



UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

Trabajo Fin de Grado

**“Sistema de monitorización continua de
datos fisiológicos para ámbitos de salud
basado en internet de las cosas”**

Alumno: Yang Chen

Tutor: Juan Antonio Holgado Terriza

Noviembre 2022

Sistema de monitorización continua de datos fisiológicos para ámbitos de salud basado en internet de las cosas

Yang Chen

Palabras clave: sistemas de monitorización, plataforma de monitorización, internet de las cosas, dispositivos inteligentes

Resumen

Hoy en día, los entornos de fuerte estrés y con enormes responsabilidades nos llevan a vivir de una forma menos saludables, por lo que resulta importante desarrollar aplicaciones o sistemas que nos permiten monitorizar la salud.

En este trabajo se va a desarrollar una plataforma de monitorización basado en internet de las cosas. El usuario llevará puesto unos dispositivos wearables que recopilarán los datos relacionados con su salud, tales como el peso, la frecuencia cardíaca, el valor de Spo2. Dichos datos, serán recogidos y almacenados por el sistema.

Posteriormente, a través de una herramienta de visualización, el usuario podrá ver sus datos de manera clara y sencilla, con la finalidad de poder hacer un seguimiento de la evolución de su salud.

Continuous monitoring system of physiological data for health based on Internet Of Things

Yang Chen

Keywords: monitoring system, monitoring platform, Internet Of Things, smart devices

Abstract

Nowadays, high stress environments and enormous responsibilities lead us to live in a less healthy way, so it is important to develop applications or systems that allow us to monitor health.

In this project, we are going to develop a monitoring platform based on the Internet of Things. The user will wear wearable devices that will collect data related to their health, such as weight, heart rate, Spo2 value. These data will be collected and stored by the system.

Subsequently, through a visualization tool, the user will be able to see their data in a clear and simple way, in order to be able to monitor the evolution of their health.

Yo, **Yang Chen**, alumno de la titulación Ingeniería de Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con NIE X7269417Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Yang Chen

Granada a 16 de Noviembre de 2022 .

D. Juan Antonio Holgado Terriza, Profesor de Lenguajes y Sistemas Informáticos del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada

Informan:

Que el presente trabajo, titulado Sistema de monitorización continua de datos fisiológicos para ámbitos de salud basado en internet de las cosas, ha sido realizado bajo su supervisión por **Yang Chen**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informen en Granada a 16 de noviembre de 2022.

Los directores:

Juan Antonio Holgado Terriza

Índice

Capítulo 1. Introducción	5
1.1 Motivación	5
1.2 Objetivos	6
1.2.1 Objetivo principal.....	6
1.2.2 Objetivos específicos.....	7
1.3 Estructura del trabajo	7
Capítulo 2. Estudio de la actividad física y el estado de salud de una persona	9
2.1 Medición de la actividad física personal y de la salud	9
2.2 Dispositivos wearables.....	11
2.3 Estudio de las variables fisiológicas.....	12
2.4 Estudio de dispositivos wearables del mercado	19
2.4.1 Relojes/pulseras inteligentes	19
2.4.2 Balanzas inteligentes	25
Capítulo 3. Estudio de las plataformas de monitorización continua.....	29
3.1 Sistemas de monitorización continua	29
3.2 Arquitectura para la monitorización continua	31
3.2.1 ¿Qué son los patrones arquitectónicos?	31
3.2.2 Patrón arquitectónico en capas	32
3.2.3 Patrón arquitectónico cliente-servidor	33
3.2.4 Arquitectura de sistemas de monitorización continua.....	35
3.3 Estudio de plataformas de monitorización continua	35
3.3.1 Base de datos	35
3.3.2 Herramientas de visualización de datos	44
3.4 Despliegue de la plataforma de monitorización	45
3.4.1 Instalación y configuración de TimescaleDB	45
3.4.1.1 Instalación	45
3.4.1.2 Configuración.....	45
3.4.2 Instalación y configuración de Grafana	46

3.4.2.1 Instalación	46
3.4.2.2 Configuración.....	46
Capítulo 4. Dispositivos basados en Withings.....	49
4.1 Estudio de los dispositivos wearables Withings	49
4.2 Infraestructura de Withings.....	52
4.3 Parámetros medibles	55
4.4 Despliegue e integración de la infraestructura Withings	59
4.5 Estudio de parámetros medibles por dispositivos Withings.....	60
4.5.1 Acceso a los datos con Postman.....	60
4.5.2 Medidas relacionadas con el reloj	64
4.5.2.1 Medidas de actividad física	64
4.5.3 Medidas relacionadas con la balanza	66
4.5.4 Clasificación de las medidas	66
4.5.4.1 Medidas de tipo resumen.....	67
4.5.4.2 Medidas instantáneas.....	68
4.5.4.3 Medidas a almacenar.....	68
Capítulo 5. Sistema MONITOR	71
5.1 Especificación informal de la plataforma.....	71
5.2 Modelado de requisitos	72
5.2.1 Requisitos funcionales.....	72
5.2.2 Requisitos no funcionales.....	72
5.2.3. Especificación del sistema.	72
5.2.3.1 Actores del sistema.....	73
5.2.3.2 Casos de uso	73
5.3 Modelado conceptual	77
5.4 Diseño del sistema MONITOR	81
5.4.1 Arquitectura del sistema.....	81
5.4.2 Diagrama entidad relación	82
5.4.3 Prototipo de la interfaz del sistema	84
5.5 Implementación del sistema	85
5.5.1 Capa de percepción.	85

5.5.2 Capa de red.....	86
5.5.3 Capa de procesamiento y almacenamiento	86
5.5.3 Capa de visualización.....	100
5.5.3.1 conexión al TimescaleDB	100
5.5.3.2 Creación de Dashboard	102
5.4 Dashboard de Grafana.....	104
5.5 Pruebas de la plataforma	107
5.5.1 Pruebas unitarias	107
5.5.2 Pruebas de integración	108
5.5.3 Pruebas de funcionamiento	110
Capítulo 6. Conclusiones	113
6.1 Conclusión general.....	113
6.2 Análisis de los objetivos del proyecto.....	115
6.3 Conclusiones personales	116
Capítulo 7. Bibliografía	117
Anexo 1. Gestión y planificación del proyecto.....	119
Anexo1.1 Metodología para el desarrollo del proyecto	119
Anexo1.2 Gestión del proyecto	119
Anexo1.3 Planificación del proyecto.	124
Anexo1.4 costes del proyecto	124
Anexo 2. Manual de uso	125
Anexo 2.1 Obtención de los tokens.....	125
Anexo 2.2 Ejecución del programa	126
Anexo 2.3 Visualización de datos	128
Anexo 3. Continuación de la sección 4.5.2 y sección 4.5.3.....	129

Capítulo 1. Introducción

1.1 Motivación

Hoy en día, vivimos en un mundo llena de nuevas tecnologías y de comidas pocas saludables, estos hacen que nuestros hábitos sean cada vez menos saludables. Los teléfonos móviles, los ordenadores y las consolas de videojuegos nos hacen pasar la mayor parte del tiempo sentados en las sillas o en el sofá, los coches y las motos nos hacen caminar menos a la hora de tener que ir a trabajar o a hacer compras. Como consecuencia, hacemos cada vez menos ejercicios y comemos cada vez más las comidas rápidas por la falta de tiempo en el trabajo. Todos estos hacen que un cuerpo o una vida saludable se convierte en algo muy importante para nosotros.

¿Pero qué es la salud? ¿Por qué la salud es tan importante para nosotros? La palabra salud proviene del latín ‘salus-utis’, cuyo significado es ‘salvación’ o ‘saludo’. La salud es un estado ideal de un individuo, un estado perfecto que es capaz de realizar de forma eficiente las funciones vitales, y libre de las enfermedades. Según la OMS (Organización Mundial de la Salud), “la salud es un estado de completo bienestar físico, mental y social, y no solamente la ausencia de afecciones o enfermedades”. Tener una vida saludable nos aporta muchos beneficios, nos permite tener una vida feliz y plena, rendir de forma máxima en nuestras actividades cotidianas, pero el más importante es que nos mantiene libre de las enfermedades, y como consecuencia, tener una vida más larga y de mejor calidad.

Ahora bien, si la salud es tan importante, ¿Cómo podemos saber en qué estado se encuentra nuestro cuerpo? ¿Cómo podemos medir la salud? Existen una serie de

parámetros fisiológicos que están relacionadas con la salud. Algunos de los parámetros más importantes son: la frecuencia cardíaca, presión arterial, temperatura corporal, la saturación de oxígeno, etc. Estos parámetros nos permiten determinar en qué estado de salud se encuentra una persona.

Hace años, cuando las personas querían medir algunos de los parámetros fisiológicos, tenían que acudir a un hospital o a algún lugar especializado que contara con los instrumentos específicos destinados a la medición de dichos parámetros. Esto no solo hacía que las personas se sintieran molestas por los desplazamientos que había que hacer cada vez que querían saber su estado de salud o incluso tener esperar largas colas para ser atendido, sino que también por el pago que había que asumir en algunos casos, lo que a largo plazo podía convertirse en un coste importante para algunas personas.

En la actualidad todo esto ha cambiado. Las nuevas tecnologías no solo nos hacen insanos, sino también tienen sus lados positivos, ya que permiten mejorar nuestros estilos de vida. Con los dispositivos corporales (wearables en inglés) como relojes inteligentes, las personas pueden medir sus parámetros fisiológicos en su casa en muy poco tiempo, e incluso pueden medirse de forma automática sin que tengamos que estar pendiente de ellos. Además de los dispositivos wearables existen también pequeños dispositivos electrónicos como las balanzas inteligentes que permiten obtener distintas medidas y almacenar los datos en la nube. Por tanto, para conocer nuestro estado de salud ya no tenemos que desplazarnos, y realizando un seguimiento de la evolución de estos parámetros podemos tener un mejor control sobre nuestra salud. En torno a estos dispositivos se ha desarrollado en el mercado una multitud de aplicaciones y plataformas de monitorización de la salud que beneficiara a las personas en general a tener una mejor percepción de lo que sucede con su salud.

Los entornos actuales de fuerte estrés y con enormes responsabilidades nos llevan a vivir de una forma menos saludable, por lo que resulta importante desarrollar plataformas y aplicaciones que permitan monitorizar la salud, en especial ante el incremento de población de adultos mayores que hay en los países occidentales.

1.2 Objetivos

1.2.1 Objetivo principal

El objetivo principal de este proyecto consiste en desarrollar una plataforma de monitorización continua basado en la recopilación de datos a través de los dispositivos inteligentes y su posterior visualización de dichos datos para posibilitar la supervisión de datos corporales fisiológicos.

La plataforma debe ser capaz de monitorizar las variables de la salud del usuario, extrayendo de forma continua los datos de los parámetros fisiológicos medidos por los distintos dispositivos a través de los APIs, y almacenarlos en la base de datos, permitiendo la visualización de la evolución de dichos datos en forma gráficas y, según distintos criterios, de manera que con la observación de dichos datos sea posible detectar algunas anomalías que surgen en nuestro cuerpo y prevenir la aparición de algunas enfermedades graves.

1.2.2 Objetivos específicos

Para lograr el objetivo principal se requiere cumplir con los siguientes objetivos específicos:

- Identificar los parámetros fisiológicos más significativos que determinan el estado de salud de una persona.
- Analizar los dispositivos wearables que hay en el mercado que puedan tener impacto en el conocimiento de parámetros fisiológicos.
- Estudiar los ecosistemas o plataformas que se utilizan para el registro de datos de dispositivos wearables que hay en el mercado.
- Desarrollar una plataforma para el registro de datos y poder supervisar la evolución de dichos datos.
- Diseñar un cuadro de mandos o dashboard que facilite la visualización de los datos fisiológicos.
- Aplicar buenas prácticas en el proceso de desarrollo y la implementación de la plataforma.

1.3 Estructura del trabajo

A continuación, vamos a presentar brevemente la estructura y los contenidos de la memoria. La memoria consta de varios capítulos, en concretos 8 capítulos.

El capítulo 1, donde encontramos ahora, es una pequeña introducción de la memoria, en la que se presentan la motivación del proyecto, una breve descripción tanto del objetivo principal como de los objetivos secundarios del trabajo, y también una explicación sobre la estructura de la memoria.

El capítulo 2 consiste en el estudio de las actividades físicas y el estado de las personas. Donde investigamos sobre cómo se puede medir la actividad física de una persona, cómo se puede medir el estado de la salud. Además, estudiamos también los dispositivos wearables que hay actualmente en el mercado y realizamos una comparación de ellos.

En el capítulo 3 se explica sobre la arquitectura de la plataforma, en la que se explica

los sistemas de monitorización y las plataformas de monitorización continua, también trata de los tipos de arquitecturas empleadas para el desarrollo de la plataforma.

El capítulo 4 es un estudio de los dispositivos que vamos a usar en el proyecto y también las infraestructuras que dispone para la recogida y almacenamiento de datos, y los parámetros que nos permiten medir.

El capítulo 5 es el desarrollo del proyecto, donde se especifican los requisitos, define la arquitectura, y se implementan el sistema y las distintas capas del sistema. También se han realizado una serie de pruebas sobre el funcionamiento del sistema.

El capítulo 6 se presenta la conclusión del trabajo y el análisis de los objetivos del proyecto.

Y finalmente en el capítulo 7 es donde encontramos todas las referencias utilizadas en el desarrollo del proyecto y la memoria.

Capítulo 2. Estudio de la actividad física y el estado de salud de una persona

2.1 Medición de la actividad física personal y de la salud

La actividad física hace referencia a todos aquellos movimientos que se producen cuando se mueve una persona o cuando realiza algún trabajo. Según la OMS, la actividad física es “*cualquier movimiento corporal producido por los músculos esqueléticos*” [oms21a]

La realización de la actividad física durante el día nos proporciona muchas ventajas desde el punto de vista de la salud. Una de las más importantes es la prevención de las enfermedades tales como el cáncer, cardiovasculares, etc. Por lo que se recomienda un nivel de realización de actividad para las personas de diferentes edades. Según la recomendación de OMS:

- Los niños y adolescentes de 5-17 años tiene que dedicar por lo menos 60 minutos al día a las actividades intensas, sobre todos aeróbicas.
- Para los adultos de 18-64 años hay que dedicar por lo menos de 150-300 minutos a la semana si se trata de actividades físicas aeróbicas moderadas, si son actividades físicas intensas, como mínimo de 17-150 minutos a la semana.

- Y para los adultos de 65 años o más la recomendación es la misma que para los adultos del grupo anterior. [oms21a]

Ahora bien, si la actividad física es tan importante para nuestra salud, ¿Cómo podemos medir las actividades físicas? ¿Cómo podemos monitorizar las actividades físicas que realizamos a lo largo del día?

Hoy en día, hay muchas herramientas que nos pueden servir para medir las actividades físicas, como los relojes, anillos inteligentes e incluso simplemente a través de un pequeño cuestionario.

En relación a los cuestionarios se ha definido el IPAQ, Cuestionario Internacional de Actividad Física, que dispone de varios versiones e idiomas que nos sirve para medir la actividad física y el tiempo dedicado. Su uso es muy recomendado en la atención primaria y son fáciles de aplicar.

CUESTIONARIO INTERNACIONAL DE ACTIVIDAD FÍSICA (IPAQ)

Nos interesa conocer el tipo de actividad física que usted realiza en su vida cotidiana. Las preguntas se referirán al tiempo que destinó a estar activo/a en los últimos 7 días. Le informamos que este cuestionario es totalmente anónimo.

Muchas gracias por su colaboración

1.- Durante los últimos 7 días, ¿en cuántos realizó actividades físicas intensas tales como levantar pesos pesados, cavar, ejercicios, hacer aeróbicos o andar rápido en bicicleta?

Días por semana (indique el número)	<input type="text"/>
Ninguna actividad física intensa (pase a la pregunta 3)	<input checked="" type="checkbox"/>

2.- Habitualmente, ¿cuánto tiempo en total dedicó a una actividad física intensa en uno de esos días?

Indique cuántas horas por día	<input type="text"/>
Indique cuántos minutos por día	<input type="text"/>
No sabe/no está seguro	<input type="checkbox"/>

3.- Durante los últimos 7 días, ¿en cuántos días hizo actividades físicas moderadas tales como transportar pesos livianos, o andar en bicicleta a velocidad regular? No incluya caminar

Días por semana (indicar el número)	<input type="text"/>
Ninguna actividad física moderada (pase a la pregunta 5)	<input checked="" type="checkbox"/>

4.- Habitualmente, ¿cuánto tiempo en total dedicó a una actividad física moderada en uno de esos días?

Indique cuántas horas por día	<input type="text"/>
Indique cuántos minutos por día	<input type="text"/>
No sabe/no está seguro	<input type="checkbox"/>

Figura 2.1 captura del IPAQ de Junta de Andalucía

En cuanto a los dispositivos wearables o corporales, están muy de moda actualmente. En este grupo encontramos las pulseras y los relojes inteligentes, los parches, etc. Son dispositivos electrónicos que nos podemos llevar en el cuerpo y que son capaces de monitorizar nuestros movimientos durante el día. Dependiendo del precio del dispositivo y el tipo de dispositivo tienen más o menos funciones.



Figura 2.2 Reloj inteligente Fitbit

Por otra parte, la OMS define el estado de salud como “un estado de completo bienestar físico, mental y social, y no solamente la ausencia de afecciones o enfermedades”. [oms21b]

¿Pero cómo se puede saber en qué estado de la salud se encuentra una persona? Se puede saber el estado de salud general de una persona utilizando simplemente la observación, pero hay otras formas mucho más eficientes utilizando algunas herramientas como los dispositivos de medición, que además nos permite cuantificar el estado de varios parámetros. Hoy en día, existen una gran variedad de dispositivos que nos permiten medir el estado de la salud, tales como los termómetros para medir la temperatura, el esfigmomanómetro o tensiómetro para medir la presión arterial, el pulsómetro para medir la frecuencia cardíaca, el respirómetro para medir la frecuencia respiratoria, etc. Muchos de ellos suelen encontrarse en una farmacia o hospital y no es normal que lo tengamos en la casa, por lo que no son accesibles siempre cuando queremos medir nuestros signos vitales

2.2 Dispositivos wearables

En los últimos años han surgido nuevos dispositivos que permiten la medición del estado de la salud como los dispositivos corporales o wearables. Son aparatos eléctricos normalmente pequeños, que podemos llevar fácilmente en alguna parte del cuerpo, como las pulseras o relojes inteligentes en la muñeca, los anillos en la mano, los colgantes en el cuello, etc. Estos permiten controlar y monitorizar de forma automática el estado de la salud y las actividades físicas de las personas. Normalmente están conectados a un terminal en la que podemos ver de forma instantánea los valores de cada parámetro registrado.

A diferencia de los dispositivos mencionados en la sección anterior, estos dispositivos si se pueden usar fuera del hospital, son más fáciles de usar y de conseguir, además las mayorías son invasivas.

Gracias a estos wearables podemos realizar un seguimiento preciso de nuestros hábitos de salud. Pero, ¿qué pueden medir los wearables? Las mediciones que pueden realizar

las podemos clasificar según el tipo de wearables:

- Relojes/pulseras inteligentes: Según el modelo y el tipo del reloj, estos permiten realizar unas mediciones u otras. Entre las que se encuentran la medición de frecuencia cardíaca, nivel de oxígeno en la sangre, la temperatura cutánea y corporal, la frecuencia respiratoria, el sueño, las actividades físicas, los pasos, etc.
- Anillos inteligentes, son capaces de medir la frecuencia cardíaca y respiratoria, la temperatura corporal, el sueño y las actividades físicas.
- Collares inteligentes, este tipo de wearables se utilizan normalmente para las mascotas, permiten medir el ritmo cardíaco, el sueño, las actividades físicas, etc.
- A parte de los wearables, existe también otro dispositivo inteligente muy interesante para controlar nuestra salud como pueden ser las básculas inteligentes que nos permiten conocer el índice de masa corporal, el porcentaje de agua que hay en nuestro cuerpo, porcentaje de masa muscular, metabolismo basal, edad metabólica, calidad muscular, etc.

2.3 Estudio de las variables fisiológicas

Las variables fisiológicas son todas aquellas medidas medibles y cuya funcionalidad es regular el funcionamiento biológico de las personas. Son muchas las variables fisiológicas, aquí vamos a estudiar algunas más importantes de ellas.

● Frecuencia cardíaca

La frecuencia cardíaca es el número de veces que se contrae el corazón durante un minuto. Conocer nuestra frecuencia cardíaca nos ayuda a controlar nuestros estados físicos y también nos permite detectar si sufrimos algún problema de la salud.

Hay varios arteriales principales en nuestro cuerpo en las que donde podemos tomar una medición más precisa, estas arteriales son las arteriales carótidas, pulso braquial, pulso temporal, pulso femoral, pulso poplíteo, arteria tibial posterior, pulso facial, arteria facial.

Hay dos formas de medir la frecuencia cardíaca, la más fácil de usar es mediante los dispositivos electrónicos de medición, para su medición basta con colocar el sensor en la muñeca o el dedo durante un tiempo determinado. Y la otra forma es medir manualmente, para ello, hay que localizar la arteria que queremos medir, y luego colocar encima los dedos índice y corazón, para contar las pulsaciones durante un minuto.

Rango de valor correcto: La frecuencia cardíaca de una persona sana, relajada, tranquila, y sentada o tumbada, suele encontrarse entre 60-100 latidos por minuto. Pero una frecuencia cardíaca inferior a 60 no siempre indica que hay un problema de la salud, ya que puede ser debido a la toma de medicamentos o porque realiza mucha actividad física.

Además de eso, también existen otros factores que pueden afectar a la frecuencia cardíaca, como temperatura, posición del cuerpo, emociones, etc.

