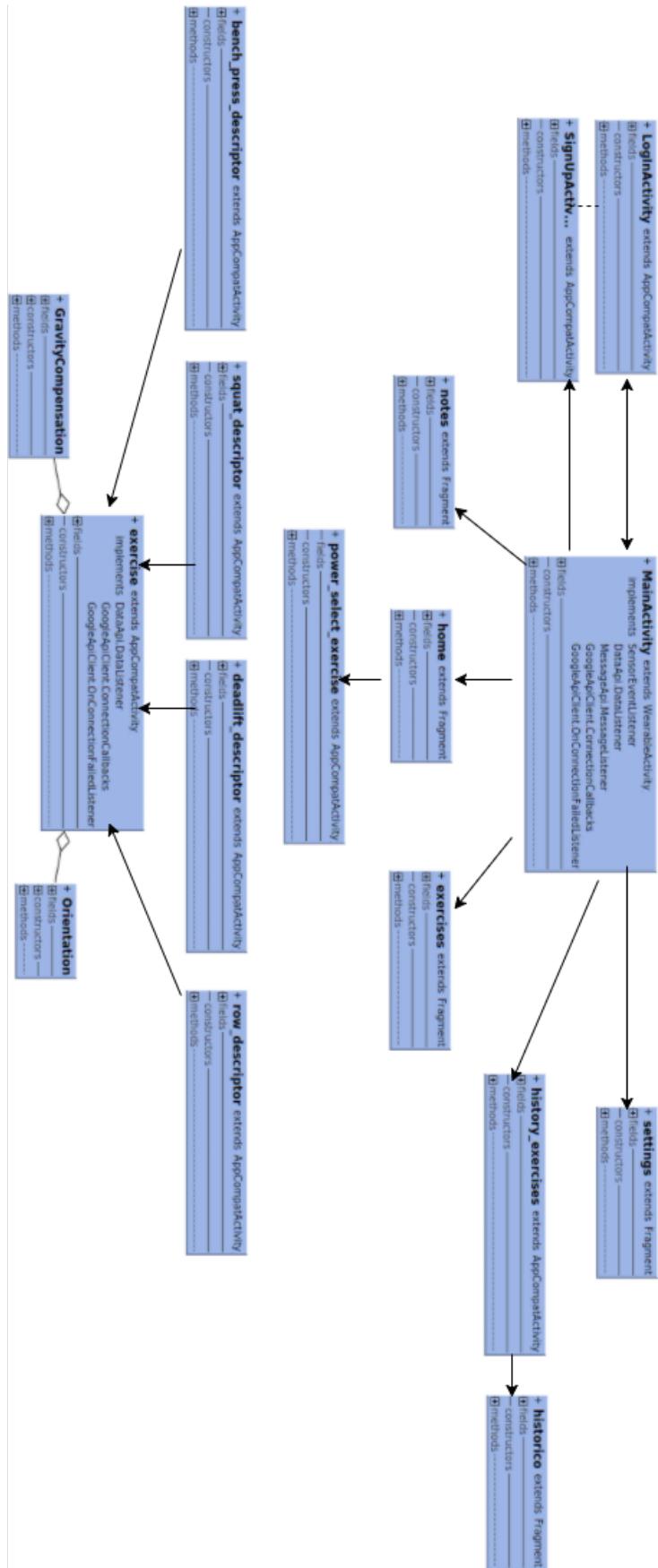


Figura 6.2: Diagrama de clases de la aplicación móvil.

6.3. Diagrama de clases aplicación wear

La finalidad de la aplicación wear es la de recoger y enviar la información, por lo que se ha decidido implementar toda su funcionalidad en una sola clase:

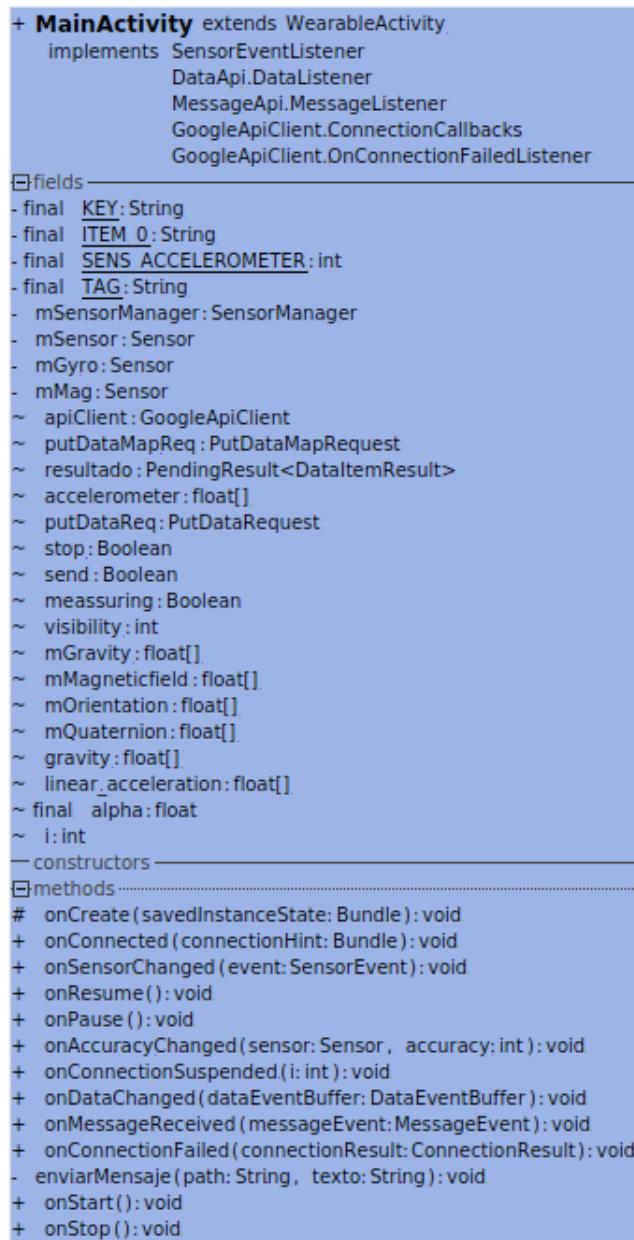


Figura 6.3: Diagrama de clases de la aplicación wear.