Según la fuente de Institutos Nacionales de la Salud / Biblioteca Nacional de Medicina de los EE. UU [Med21], la frecuencia cardíaca normal en reposo según la edad es:

- 0 – 1 mes:	70-190 latidos por minuto
- 1 – 11 meses:	80-160 latidos por minuto
- 1 – 2 años:	80-130 latidos por minuto
- 3 – 4 años:	80-120 latidos por minuto
- 5 – 6 años:	75-115 latidos por minuto
- 7 – 9 años:	70-110 latidos por minuto
- 10 años o más:	60-100 latidos por minuto
- Atletas bien entrenados:	40-60 latidos por minuto

- **SpO₂** (saturación de oxígeno en la sangre)

La saturación de oxígeno en la sangre se refiere a la cantidad de oxígeno que está circulando en la sangre. Cuando bombea sangre el corazón, el oxígeno se unen a los glóbulos rojos que hay en nuestro cuerpo y se reparten así por todo el cuerpo. Un nivel óptimo de saturación de oxígeno permite que las células de nuestro cuerpo obtienen la cantidad suficiente de oxígenos.

Para medir el nivel de oxígeno en la sangre podemos recurrir a dos formas. Una de ella es mediante una gasometría arterial o ABG. Consiste en tomar una muestra de sangre en una arteria, normalmente es de la muñeca, y de esa muestra se medirán los niveles de gases y su acidez. La ventaja de este procedimiento es el resultado suele ser muy exacto.

La otra forma es utilizar dispositivos electrónicos como oxímetro de pulso, estos dispositivos mide el oxígeno en la sangre a través de la absorción de la luz mediante pulso de la persona. Las ventajas de este método se encuentran que es más fácil y rápida de realizar y la persona que realiza no sufre dolor.

Rango de valor correcto: El nivel de oxígeno en la sangre de una persona normal es de 95%-100%.

Cuando este nivel se encuentra por debajo de 90%, es posible provocar hipoxemia, como consecuencia, a la persona afectada tiene más dificultad a la hora de respirar. Cuando el nivel de oxígeno se encuentra por debajo del 80%, se dice que tiene hipoxemia severa.

La hipoxemia puede ser causado por:

- La falta de oxígeno en el aire

- Incapacidad de los pulmones, que no son capaces de enviar oxígenos a todas las células.
- Incapacidad del torrente sanguíneo para circular a los pulmones.
- Y también puede ser causado por afecciones y situaciones médicas como sama, enfermedades del corazón, altitud elevada, anemia, enfisema, etc.

● Tensión arterial

La tensión arterial o la presión arterial es la fuerza que ejerce la sangre contra las paredes de las arterias. La presión arterial es más alta cuando late el corazón (presión sistólica), y más baja cuando el corazón se encuentra en reposo (presión diastólica). Estos dos valores son usados en la lectura de la presión arterial. Por ejemplo 120/80, el primer valor indica el valor de la presión sistólica y el segundo, presión diastólica.

La hipertensión generalmente no tiene síntomas. Por lo que la única manera de detectarla es mediante medición regulares.

Existen varias formas para la medición de la tensión arterial. Podemos usar:

- Esfigmomanómetro de mercurio, es el método más exacto con menos errores.
- Esfigmomanómetro de aire, este método es el más usado, la medición es preciso también, aunque no tan exacto como el anterior.
- Para la medición de la tensión arterial usando estos dos métodos, es necesario que el manguito del esfigmomanómetro esté colocado en la altura del corazón. Una vez colocado, se coloca la campana del fonendo donde está el latido arterial. Despues se inflama el manguito hasta llegar a una presión de 180 mm Hg o 200 mm Hg. Y finalmente se desinfla poco a poco para tomar la medida de la presión sistólica y diastólica.
- Aparato electrónico, este tipo de aparato se usa mucho para la realización de autocontrol. Es fácil de usar y no necesita fonendoscopio, se mide la presión mediante el detector de pulso. Es menos exacto y es muy sensible a ruidos y movimientos.
- La auto medición de la presión arterial suele realizarse durante la mañana y la noche, antes de realizar la medición, se recomienda reposar 3 minutos.

Rango de valor correcto: la tensión arterial normal de una persona adulta es 120/80. Cuando el valor de la presión sistólica se encuentra por encima del 140 y el valor de la presión diastólica está por encima de 90, es posible que la persona tiene problemas de hipertensión.

- Normal: presión arterial sistólica menor de 120 y presión arterial diastólica menor de 80.
- Presión arterial alta sin otros factores de riesgo cardíaco: presión arterial sistólica 140 o mayor o presión arterial diastólica 90 o mayor

- Presión arterial alta con otros factores de riesgo cardíaco: presión arterial sistólica 130 o mayor o presión arterial diastólica 80 o mayor
- Presión arterial peligrosamente alta: presión arterial sistólica 180 o mayor y presión arterial diastólica 120 o mayor

● **Electrocardiograma (ECG)**

El electrocardiograma registra la actividad eléctrica producida por el corazón en cada latido cardiaco. Es una prueba sin dolor que sirve para detectar rápidamente los problemas del corazón y controlar su salud. Las actividades eléctricas se registran a través de la superficie del cuerpo y el resultado se representa en el papel en forma de gráfica.

Para registrar dicha actividad eléctrica se puede usar un electrocardiógrafo o algunos dispositivos eléctricos wearables.

Para el uso de la máquina electrocardiógrafo es necesario tener como mínimo 10 electrodos que irán unidos al electrocardiógrafo mediante unos cables. Estos electrodos se colocan sobre la piel del paciente en diferentes puntos del cuerpo del paciente. Con esos 10 electrodos se puede conseguir 12 derivaciones, que se transforma en 12 trazados de los impulsos eléctricos del corazón. Existe la posibilidad de añadir más electrodos para conseguir más derivaciones, pero lo normal es que se usan 10 electrodos.

El electrocardiograma de las personas sanas sigue un trazado particular, podemos detectar si hay problemas del corazón cuando sufre variaciones en dicho trazado. Se pueden usar para medir la regularidad de los latidos, el ritmo cardíaco, posición, tamaño de las aurículas y ventrículos, y daños en el corazón.

Rango de valor correcto:

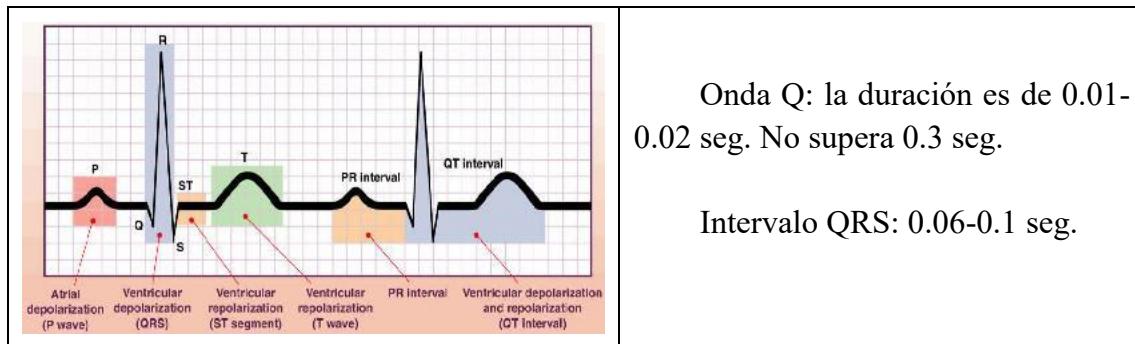


Figura 2.3 Captura de un ECG normal (fuente figura: <http://www.dalcame.com/ecg.html#.Y3RJ-XbMKHu>)

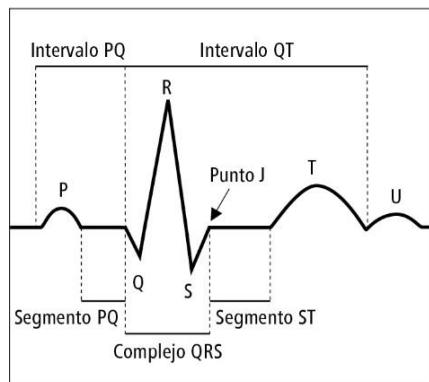


Figura 2.4 Esquema del trazado normal del ECG (fuente figura: <http://www.dalcame.com/ecg.html#.Y3RJ-XbMKHu>)

● Sueño

El sueño es un proceso biológico complejo que nos permite restablecer las funciones físicas y psicológicas de nuestro cuerpo para mantenernos en el pleno rendimiento. Sin una suficiente cantidad de sueño, es posible reducir nuestra capacidad de concentración y de razonamiento.

Para la medición del sueño podemos usar diversos dispositivos tales como pulseras cuantificadoras, los relojes inteligentes o accesorios específicos. Estos dispositivos miden el sueño a través de los sensores de movimiento, presión, y de cardíaco.

Rango de valor correcto: se considera que una cantidad suficiente de sueño que debe tener una persona es de 7-9 horas. Durante la noche, el sueño pasa por dos fases principales, el sueño de no movimiento rápido de los ojos (NREM) que se divide en varios subfases, y sueño de movimiento rápido de los ojos (REM). Estas fases se repiten durante la noche cuando estamos durmiendo.

Fase 1 etapa de adormecimiento, corresponde aproximadamente a los primeros 10 minutos del sueño, es una etapa de transición de desde período de vigilia hasta que adormecemos.

Fase 2 etapa de sueño ligero, ocupa aproximadamente el 50% del sueño. En esta etapa, el cuerpo va desconectando progresivamente, la respiración y el ritmo cardíaco se van ralentizando.

Fase 3 etapa de transición, es una fase corta que dura sólo 2 o 3 minutos, en esta etapa, el cuerpo se encuentra en el estado de relajación profunda, y es donde se encuentra el pico de segregación de hormonas de crecimiento.

Fase 4 etapa de sueño profundo, ocupa aproximadamente 20% de nuestro ciclo de sueño. Durante esta fase, el ritmo respiratorio es bajo, y la presión arterial suele

descender entre un 10 y un 30%. La persona que está en esta etapa suele ser más difícil de despertar.

Fase de sueño REM, ocupa un 25% del ciclo de sueño, entre 15 y 30 minutos. Esta fase está caracterizada por una alta actividad cerebral. Es la fase en la que soñamos y captamos informaciones del exterior.

● **Temperatura corporal**

La temperatura corporal del interior del cuerpo, varía de forma natural según personas y el medio. Está regulada por un proceso que contiene 3 mecanismos:

- Termorreceptores, que están situados en la piel y en el núcleo preóptico del hipotálamo.
- Efectos termorreguladores, que se basa en la sudación y la vasodilatación.
- Área de control localizada en el cerebro.

Para medir la temperatura corporal podemos usar

- Un termómetro clásico, colocándolo en la boca, el recto o en la axila.
- Un termómetro de tira plástica, que cambia de color para mostrar la temperatura, se puede colocar en la frente o en la boca.
- Un termómetro eléctrico, que están disponible para el uso en el oído o en la frente.

Rango de valor correcto: La temperatura normal en el ano es de aproximadamente 37.5 °C. En la boca es de aproximadamente 36.8°C y 36.5°C en la axila. Cuando la temperatura se supera a la temperatura normal, significa que tiene fiebre, y la fiebre puede ser indicio de coágulos de sangre, cáncer, infecciones, etc.

Los factores que pueden afectar a la temperatura corporal son la edad, la hora del día, el sexo, el ejercicio físico, el estrés, enfermedades, temperatura del ambiente, etc.

Otro parámetro que está muy relacionado en la temperatura cutánea es la temperatura de la superficie de nuestra piel, suele variar más que la temperatura corporal

Rango de valor correcto: Su valor normal suele encontrarse alrededor de 33.5 °C

● **Caloría**

La caloría es la energía que necesita nuestro cuerpo para funcionar durante el día. Las calorías miden el valor energético que contienen los alimentos.

Nuestro cuerpo gasta energía para poder llevar a cabo todas sus funciones, pero existe también un gasto de energía por parte de nuestro cuerpo aun estando en reposo. Ya que quemamos energía cuando respiramos, el corazón late, etc. Por lo

que necesitamos una determinada cantidad de calorías al día para vivir, que se conoce como la tasa metabólica basal (TMB), que es la cantidad mínima de energía que el cuerpo necesita para mantener las funciones básicas.

La tasa metabólica se puede calcular a partir de la fórmula de Harris-Benedict [Axa21]:

- TMB en hombres = $(10 * \text{peso de Kg}) + (6.25 * \text{altura en cm}) - (5 * \text{edad en años}) + 5$
- TMB en mujeres = $(10 * \text{peso de Kg}) + (6.25 * \text{altura en cm}) - (5 * \text{edad en años}) - 161$

Normalmente la cantidad de calorías que necesitamos al día según la edad son [Axa21]:

- Hasta 18 años: $17.3 * \text{peso corporal} + 651$
- De 19 a 30 años: $15.3 * \text{peso corporal} + 679$
- De 31 a 60 años: $11.6 * \text{peso corporal} + 879$
- Más de 60 años: $13.5 * \text{peso corporal} + 487$

● Pasos

La cantidad de pasos que realizamos cada día sirve para evaluar el nivel de nuestra actividad física. Se puede medir mediante un podómetro o dispositivos electrónicos como un móvil, reloj inteligente, etc.

El número de pasos recomendados al día varía según si se trata de niños o adultos.

- Para los niños, se recomienda un mínimo de 13.000 a 15.000 pasos al día, y de ellos, por lo menos 6.000 pasos deben realizarse con una intensidad moderada.
- Para los adolescentes se recomienda unos 11.000 a 12.000 pasos al día, y la mitad de esos deben realizarse con intensidad moderada.
- Para adultos se recomienda unos 10.000 pasos diarios, de ellos al menos 3.000 pasos deben realizarse con una intensidad moderada, y se sugiere repetirlos 5 días a la semana. Dar menos de 5.000 pasos diarios aumenta la posibilidad de sufrir enfermedades cardiovasculares y metabólicas.
- Para los adultos mayores, se recomienda no bajar de 7.000 pasos diarios, y 3.000 de esos deben realizarse con una intensidad moderada durante cinco días a la semana.

● Frecuencia respiratoria

Es el número de respiraciones que realiza una persona en un periodo específico, generalmente durante un minuto. La frecuencia respiratoria está regulada por el sistema nervioso.

No debemos confundirlo con la frecuencia ventilatoria, la frecuencia respiratoria es el proceso fisiológico de la respiración, mientras que la ventilatoria es el proceso

mecánico.

Para medir la frecuencia respiratoria se puede utilizar los dispositivos electrónicos o de forma manual, contando el número de respiraciones que realizamos durante un minuto cuando estamos quieto. Para obtener una medición más precisa, se recomienda medirla cuando estamos sentado en una silla o en la cama.

Rango de valor correcto: La frecuencia respiratoria normal de una persona varía según la edad y el sexo [Wik21]:

- Recién nacido (hasta los 12 meses): de 30-60 respiraciones por minuto
- Infante (1 a 3 años): de 24-40 respiraciones por minuto
- Preescolar (3 a 6 años): de 22-34 respiraciones por minuto
- Escolar (de 6 a 13 años): de 18-30 respiraciones por minuto
- Adolescentes y adultos: de 12-18 respiraciones por minuto.

Cuando las respiraciones por minuto se encuentran por encima de lo normal, se dice que tiene taquipnea. Por lo contrario, cuando se encuentran por debajo de lo normal, se dice que hay bradipnea.

Una respiración rápida puede estar causado por muchos factores tales como asma, asfixia, enfermedad pulmonar, insuficiencia cardíaca, infección en las vías respiratorias, neumonía, etc.

- **Actividad electrodérmica de la piel (EDA)**

Es la variación de las propiedades eléctricas de la piel al producirse sudor. Se puede medir mediante una corriente continua de baja intensidad no invasiva.

2.4 Estudio de dispositivos wearables del mercado

Como ya hemos comentado en los apartados anteriores, hoy en día existen muchos dispositivos wearables de diferentes tipos. En esta sección vamos a estudiar algunos de estos dispositivos más relevantes para nuestro proyecto.

2.4.1 Relojes/pulseras inteligentes

- Xiaomi Mi Band 6 (Url: <https://www.mi.com/es/product/mi-smart-band-6/>). Es la pulsera inteligente de la última generación de la empresa Xiaomi. Está compuesto de dos partes, la correa y una carcasa hecha de policarbonato. Esta carcasa es la parte principal de la pulsera y lleva un sensor de 6 ejes de alta precisión, un sensor de oxígeno en la sangre, y un sensor de frecuencia cardíaca

PPG. Esta pulsera es capaz de medir la frecuencia cardíaca, el estrés, el sueño, la saturación de oxígeno en la sangre, los pasos y las actividades físicas.



Figura 2.5 Mi Band 6

- Polar Ignite 2 (Url: <https://www.polar.com/es/ignite2>). Es un reloj inteligente deportivo, la segunda generación de relojes con GPS de la empresa Polar. El reloj tiene una pantalla táctil circular con lente de vidrio Dragontrail a color con un sensor de luz ambiental. Cuenta con 4 sensores ópticos Polar Precision con 9 bombillos LED y 4 electrodos. Gracias a estos sensores el reloj nos permite medir la presión arterial bajo el agua, la frecuencia cardíaca, el sueño, las rutinas de ejercicios, uso de fuente de energía, los pasos y la altitud.



Figura 2.6 Polar Ignite 2

- Apple Watch Series 6 (Url: <https://www.apple.com/es/newsroom/2020/09/apple-watch-series-6-delivers-breakthrough-wellness-and-fitness-capabilities/>). Es el reloj inteligente de la empresa Apple. Tiene una pantalla táctil de Retina OLED LTPO siempre activa con un brillo de 1000nits. Utiliza procesadores S6 de 64 bits de doble núcleo, el chip inalámbrico W3 y chip de banda ultra ancha U1. En cuanto a los sensores dispone de un sensor eléctrico de frecuencia cardíaca, un sensor

óptico de frecuencia cardíaca de segunda generación, un sensor de oxígeno en sangre, un sensor de luz ambiental, acelerómetro, giroscopio, altímetro y una brújula.

Los parámetros que son capaces de medir son el nivel de saturación de oxígeno en la sangre, la frecuencia cardíaca, el sueño, las calorías quemadas, la latitud, las actividades físicas.



Figura 2.7 Apple Watch 6

- HuaWei Watch Fit. (Url: <https://consumer.huawei.com/es/wearables/watch-fit/>) Es el reloj inteligente de la marca HuaWei. Su diseño es rectangular y alargado, con una pantalla táctil AMOLED de 1.64 pulgadas. Dispone de un sensor IMU de 6 ejes, un sensor óptico de frecuencia cardíaca, un sensor de luz ambiental y un sensor capacitivo. Con HuaWei Watch Fit, podemos monitorizar la frecuencia cardíaca, la estimación de VO2max (la cantidad máxima de oxígeno que el cuerpo puede absorber), la saturación de oxígeno en la sangre, el sueño, el estrés, las actividades físicas y las calorías quemadas.

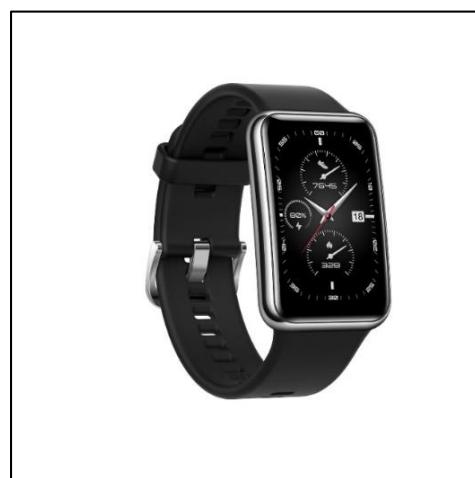


Figura 2.8 HuaWei Watch Fit

- Fitbit Sense (Url: <https://www.fitbit.com/global/be/products/smartwatches/sense>). Es uno de los relojes dedicados a la salud lanzado por la marca FitBit. Su diseño es cuadrado con esquina redonda y posee una pantalla táctil AMOLED con opción de encender siempre o de forma automática. Dispone de un micrófono y altavoz que nos permite interactuar con Alexa. Cuenta con un sensor óptico de frecuencia cardíaca, sensor de temperatura cutánea, sensor de luz ambiental y sensores eléctricos con múltiples funciones compatibles con la aplicación Escáner EDA. También incorpora un giroscopio, un altímetro y un acelerómetro de 3 ejes. Es capaz de monitorizar nuestro estilo de vida tanto de día como de noche. Permite medir de forma continua el pulso, la frecuencia cardíaca. También puede medir el nivel de estrés, el sueño, la saturación de oxígeno en sangre, la temperatura corporal, las actividades físicas, los pasos, la altitud y las calorías quemadas.



Figura 2.9 Fitbit sense

- OURA Ring (Url: <https://ouraring.com/es/product/horizon-silver>). Es un dispositivo wearable con forma de anillo de la compañía Oura Health. Tiene un sistema de medición avanzado que utiliza algoritmo de medición propios. Posee un procesador de doble núcleo, MCU de potencia ultrabaja basada en ARM Cortex, sensor de temperatura corporal, sensor que detecta la forma de onda del pulso y la amplitud del pulso con ayuda del sensor infrarrojo PPG. Nos permite monitorizar la frecuencia cardíaca, la frecuencia respiratoria, la temperatura corporal, el sueño, los pasos, las actividades físicas y las calorías quemadas.



Figura 2.10 Oura ring

- Withings ScanWatch (Url: <https://www.withings.com/es/es/scanwatch>). Es una de los relojes inteligentes lanzados por la marca Withings. A diferencia con respecto a los otros relojes, el ScanWatch es un reloj híbrido que está compuesto por un reloj analógico con una pequeña pantalla que permite la visualización de las informaciones. Incorpora un sensor de ritmo cardíaco PPG con lectura de SpO₂, un acelerómetro de 3 ejes y unos electrodos de acero inoxidable. ScanWatch es capaz de detectar las variaciones respiratorias, la fibrilación auricular, la frecuencia cardíaca, el sueño, la saturación de oxígeno en la sangre, los pasos, la altitud, las actividades físicas y las calorías quemadas.



Figura 2.11 ScanWatch(izquierda) y E4 wristband(derecha)

- E4 wristband (Url: <https://www.empatica.com/en-eu/research/e4/>). Es un dispositivo wearable desarrollado por la compañía Empatica. Se trata de un dispositivo de grado médico que permite la obtención de datos fisiológicos en tiempo real. Cumple una serie de normativas, una de ellas es el Certificado CE. Cuenta con un sensor PPG, un sensor EDA, un acelerómetro de 3 ejes, un reloj interno de tiempo real y una termopila infrarroja. Tiene dos modos de funcionamiento, un modo de transmisión que se utiliza para visualizar los datos en tiempo real, y un modo de grabación, que sirve para medir los datos y almacenarlos en la memoria. Los datos que pueden medir son el pulso de volumen sanguíneo (BVP), que es usado en la determinación de la frecuencia cardíaca y otras características cardiovasculares. La excitación del sistema nervioso, que se miden a través de los cambios que fluctúan en las propiedades eléctricas de la piel. También puede captar las actividades basado en movimiento y la temperatura cutánea.

Para poder verlos de forma más clara las características de cada dispositivo wearable y poder hacer una comparación, vamos a mostrar una tabla comparativa de los

dispositivos wearables mencionados anteriormente.

	Sensores	SDK & Programable	Parámetros a medir	Software	Lugar de almacenamiento	Coste
Apple Watch S6	De oxígeno en sangre, de frecuencia cardiaca, óptico, de luz ambiental	Si Si	SpO2, ECG, calorías, altitud, pasos, sueño, lavado de manos	WatchOs 7	En el móvil En el cloud	€ 427
E4 wristband	EDA, PPG, de temperatura, acelerómetro	Si Si	EDA, frecuencia cardiaca, movimientos, temperatura, actividad física, respiración, SpO2	-	En el dispositivo En el dispositivo emparejado En en cloud	\$ 1690.00
Fitbit Sense	De frecuencia cardiaca, eléctricos, de temperatura cutánea, de luz ambiental, giroscopio, altímetro, acelerómetro	Si Si	Estrés, actividad electrodémica, temperatura cutánea, SpO2, frecuencia cardiaca, frecuencia respiratoria	Fitbit Os 5.0	En el dispositivo En el móvil	€ 299.95
HuaWei Watch Fit	Acelerómetro, giroscopio, de frecuencia cardiaca, capacitivo, de luz ambiental	Si Si	Frecuencia cardiaca, VO2 max, SpO2, sueño, estrés	Wear Os by Google	En el móvil En el cloud	€ 79
Oura Ring	Acelerómetro, giroscopio, NTC de temperatura, infrarrojo, de fotodiodo	Si Si	Sueño, frecuencia cardiaca, temperatura corporal, frecuencia respiratoria, pasos, calorías	-	En el móvil En el dispositivo	€ 314
Polar Ignite 2	Polar Precision Prime	Si Si	Frecuencia cardiaca Fuente de energía Natación, sueño, pasos, altitud	Polar's OS	En el dispositivo En el cloud	€ 229.90
Withings ScanWatch	Acelerómetro, de ritmo cardiaco, de electrodo	Si Si	Frecuencia cardiaca y respiratoria, electocardiograma, SpO2, sueño, pasos, distancia, calorías, natación, altitud	-	En el móvil	Desde € 279.95
XiaoMi Band 6	Acelerómetro, giroscopio, de frecuencia cardiaca	Si Si	Frecuencia cardiaca, sueño, salud femenina, SpO2, estrés, ejercicio de respiración, pasos	-	En el dispositivo En el móvil En el cloud	€ 44.99

Tabla 2.1 Comparación de los dispositivos wearables

A partir del estudio realizado, en mi opinión, el E4 wristband, el Fitbit Sense y el Withings ScanWatch son los 3 mejores dispositivos de entre todos. Estos dispositivos cuentan con una variedad de sensores que pueden detectar más datos de la salud que nos pueden interesar. Para la realización del proyecto he escogido el dispositivo Withings ScanWatch, no solo porque permite obtener un mayor número de medidas, sino también por el nivel de precisión de los datos registrados. De hecho, Withings es una empresa que posee muchos dispositivos testados y validados clínicamente, y entre ellos se encuentra el ScanWatch. Otro dispositivo interesante sería el E4 wristband, pero lamentablemente ya no lo podemos conseguir debido a que ya no se vende actualmente.

2.4.2 Balanzas inteligentes

A veces no es suficiente la información que nos aportan los relojes inteligentes, sino que es necesario usar otros dispositivos que nos proporcionan medidas que pueden complementar a los obtenidos por relojes como ocurre con las balanzas. Con estos dispositivos podemos medir, por ejemplo, la grasa corporal, el peso, la masa ósea, etc. Actualmente, las básculas inteligentes que podemos encontrar en el mercado no se diferencian mucho en su diseño. Todos tienen el aspecto de una báscula normal. El componente hardware y los sensores que poseen son la mayor diferencia que hay entre ellos.

- Fitbit aria air (Url : <https://www.fitbit.com/global/es/products/scales/aria-air>). Es la báscula inteligente de la marca Fitbit que se conecta al dispositivo móvil mediante la conexión Bluetooth. Mide el peso con los cuatro células de carga que posee y lo visualiza en la pantalla que incorpora la balanza. Para tener una mayor experiencia de uso, es necesario conectarlo con la aplicación Fitbit. Conectado a la aplicación, podemos ver información acerca del peso, el IMC, y registrar los datos del entrenamiento, del sueño y de la nutrición.



Figura 2.12 Fitbit aria air

- Mi Body Composition Scale (Url: <https://www.mi.com/es/mi-body-composition-scale/>). Báscula inteligente de Xiaomi con una precisión de 50g que soporta hasta 150kg de peso. Se conecta a los dispositivos móviles mediante la tecnología Bluetooth 4.0. Gracias a sus electrodos de acero inoxidable junto con la tecnología BIA (Análisis de Impedancia Bioeléctrica) y una serie de algoritmos, es capaz de medir hasta 10 datos. Además del peso, mide también el IMC, la masa muscular, la masa ósea, la grasa corporal, la grasa visceral, el agua, el metabolismo basal y la puntuación corporal.

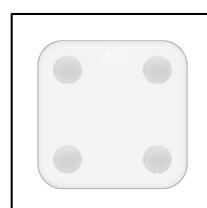


Figura 2.13 Mi Body Scale

- OMRON VIVA (Url: <https://www.omron-healthcare.es/es/basculas-digitales/VIVA.html>). Balanza inteligente de OMRON con tecnología BIA y una validación clínica. Se sincroniza con la aplicación del móvil mediante Bluetooth, en el caso de no poder conectarse al móvil, permite almacenar en su memoria local las mediciones de como máximo 30 días por usuario hasta que se conecta de nuevo y realizar una sincronización con la aplicación. Las medidas que puede medir son el peso, la grasa corporal, el IMC, la grasa visceral, la tasa de metabolismo basal y el músculo esquelético.



Figura 2.14 OMRON VIVA

- Medisana BS450 Connect (Url: <https://www.medisana.es/tienda/bs-450-connect-bascula-con-analisis-corporal/>). Báscula inteligente de Medisana con conexión Bluetooth y 30 almacenamientos de datos para cada uno de los ocho usuarios. Cuenta con electrodos de sensibilidad alta de acero inoxidable y cuatro sensores de galgas extensométricas. Permite medir el peso, el IMC, la grasa corporal, el agua corporal, el porcentaje muscular y el peso óseo.



Figura 2.15 Medisana BS450 Connect

- Withings Body+ (Url: <https://www.withings.com/es/es/body-plus>). Báscula de la marca Withings testado clínicamente que además de la conectividad Bluetooth, ofrece también la sincronización con Health Mate a través del WIFI, sin tener que conectarse al móvil. Tiene cuatro sensores de peso con alta precisión, detector de posición corporal patentado, y pesaje de alta precisión gracias a la tecnología Position Control. Puede realizar mediciones del peso, la masa muscular, la masa ósea, la masa grasa, y la masa libre de grasa.



Figura 2.16 Withings Body+

- Withings Body Cardio (Url: <https://www.withings.com/es/es/body-cardio>). Otra báscula inteligente de la marca Withings cuyo el desarrollo cuenta con la colaboración de varios cardiólogos. Se puede decir que es una versión mejorada del modelo Body+, además de las características que tiene el modelo anterior, posee la tecnología para la medición de frecuencia cardíaca patentada, también permite obtener la velocidad de onda de pulso.



Figura 2.17 Withings Body Cardio

De la misma forma con los relojes y pulseras, vamos a realizar una tabla comparativa de estas básculas inteligentes.

	Sensores	SDK & Programable	Conectividad	Lugar de almacenamiento	Coste
Fitbit aria air	4 células de carga Electrodos de acero	si	Bluetooth	En la nube	59.95€
Mi body Composition Scale	inoxidable, tecnología BIA	si	Bluetooth 4.0	En la nube	34.99€
OMRON VIVA	Electrodos de acero inoxidable, tecnología BIA	si	Bluetooth	En la nube En el dispositivo	109€
Medisana BS450 Connect	Electrodos de acero inoxidable, sensores de galgas extensométricas	si	Bluetooth	En la nube En el dispositivo	65€
Withings Body+	4 sensores de peso de alta precisión, detector de posición corporal, pesaje de alta precisión	si	Bluetooth WIFI	En la nube	99.95€
Withings Body Cardio	4 sensores de peso, detector de posición corporal, pesaje de alta precisión, sensor de frecuencia cardíaca	si	Bluetooth WIFI	En la nube	179.95€

Tabla 2.2 Tabla comparativa de las básculas

Capítulo 3. Estudio de las plataformas de monitorización continua

3.1 Sistemas de monitorización continua

Antes de explicar que son los sistemas de monitorización continua, vamos a describir primero un poco sobre en qué consiste la monitorización continua en el ámbito de la salud. La monitorización continua es un proceso sistemático y continuo que tiene como objetivo recoger los parámetros fisiológicos del usuario y registrarlos en algún lugar para poder luego mostrarlos.

Un sistema de monitorización continua es un sistema que utiliza los dispositivos wearables con sensores para recolectar de forma automática o manual y continua las medidas fisiológicas del usuario y que registran los datos recogidos para que posteriormente se pueda visualizar de forma clara los resultados para poder estudiar la evolución temporal de dichos parámetros físicos o fisiológicos.

A continuación, se muestra un esquema general de los sistemas de monitorización continua, formadas por los subsistemas de medición de datos, de registros de datos y de visualización de datos.

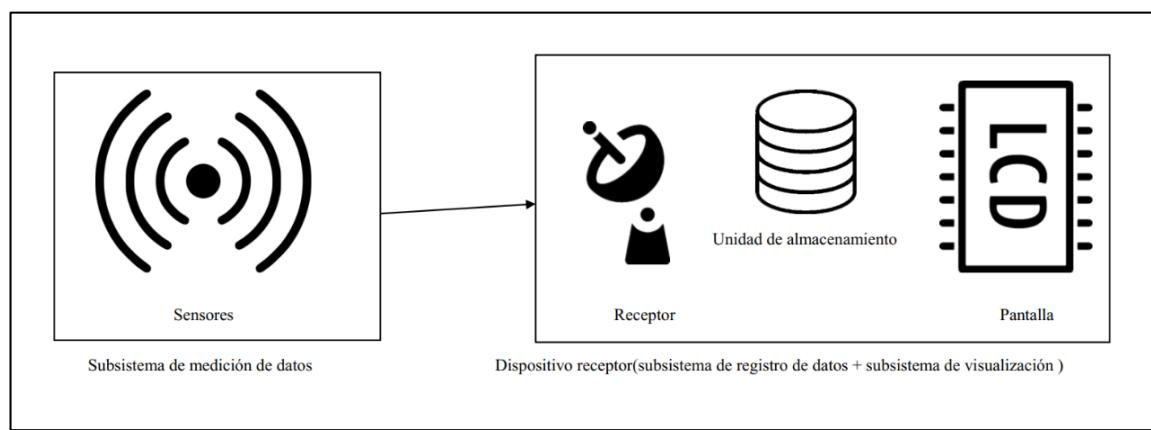


Figura 3.1 Esquema general de un Sistema de monitorización

Hoy en día, existen muchos sistemas de monitorización en el mercado, como:

- Sistema de monitorización continua de glucosa. Son dispositivos que miden el nivel de glucemia intersticial de forma continua. Está compuesto por dos piezas, una pieza sensor flexible que se inserta en la piel, éste lleva asociado también un transmisor, que se encarga de enviar la señal a la otra pieza, que es el receptor, el cual muestra el resultado en la pantalla de un dispositivo compatible.
Hay dos formas de monitorización. En la primera forma o monitorización en tiempo real en la que da una lectura en tiempo real y directa, es capaz de hacer un trazado de 24 horas, y permite avisar al paciente cuando su nivel de glucemia es demasiado alto, hiperglucemias, o demasiado bajo, hipoglucemias. La otra forma es la monitorización flash, en la que se realiza la medición cuando se acerca al sensor el receptor de la señal. Da una lectura a tiempo real y permite un trazado de últimas ocho horas. No permite la configuración de alarmas.



Figura 3.2 Sistema de monitorización de glucosa

- Monitor Holter. Es una máquina que es capaz de medir los ritmos cardíacos de forma continua durante las 24 horas del día. Consiste en unos parches pequeños, llamados electrodos, que se colocan en el pecho del paciente, que se van a encargar

de detectar los latidos del corazón. Dichos electrodos mediante unos alambres se conectan a un monitor de registro, donde se almacena los resultados registrados. Se recomienda llevar puesto durante unos 24 a 48 horas. Mientras el paciente lleva el dispositivo y esté funcionando, registra todas las actividades que realizan el usuario durante el día. Su uso permite la observación de los trastornos cardíacos como taquicardia, fibrilación auricular, extrasístoles, isquemia, etc.

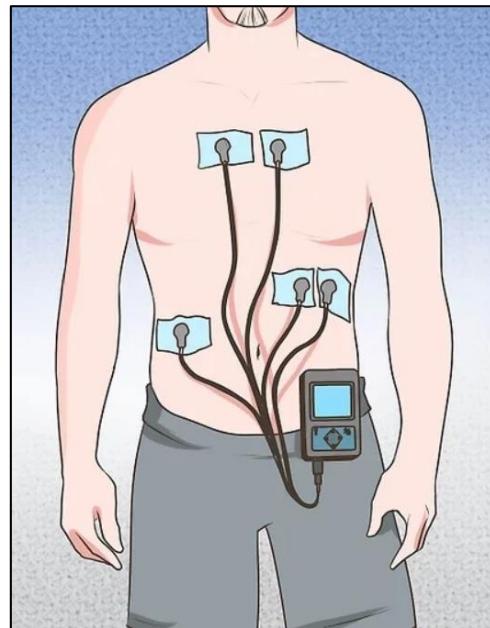


Figura 3.3 Monitor Holter

- Además de estos dos, existen también sistemas de monitorización de presión arterial, de temperatura corporal, etc.

3.2 Arquitectura para la monitorización continua

Antes de empezar a desarrollar una aplicación o software, lo primero que debemos hacer es seleccionar entre todas las arquitecturas que hay, una que se adecua a nuestro diseño, que es capaz de ofrecernos las funcionalidades y las calidades que deseamos.

3.2.1 ¿Qué son los patrones arquitectónicos?

Los patrones arquitectónicos son unas plantillas que nos sirven para resolver los problemas de diseño en contextos particulares durante la fase de desarrollo de software.

Según Christopher Alexander, “*cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces*”. Esta

definición está referido a patrones en ciudades, pero también lo podemos aplicar en el ámbito de desarrollo software. [Pat22]

Como dice Gamma, “*un patrón es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular*”. [Pat22]

Existen muchos patrones arquitectónicos, Entre los más comunes se encuentran el patrón cliente-servidor, el patrón maestro-esclavo, el patrón de filtro de tubería, el patrón de agente, el patrón de modelo-vista-controlador, el patrón de capas, etc.

Aquí vamos a describir brevemente algunos tipos de arquitecturas que podemos encontrar en un sistema de monitorización continua.

3.2.2 Patrón arquitectónico en capas

Es uno de los patrones arquitectónicos más comunes y más fácil de implementar, se encuentra en muchos programas software. Consiste en dividir el programa en varios niveles o capas, cada una de estas capas desempeña una función específica dentro del programa.

Las capas son independientes y están organizados de forma horizontal, de manera que todas las capas están interconectadas entre sí, y que cada una de las capas proporciona servicios a la capa directamente superior a él y sólo puede comunicarse con las capas que están conectados a él.

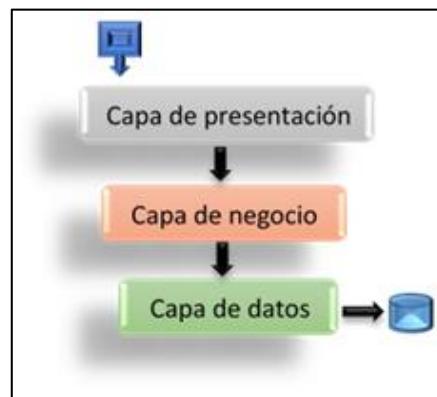


Figura 3.4 arquitectura en 3 capas

Normalmente, cuando los programadores aplican este tipo de arquitectura, suelen implementarlo en 3 capas, capa de presentación, capa de negocio o de servicio, y capa de datos. Pero eso no significa que todas las aplicaciones que utilizan este patrón arquitectónico tengan que dividirlo en 3 capas, según la situación y el tipo de programa, el programador puede decidir usar más capas o menos capas.

- Capa de presentación. Esta capa es conocida también como interfaz de usuario, es la que se ve el usuario, su función es presentar al usuario el sistema, comunicación de las informaciones y la captura de información del usuario.
- Capa de negocio o de servicio. Es la capa que establecen las reglas de la aplicación, donde se encuentran todas las funcionalidades. Se encarga de comunicar con la capa de presentación con el fin de recibir las peticiones que envían los usuarios y de mostrarles el resultado de la petición. Una vez recibida, se comunica con la capa de datos para el almacenamiento o recuperación de los datos.
- Capa de datos. Es la capa donde se encuentran los datos, y se encarga de los accesos a esos mismos datos. Normalmente está compuesto por uno o varios gestores de bases de datos.

Las ventajas de este patrón arquitectónico son:

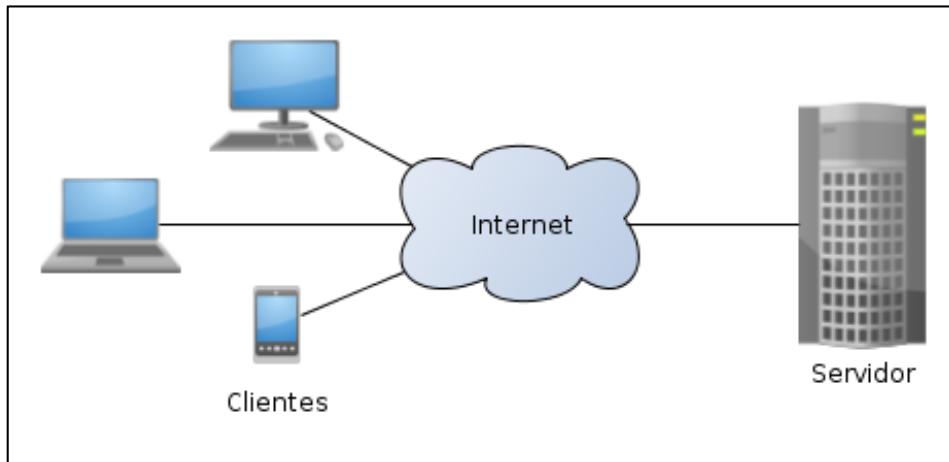
- La simplicidad, ya que este patrón es fácil de desarrollar.
- Facilidad de comprobar, ya que el programa está desarrollado por capas, por lo que se puede comprobar cada una de las capas de forma separada.
- Mantenible, como cada capa se encarga de una función específica, es fácil de encontrar donde está el problema o donde hay que hacer cambios.
- Mayor seguridad, al estar separado por capaz, los servidores se encuentran aislados en las redes haciendo que los ataques sean más difíciles.

Algunas de las desventajas de esta arquitectura son:

- Difícil de escalar.
- Tolerancia a fallo, si alguna capa surge algún problema, todas las capas superiores también se ven afectados por ese problema.

3.2.3 Patrón arquitectónico cliente-servidor

El patrón cliente-servidor es una arquitectura distribuida compuesto por dos componentes, el cliente y el servidor. Los dos componentes están distribuidos en computadores diferentes, aunque en algunos casos pueden estar en la misma computadora, y se comunican a través de una red. En esta arquitectura, las tareas están repartidas entre los dos componentes. El cliente es el componente encargado de solicitar servicios o recursos al servidor, y un servidor es una máquina donde se almacena los recursos y da la respuesta a la solicitud de los clientes.