6.4. Casos de uso

En esta sección se va a detallar los casos de uso de la aplicación, los cuales encajan con los requisitos funcionales descritos en el capítulo 3 de este trabajo.

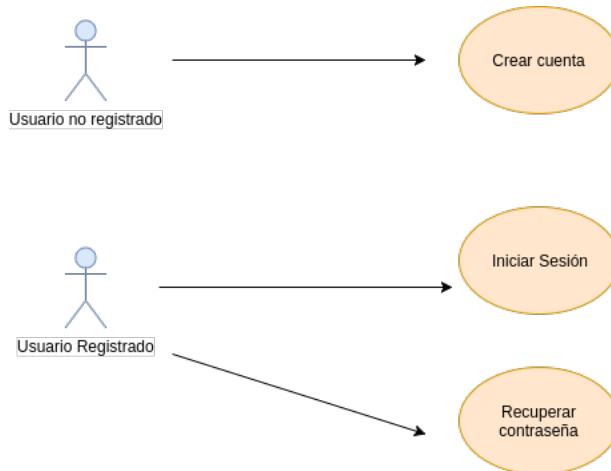


Figura 6.4: Caso de uso 1 - Credenciales.

Crear cuenta

Cuadro 6.1: Caso de uso - Crear cuenta

Titulo	Crear cuenta
Descripción	El usuario puede crearse una cuenta en la aplicación introduciendo el email y una contraseña.
Actor	Usuario no registrado
Precondición	Que el email no se encuentre ya en el sistema
Secuencia	<ul style="list-style-type: none"> - El usuario instala la aplicación. - El usuario entra en la aplicación. - Introducir los campos - Pulsar el botón de registro
Post condición	El usuario entra en la aplicación y ya dispone de credenciales
Comentarios	El campo email debe ser un email válido y accesible, pues se utilizará para restaurar la contraseña.

Iniciar sesión

Cuadro 6.2: Caso de uso - Inicio sesión

Título	Iniciar sesión
Descripción	El usuario podrá iniciar sesión utilizando los credenciales que creó previamente.
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Introducir los campos - Pulsar el botón de inicio de sesión
Post condición	El usuario se encuentra dentro de la aplicación
Comentarios	El usuario debe memorizar su email y contraseña.

Recuperar contraseña

Cuadro 6.3: Caso de uso - Recuperar contraseña

Título	Recuperar Contraseña
Descripción	El usuario podrá recuperar la contraseña utilizando los credenciales que creó previamente (email).
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Introduce el campo email - Pulsar el botón de recuperar contraseña <p>Accede a su email y recupera la contraseña</p>
Post condición	El usuario obtiene su contraseña
Comentarios	El usuario debe memorizar su email y contraseña.

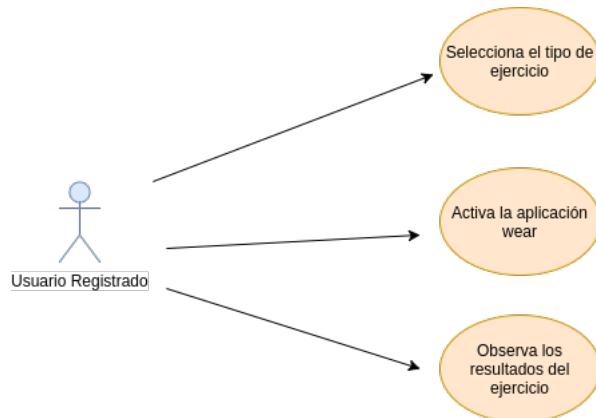


Figura 6.5: Caso de uso 2 - Realización del ejercicio.

Selección del tipo de ejercicio

Cuadro 6.4: Caso de uso - Seleccionar ejercicio

Titulo	Selección del tipo de ejercicio
Descipción	El usuario podrá seleccionar el tipo de ejercicio que va a realizar.
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Pulsar la pestaña de ejercicio - Selecciona el tipo de ejercicio deseado
Post condición	El usuario se encuentra con la descripción del ejercicio
Comentarios	El usuario debe leer la descripción para realizar el ejercicio correctamente y de manera segura

Inicio de la monitorización del ejercicio

Cuadro 6.5: Caso de uso - Inicio de la monitorización del ejercicio

Titulo	Activar la monitorización de la aplicación wear
Descipción	El usuario podrá indicar cuando va a comenzar el ejercicio.
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema y haya seleccionado un ejercicio.
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Pulsar la pestaña de ejercicio - Selecciona el tipo de ejercicio deseado - Inicia la aplicación wear. - Selecciona el botón de inicio.
Post condición	El usuario se encuentra realizando el ejercicio
Comentarios	El usuario debe leer la descripción para realizar el ejercicio correctamente y de manera segura