*Figura 3.5 Esquema arquitectura cliente-servidor (fuente figura:
<https://upload.wikimedia.org/wikipedia/commons/1/1c/Cliente-Servidor.png>)*

Según el tamaño del lado cliente y del lado servidor podemos clasificarla en:

- Cliente pesado y servidor ligero. En este tipo las máquinas clientes son más potentes que el servidor. La mayoría de las cargas de cómputos se realizan en el lado de los clientes y se accede de forma esporádica al servidor. Mientras que el servidor es utilizado para las tareas ligeras como el hospedaje de Sistema Gestor de Bases de Datos.
- Servidor pesado y cliente ligero. Es el contrario del tipo anterior, necesitamos un servidor potente. La mayoría de las funciones y los cómputos de la aplicación se ejecutan en el lado del servidor, y el cliente se emplea para tareas ligeras de nivel de presentación.

Las ventajas de este patrón arquitectónico son:

- Escalabilidad, tanto el cliente como el servidor se puede aumentar de forma sencilla y en cualquier momento.
- Tolerancia al fallo, al estar separado las funcionalidades, si falla un cliente o servidor no parará la ejecución del sistema.
- Código reutilizable, el código que implementa un servicio determinado puede ser reutilizado en varios servidores.

Y las desventajas son:

- La congestión del tráfico de la red causada por la elevada cantidad de peticiones realizadas simultáneamente por los clientes al servidor.
- La seguridad, debido a que se comparten informaciones entre los servidores y clientes por la red.
- El coste, generalmente este tipo de arquitectura necesita un servidor específico y potente.

3.2.4 Arquitectura de sistemas de monitorización continua

Un sistema de monitorización continua, como los que se utilizan por parte de los dispositivos wearables del mercado, está basado en el uso de computación de la nube. Como los sistemas de internet de las cosas suele contener los siguientes subsistemas:

- Subsistema de medición de datos. Generalmente está compuesto por los dispositivos wearables que captan los datos a través de los sensores que llevan. Estos datos capturados se envían al subsistema de persistencia.
- Subsistema de procesamiento y persistencia. En este subsistema los datos recibidos del subsistema de medición son procesados por la unidad de procesamiento y se almacenan en la unidad de almacenamiento, que puede ser una base de datos.
- Subsistema de visualización. Su función principal es la visualización de los datos almacenados. A través de unas herramientas o aplicaciones de visualización, muestra los datos a los usuarios de forma clara y fácil de interpretar.

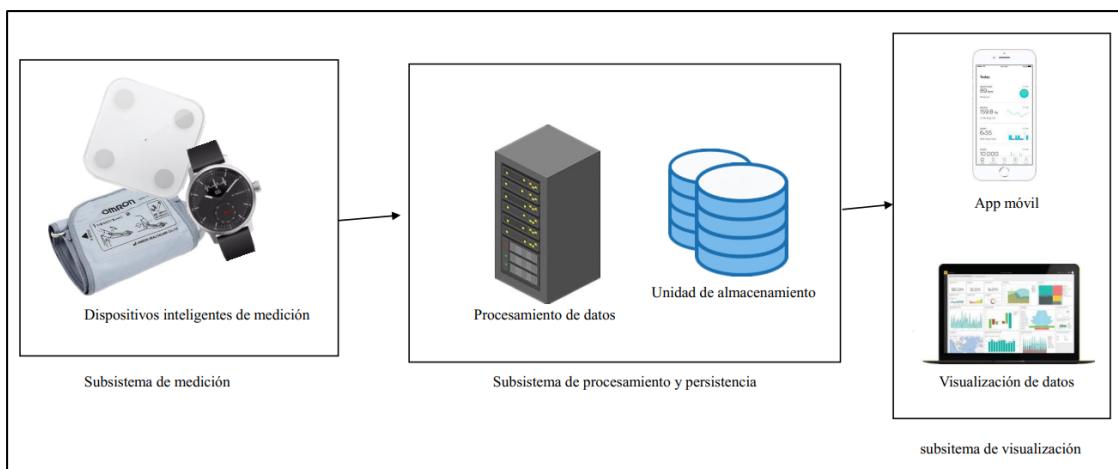


Figura 3.6 Esquema arquitectura de sistema de monitorización continua (IoT)

3.3 Estudio de plataformas de monitorización continua

En los sistemas de monitorización continua es imprescindible tener una base de datos donde podemos guardar los registros y una herramienta de visualización que nos permite ver de forma simple y clara los distintos datos.

3.3.1 Base de datos

Una base de datos es una herramienta que sirve para recopilar y organizar aquellos informaciones o datos que pertenecen a un mismo contexto. En él podemos almacenar

datos de personas, de pedidos, de inventarios, de comidas, etc. Para el manejo de las bases de datos podemos usar los SGBD, sistemas gestores de bases de datos, que son programas cuyo objetivo es facilitar el uso de las bases de datos permitiendo el rápido almacenamiento y acceso de datos de forma estructurada.

Algunos tipos de bases de datos clasificados según sus modelos son:

- Bases de datos jerárquicas, los datos se almacenan en forma de árbol, en la que un nodo de información padre puede tener cero, uno o varios hijos.
- Bases de datos de red. Es muy parecido al modelo jerárquico. La diferencia es que, en este modelo, un nodo puede tener varios padres.
- Bases de datos transaccionales, están destinados a recibir y enviar los datos a altas velocidades.
- Bases de datos relacional. Es el modelo más utilizado en el que los datos se almacenan en tablas formadas por registros y campos. Las distintas tablas pueden estar unidos mediante claves que representan las relaciones que existen entre ellos.
- Bases de datos de series temporales. Es un tipo de base de datos profesional dedicada para el tratamiento de datos de series temporales, donde la ordenación y organización de dichos datos están optimizados. Este modelo es muy útil para el monitorear carga de trabajo, comportamientos de los procesos, los datos de los sensores, etc.

A continuación, vamos a hacer un estudio de los distintos bases de datos de series temporales.

- InfluxDB

Web: <https://www.influxdata.com/>



Figura 3.7 Logo de InfluxDB

InfluxDB es un servidor de código abierto de base de dato de series de tiempo desarrollada por InfluxData. Está escrito en Go y optimizado para almacenar y recuperar los datos de forma rápida. Y está diseñado específicamente para manejar los datos con marcas de tiempo producidos por los sensores, aplicaciones e infraestructuras.

Utiliza InfluxQL, que es muy similar a un lenguaje de consulta de estructura, para

interactuar con los datos. Es capaz de manejar millones de puntos de datos en un segundo y tiene políticas de retención para eliminar automáticamente los datos obsoletos. Dispone de soporte para procesar datos desde Graphite, y una API poderosa.

Los datos almacenados en el InfluxDB se encuentran en la nube y tienen un formato basado en un texto que proporciona la medición, un conjunto de etiquetas, un conjunto de campos y la marca de tiempo.

```
<measurement>[,<tag_key>=<tag_value>[,<tag_key>=<tag_value>]]  
<field_key>=<field_value>[,<field_key>=<field_value>] [<timestamp>]
```

Ejemplo:

```
myMeasurement, tag1=value1,tag2=value2 fieldKey="fieldValue"  
1556813561098000000
```

La marca de tiempo es opcional, si no se proporciona una marca de tiempo, se usa por defecto la hora del sistema de la máquina host.

Se puede utilizar el agente Telegraf para recolectar los datos y eventos de bases de datos, sistemas y sensores de IoT, y luego enviarlas al InfluxDB.

Para la visualización de los datos almacenados en InfluxDb se puede utilizar Grafana, que es un dashboard que se encarga de mostrar las informaciones almacenadas.

A través del Kapacitor, que es un motor de procesamiento de datos nativo para InfluxDB, los usuarios pueden configurar las alertas, ejecutar jobs y detectar anomalías en los datos fácilmente

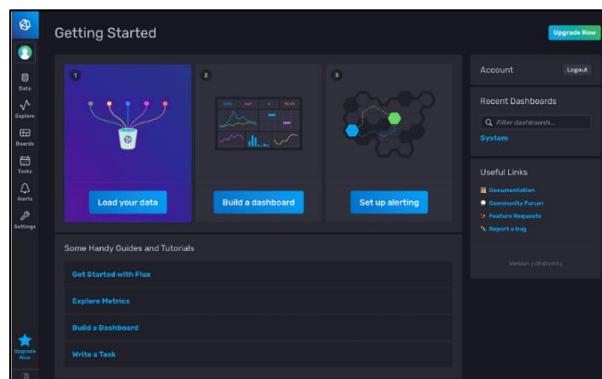


Figura 3.8 InfluxDB versión cloud(fuente figura: <https://www.influxdata.com/get-influxdb/>)

- Prometheus

Web: <https://prometheus.io/>



Figura 3.9 Logo de Prometheus

Prometheus es un sistema de monitoreo y alerta de sistema de código abierto basado en métricas construido por SoundCloud. Algunas de las características de Prometheus son:

- Modelo de datos multidimensional con datos de series de tiempo identificados por un nombre y los pares clave/valor.
- Utiliza PromQL, que es un lenguaje de consulta flexible.
- La recopilación de datos ocurre a través de HTTP.
- La inserción de datos se puede hacer a través de una puerta de enlace intermedia.
- Múltiples modos de soporte de gráficos y tableros

Prometheus cuentan con una gran cantidad de bibliotecas cliente, que permiten una fácil instrumentación de los servicios. Y también se puede implementar fácilmente bibliotecas personalizadas.

Desde un servidor de Prometheus se puede acceder al API HTTP, el formato de la respuesta de la API es JSON.

Prometheus almacena los datos de forma local, cada serie de tiempo está identificado por un nombre de la métrica y unas etiquetas o pares llave/valor
`<metric name>{<label name>=<label value>, ...}`

Ejemplo:

```
probe_success{instance=>>192.168.1.15",job=>>servidor-01"}
```

Prometheus dispone de una gran cantidad de exportadores que permiten exportar las métricas de sistemas de terceros como métricas de Prometheus, como exportar datos de Windows, Linux, Java, Base de datos, API, etc.

Para la visualización de los datos almacenados en el Prometheus se puede utilizar la herramienta Grafana, desde 2015 se incluye la fuente de datos de Grafana para Prometheus.

- TimescaleDB

Web: <https://www.timescale.com/>



Figura 3.10 Logo de TimescaleDB

TimescaleDB es una base de datos relacional para datos de series de tiempo de código abierto. Esta base de dato está basada en PostgreSQL, por lo que se puede aprovechar las extensiones de PostgreSQL. Existen dos versiones, una versión de software en la que nos podemos instalar en nuestro servidor, y otra versión Cloud, en la que no hace falta la instalación para su uso, y está completamente administrado por TimescaleDB.

TimescaleDB aprovecha la paralelización de consultas, los agregados continuos y otras optimizaciones de rendimiento, esto hace que su rendimiento es muchas veces más rápido que otras bases de datos como MongoDB, InfluxDB, etc.

TimescaleDB es capaz de escribir millones de puntos de datos por segundo, y almacenar terabytes de datos en un solo nodo o petabytes en varios nodos.

TimescaleDB permite almacenar los datos relacionales junto con los datos de series de tiempo. Se puede centralizar el almacenamiento de datos de sensores, aplicaciones y series de tiempo, correlacionar las métricas con datos comerciales y del sistema de registro.

Los datos de series de tiempo almacenados en TimescaleDB suele tener el formato de

Marca de tiempo | Etiqueta | valores

Ejemplo:

time | device_id |temperature | humidity

Las APIs que soporta el TimescaleDB las podemos encontrar en la URL <https://docs.timescale.com/api/latest/#apis-grouped-by-related-feature>

Para la visualización de los datos almacenados en la base de datos se puede usar igual que las dos bases de datos anteriores, la herramienta Grafana, que se puede crear paneles y luego conectarlos al TimescaleDB.

- Graphite

Web: <https://graphiteapp.org/>



Figura 3.11. Logo de Graphite

Graphite es una herramienta de monitoreo que funciona tanto en hardware como en infraestructuras nubes. Fue diseñado y escrito por Chris Davis en Orbitz en 2006. Tiene dos funcionalidades, una es almacenar datos numéricos de series de tiempo, y la otra es la representación gráfica de datos de tiempo bajo demanda.

Graphite consta de tres componentes software:

- Carbon: es un servicio de alto rendimiento que escucha datos de series temporales. Recibe los datos, los agrega y los conserva en el disco.
- Whisper: es una biblioteca de base de datos simple para almacenar datos de series de tiempo.
- Graphite-web: es la interfaz de usuario y API de Graphite para renderizar gráficos y paneles.

Algunas características del Graphite son:

- El formato de métricas en el que se envían los datos es sencillo.
- Tiene integrada una API para visualizar los datos y crear cuadros, gráficos, etc.
- Proporciona un amplio conjunto de biblioteca estadísticas y funciones de representación.

Cada serie de datos almacenado en Graphite tiene un identificador único, y está formado por un nombre de la métrica y opcionalmente, un conjunto de etiquetas.
<metric path> <metric value> <metric timestamp>

Graphite está basado en bases de datos de tamaño fijo, por lo que hay que

configurar de antemano cuántos datos se quiere almacenar y con qué nivel de precisión. Tal como almacenar datos con una precisión de 1 minuto durante 2 horas.

Graphite ofrece por un lado API de Render URL que permite generar gráficos y recuperar datos sin procesar. Y, por otro lado, ofrece API de métricas, que son útiles para encontrar y enumerar las métricas disponibles en el sistema.

- QuestDB

Web: <https://questdb.io/>



Figura 3.12 Logo de QuestDB

QuestDB es una base de datos relacional orientada a columna de código abierto que es capaz de analizar en tiempo real los datos de series de tiempo. Está diseñada desde cero y libre de las dependencias.

QuestDB está

- Construido para el rendimiento, tiene análisis optimizados para SIMD, acceso basado en filas y columnas, ejecución de consultas vectorizadas.
- Optimizado para series de tiempo, tiene un modelo relacional para series de tiempo, los datos están almacenados en orden cronológico, tiene un protocolo de la línea de Fast InfluxDB
- Implementado con SQL, tiene series de tiempo y uniones relacionales es compatible con Postgres, subconsultas ilimitadas, y tiene incorporado el optimizador de SQL.

QuestDB mejora ANSI SQL con extensiones de series de tiempo para manipular datos con marcas de tiempo. Puede

- Filtrar y buscar marcas de tiempo específicas con “WHERE”
- Crear cubos de tiempo y agregar por intervalo mediante “SAMPLE BY”
- Buscar series de tiempo desde los valores más recientes hasta los más antiguos con “LATEST BY”
- Unir dos tablas según la marca de tiempo con “ASOS JOIN”

Algunas características de QuestDB son:

- Tiene una consola interactiva para importar datos usando arrastrar y soltar y consultarlos.
- Es compatible con sistemas nativos de la nube, locales o integrados.

- Proporciona integración empresarial.
- Proporciona información en tiempo real mediante análisis operativos y predictivos.

Se puede usar varios APIs, tales como REST API, de Postgres, de InfluxDB y Java.

Los datos almacenados en la base de dato tienen un formato similar a los del SQL.
Se puede recopilar también datos con la herramienta telegraf.

Para la visualización de datos se puede utilizar la herramienta Grafana

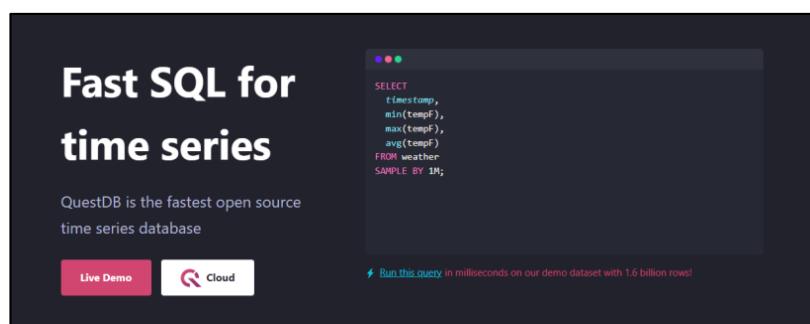


Figura 3.13 Captura de web de QuestDB

- OpenTSDB

Web: <http://opentsdb.net/>



Figura 3.14 Logo de OpenTSDB

OpenTSDB es una base de datos de series de tiempo escalable escrito en Hbase. Es capaz de almacenar billones de puntos de datos a millones de escrituras por segundo.

OpenTSDB consta de un demonio de serie temporal (TSD) y un conjunto de utilidades de línea de comandos. La interacción con OpenTSDB se realizan mediante la ejecución de uno o varios TSD. Cada uno de los TSD es independiente de otros, por lo que se puede ejecutar tanto TSD como hace falta para manejar cualquier carga. El TSD utiliza la base de datos de código abierto de Hbase o el servicio de Google Bigtable. Para comunicarse con el TSD, se puede utilizar un protocolo simple de estilo telnet, una API HTTP o una GUI incorporada simple.

Algunas de las características de OpenTSDB son:

- Puede agregar, filtrar y reducir la resolución de métricas a una velocidad vertiginosa.
- Almacena y escribe datos con precisión de milisegundos.
- Se puede escalar con facilidad.
- Utiliza GUI para generar gráficos.

A la hora de insertar datos a la base de datos, se puede realizar varias herramientas, tales como flume, collectd, vacuumetrix, para extraer datos de otras fuentes a OpenTSDB. Los datos almacenados en la base de datos tienen el formato de:

- Un nombre de métrica.
- Una marca de tiempo UNIX.
- Un valor, un evento con formato JSON o un histograma/resumen.
- Un conjunto de etiquetas, pares clave-valor, que describen la serie de tiempo correspondiente.

Para la visualización de datos, se puede usar como las demás bases de datos, la herramienta Grafana. La otra opción es utilizar el interfaz de usuario simple que viene incorporada el OpenTSDB, que permite seleccionar una o más métricas y etiquetas para generar gráfico.

A continuación, vamos a mostrar una tabla comparativa de estas bases de datos:

	Coste	API	Servidor y nube	Visualización de datos	otros
<u>InfluxDB</u>	Código abierto	API HTTP API InfluxDB-v2	Disponible para servidor y nube	Posibilidad de representar los datos mediante herramientas como Grafana	Compatibilidad de Flux. Los datos se almacenan en la nube
<u>Prometheus</u>	Código abierto	API HTTP	Versión para instalar en servidor	Posibilidad de representar los datos mediante herramientas como Grafana	Los datos se almacenan localmente. Gran cantidad de biblioteca
<u>TimescaleDB</u>	Código abierto	API con distintas funcionalidades como retención de datos, automatización, agregados continuos etc.	Disponible para servidor y nube	Posibilidad de representar los datos mediante herramientas como Grafana	Basado en PostgreSQL. Almacena datos relacionales junto con los datos de series de tiempo
<u>Graphite</u>	Código abierto	Graphite API API HTTP	Disponible para servidor y nube	Genera gráfica con su propio API de Render URL	Incorpora API para visualización de datos Amplio biblioteca
<u>QuestDB</u>	Código abierto	REST API API de Postgres API de InfluxDB	Versión para instalar en servidor	Possibilidad de representar los datos mediante herramientas como Grafana	Implementado con SQL Modelo relacional
<u>OpenTSDB</u>	Código abierto	API HTTP	Versión para instalar en servidor	Possibilidad de representar los datos mediante herramientas como Grafana	Escalable Incorpora una GUI Consta de demonio de serie temporal

Tabla 3.1 Tabla comparativa de BDs

Básicamente no hay muchas diferencias entre las bases de datos que hemos mencionados anteriormente. Todos son de código abierto y permiten almacenar datos

de series de tiempo y disponen de versión para instalar en el propio servidor. La diferencia que hay entre ellas son las APIs que soportan y las bibliotecas que tienen, esto nos proporcionarán más o menos funcionalidades. Podría ser interesante la API de generación de gráfica del Graphite, pero como los demás tienen la posibilidad de representar los datos mediante Grafana, que nos dan muchas más posibilidades a la hora de la visualización, sería poco relevante esta característica.

Entre todas estas bases de datos, hemos escogidos el TimescaleDB, ya que está basado en PostgreSQL. A la hora de poder aprovechar las extensiones de PostgreSQL, también permite almacenar datos relacionales junto con los datos de series temporales, ideal para nuestro proyecto, donde hay datos de ambos tipos.

3.3.2 Herramientas de visualización de datos

La visualización de datos se refiere a la representación gráfica de las informaciones y datos utilizando diferentes elementos, que pueden ser gráficos, mapas, diagramas, cuadro de mando, dashboard, etc. A través de estas representaciones, cualquier persona sería capaz de comprender los datos e interpretarlos sin dificultad.

Hay dos opciones que podemos optar para representar los datos de un sistema:

- Implementar un front end de una aplicación web para la visualización de datos. La implementación se puede usar cualquier lenguaje. El principal inconveniente de esta opción es la necesidad de un conocimiento bastante avanzada sobre la programación, a pesar de que podemos encontrar una multitud de plantillas y herramientas en la web que nos pueden servir para su desarrollo.
- Usando una herramienta de visualización de dashboard disponible en el mercado. Esta es la opción más sencilla, el uso de estas herramientas normalmente no requiere un conocimiento muy elevado, cualquier persona con un poco de conocimiento específico sobre el tema es capaz de usarlas.

Una de las herramientas de visualización muy conocidas es la Grafana. Grafana es una herramienta de código abierto con licencia Apache 2.0, escrita en Lenguaje Go y creada por el desarrollador sueco Torkel Ödegaard. Con esta herramienta, podemos crear representaciones dinámicas e interactivas. Algunas de sus características son:

- Admite una gran variedad de bases de datos tales como MySQL, PostgreSQL, InfluxDB, OpenTSDB, etc.
- Se puede ejecutar en diferentes plataformas, MacOS, Windows y Linux.
- Dispone de una serie de gráficos elegantes y paneles dinámicos, reutilizables y con numerosas opciones de visualización.
- Los paneles con funcionalidades distintos pueden agruparse en un tablero.
- Permite la configuración de alertas, cuando cumple alguna de las condiciones que hemos establecido con anterioridad, enviará una alarma por el medio de comunicación que tenemos configurado.



Figura 3.15 Captura del Demo de Grafana

3.4 Despliegue de la plataforma de monitorización

3.4.1 Instalación y configuración de TimescaleDB

3.4.1.1 Instalación

Para la instalación del TimescaleDB hay que instalar primero el PostgreSQL debido a que el TimescaleDB es una extensión del PostgreSQL.

La instalación del PostgreSQL es bastante sencilla, sólo tenemos que seguir los pasos que hay en la documentación oficial del PostgreSQL, la podemos encontrar en el siguiente enlace:

<https://docs.timescale.com/install/latest/self-hosted/installation-source/#configure-postgresql-after-installing-from-source>

3.4.1.2 Configuración

Una vez instalado, vamos a configurar el PostgreSQL. Hay dos maneras de hacerla, una es de forma manual editando el archivo postgresql.conf, y la otra es de forma automática con la herramienta timescale-tune, lo que hace esta herramienta es leer el archivo postgresql.conf, y de una forma interactiva, nos sugerirá los cambios que podemos realizar. Para la ejecución de este asistente, tenemos que usar el comando

Sudo timescaledb-tune

Terminada la ejecución del asistente, deberíamos de tener ya configurada el PostgreSQL. Pero hasta ahora sólo nos permite almacenar los datos relacionales, para poder almacenar datos de series de tiempo, es necesario habilitar la extensión de TimescaleDB. Para habilitarla basta con conectar a base de datos y ejecutar el comando:

```
create extension if not exist timescaledb cascade;
```

Este comando nos permite la creación de hypertabla, una abstracción de nivel más alto de muchas tablas individuales. Las hypertablas son las estructuras principales con las que trabaja TimescaleDB.

Para crear una hypertabla, tenemos que crear primero una tabla PostgreSQL normal y luego convertirla en hypertabla con la función `create_hypertable`.

```
create table conditions (
    time      TIMESTAMP WITH TIME ZONE NOT NULL,
    device_id TEXT,
    temperature NUMERIC,
    humidityNUMERIC
);
```

Creada la tabla SQL normal, vamos a convertirla ahora en una hypertabla.

```
select create_hypertable('conditions', 'time');
```

El comando anterior invoca la función `create_hypertable()`, crea una hypertabla de TimescaleDB a partir de una tabla de PostgreSQL y la sustituye.

3.4.2 Instalación y configuración de Grafana

3.4.2.1 Instalación

Para instalar Grafana, podemos usar la herramienta apt, lo primero que tenemos que hacer es añadir los repositorios de Grafana a nuestro sistema, y luego ejecutar el comando:

```
sudo apt install grafana
```

3.4.2.2 Configuración

Instalada la herramienta, tenemos que habilitar el servicio usando el comando:

```
Sudo systemctl start grafana-server.service
```

Si queremos comprobar su funcionamiento, podemos ejecutar:

Sudo systemctl status grafana-server.service

Después de habilitar el servicio, ya podemos acceder al Grafana, para eso tenemos que abrir el navegador e ir a la página de Grafana mediante el link <http://GRAFANAIP:puerto> o <http://localhost:puerto> y loqueamos con usuario admin y password admin. Si es la primera vez que logueamos, nos pedirá cambiar la contraseña por seguridad.

Con eso ya tenemos configurado la Grafana, pero para poder visualizar los datos es necesario conectarse primero a algunas bases de datos.

Capítulo 4. Dispositivos basados en Withings

4.1 Estudio de los dispositivos wearables Withings

Withings (<https://www.withings.com/es/es/>) es una empresa de electrónica de consumo francesa fundada por Éric Carreel, Cédric Hutchings y Fred Potter en 2008. Está dedicado a la fabricación de dispositivos inteligentes para la monitorización de salud humana. Desde su fundación hasta ahora, ha lanzado una gran variedad de productos de distintos tipos. Según sus categorías, podemos clasificarlos en:

- Pulseras inteligentes, en esta categoría encontramos el Pulse HR y el Withings Pulse, son pulseras que poseen unos sensores encargados de recopilar las métricas del usuario y una pantalla táctil con la finalidad de mostrar notificaciones y las métricas recogidas del usuario. Disponen de conexión Bluetooth para conectar a los teléfonos móviles.

Algunos de los parámetros que pueden medir son la frecuencia cardíaca, la distancia recorrida, los pasos realizados, etc.



Figura 4.1 Pulse HR

- Relojes inteligentes, en esta categoría se encuentran una gran variedad de relojes, como el Move ECG, Steel, Steel HR, ScanWatch, etc. Estos relojes pueden clasificarse a su vez en relojes analógicos, por ejemplo, Withings Move ECG, o híbridos, como ScanWatch.
 - Los relojes analógicos tienen un diseño de un reloj tradicional, se diferencian en que cuentan con unos sensores y la conexión Bluetooth. Gracias a los sensores que llevan, son capaces de medir ECG, registrar las actividades diarias etc.
 - Los relojes híbridos son relojes que combinan el reloj tradicional con digital, además de tener el diseño de un reloj analógico, vienen equipados con una pequeña pantalla circular que sirve para mostrar las notificaciones y las informaciones. Llevan más sensores que los de tipo analógicos, por lo que son más las medidas que pueden medir, como el Vo2Max, seguimiento de sueño, SpO2, etc.



Figura 4.2 Steel Hr (izquierda) y ScanWatch Horizon (derecha)

- Básculas inteligentes, aquí podemos encontrar el Body Cardio, Body+, Body comp, etc. Son básculas que, una vez realizado la medición, a través de la conexión WIFI, envía los resultados a la aplicación de los usuarios. Las características generales de estos productos son la conectividad Bluetooth y WIFI, posee detector de posición corporal a través de la tecnología Position Control, que permite una mayor precisión del pesaje, tienen tecnología de medición de frecuencia cardíaca patentada, son capaces de reconocer hasta 8 usuarios diferentes, etc.

Además del peso e IMC que pueden medir una báscula inteligente normal, son capaces de medir la frecuencia cardíaca, masa muscular, masa ósea, velocidad de onda de pulso, el nivel de agua corporal, etc.



Figura 4.3 Body+ (izquierda) y Body Cardio (derecha)

- Tensiómetros, el BPM Connect y el BPM Core, igual que las básculas de Withings, estos tensiómetros disponen también de la conectividad Bluetooth, la conexión WIFI, que permite sincronizar los datos de la medición inmediatamente después de su uso. Sus características más destacables son:
 - Fácil de usar, sólo hay que pulsar un único botón para todas las acciones, tanto para sincronizar con el dispositivo por primera vez, como para realizar las mediciones.
 - Fácil de entender, además de mostrar las lecturas obtenidas en la medición con luces LED grandes en la pantalla, disponible también una retroalimentación de resultados mediante luces de colores, por lo que se puede saber fácilmente nuestro estado antes de abrir la aplicación.
 - Fácil de transportar, sus estructuras en una sola pieza y el manguito suave hace que su transporte sea más ligero.
 - Rendimiento validado clínicamente.

Algunos de las métricas que son capaces de medir son las tensiones arteriales, frecuencia cardíaca, ECG, detección de la fibrilación auricular, etc. Los resultados obtenidos se pueden compartir fácilmente con un médico o especialista.



Figura 4.4 BPM Connect (izquierda) y BPM Core (derecha)

- Monitor de sueño, Sleep analyzer, es un dispositivo para el control de sueño que es capaz de proporcionar resultados con una precisión clínica. No es necesario llevar ningún dispositivo para la medición, consiste en una pequeña esterilla que se coloca por debajo del colchón por la zona del pecho cuando dormimos. Dispone de conectividad Bluetooth 4.0 y la función de sincronización automática mediante WIFI. Posee un sensor neumático y un sensor de sonido, que miden la frecuencia cardíaca e identifica las señales de audio. Una de la característica más destacada de este dispositivo es la detección de apnea del sueño, además de esto, también monitoriza la duración del sueño, los ciclos del sueño, la frecuencia cardíaca, el ronquido, etc.

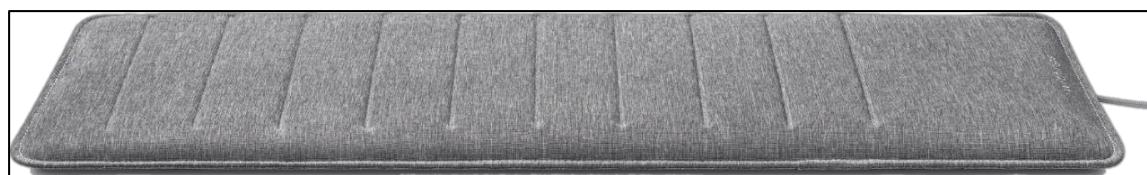


Figura 4.5 Sleep Analyzer

4.2 Infraestructura de Withings

Para el acceso de los datos almacenados en la nube, Withings nos proporciona una API pública, que tiene dos planes:

- Plan estándar o gratis, está limitado a 5000 usuarios activos y 120 solicitudes por minuto.
- Plan de empresa, en lugar de 5000 usuarios activos que tiene el plan estándar, este plan tiene un numero de usuario activos ilimitados, y frente a 120 solicitudes por minutos, este dispone de 1500 solicitudes por minutos. Además, tiene una entrega de notificación de actualización de datos más rápidas y dispone de soporte técnico de Withings.

Ambos planes nos permiten acceder a los datos de la salud de los usuarios que usan el dispositivo y extraerlos de la nube del Withings. Los pasos que tenemos que seguir para usar la API pública son:

- Creación de la cuenta del desarrollador. Antes de nada, lo primero que tenemos que hacer es crearnos una cuenta Withings para nuestra organización, esta cuenta nos proporcionará un `client_id` y un `client_secret` que serán utilizados más tarde para el acceso de los servicios Withings.

El `client_id` y el `client_secret` podrá ser actualizado posteriormente en el panel del desarrollador, cuyo url es <https://developer.withings.com/dashboard/>, encontramos aquí también restos de las informaciones asociadas a la cuenta.

- Generación de firma, algunas API de Withings puede solicitar un valor hash como firma para autenticar al socio, que consta de un `action`, `client_id` y un `nonce`. Para generar dicha firma, se puede usar el servicio Signature v2 – Getnonce.
- Solicitud de permiso. Para el acceso a los datos de salud de los usuarios de la cuenta, debemos pedir permisos de aplicación al usuario mediante el método de autorización de la API OAuth2 de Withings, que como resultado obtenemos un `Access_token` y un `refresh_token`.
- Extracción de datos. Una vez obtenido el permiso y el `Access_token`, ya podemos empezar a extraer los datos de la nube usando los distintos servicios que ofrece la API, como Measure v2 – Getactivity, Measure v2 – Getworkouts, Heart – Get, User v2 – Get, etc.

A continuación, vamos a describir brevemente el protocolo OAuth 2. El OAuth 2 es un protocolo de autorización que permite a las aplicaciones obtener acceso a las cuentas de los usuarios delegando la autenticación de usuario al servicio que gestiona las cuentas. En este protocolo, podemos identificar 4 roles:

- Propietario del recurso, es el usuario que autoriza a una aplicación para acceder a su cuenta, dicho acceso está limitado por el alcance o scope que define el usuario a la hora de dar la autorización.
- Cliente, es la aplicación que intenta acceder a la cuenta del usuario.
- Servidor de recurso, es el servidor donde se encuentran las cuentas de los usuarios.
- Servidor de autorización, es el servidor que identifica la identidad del usuario, una vez verificado la identidad, genera un token de acceso.
- Podemos referirnos al servidor de recurso junto con el servidor de autorización con el nombre de servicio o API.

El flujo genérico del protocolo OAuth2 consiste en:

1. La aplicación solicita una autorización con el fin de poder acceder a los recursos de un usuario.

2. El usuario otorga la autorización a la aplicación aceptando la solicitud, y la aplicación recibe dicha autorización.
3. Con la identidad propia y la autorización obtenida previamente, la aplicación solicita al servidor de autorización el token de acceso.
4. Si los datos proporcionados por la aplicación son todos correctos, el servidor de autorización proporciona un token de acceso a la aplicación, y con esto, se finaliza el proceso de autorización.
5. Ahora con el token de acceso obtenido anteriormente, la aplicación puede solicitar algún recurso al servidor de recurso.
6. Si el token es válido, el servidor de recurso entrega a la aplicación el recurso solicitado.
7. Cuando se haya caducado el access_token, la aplicación puede pedir un nuevo access_token al servidor de autorización.

En el siguiente diagrama podemos ver el flujo que sigue el cliente a la hora de solicitar el permiso y los recursos.

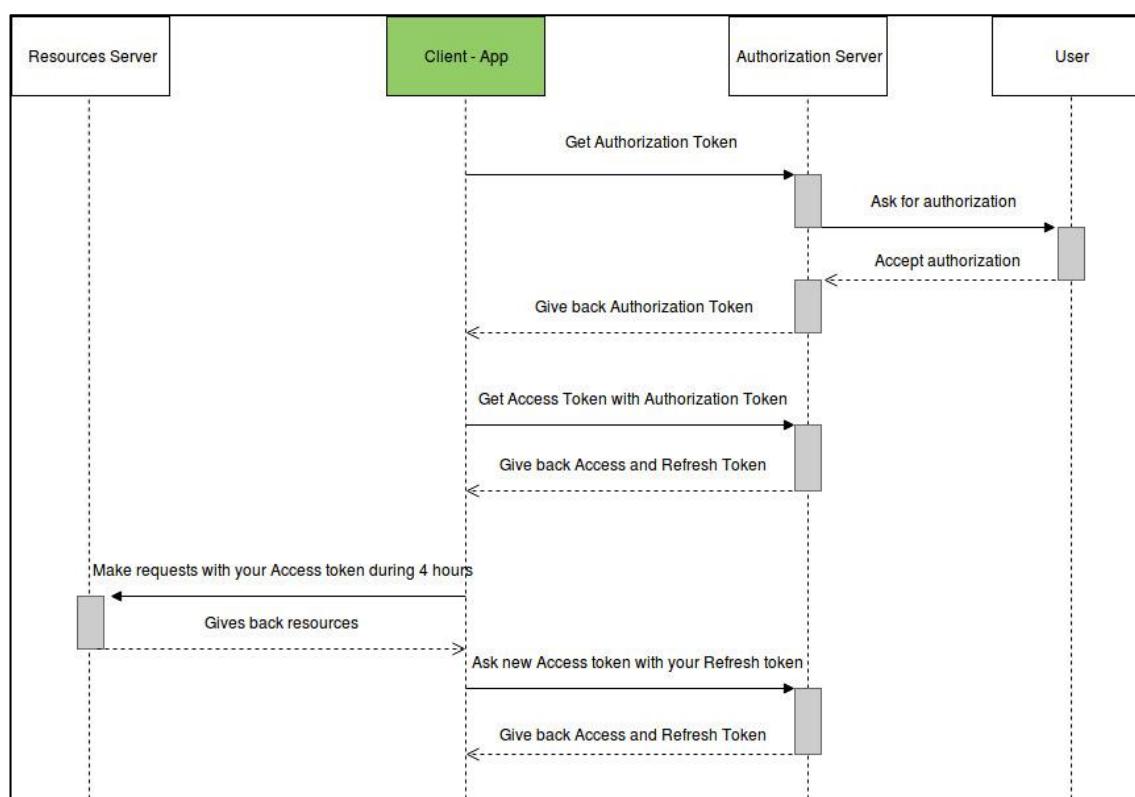


Figura4.6 Diagrama Oauth 2 (Fuente: <https://developer.withings.com/developer-guide/v3/integration-guide/public-health-data-api/get-access/oauth-web-flow>)

Los scopes que el usuario puede definir durante el otorgamiento de autorización son:

- User.activity, permite obtener datos relacionados con la actividad física y de sueño del usuario.
- User.metrics, permite obtener datos relacionados con las mediciones puntuales del

usuario.

- User.info, permite obtener datos personales del usuario
- User.sleepevents, usado para recibir notificaciones cuando ocurre un evento relacionado con el sueño. [Wit22a]

4.3 Parámetros medibles

Como ya hemos visto en el punto 4.1, los dispositivos Withings permiten medir diversos parámetros fisiológicos, pero no todos tienen la misma capacidad de medición, dependiendo de su hardware, uno puede medir más parámetros o menos, a continuación, vamos a mostrar una tabla de las variables que se pueden medir dependiendo de la categoría y modelo de dispositivos, extraída de la página web de Withings.

- Relojos inteligentes

Productos	Medidas	Servicios	Scopes
Move Steel Activité Pop Go	● Pasos ● Distancia ● Calorías ● Intensidad	● Measure v2 – Getactivity ● Measure v2 – Getintradayactivity	● User.activity
	● Entrenamientos	● Measure v2 – Getworkouts	● User.activity
	● Duración sueño ● Duración estado sueño - Despierto - Ligero - Profundo ● Sleep wakeup counts	● Sleep v2 – Get ● Sleep v2 – Getsummary	● User.activity
Pulse 0x	● Pasos ● Distancia ● Calorías ● Intensidad	● Measure v2 – Getactivity ● Measure v2 – Getintradayactivity	● User.activity
	● Entrenamientos	● Measure v2 – Getworkouts	● User.activity
	● Manual SpO2	● Measure v2 – Getmeas	● User.metrics
Steel HR Pulse HR	● Pasos ● Distancia ● Calorías ● Intensidad ● Frecuencia	● Measure v2 – Getactivity ● Measure v2 – Getintradayactivity	● User.activity

	cardíaca continua		
	● Entrenamientos	● Measure v2 – Getworkouts	● User.activity
	● Frecuencia cardíaca puntual ● VO2max	● Measure v2 – Getmeas	● User.metrics
	● Duración sueño ● Duración estado sueño - Despierto - Ligero - Profundo ● Sleep wakeup counts ● Frecuencia cardíaca en sueño	● Sleep v2 – Get ● Sleep v2 - Getsummary	● User.activity
Scanwatch	● Pasos ● Distancia ● Calorías ● Intensidad ● Swim strokes ● Pool laps ● Elevación ● Frecuencia cardíaca continua ● SpO2 auto	● Measure v2 – Getactivity ● Measure v2 – Getintradayactivity	● User.activity
	● Entrenamientos	● Measure v2 – Getworkouts	● User.activity
	● Frecuencia cardíaca puntual ● VO2max ● SpO2 manual ● Duración intervalo QRS ● Duración intervalo QT ● Duración intervalo QTC ● Duración intervalo PR	● Measure – Getmeas	● User.metrics

	<ul style="list-style-type: none"> ● Fibrilación auricular de PPG 		
	<ul style="list-style-type: none"> ● Duración sueño ● Duración estado sueño <ul style="list-style-type: none"> - Despierto - Ligero - Profundo - Rem ● Sleep wakeup counts ● Frecuencia cardíaca en sueño 	<ul style="list-style-type: none"> ● Sleep v2 – Get ● Sleep v2 – Getsummary 	<ul style="list-style-type: none"> ● User.activity
	<ul style="list-style-type: none"> ● Señal ECG ● Fibrilación auricular de ECG 	<ul style="list-style-type: none"> ● Heart – Get ● Heart – List 	<ul style="list-style-type: none"> ● User.metrics

Tabla 4.1 Parámetros medibles por los relojes

- Básculas inteligentes

Productos	Medidas	Servicios	Scopes
Body WBS01 Wireless Scale Smart Kid Scale	<ul style="list-style-type: none"> ● Weight 	<ul style="list-style-type: none"> ● Measure – Getmeas 	<ul style="list-style-type: none"> ● User.metrics
Body+	<ul style="list-style-type: none"> ● Weight ● Masa muscular ● Masa ósea ● Grasa corporal ● Masa libre de grasa ● Fat ratio 	<ul style="list-style-type: none"> ● Measure – Getmeas 	<ul style="list-style-type: none"> ● User.metrics
Smart Body Analyzer	<ul style="list-style-type: none"> ● Peso ● Masa muscular ● Masa ósea ● Grasa corporal ● Masa libre de grasa ● Fat ratio 	<ul style="list-style-type: none"> ● Measure – Getmeas 	<ul style="list-style-type: none"> ● User.metrics

	<ul style="list-style-type: none"> ● Frecuencia cardíaca 		
Body Cardio	<ul style="list-style-type: none"> ● Peso ● Masa muscular ● Masa ósea ● Grasa corporal ● Masa libre de grasa ● Fat ratio ● Frecuencia cardíaca ● Velocidad de la onda de pulso 	<ul style="list-style-type: none"> ● Measure – Getmeas 	<ul style="list-style-type: none"> ● User.metrics

Tabla 4.2 Parámetros medibles por las básculas

- Dispositivos de monitorización de sueño

Productos	Medidas	Servicios	Scopes
Aura Sensor	<ul style="list-style-type: none"> ● Duración sueño ● Estado sueño ● Duración estado sueño <ul style="list-style-type: none"> - Despierto - Ligero - Profundo - REM ● Sleep wakeup counts 	<ul style="list-style-type: none"> ● Sleep v2 – Get ● Sleep v2 – Getsummary 	<ul style="list-style-type: none"> ● User.activity
Sleep Sensor	<ul style="list-style-type: none"> ● Duración sueño ● Estado sueño ● Duración estado sueño <ul style="list-style-type: none"> - Despierto - Ligero - Profundo - REM ● Sleep wakeup counts ● Frecuencia 	<ul style="list-style-type: none"> ● Sleep v2 – Get ● Sleep v2 – Getsummary 	<ul style="list-style-type: none"> ● User.activity

	cardíaca ● Ritmo respiratorio ● Ronquidos		
--	--	--	--

Tabla 4.3 Parámetros medibles por los dispositivos de monitorización de sueño

- Monitores de presión arterial

Productos	Medidas	Servicios	Scopes
Blood Pressure Monitor	● Presión arterial diastólica		
Wireless Blood Pressure Monitor	● Presión arterial sistólica	● Measure – Getmeas	● User.metrics
Blood Pressure Monitor Travel	● Frecuencia cardíaca		

Tabla 4.4 Parámetros medibles por monitores de presión arterial

- Termómetros

Productos	Medidas	Servicios	Scopes
Thermo	● Temperatura ● Temperatura corporal ● Temperatura cutánea	● Measure – Getmeas	● User.metrics

Tabla 4.5 Parámetros medibles por los termómetros

4.4 Despliegue e integración de la infraestructura Withings

La infraestructura de Withings consta de un subsistema de medición de datos, el gateway, la API Withings y el subsistema de visualización. A continuación, se encuentra un esquema de la infraestructura.