Visualización de los resultados

Cuadro 6.6: Caso de uso - Visualización de los resultados

Título	Visualización de los resultados
Descripción	El usuario podrá visualizar la potencia realizada en tiempo real.
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema, haya seleccionado un ejercicio y haya activado la aplicación wear.
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Pulsar la pestaña de ejercicio - Selecciona el tipo de ejercicio deseado - Inicia la aplicación wear. - Selecciona el botón de inicio. - Observa como se procesa la información en tiempo real.
Post condición	El usuario termina la serie.
Comentarios	El usuario debe leer la descripción para realizar el ejercicio correctamente y de manera segura

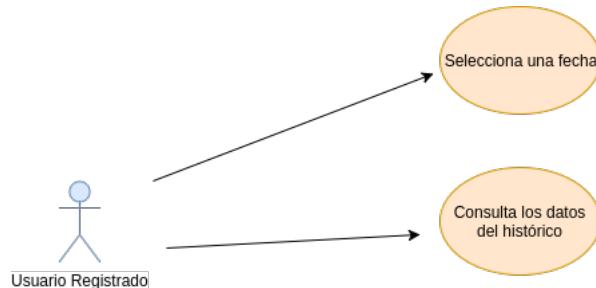


Figura 6.6: Caso de uso 3 - Consulta del histórico.

Selección de una fecha

Cuadro 6.7: Caso de uso - Selección de una fecha

Título	Selección de una fecha
Descripción	El usuario podrá seleccionar una fecha para realizar una consulta.
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema.
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Pulsar la pestaña de historico - Selecciona una fecha del calendario
Post condición	El usuario visualiza los datos para dicha entrada.
Comentarios	Si el usuario no realizó ninguna actividad durante dicho día no verá información.

Consulta de los datos del histórico

Cuadro 6.8: Caso de uso - Consulta de los datos del histórico

Título	Consulta de los datos del histórico
Descripción	El usuario podrá visualizar su entrenamiento de un día dado.
Actor	Usuario registrado
Precondición	Que el usuario se encuentre registrado en el sistema y haya seleccionado una fecha.
Secuencia	<ul style="list-style-type: none"> - El usuario entra en la aplicación - Pulsar la pestaña de historico - Selecciona una fecha del calendario -Visualiza sus datos el dia seleccionado
Post condición	El usuario visualiza los datos para dicha entrada.
Comentarios	Si el usuario no realizó ninguna actividad durante dicho día no verá información.

6.5. Diseño de la base de datos

Como se ha comentado en el capítulo 2 de este documento. La base de datos se va a almacenar en Firebase, que utiliza una estructura no-SQL, utilizando un formato JSON, por lo que se ha decidido almacenar la base de datos de la siguiente manera, con el fin de agilizar la escritura/lectura de los datos de una manera rápida:

Cuadro 6.9: Estructura de la base de datos

```
Usuarios : {  
    IdUsuario : {  
        FechaActual : {  
            Tipo ejercicio : {  
                Hora : {  
                    Datos: []  
                }  
            }  
        }  
    }  
}
```

Capítulo 7

Implementación

En esta sección vamos a tratar como se ha implementado el proyecto sobre el cual se basa esta memoria y sus principales módulos. Contiene 3 secciones. La primera llamada Hito 1: Asistente entrenamiento potencia 7.1, describe los pasos realizados para implementar el ecosistema de la aplicación móvil. La segunda, denominada Hito 2: Uso del servidor, describe los pasos realizados para implementar la conexión con el servidor. Finalmente, la sección Capturas 7.3, contiene capturas sobre la aplicación.

La arquitectura del proyecto sigue el siguiente esquema, el cual se ha desarrollado a lo largo de las dos historias de usuario:

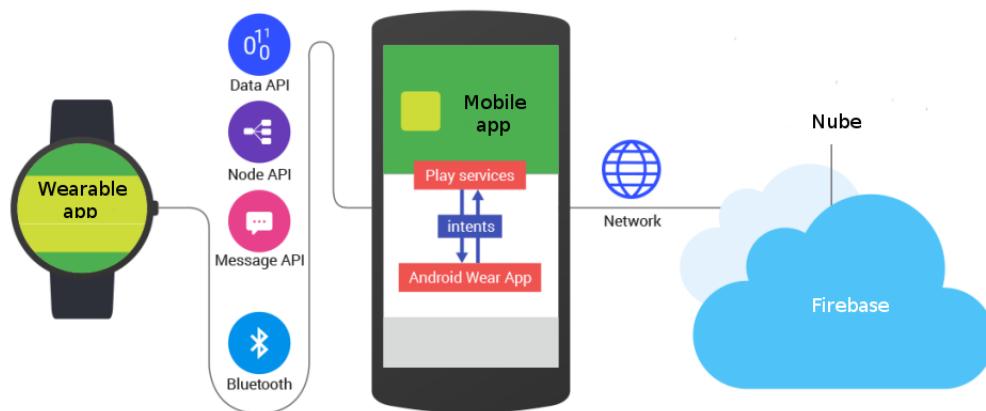


Figura 7.1: Diagrama de clases de la aplicación móvil.

El proyecto ha sido desarrollado en Android, por lo que la parte lógica ha sido

totalmente programada en Java y utiliza XML para crear las distintas vistas.

7.1. Hito 1: Asistente entrenamiento potencia

El primer paso fue la creación de dos aplicaciones básicas, las cuales incluían una actividad MainActivity, sin ninguna funcionalidad. Una vez creadas ambas aplicaciones había que establecer un cauce de comunicación entre ambas aplicaciones, para ello se ha utilizado la API GoogleApiClient.

Debemos tener en cuenta de que para realizar la conexión, debe haberse instalado previamente la aplicación Android Wear y debe estar enlazada al smartwatch que deseemos usar. Una vez hemos creado una instancia de GoogleApiClient, debemos conectar a la API Wearable API. Se debe establecer la conexión tanto en la aplicación wear como en la móvil para crear el cauce de comunicación.

Tras la creación de un cauce de comunicación entre ambas aplicaciones, se procedió a la obtención de los datos de los sensores. Para ello se utilizó la API de Android y el tipo de sensor TYPE_LINEAR_ACCELERATION. Que como se pudo ver en el capítulo 5, fue el método que menor error produjo en las muestras.

De manera paralela, se creó una selección de ejercicios disponibles en la aplicación móvil, añadiendo una pequeña descripción a cada ejercicio.

Una vez obtenidos los datos, se procedió al envío de los mismos en la aplicación Wear, haciendo uso del cauce que se había creado previamente. Utilizando la API DataApi, se creó un objeto DataItem, el cual mediante peticiones se utiliza para el envío de datos sincronizados.

A su vez, se procedió a la creación de un listener para el DataItem (utilizando también la API DataApi) en la aplicación móvil, de modo que los datos fueran recibidos en la aplicación móvil.

Todo este proceso se realizó sin ningún control del envío y recepción, siendo estos realizados de manera automática. Por lo que tras estos pasos, se procedió a crear una interfaz de control en la aplicación wear para poder iniciar y detener el envío de información. Dando por finalizada la aplicación Wear.

Estando la comunicación establecida y una vez recibidos los datos se procedió al filtrado de los datos por medio de una ventana de discriminación con el fin de eliminar parte del ruido de los datos.

Posteriormente, con los datos procesados, se procedió al cálculo de la velocidad utilizando un algoritmo de integración. Representando la velocidad calculada haciendo uso de la librería GraphView[26]. Cuyo código puede ser consultado en la página del desarrollador y se encuentra bajo licencia Apache v2.

Finalmente, se introdujo un campo para que el usuario introduzca el peso con el que va a realizar el ejercicio y con estos datos, se produjo a calcular la potencia obtenida en cada momento por el usuario.

Estando así la funcionalidad principal de la aplicación completa y totalmente operativa. Sin embargo, carecía de una interfaz gráfica que facilitase la experiencia del usuario. Por lo que se procedió a la creación de un menú en base a Bottom navigation de material design[27]. Utilizando fragments para las distintas vistas del menú.

7.2. Hito 2: Uso del servidor

En este hito se centró en ampliar la funcionalidad de la aplicación por medio de un servidor.

El primer paso fue crear un proyecto en la consola de Firebase.

Una vez importado el paquete Firebase en la aplicación móvil, se utilizó la librería auth para la creación de usuarios (previa activación en la consola de Firebase), inicio de sesión y mantenimiento de la sesión.

Para la obtención de credenciales se crearon dos vistas, una para el registro y otra para el inicio de sesión. En las cuales se obtienen los datos del usuario.

Se utiliza la librería database para las operaciones de escritura lectura de la base de datos, siguiendo el esquema que se diseño en la fase de diseño de la base de datos. Escribiendo en la base de datos al realizar el ejercicio y creando una nueva vista para recuperar los datos guardados en la base de datos, utilizando el criterio de la fecha como clave de búsqueda.

Además, cuando el sistema se encuentre sin conexión a la red y se esté realizando un ejercicio. El sistema guardará los datos en almacenamiento local, procediendo al envío de los mismos en cuanto se disponga de conexión y liberando así el espacio ocupado.

7.3. Capturas

Tras la finalización de los dos hitos, a continuación se adjuntan capturas de pantalla de la aplicación.

Android Wear

En la aplicación de Adroid Wear solo contamos con una vista, la cual va cambiando si el usuario quiere comenzar o terminar un ejercicio.



Figura 7.2: Pantalla principal Wear

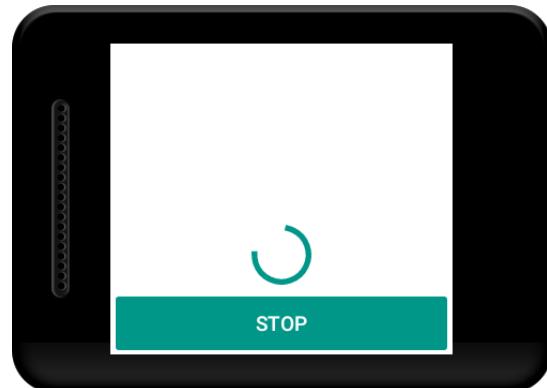


Figura 7.3: Realizar medición Wear

Android

Las vistas más importantes de la aplicación móvil son:

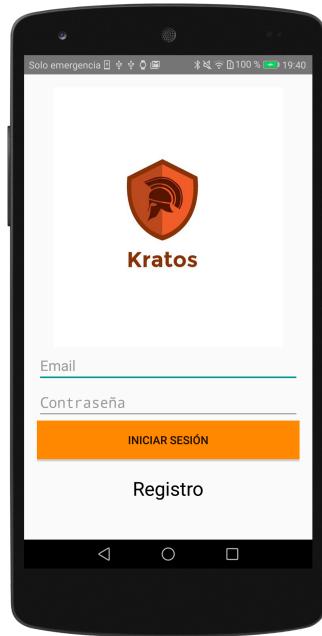


Figura 7.4: Pantalla inicio sesión móvil



Figura 7.5: Pantalla principal móvil

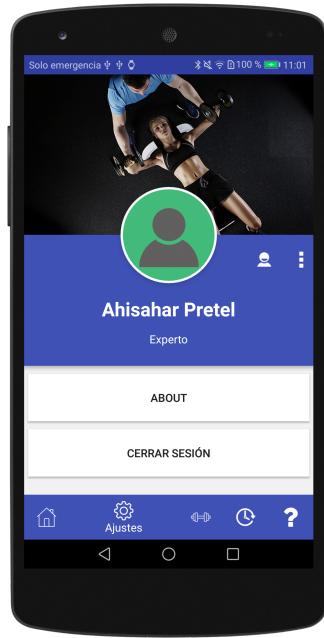


Figura 7.6: Vista menú móvil

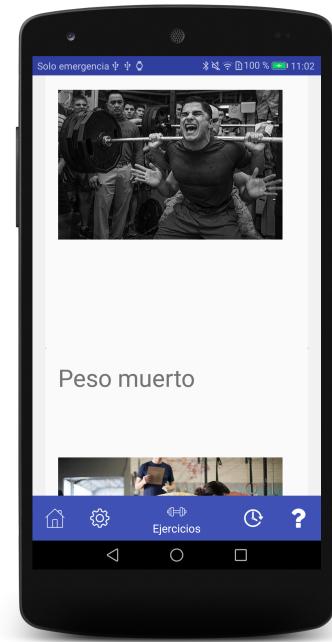


Figura 7.7: Vista selección ejercicio móvil



Figura 7.8: Vista descripción ejercicio móvil



Figura 7.9: Vista inicial ejercicio móvil

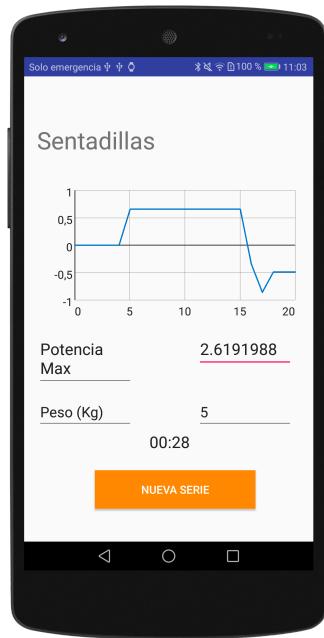


Figura 7.10: Vista realización ejercicio móvil

Capítulo 8

Pruebas

En este capítulo se va a analizar como se comporta la aplicación frente a un escenario real, con el fin de apreciar la fiabilidad con la que representa la potencia.

Para ello se han realizado unas mediciones con la aplicación en una máquina 'multi-power' (smith machine), utilizando una escala para medir la distancia con respecto al tiempo, haciendo uso de una cámara (no de alta velocidad). Se ha procedido a calcular la velocidad real media y la velocidad calculada, con el fin de ver cual es la diferencia. Puesto que el cálculo de la velocidad es el que más error implica en nuestro sistema.



Figura 8.1: Realización del primer test de la aplicación.

Test 1

Se realizó una serie con 3 repeticiones.

Cuadro 8.1: Test 1

Tiempo (s)	Velocidad media real (m/s)	Velocidad media calculada (m/s)	Diferencia
1	0,175	0	0,175
2	0,175	0,11	0,065
3	0,175	0,19	0,015
4	0,08	0,21	0,1225
5	0,08	0,35	0,27
6	0,262	0,35	0,08
7	0,087	0,76	0,67
8	0,22	0,28	0,06
9	0,17	0,28	0,11
10	0,25	0,43	0,18

Test 2

Esta vez se realizó una sola repetición. Para evaluar el comportamiento de la aplicación a baja velocidad.

Cuadro 8.2: Test 2

Tiempo (s)	Velocidad media real (m/s)	Velocidad media calculada (m/s)	Diferencia
1	0	0	0
2	0	0	0
3	0	0,190	0,190
4	0	0,26	0,26
5	0,05	0,26	0,21
6	0,10	0,26	0,16
7	0,076	0,15	0,12
8	0,05	0,15	0,10
9	0,05	0,15	0,10
10	0,03	0,27	0,24

Test 3

Se realizaron 15 repeticiones a lo largo de 40 segundos. Evaluando así el comportamiento de la aplicación en series largas.

Cuadro 8.3: Test 3

Tiempo (s)	Velocidad media real (m/s)	Velocidad media calculada (m/s)	Diferencia
0	0	0	0
5	0,66	0,57	0,09
10	0,96	0,61	0,35
15	0,56	1,43	0,87
20	0,54	1,16	0,62
25	0,24	1,94	1,7
30	0,64	2,00	1,36
35	0,76	2,31	1,55
40	0,40	2,04	1,64

Se puede ver que la aplicación se comporta de manera correcta en tiempo real, sin retraso apreciable por el usuario. Sin embargo la aplicación parece ser poco precisa cuando la velocidad del ejercicio es muy baja y cuando se produce desaceleración. Además la diferencia entre los valores calculados y reales aumenta con el tiempo, produciendo así un error bastante elevado. Esto es debido a que la estimación de la velocidad actual por medio de la integración diverge alejándose de la realidad rápidamente debido al ruido.

A falta de más pruebas, la aplicación es funcional, aunque su cálculo de la potencia es aproximado y contiene un gran margen de error. Por lo que no podemos garantizar que la aplicación realice la monitorización de manera correcta, especialmente cuando el tiempo de la serie es elevado.