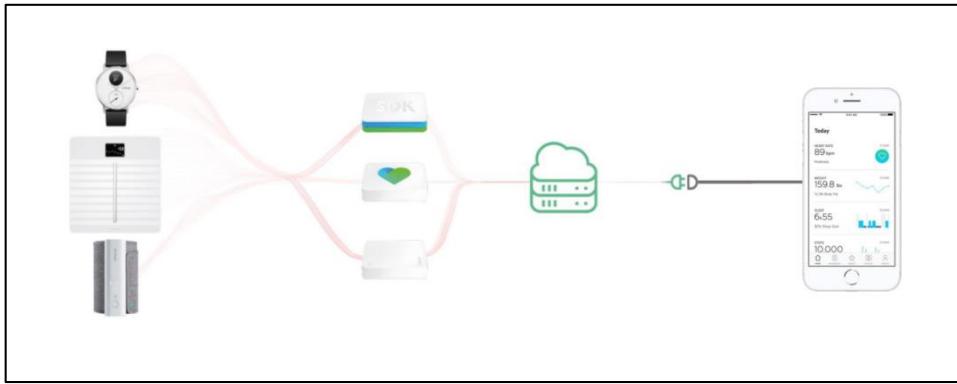


Figura 4.7 Esquema de la infraestructura Withings capturado en la web del Withings developer

El subsistema de medición está compuesto por uno o varios dispositivos inteligentes del Withings. Los dispositivos wearables registrarán de forma continua todas las actividades diarias del usuario, como los datos de entrenamiento, de la frecuencia cardíaca, del sueño en el caso de llevarlo puesto durante la noche, etc. Y el resto de los dispositivos medirá puntualmente los parámetros fisiológicos cuando el usuario haya hecho la medición con ellos.

Estos datos registrados por los dispositivos a través de un gateway, puede ser un SDK (Kit de Desarrollo de Software), el dispositivo móvil del usuario o bien el router que tenemos en la casa, son enviados a la API del Withings, lugar donde se encuentra la nube de almacenamiento de datos de los usuarios.

Sincronizado los datos, con el subsistema de visualización, la aplicación móvil Health Mate del Withings, el usuario puede consultar dichos datos en cualquier momento y visualizarlos de una forma clara y personalizada.

4.5 Estudio de parámetros medibles por dispositivos Withings

Son muchas los parámetros que se pueden medir con los dispositivos de Withings tales como el peso, la altura, la frecuencia cardíaca, la temperatura, la masa muscular, etc. Para poder medir todos estos parámetros suelen utilizarse distintos tipos de dispositivos. En este documento, vamos a mostrar un poco sobre cómo se puede obtener los datos de cada uno de los dispositivos Withings que se han manejado; concretamente un reloj y una balanza inteligente.

4.5.1 Acceso a los datos con Postman

Los datos recogidos por los dispositivos son almacenados en la nube de Withings, las

podemos obtener fácilmente usando la colección de Postman que nos proporciona el propio fabricante, esta colección nos permite realizar peticiones al servidor de recurso usando el token de acceso y recuperar los datos. Su uso es muy intuitivo y fácil, aquí vamos a mostrar algún ejemplo de realizar peticiones con Postman.

- Obtención de datos de actividad

Primero tenemos que seleccionar las peticiones que hay en el workspaces el Measure v2 – Get activity. En el body, tenemos que llenar los parámetros necesarios para este método, los cuales se pueden encontrar en la sección siguiente. [Wit22c]

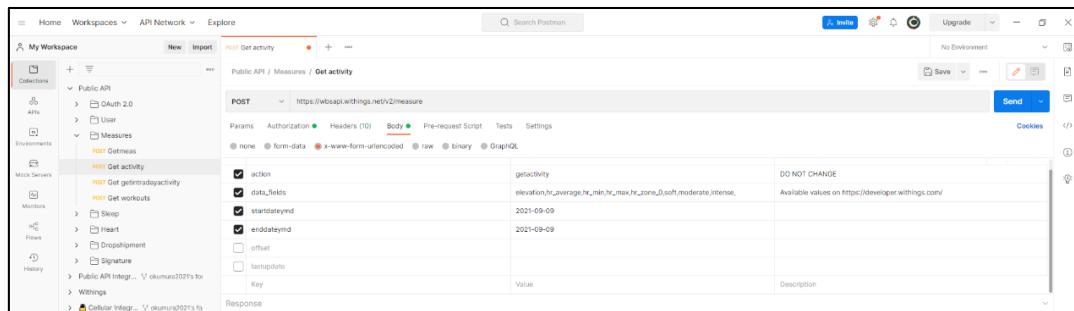


Figura 4.8 captura de Postman sobre método Getactivity

Rellanado los parámetros, antes de lanzar la petición, en la pestaña Authorization debemos poner el access_token que hemos obtenido con el protocolo OAuth 2.

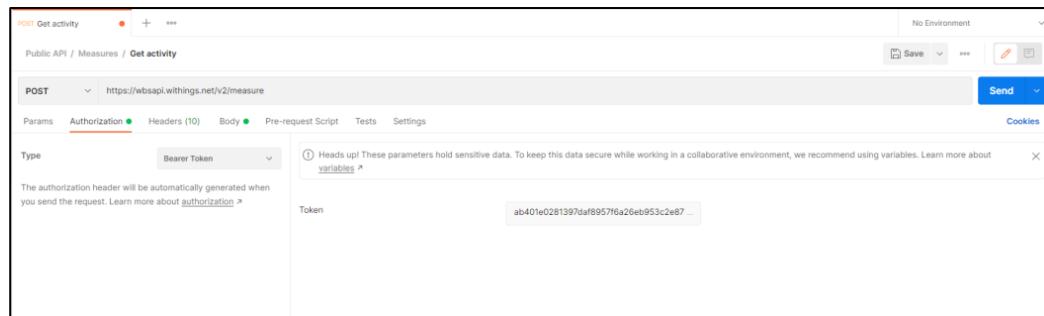


Figura 4.9 access_token

Ahora ya podemos enviar la petición dando clic al botón send. Si todos los datos son correctos, el servidor debería devolvernos una respuesta en Json con los datos solicitados.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

```

Figura 4.10 resultado del getactivity

Para el resto de los métodos se hacen de la misma manera.

- Obtención de datos de sueño

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

```

Figura 4.11 petición y resultado de Sleep v2 - Get

POST https://wbsapi.withings.net/v2/sleep

KEY	VALUE	DESCRIPTION
action	getsummary	DO NOT CHANGE
startdateymd	2021-11-24	
enddateymd	2021-11-25	
data_fields	sleep_efficiency,total_sleep_time,wakeup_latency,total_timeinbed,nb	Available values on https://developer.withings.com/
lastupdate		

```

3 "body": {
4   "series": [
5     {
6       "id": 2448331913,
7       "timezone": "Europe/Madrid",
8       "model": 16,
9       "model_id": 93,
10      "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",
11      "startdate": 1657718400,
12      "enddate": 1657741168,
13      "date": "2021-11-24",
14      "data": [
15        {
16          "total_timeinbed": 22680,
17          "total_sleep_time": 21480,
18          "sleep_efficiency": 0.95,
19          "wakeup_latency": 360,
20          "no_rem_episodes": 0,
21          "lightsleepduration": 9180,
22          "deepsleepduration": 12300,
23          "hr_average": 61,
24          "hr_max": 68
25        },
26      ],
27      "created": 1637814188,
28    }
29  ],
30 }
31 
```

Figura 4.12 petición y respuesta de sleep v2 – Getsummary

● Obtención de señales de ecg

POST https://wbsapi.withings.net/v2/heart

KEY	VALUE	DESCRIPTION
action	list	DO NOT CHANGE
startdate	1663722000	
enddate	1663765200	
data_fields		

```

1 "status": 0,
2 "body": {
3   "series": [
4     {
5       "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",
6       "model": 93,
7       "ecg": {
8         "signalid": 165052981,
9         "afib": 0
10      },
11      "heart_rate": 65,
12      "timestamp": 1663745210,
13      "modified": 1663969333
14    },
15    {
16      "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",
17      "model": 93,
18      "ecg": {
19        "signalid": 164243666,
20        "afib": 0
21      },
22      "modified": 1663969333
23    }
24  ]
25 } 
```

Figura 4.13 petición y respuesta de Heart v2 – List

The screenshot shows a Postman interface with the following details:

- Request URL:** https://wbsapi.withings.net/v2/heart
- Method:** POST
- Body:** x-www-form-urlencoded
- Params:**

KEY	VALUE	DESCRIPTION
action	get	DO NOT CHANGE
signalid	165052981	Get signalid with v2 Heart - List
- Response Headers:**
 - Status: 200 OK
 - Time: 207 ms
 - Size: 28.54 KB
 - Save Response
- Response Body (Pretty JSON):**

```

1
2   "status": 0,
3   "body": {
4     "signal": [
5       1,
6       2,
7       -2,
8       -1,
9       -11,
10      -14,
11      -14,
12      -8,
13      -8,
14      -7,
15      -2,
16      2,
17      4,
18      -1,
19      -1,
20      2,
21      -6
22

```

Figura 4.14 petición y respuesta de Heart v2 – Get

4.5.2 Medidas relacionadas con el reloj

4.5.2.1 Medidas de actividad física

● Steps:

Proporciona el número de pasos que damos durante un día o un periodo de tiempo según el método que usamos.

➤ Measure v2 – Getactivity

Los parámetros necesarios son:

- Action, un string que toma el valor getactivity
- Startdateymd, es la fecha inicio de los datos
- Enddateymd, es la fecha fin de los datos
- Data_fields, es una lista de string que indica los datos de las medidas que queremos recuperar, en este caso es steps.

La respuesta devuelta por el servidor tiene la siguiente estructura:

```
{  
    "status": 0,  
    "body": {  
        "activities": [  
            {  
                "steps": 913,  
                "deviceid": null,  
                "hash_deviceid": null,  
                "timezone": "Europe/Madrid",  
                "date": "2021-09-09",  
                "brand": 18,  
                "is_tracker": true  
            },  
            {  
                "steps": 2246,  
                "deviceid": null,  
                "hash_deviceid": null,  
                "timezone": "Europe/Madrid",  
                "date": "2021-09-10",  
                "brand": 18,  
                "is_tracker": true  
            }  
        ],  
        "more": false,  
        "offset": 200  
    }  
}
```

Figura 4.15 Respuesta de la petición de steps Getactivity

➤ Measure v2 – Getintradayactivity

Los parámetros query para esta petición son:

- Action, es un string que toma el valor getintradayactivity
- Startdate, es un entero que indica el tiempo de inicio
- Enddate, es un entero que indica el tiempo fin
- Data_field, es una lista de string que indica las medidas que deseamos obtener, en este caso steps

Si la diferencia de horas entre startdate y enddate es mayor que 24 horas, solo se devuelven los datos capturados durante las primeras 24 horas desde startdate.

Y si no se proporciona el startdate y enddate, se devuelve por defecto los datos más recientes.

La respuesta devuelta por el servidor tiene la siguiente estructura:

```
{  
    "status": 0,  
    "body": {  
        "series": {  
            "1631146200": {  
                "steps": 6,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631146260": {  
                "steps": 6,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631150340": {  
                "steps": 8,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            . . .  
            . . .  
            . . .  
            "1631225700": {  
                "steps": 8,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631225940": {  
                "steps": 13,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631226480": {  
                "steps": 9,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            }  
        }  
    }  
}
```

Figura 4.16 Respuesta de la petición de Getintradayactivity

Aquí solo vamos a mostrar la descripción de una medida para no hacer demasiado largo esta sección, más detalles se encuentra en el anexo 3.

4.5.3 Medidas relacionadas con la balanza

Igual que la sección anterior, se puede ver en el anexo 3.

4.5.4 Clasificación de las medidas

Una vez realizado el análisis de los distintos parámetros que podemos recuperar desde el servidor, ya podemos clasificarlos según si es un parámetro instantáneo o es de tipo

resumen, y decidir que parámetros escogemos para almacenar en nuestra base de datos.

4.5.4.1 Medidas de tipo resumen

Las medidas de tipo resumen son las que nos muestran los datos de un determinado parámetro en un periodo de tiempo determinado o en un día concreto. Principalmente son las que están relacionadas con el sueño y las actividades. Las podemos obtener usando el método Getactivity y Getsummary, Las cuales son:

De actividad

- Steps
- Distance
- Calories
- Intensity
- Elevation
- Workouts

De sueño

- Nb_rem_episodes
- Sleep_efficiency
- Sleep_latency
- Total_sleep_time
- Total_timeinbed
- Wakeup_latency
- Waso
- Apnea_hypopnea_index
- Breathing_disturbances_intensity
- Asleepduratin
- Deepsleepduration
- Hr_average, hr_max, hr_min,
- Lightsleepduration
- Night_events
- Out_of_bed_count
- Remsleepduration
- Rr_average, rr_max, rr_min
- Sleep_score
- Snoring
- Snoringepisodecount
- Wakeupcount
- Wakeupduration

4.5.4.2 Medidas instantáneas

Son aquellas medidas que se miden en un tiempo (segundo) concreto o durante un periodo de tiempo muy corto, como la medición de ECG. Las cuales son:

De actividad

- Steps
- Distance
- Calories
- Intensity
- Elevation

De salud

- Heart_rate
- SpO2_auto
- VO2max
- Manual SpO2
- Sleep heart rate
- Rr
- Snoring
- Sdnn_1
- Rmssd
- Ecg
- QRS Interval duration
- PR Interval duration
- QT Interval duration
- QTC Interval duration

Medidas generales

- Weight
- Muscle mass
- Bone mass
- Fat mass
- Fat free mass
- Fat ratio
- Pulse wave velocity

4.5.4.3 Medidas a almacenar

Después del análisis de los datos, creo que serían interesantes para nuestro proyecto recoger los siguientes datos.

- Las que son tipo resumen, que nos devuelven el resumen en un día:

Medidas de actividad:

- Steps
- Distance
- Calories
- Elevation
- Workouts
- Intensidad

Medidas de la salud:

- Sleep_efficiency
- Total_sleep_time
- Deepsleepduration
- Hr_average, hr_max, hr_min,
- Lightsleepduration
- Remsleepduration
- Rr_average, rr_max, rr_min
- Sleep_score
- Snoring
- Snoringepisodecount

- Las medidas instantáneas:

Medidas de la salud:

- Heart_rate
- SpO2_auto
- VO2max
- Manual SpO2
- Ecg
- QRS Interval duration
- PR Interval duration
- QT Interval duration
- QTC Interval duration
- Weight
- Muscle mass
- Bone mass
- Fat mass
- Fat free mass
- Fat ratio
- Pulse wave velocity

Capítulo 5. Sistema MONITOR

5.1 Especificación informal de la plataforma

La plataforma que se va a desarrollar es una plataforma de monitorización continua de salud de las personas. Durante el día, el usuario llevará puesto uno o varios dispositivos inteligente wearable, que recopilarán los datos relacionados con la salud del usuario, las actividades diarias realizados por el usuario y los entrenamientos en caso de si los hubiera practicado.

Durante la noche, al acostarse, el dispositivo realizará un seguimiento del estado del sueño del usuario, tales como las fases que se encuentran, la frecuencia cardíaca en sueño, la tasa de respiración, etc.

Además de esos datos, el dispositivo permite también la medición del peso, la masa muscular, la masa ósea, el porcentaje del agua en el cuerpo, etc.

Los datos recogidos por estos dispositivos quedarán almacenados en el móvil vinculado del usuario y también se almacenará en una nube de almacenamiento de datos que proporciona la empresa de los dispositivos. Posteriormente estos datos podrán ser recuperados y visualizados.

La plataforma recuperará de forma automática y continua los datos del usuario registrados en dicha nube y los almacenan en su base de datos. Para estudiar la evolución de los datos registrados, se utilizará un cuadro de mandos o dashboard, que permite al usuario una mejor visualización de estos datos y hacer un seguimiento más

fácil.

5.2 Modelado de requisitos

En este punto se va a listar tanto los requisitos funcionales como no funcionales del sistema.

5.2.1 Requisitos funcionales.

- RF1. El sistema debe ser capaz de realizar medidas continuas con los dispositivos en los usuarios.
- RF2. El sistema debe registrar las medidas que se van obteniendo en los dispositivos.
- RF3. El sistema almacenará automáticamente en su base de datos los datos obtenidos según sus categorías.
- RF4. El sistema permitirá la consulta de datos.
- RF5. El sistema permitirá la visualización de los datos almacenados.
- RF6. El sistema permitirá crear y modificar el Dashboard con distintos datos.

5.2.2 Requisitos no funcionales.

- RNF1. El sistema debe ser capaz de operar adecuadamente ante una gran cantidad de datos recibidos.
- RNF2. El sistema permite obtener los datos de salud del usuario de forma automática y continua.
- RNF3. La plataforma debe ser fácil de usar para usuario con poca experiencia.
- RNF4. El sistema debe estar disponible mayor tiempo posible.
- RNF5. El sistema debe poder ejecutarse en Windows y Linux.
- RNF6. Los datos de series de tiempo deben tener obligatorio un timestamp.
- RNF7. Cada uno de los datos almacenados debe estar asociado a un número de identificación único.

5.2.3. Especificación del sistema.

El sistema MONITOR está compuesto por dos partes diferenciadas: a) Infraestructura de Withings y b) Sistema Central.

La infraestructura de Withings incluye la plataforma Withings que se ha utilizado para el desarrollo del proyecto y que provee herramientas para la obtención de medidas de salud y de actividad física, así como un almacenamiento temporal de los mismos asociadas a los dispositivos utilizados y el usuario que los utiliza.

El sistema central se encarga de acceder a los datos, registrarlos, procesarlos y visualizarlos de acuerdo a los intereses personales de los usuarios.

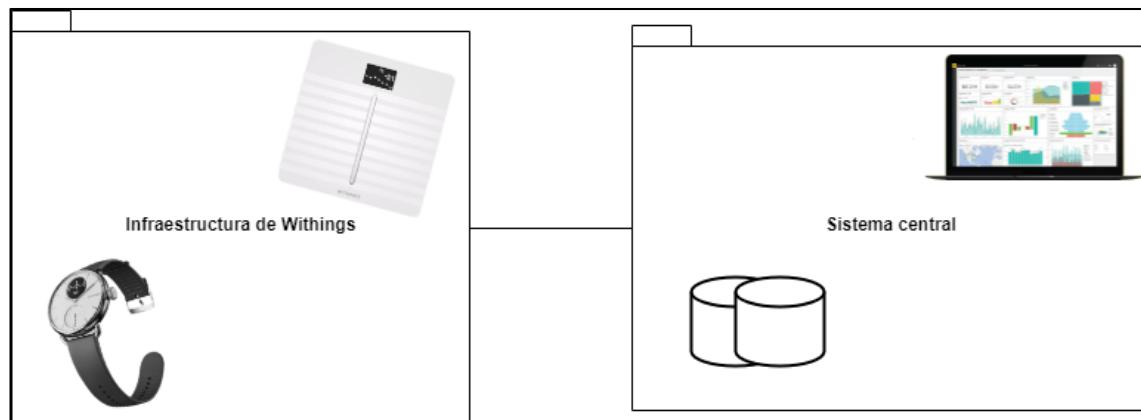


Figura 5.1 infraestructura Withings y sistema central

A continuación, se detalla la interacción de la infraestructura de Withings con el sistema central que es la parte principal que se ha desarrollado.

5.2.3.1 Actores del sistema

Los actores que podemos encontrar en el sistema son

- Withings, la infraestructura de Withings. Encargada de recolectar los datos registrados por los dispositivos.
- Usuario, el usuario de nuestro sistema.

5.2.3.2 Casos de uso

Los casos de uso los vamos a separar por paquetes en base a la funcionalidad y al actor del sistema.

- Configurar dispositivos wearables. El usuario se configura los dispositivos wearables para poder medir los datos y transmitirlos a la nube de Withings.

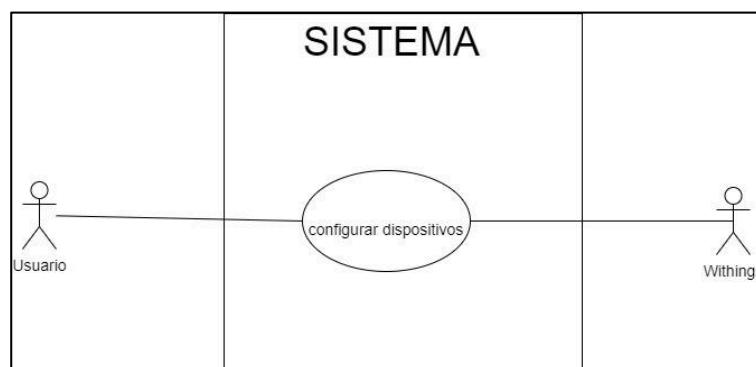


Figura 5.2 Caso de uso – configuración de dispositivo

- Obtener medida. El usuario pide el valor de alguna medida al Withings.