Capítulo 9

Conclusiones y trabajo futuro

Conclusiones

A la hora de realizar el proyecto, se partía con una serie de objetivos en mente, los cuales se debían conseguir para el correcto funcionamiento de la aplicación. Una vez finalizado el proyecto, no se puede afirmar que los objetivos se hayan conseguido. Pues aún habiendo obtenido una aplicación totalmente operativa y completa, no podemos garantizar que la monitorización del ejercicio sea correcta.

Esto es debido a que la desviación producida al calcular la velocidad diverge rápidamente, a causa del ruido en función del tiempo. Por lo que el error de offset entre la potencia real y potencia monitorizada, aumentará rápidamente conforme el tiempo de realización del ejercicio aumenta. Además, la frecuencia de muestreo en Android puede cambiar a lo largo del tiempo (recibiendo los datos antes o después de la tasa especificada) y no puede ser controlada [16], produciendo así incoherencia en el cálculo de la velocidad. Lo cual produce que la aplicación solo pueda ser usada en series de 1 a 3 repeticiones y, consecuentemente, inviable en la mayoría de entrenamientos.

Dicha desviación podría ser corregida realizando la asunción de patrones conocidos, como que el usuario realice una parada entre repeticiones. Pero en dicho caso la rutina del usuario estaría condicionada, por lo que no es viable.

Se debe destacar que para la realización de este trabajo fue necesario realizar un estudio sobre la monitorización del movimiento utilizando sensores en un escenario 3D, sus limitaciones y métodos de calibración. Implementando los métodos más prometedores y comparandolos. Sorprendentemente, el algoritmo de orientación de Madgwick, siendo el más prometedor de todos, fue el que peores resultados obtuvo. Aparentemente, sobre el sistema operativo Android parece no funcionar tan bien, debido principalmente a que la frecuencia de muestreo en Android es orientativa, el desarrollador no tiene control absoluto sobre la frecuencia de muestreo y además, varía con el tiempo. Lo cual afecta mucho a la precisión de algunos algoritmos de calibración y posicionamiento.

Trabajo futuro

La aplicación aún siendo funcional presenta una serie de limitaciones. Las cuales se podrán intentar resolver con las siguientes propuestas (ordenadas por prioridad):

- Mejora de la calibración de los sensores. Utilizando otras técnicas con el fin de obtener mayor precisión.
- Empleo de sensores de mayor calidad.
- Creación de un backend propio con una API Rest. Si bien es cierto que Firebase nos proporciona muchas funcionalidades y servicios. El plan gratuito del servicio habilita hasta 100 conexiones simultáneas. Lo cual se traduce en un cuello de botella.
- Mejora de las interfaces de usuario. Tanto de la aplicación móvil como wear, con el fin de mejorar la experiencia de usuario.

Bibliografía

- [1] NXP Implementing Positioning Algorithms Using Accelerometers. <https://www.nxp.com/docs/en/application-note/an3397.pdf/>, Accedido el 21 de Noviembre de 2017.
- [2] El Androide. <https://elandroidelibre.elespanol.com/2015/01/como-instalar-cualquier-aplicacion-en-android-wear.html>, Accedido el 22 de Noviembre de 2017.
- [3] Paton C.D and Hopkins W.G. Combining explosive and high-resistance training improves performance in competitive cyclists. Technical report, Journal of Strength and Conditioning Research 826–830, 2005.
- [4] Docherty D. and Sporer B. A proposed model for examining the interference phenomenon between concurrent aerobic and strength training. Technical report, Sports Medicine Volume 30, Issue 6, pp 385–394, 2000.
- [5] Robert C. Hickson. Interference of strength development by simultaneously training for strength and endurance. Technical report, University of Illinois at Chicago, 1980.
- [6] Brad J Schoenfeld. The mechanisms of muscle hypertrophy and their application to resistance training. Technical report, Journal of Strength and Conditioning Research 24, 2010.
- [7] Gord Sleivert John Cronin. Challenges in understanding the influence of maximal power training on improving athletic performance. Technical report, Sports Medicine pp 213–234, 2005.
- [8] Prue Cormie, Grant O McCaulley, N Travis Triplett, and Jeffrey M McBride. Optimal loading for maximal power output during lower-body resistance exercises. 39:340–9, 02 2007.
- [9] MuscleLab. <http://www.ergotest.com/?product=product-5>, Accedido el 22 de Noviembre de 2017.
- [10] Push. <https://www.trainwithpush.com/push-band/>, Accedido el 22 de Noviembre de 2017.