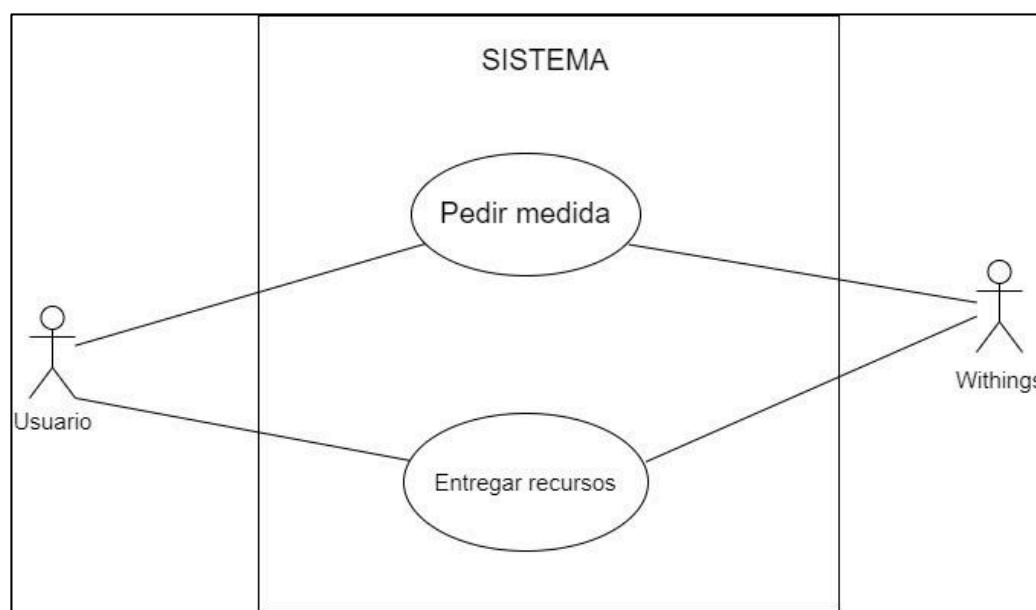


Figura 5.3 Caso de uso - obtener medida

- Registrar medida. El sistema MONITOR pide determinada medida al Withings y almacena su valor en base de datos local.

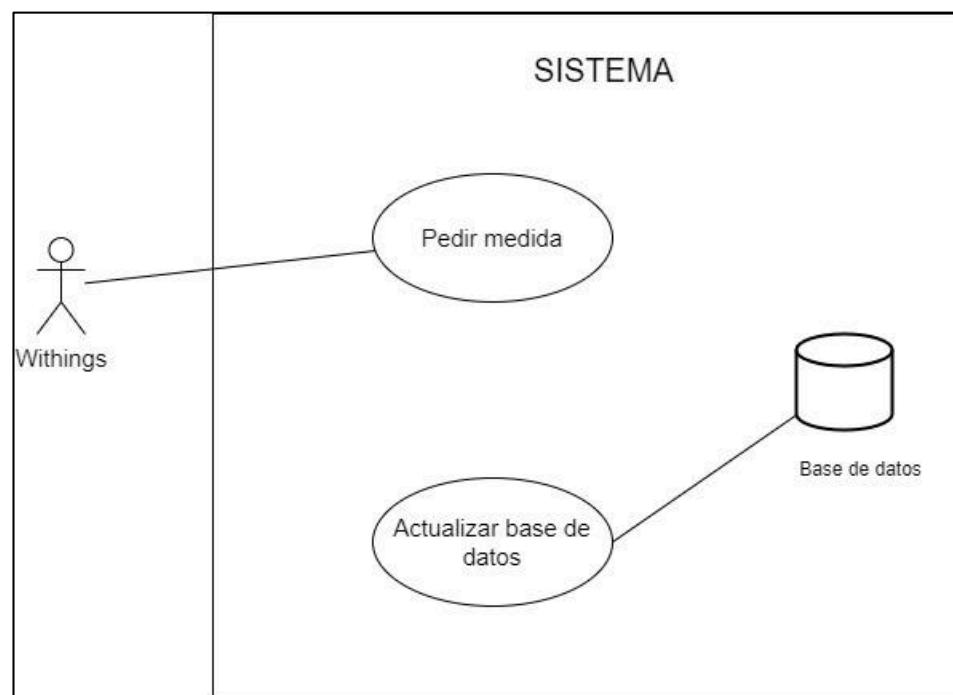


Figura 5.4 Caso de uso – registrar medida

- Visualizar medida. El usuario visualiza los datos de determinadas medidas.

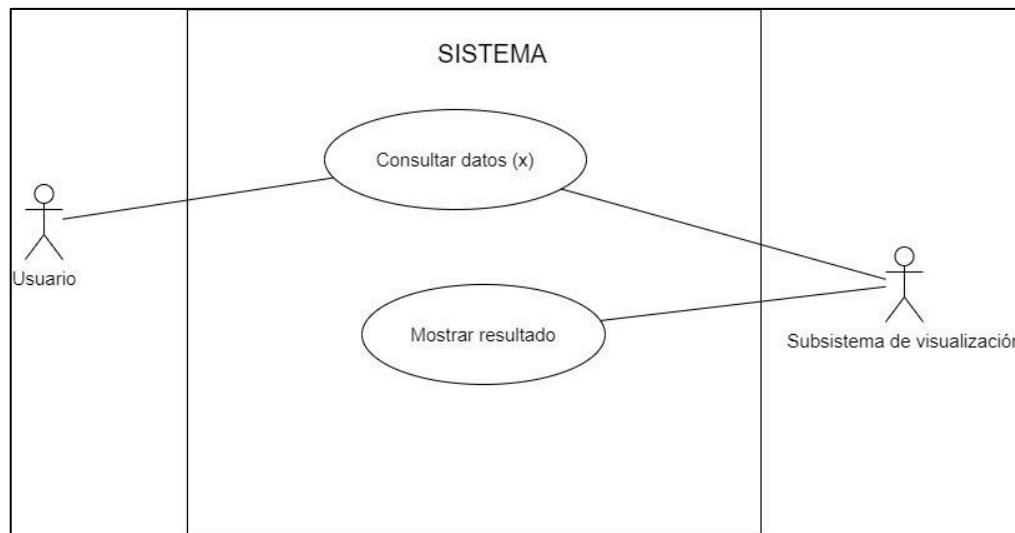


Figura 5.5 Caso de uso – visualizar medida

- Consultar parámetros. El usuario consulta los parámetros asociados a alguna medida.

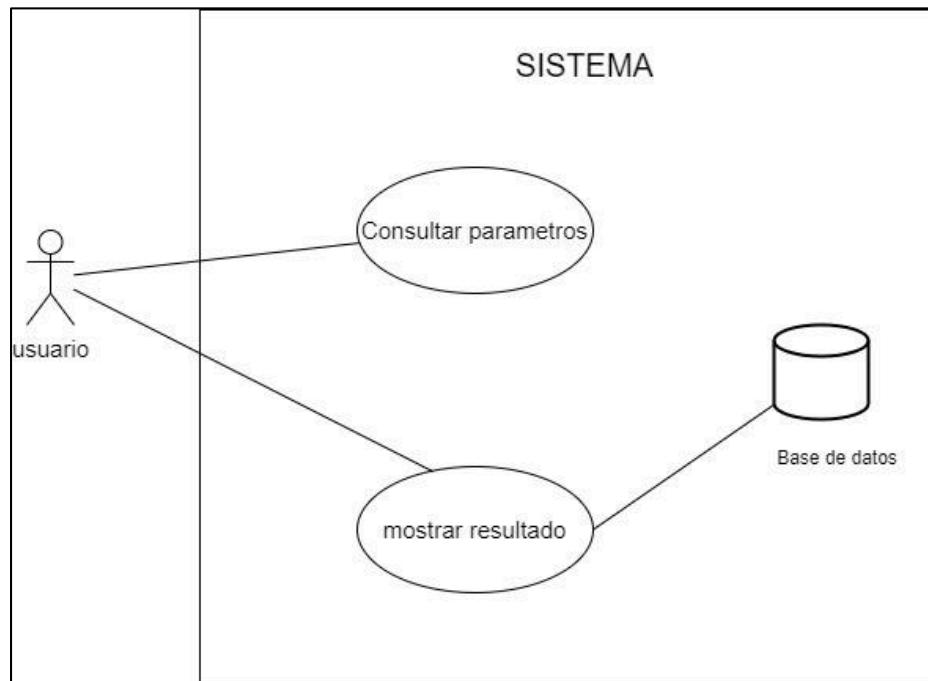


Figura 5.6 Caso de uso – consultar parámetros

- Crear dashboard. El usuario crea el cuadro de mando según sus intereses.

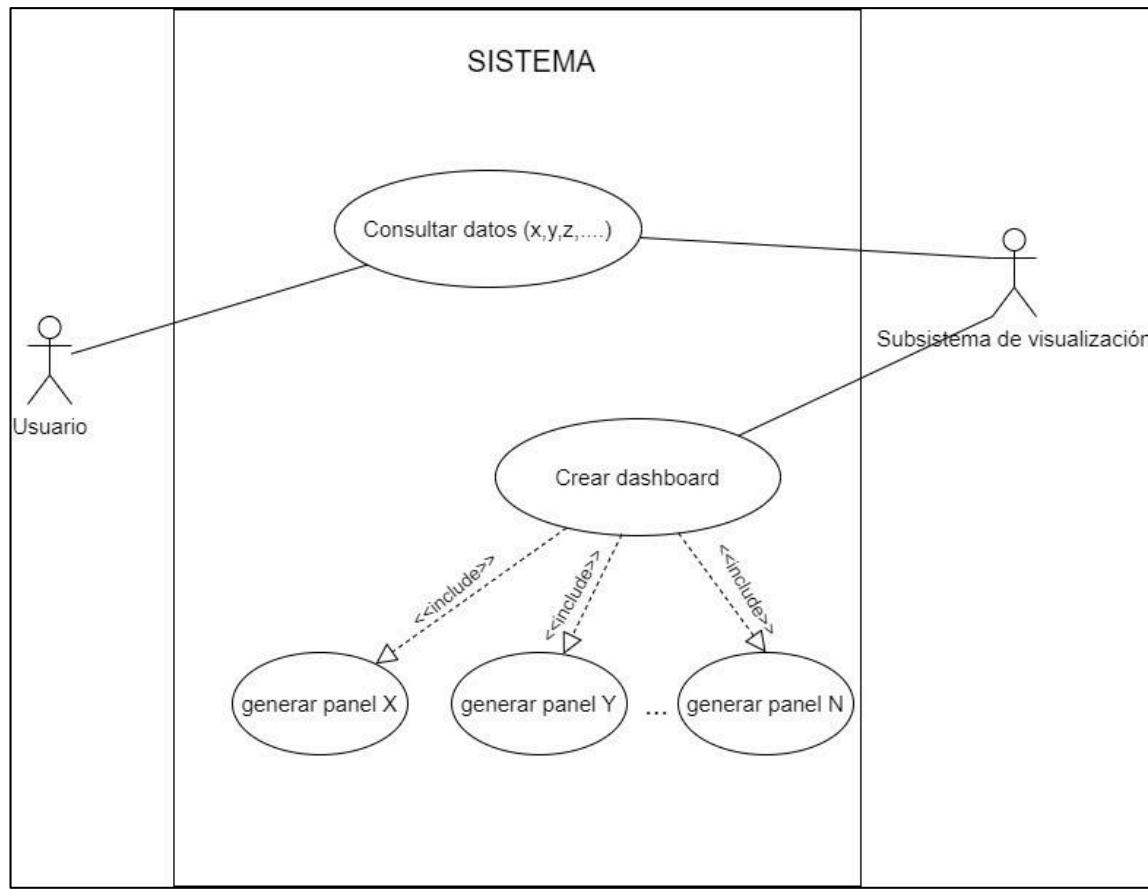


Figura 5.7 Caso de uso – crear dashboard

5.3 Modelado conceptual

En este punto se muestra el diagrama de clase de todos los conceptos usados en el sistema.

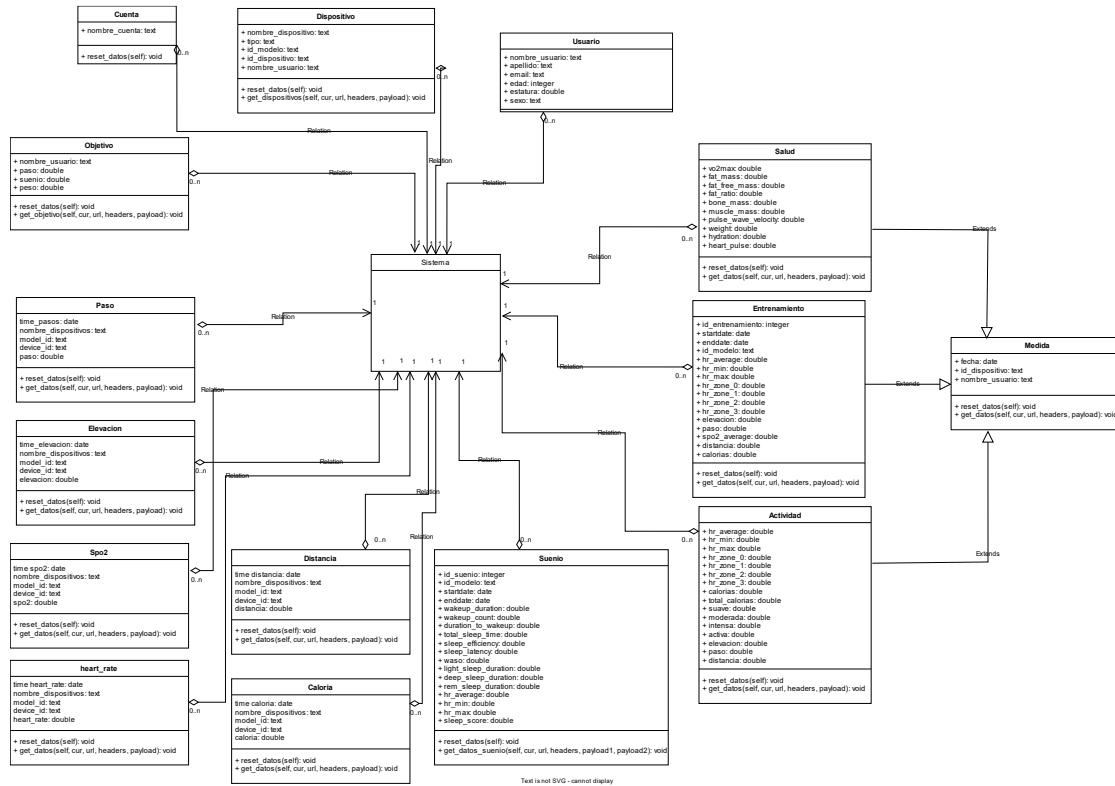


Figura 5.8 Diagrama de clase del sistema

Nombre clase	Atributos	Responsabilidad
Cuenta	nombre_cuenta	Almacena informaciones acerca de la cuenta de los usuarios
Usuario	nombre_usuario apellido email edad estatura sexo	Almacena datos relacionados con los usuarios
Objetivo	nombre_usuario paso suenio peso	Almacena informaciones sobre el objetivo de cada usuario
Dispositivo	nombre_dispositivo	Contiene informaciones

	tipo id_modelo id_dispositivo nombre_usuario	sobre los dispositivos de los usuarios
Suenio	id_suenio id_modelo startdate enddate wakeup_duration wakeup_count duration_to_wakeup total_sleep_time sleep_efficiency sleep_latency waso light_sleep_duration deep_sleep_duration rem_sleep_duration hr_average hr_min hr_max sleep_score	Almacena los datos relacionados con el sueño del usuario, tanto las fases del sueño como la duración de cada fase. Y también de los demás datos como la eficiencia del sueño, la frecuencia cardíaca, etc.
Paso	time_pasos nombre_dispositivos model_id device_id paso	Contiene los registros de los datos de paso medido con alta frecuencia durante el día.
Distancia	time_pasos nombre_dispositivos model_id device_id distancia	Contiene los registros de los datos de distancia medido con alta frecuencia durante el día.
Elevacion	time_pasos nombre_dispositivos model_id device_id elevacion	Contiene los registros de los datos de elevacion medido con alta frecuencia durante el día.
Caloria	time_pasos nombre_dispositivos model_id device_id caloria	Contiene los registros de los datos de caloria medido con alta frecuencia durante el día.
Spo2	time_pasos	Contiene los registros

	nombre_dispositivos model_id device_id spo2	de los datos de spo2 medido con alta frecuencia durante el día.
Heart_rate	time_pasos nombre_dispositivos model_id device_id heart_rate	Contiene los registros de los datos de frecuencia cardíaca medido con alta frecuencia durante el día.
Ecg	id_modelo id_dispositivo nombre_usuario nombre_dispositivo id_senial senial qrs qt pr qtc heart_rate	En esta clase se guardan los datos acerca de la medición del ecg, tanto de la frecuencia cardíaca como las señales de ecg.
Medida	fecha id_dispositivo nombre_usuario	La clase general de las clases Actividad, Entrenamiento y Salud.
Salud	vo2max fat_mass fat_free_mass fat_ratio bone_mass muscle_mass pulse_wave_velocity weight hydration heart_pulse	Hereda la clase Medidas, se encarga de almacenar datos generales de la salud como el peso, vo2max, hidración, etc.
Entrenamiento	id_entrenamiento startdate enddate id_modelo hr_average hr_min hr_max hr_zone_0 hr_zone_1	Hereda de la clase Medidas, contiene informaciones acerca de los entrenamientos realizados por el usuario.

	hr_zone_2 hr_zone_3 elevacion paso spo2_average distancia calorias	
Actividad	hr_average hr_min hr_max hr_zone_0 hr_zone_1 hr_zone_2 hr_zone_3 calorias total_calorias suave moderada intensa activa elevacion paso distancia	Hereda la clase Medidas, almacena datos relacionados con las actividades diarios realizados por el usuario, los datos se registrarán una vez al día

Tabla 5.1 Tabla descriptiva de las clases

Nombre relación	Clase involucradas	Observación
actividad_hereda_medida	Medida, Actividad	La clase Actividad es una subclase de la Medida
entrenamiento_hereda_medida	Medida, Entrenamiento	La clase Entrenamiento es una subclase de Medida
salud_hereda_medida	Medida, Salud	La clase Salud es una subclase de Medida

Tabla 5.2 Tabla descriptiva de las relaciones entre clases

5.4 Diseño del sistema MONITOR

5.4.1 Arquitectura del sistema

Para diseñar el sistema, hemos adoptado una arquitectura de monitorización continua formada por varias capas integrando la infraestructura de Withings. A través de la API del Withings tendremos acceso a los datos de salud de los usuarios.

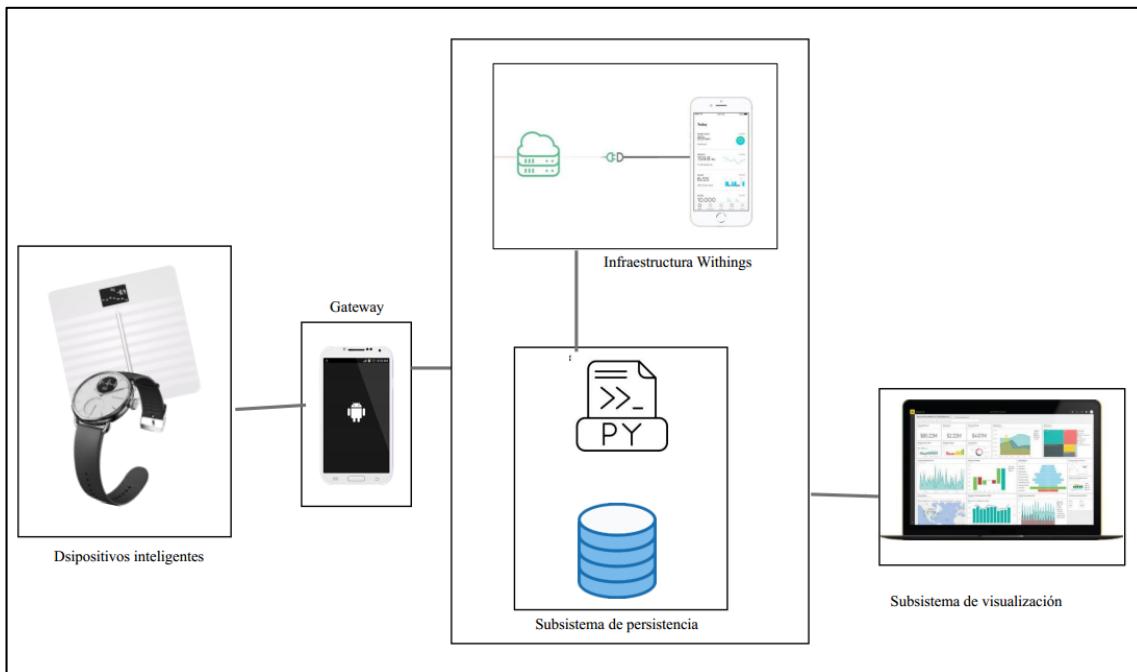


Figura 5.9 Arquitectura del sistema MONITOR.

La arquitectura se compone de cuatro capas: la capa de percepción, la capa de red, la capa de procesamiento y almacenamiento y la capa de visualización.

- En la capa de percepción se encuentra el subsistema de medición de datos, constituidos por los dispositivos inteligentes que medirán los datos continuamente. Los datos recogidos por el subsistema son transmitidos a través de un Gateway a la API del Withings, donde serán almacenados en la nube.
- La capa de red funciona como conector entre la capa de percepción y el servidor de datos. Puede ser un móvil o un router el intermediario para que los datos de los dispositivos móviles lleguen a la capa de procesamiento y almacenamiento.
- En la capa de procesamiento y almacenamiento se encuentra el subsistema de persistencia que se compone de dos partes: a) subsistema de almacenamiento de Withings y b) subsistema de procesamiento y almacenamiento de MONITOR. Esta capa se encarga de extraer los datos almacenados en la nube de Withings de forma automatizada y continua mediante un programa escrito en lenguaje Python, y almacenarlos en la base de datos de la plataforma, el timescaleDB.