- [11] Beast. <https://www.thisisbeast.com/en/products>, Accedido el 22 de Noviembre de 2017.
- [12] Google docs. Googleapiclient. Technical report, Alphabet Inc., Accedido el 7 de noviembre 2017.
- [13] Google docs. Sending and syncing data android wear. Technical report, Alphabet Inc., Accedido el 7 de noviembre 2017.
- [14] Google docs. Sending and syncing data android wear. Technical report, Alphabet Inc., Accedido el 7 de noviembre 2017.
- [15] Google. Firebase. Technical report, Alphabet Inc.
- [16] Google docs. Motion sensors. Technical report, Alphabet Inc.
- [17] Wikipedia. Accelerometer. Technical report, Wikipedia foundation.
- [18] Google docs. Motion sensors. Technical report, Alphabet Inc.
- [19] The Guardian. <https://www.theguardian.com/environment/2013/jan/13/lifespan-laptop-pc-planned-obsolescence>, Accedido el 22 de Noviembre de 2017.
- [20] The New York Times. <https://www.nytimes.com/2014/03/13/technology/personaltech/the-radical-concept-of-longevity-in-a-smartphone.html>, Accedido el 22 de Noviembre de 2017.
- [21] Android Authority. <https://www.androidauthority.com/long-smart-watch-last-403968/>, Accedido el 22 de Noviembre de 2017.
- [22] Indeed. <https://www.indeed.es/salaries/desarrollador/a-junior-salaries>, Accedido el 21 de Noviembre de 2017.
- [23] Sebastian O. H. Madgwick ; Andrew J. L. Harrison ; Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. Technical report, IEEE, 2011.
- [24] Jack B.Kuipers. Quaternion and rotation sequences. 1999.
- [25] Will Tan. <https://github.com/twillzy/senses-tw>, Accedido el 22 de Noviembre de 2017.
- [26] Jonas Gehring. <http://www.android-graphview.org>, Accedido el 21 de Noviembre de 2017.
- [27] Material Design. <https://material.io/guidelines/components/bottom-navigation.html#bottom-navigation-specs>, Accedido el 21 de Noviembre de 2017.

Apéndice A

Manual de usuario

A.1. Instalación

Para poder instalar la aplicación será necesario disponer de una versión de Android igual o superior a la versión 4.4 de Android, puesto que necesitamos compatibilidad con Android Wear. Se debe además disponer de un dispositivo que utilice Android Wear 1.0 o superior.

Móvil

Para instalar la aplicación en un terminal se debe transferir a dicho terminal el archivo apk. Una vez se encuentra en el dispositivo, se debe habilitar la instalación desde orígenes desconocidos. Para ello se deberá acceder a la opción de ajustes/seguridad y activar la opción de aplicaciones de origen desconocido.

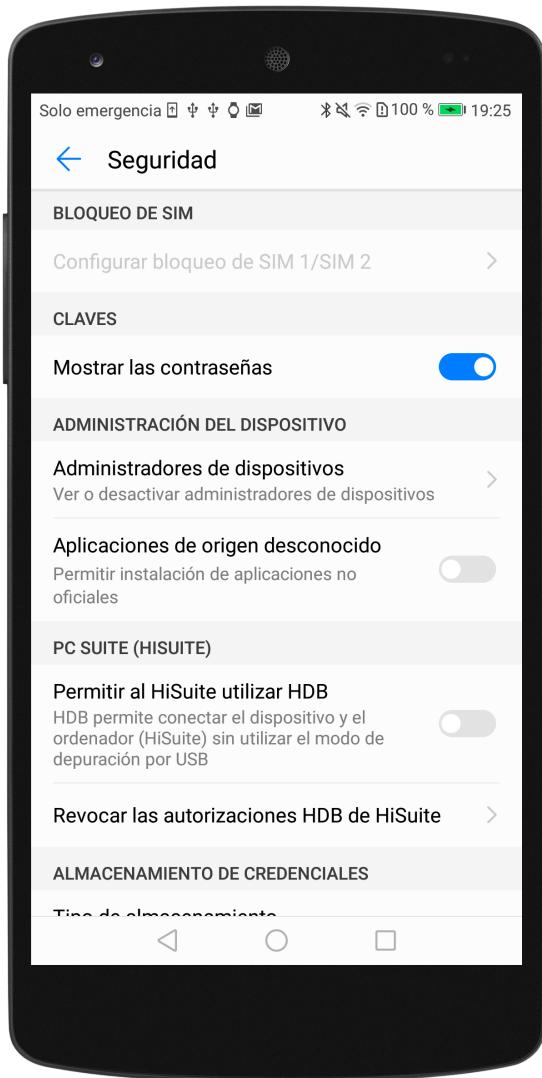


Figura A.1: Instalación desde origen desconocido.

Una vez esta la opción habilitada, simplemente se tendrá que seleccionar el apk desde el explorador de archivos por defecto y se procederá a su instalación.

Wear

Para instalar la aplicación en el dispositivo wearable, se deberá activar las opciones de desarrollador del dispositivo, para ello se debe acceder a la opción de información y seleccionando repetidas veces sobre el número de compilación se podrá acceder a las opciones de desarrollador. Una vez activadas las opciones de desarrollador se deberá permitir la depuración en el dispositivo.

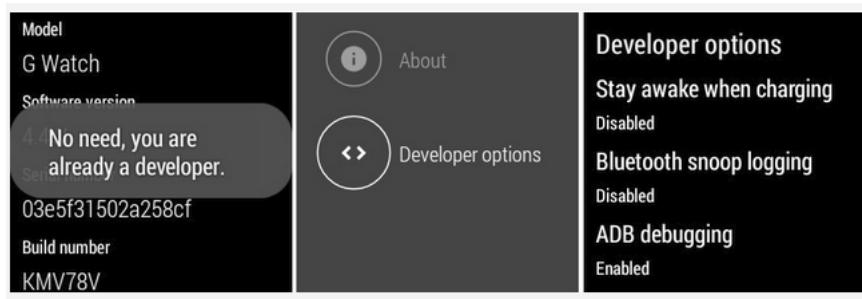


Figura A.2: Modo depuración Android Wear [2].

Con el archivo apk en el ordenador y en el directorio actual, solo quedará ejecutar en un terminal:

```
$adb forward tcp:4444 localabstract:/adb-hub  
$adb connect localhost:4444  
$adb -e install app.apk
```

A.2. Creación de usuario

Para crear un usuario simplemente se debe abrir la aplicación, hacer click en Registro y llenar las credenciales necesarias.

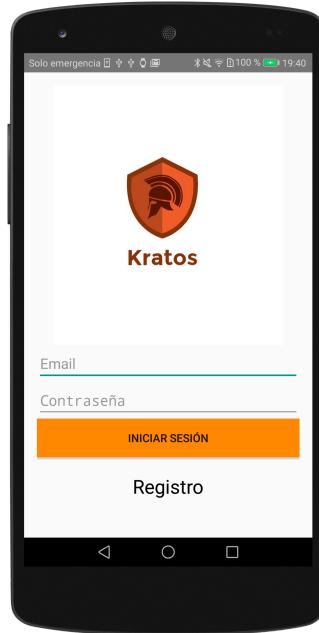


Figura A.3: Registro 1.

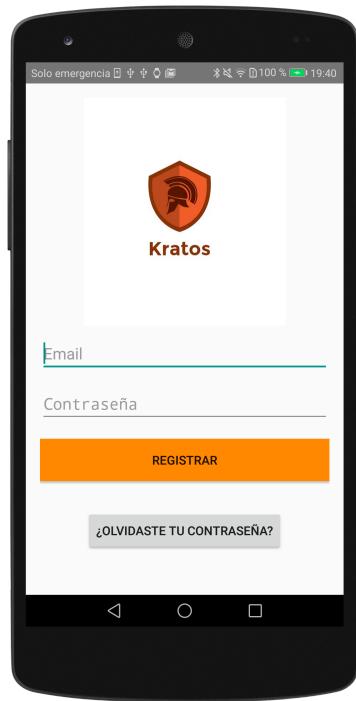


Figura A.4: Registro 2.

A.3. Realización ejercicio

Para realizar la monitorización del ejercicio se debe seleccionar la pestaña ejercicio, seleccionar el ejercicio deseado y empezar el ejercicio. De manera paralela una vez hayamos llegado a la Ejemplo de monitorización, activaremos la aplicación wear y se procederá a realizar el ejercicio.

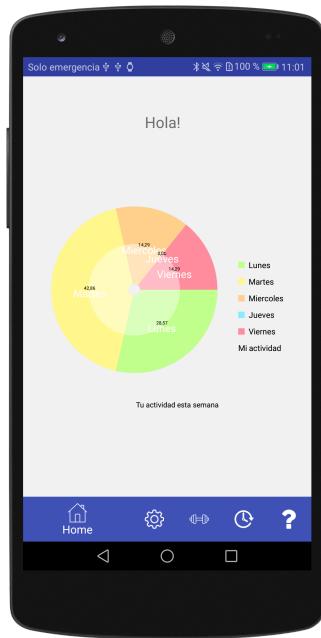


Figura A.5: Pantalla principal móvil

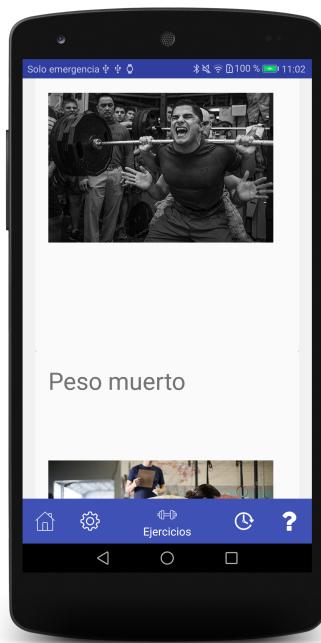


Figura A.6: Ejemplo selección ejercicio móvil



Figura A.7: Ejemplo descripción ejercicio móvil

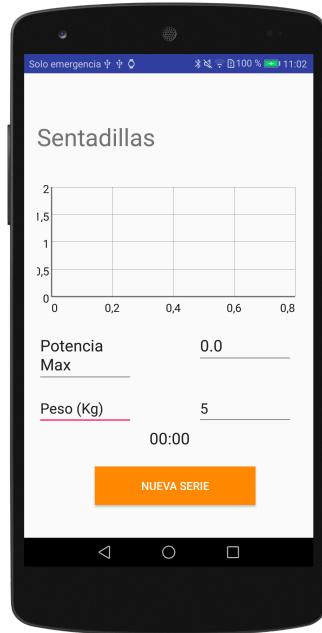


Figura A.8: Ejemplo iniciar ejercicio móvil

Se debe activar la aplicación wear e iniciar el ejercicio:



Figura A.9: Pantalla principal wear



Figura A.10: Realizar medición wear

Finalmente monitorizando así el ejercicio

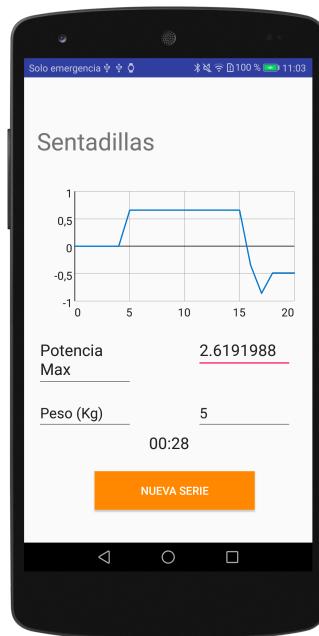


Figura A.11: Ejemplo realización ejercicio móvil

A.4. Consulta del histórico

Para realizar una consulta simplemente se debe que seleccionar en el menú de navegación la pestaña histórico y seleccionar un día del calendario.



Figura A.12: Consulta del histórico