- Y por último se encuentra la capa de visualización, donde se sitúa el subsistema de visualización. Para la visualización de los datos se va a usar la herramienta Grafana. Conectado a la base de datos TimescaleDB, se puede consultar las informaciones guardadas y construir los dashboards personalizados de los usuarios.

5.4.2 Diagrama entidad relación

Para almacenar los datos recogidos de la nube de Withings, es necesario crear primero la base de datos. Para ello, vamos a usar un modelo entidad-relación.

En nuestro proyecto, podemos identificar las siguientes entidades:

- Cuenta, donde se almacena las informaciones de la cuenta de los usuarios
- Usuario, almacena información generales del usuario, tales como nombre, email, etc.
- Objetivo, almacena el objetivo definido de cada usuario.
- Dispositivos, almacena las informaciones de los dispositivos de los usuarios.
- Spo2, guarda las informaciones sobre el valor de spo2 medido durante el día.
- Elevacion, almacena los datos de la elevación realizado por el usuario durante el día.
- Heart_rate, registra los valores de la frecuencia cardíaca del usuario medidos a alta frecuencia.
- Calorias, almacena la cantidad de calorías quemadas del usuario durante el día.
- Distancia, almacena la distancia recorrida por el usuario durante el día.
- Pasos, registra el número de pasos dado por el usuario durante el día.
- Actividad, almacena la información general de una determinada actividad, como la fecha de la actividad, el dispositivo de medición, etc.
- Hr_actividad, almacena las informaciones acerca de la frecuencia cardíaca del usuario en un día, como la media, el mínimo, el máximo de la frecuencia cardíaca.
- Calorias_actividad, guarda la cantidad de las calorías quemadas por el usuario en un día.
- Paso_actividad, guarda el número de los pasos dados por el usuario en un día.
- Intensidad_actividad, guarda el tiempo de la intensidad de las actividades realizadas en un día.
- Distancia_Actividad, almacena la distancia recorrida por el usuario en un día.
- Elevación_actividad, guarda la elevación realizada por el usuario en un día.
- Ecg, guarda los datos generales de una medición de ECG, como la fecha, qrs, qt, el id de las señales, etc.
- Senial_ecg, almacena los valores de las señales de una medición de ECG.
- Suenio, registra los datos generales del sueño como fecha de inicio y fecha de finalización.
- Datos_suenio, guarda los datos relacionados del sueño como tiempo total de sueño, tiempo de cada fase de sueño, la eficiencia del sueño, etc.
- Hr_suenio, almacena las frecuencias cardíacas medidas durante el sueño.

- Entrenamiento, almacena la información general de un determinado entrenamiento, como la fecha de inicio y finalización del entrenamiento, el dispositivo de medición, etc.
- Elevación_entrenamiento, almacena datos acerca de la elevación realizado durante un determinado entrenamiento.
- Hr_entrenamiento, almacena datos acerca de la frecuencia cardíaca durante un determinado entrenamiento.
- Pasos_entrenamiento, almacena el número de pasos realizados durante un determinado entrenamiento.
- Distancia_entrenamiento, almacena la distancia recorrida durante un determinado entrenamiento.
- Spo2_average_entrenamiento, almacena el valor medio de los spo2 medidos durante un determinado entrenamiento.
- Calorias_entrenamiento, guarda la cantidad de calorías quemadas durante un entrenamiento determinado.
- Salud, almacena la información general de los datos de salud general, como fecha de medición.
- Weight, guarda la información sobre el peso del usuario.
- Pulse_wave_velocity, guarda la información acerca de la velocidad de onda de pulso.
- Vo2max, guarda el valor de vo2max.
- Heart_pulse, guarda el valor de heart_pulse.
- Hydration, almacena el porcentaje de agua en el cuerpo.
- Fat_ratio, almacena datos sobre la proporción de grasa.
- Fat_mass, almacena datos sobre la masa corporal.
- Bone_mass, almacena datos sobre la masa ósea.
- Muscle_mass, almacena datos sobre la masa muscular,
- Fat_free_mass, almacena datos sobre la masa libre de grasa.

Las relaciones que se pueden establecer entre estas entidades son:

- Una cuenta puede tener uno o varios usuarios.
- Cada usuario puede establecer uno o varios objetivos.
- Un usuario puede disponer de uno o varios dispositivos
- Un dispositivo puede registrar uno o varios datos de Actividad, Entrenamiento, ECG, Salud, Sueño.
- Cada actividad está compuesta por cero o una medición de Elevación, Distancia, Calorías, Pasos, Intensidad, Hr.
- Cada entrenamiento está compuesto por cero o una medición de Elevación, Distancia, Calorías, Pasos, Hr, Spo2.
- ECG, contiene una o varias mediciones de señal de ECG y Hr.
- La salud está compuesta por una o varias mediciones de Weight, Pulse_wave_velocity, Vo2max, Heart_pulde, Hydration, Fat_ratio, Fat_mass, Bone_mass, Muscle_mass, Fat_free_mass, imc, tmb, agua y proteína.

A continuación, se muestra el diagrama de la entidad relación, para verlo mejor se puede encontrar dicho diagrama en la misma carpeta en formato de imagen.

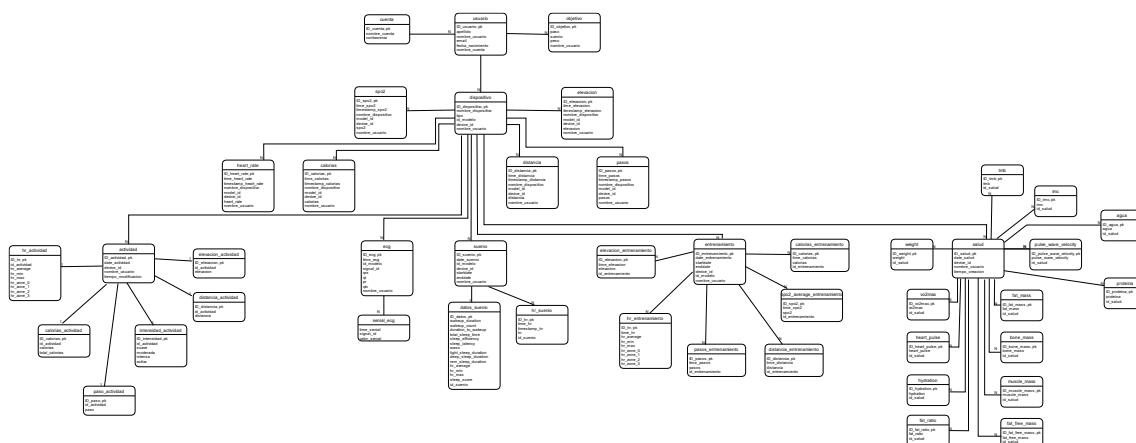


Figura 5.10 Diagrama de entidad relación

5.4.3 Prototipo de la interfaz del sistema

Aquí se encuentra un prototipo inicial sobre la interfaz del usuario sobre cómo se verían los datos en la plataforma.

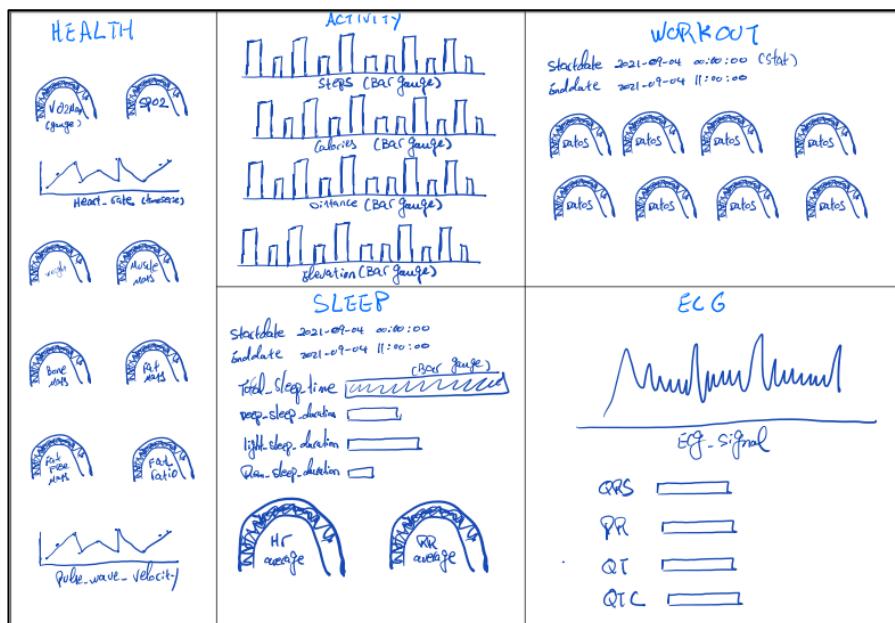


Figura 5.11 Boceto del Dashboard

Este prototipo inicial se ha diseñado para mostrar de manera más simple y clara las distintas medidas de salud al usuario.

En la parte izquierda podemos ver las medidas relacionadas con la salud general, como

el peso, masa ósea, masa muscular, etc.

En la parte central encontramos los datos acumulados de los pasos realizados, las calorías quemadas, la elevación realizada, etc. durante un día concreto. Por debajo se sitúa los datos relacionados al sueño.

Y en la parte derecha encontramos con los datos de entrenamientos y de ECG.

5.5 Implementación del sistema

5.5.1 Capa de percepción.

En esta capa se encuentra el nivel físico del sistema cuya la función principal es recoger las informaciones del usuario y de su entorno. La adquisición de los datos se realiza a través de los dispositivos inteligentes que llevan unos sensores específicos. Para nuestro proyecto, el dispositivo principal que recogerán los datos sería el reloj inteligente de Withings, el modelo ScanWatch.

Al llevarlo puesto durante el día y noche, el reloj registrará de forma automática los datos de frecuencia cardíaca, pasos, calorías, elevación, distancia, spo2, y de los entrenamientos del usuario de forma continua. En caso de datos de entrenamiento, antes de empezar, el usuario debe indicar al reloj de forma manual la intención de la realización de un entrenamiento, para un mejor registro de datos se recomienda indicar sobre los tipos de entrenamientos predefinidos por el sistema el que se va a realizar. Una vez terminado, debe notificar también al reloj de su finalización. Así, el reloj recogerá todos los datos medibles durante ese período de tiempo.

En relación a los datos del sueño, el reloj detectará si el usuario está despierto o se encuentra en alguna fase del sueño, y registra el tiempo que dura en cada caso. Además, también mide la frecuencia cardíaca y el ronquido durante el sueño.

Para complementar los datos medidos por el reloj, hemos usado la báscula inteligente de Withings, el modelo Body Cardio. Con esta báscula, el usuario podrá medir de forma puntual el peso, la masa muscular, la masa ósea, la masa corporal, etc.

En principio la implementación de programas y transmisión de datos a la infraestructura de withings es la propia desarrollada por Withings. Lo único que tenemos que hacer en este caso es configurar correctamente los dispositivos wearables para que realicen mediciones continuas de los parámetros que se vayan a medir.

5.5.2 Capa de red.

Esta capa se encarga de conectar los dispositivos inteligentes al servidor de almacenamiento de datos. En este caso, la implementación también depende de Withings.

Los datos recabados por los dispositivos de la capa de percepción se quedarán almacenados en el móvil del usuario, y a través de la aplicación Health Mate. Dichos datos son sincronizados con la API del Withings cada cierto intervalo de tiempo definido por Withings.

En caso de báscula Body Cardio, es posible sincronizar con el servidor de Withings a través del WIFI sin tener que estar conectado a un móvil. Estos datos se quedarán almacenados en la nube de Withings.

5.5.3 Capa de procesamiento y almacenamiento

La funcionalidad principal de esta capa es extraer los datos almacenados en la nube de Withings y almacenarlos en TimescaleDB.

Para conectar a la API de Withings es necesario solicitar el permiso usando el protocolo de autorización Oauth 2.0 y obtener el access_token para poder acceder a los recursos de la nube.

En la guía de integración de la API de Withings podemos encontrar un proyecto de ejemplo escrito en Python que nos pueden ayudar a implementar el flujo de OAuth 2.0.(poner cita). El proyecto lanza un pequeño servidor web Flask que contiene lo básico para poder conectarse a la API de Withings.

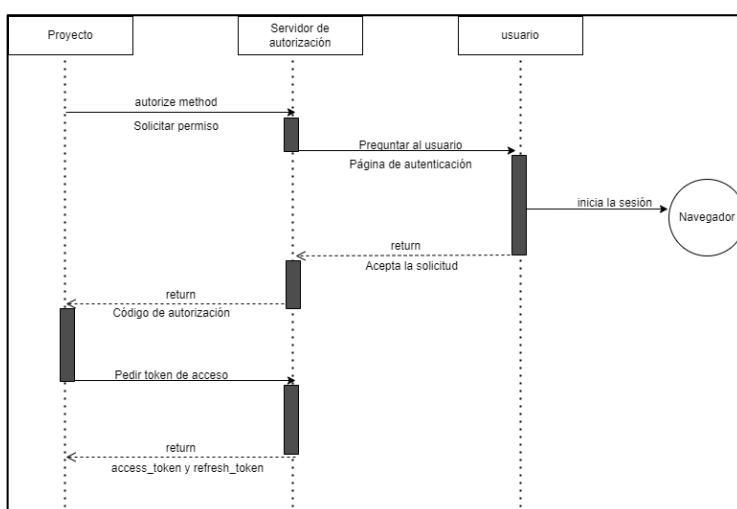


Figura 5.12 Diagrama del flujo del proyecto Withings

Cuando lanzamos el proyecto para conectarse a la API, básicamente lo que está ocurriendo es con el *authorize method* de la API, se solicita al usuario los permisos dirigiéndole a una página de autenticación de Withings. El usuario inicia la sesión con su cuenta y acepta la solicitud. Una vez aceptado la solicitud se le redirige al *redirect uri* proporcionado previamente, que contendrá un código de autorización, *authorization code*.

Con este código, usando la acción `requesttoken` se obtiene el `access_token` y el `refresh_token`. Cuando el `access_token` haya caducado, se puede usar el `refresh_token` para obtener un `access_token` y un `refresh_token` nuevo.

Los tokens obtenidos en el paso anterior, `access_token` y `refresh_token`, los tenemos que pasar como argumentos al programa Python implementado. El programa se encarga de pedir a la API de Withings usando el `access_token` los datos subidos por los dispositivos de la capa de percepción. Recuperado dichos datos, se almacenarán en la base de datos TimescaleDB. El programa ejecutará en segundo plano y pedirá de forma automática los datos cada cuatro horas, antes de realizar la petición de datos, el programa comprobará si todavía es válido el `access_token`. Si ha pasado más de 10800 segundos desde que obtenemos el `access_token`, el token deja de ser válido ya que el `access_token` caduca después de 3 horas desde su obtención. En este caso el programa pedirá un nuevo `access_token` usando el `refresh_token`, y actualiza el valor de ambos tokens una vez recibido la respuesta del servidor de autorización.

Durante el proceso de recuperación de datos, cuando se registra un nuevo valor sobre el peso del usuario, se calculará, a partir de este dato junto con los otros ya registrados, los valores del índice de masa corporal, la tasa de metabolismo basal, el porcentaje de proteínas corporal y el porcentaje de agua.

Para el cálculo del índice de masa corporal se ha usado la fórmula: $\text{peso(kg)} / [\text{estatura(m)}^2]$. El resultado obtenido lo podemos interpretar usando la siguiente lista:

- Por debajo de 18.5, el nivel de peso es bajo peso.
- De 18.5 – 24.9, el nivel de peso es normal.
- De 25.0 – 29.9, el nivel de peso es sobrepeso.
- De 30.0 o más, el nivel de peso es obesidad. [referencia]

Para calcular la tasa de metabolismo basal hemos usado la fórmula que se ha comentado en el capítulo 2:

- TMB en hombres = $(10 * \text{peso de Kg}) + (6.25 * \text{altura en cm}) - (5 * \text{edad en años}) + 5$
- TMB en mujeres = $(10 * \text{peso de Kg}) + (6.25 * \text{altura en cm}) - (5 * \text{edad en años}) - 161$

A partir de este valor calculado, se puede obtener la cantidad mínima de energía de una persona dependiendo de la frecuencia de actividad que realiza:

- Poco o ningún ejercicio = TMB * 1.2
- 1 – 3 días por semana = TMB * 1.375
- 3 – 5 días por semana = TMB * 1.55
- 6 – 7 días por semana = TMB * 1.725
- Muy fuerte = TMB * 1.9

En relación al porcentaje de proteínas y de agua en el cuerpo no hay una fórmula en concreto, pero como sabemos que el músculo está formado aproximadamente por 73% de agua y unos 27% de proteínas, podemos calcularlos fácilmente con la masa muscular.

Los métodos principales del programa son:

El funcionamiento de estos métodos es muy similar todos ellos.

En cada método se realiza una petición de datos al servidor Withings y recibe una respuesta. De dicha respuesta comprobamos si contiene datos de las medidas o no.

Si la respuesta no tiene datos, pasamos a la siguiente petición.

En caso de si hay datos, a partir del Json devuelto por el servidor, extraemos los datos que corresponden a las medidas y los almacenamos en su variable correspondiente.

Una vez obtenido los valores, procedemos a almacenar los datos en la base de datos, cada medida en su tabla.

- El método de obtención de datos de salud general.

```
214
215      # -----ontencion de datos de salud -----
216      salud.get_datos(cur, url_health, headers, payload_health)
217
```

- *Figura 5.13 llamada del método obtención de salud en el main*

```

31     def get_datos(self, cur, url, headers, payload):
32         response_health = requests.request("POST", url, headers=headers, data=payload)
33
34         health_data = response_health.json()
35         health_info = health_data['body'][‘measuregrps’]
36
37         if not health_info:
38             print("no se ha obtenido datos sobre la salud del usuario.")
39         else:
40             for series in health_info:
41                 self.reset_datos() #resetear datos
42                 if ‘date’ in series:
43                     epoch_time = series[‘date’]
44                     self.fecha = datetime.datetime.fromtimestamp(epoch_time).date()
45                 if ‘created’ in series:
46                     tiempo_creacion = series[‘created’]
47                 if ‘deviceid’ in series:
48                     self.id_dispositivo = series[‘deviceid’]
49
50                 for series_data in series[‘measures’]:
51                     if series_data[‘type’] == 54:
52                         spo2 = series_data[‘value’]
53                     elif series_data[‘type’] == 123:
54                         self.vo2max = series_data[‘value’]
55                     elif series_data[‘type’] == 1:
56                         self.weight = series_data[‘value’] / 1000
57                     elif series_data[‘type’] == 76:
58                         self.muscle_mass = series_data[‘value’] / 100
59                     elif series_data[‘type’] == 88:

```

Figura 5.14 Captura del método de obtención de datos de salud

```

# comprobamos si ya tenemos datos de la misma fecha en la base de datos
sql = "SELECT * FROM salud WHERE date_salud = '{0}'".format(self.fecha)
cur.execute(sql)
result = cur.fetchall()
if not result: # si no tenemos, guardamos directamente en la BD
    # almacenar datos en la base de dato
    # print("no tenemos datos en la BD, insertamos los datos")

    # 1. averiguamos el nombre del usuario mediante deviceid obtenido del json
    sql = "SELECT * FROM dispositivo WHERE device_id = '{0}'".format(self.id_dispositivo)
    cur.execute(sql)
    result = cur.fetchall()
    if result is None: # si no se ha encontrado el id del dispositivo en la base de datos, el nombre del dispositivo es None
        print('no se ha encontrado dispositivo')
    else:
        for id_d, nombre_disp, tipo, id_modelo, device_id, nombre_u in result:
            self.nombre_usuario = nombre_u

    # 2. guardamos datos de la salud
    sql = "INSERT INTO salud(date_salud, device_id, nombre_usuario, time_created) VALUES (%s, %s, %s, %s)"
    val = (self.fecha, self.id_dispositivo, self.nombre_usuario, tiempo_creacion)
    cur.execute(sql, val)

    # obtenemos el id del ultimo insertado
    sql = "SELECT * FROM salud ORDER BY id_salud DESC LIMIT 1"
    cur.execute(sql)
    result = cur.fetchall()
    if not result:
        print('error en la obtencion del id_salud')
    else:
        for id_s, dia, id_d, nombre_u, creacion in result:
            #print(id_s, dia, id_d, nombre_u, creacion)
            id_sa = id_s

    # 3. guardamos datos de vo2max
    if self.vo2max != 0:
        sql = "INSERT INTO vo2max(vo2max, id_salud) VALUES (%s, %s)"
        val = (self.vo2max, id_sa)
        cur.execute(sql, val)

```

Figura 5.15 Captura del método de inserción de datos de salud

- El método de obtención de datos de sueño.

```

220
221      # -----obtencion de datos de sueño -----
222      suenio.get_datos_suenio(cur, url_sleep, headers, payload_sleep_sum, payload_sleep)
223

```

Figura 5.16 Llamada del método obtención de datos de sueño en el main

```

def get_datos_suenio(self, cur, url, headers, payload1, payload2):
    print()
    print("-----peticion de Suenio resumen-----")
    print()
    response_sleep_sum = requests.request("POST", url, headers=headers, data=payload1)

    sleep_sum_data = response_sleep_sum.json()
    sleep_sum_info = sleep_sum_data['body']['series']

    if not sleep_sum_info:
        print("no se ha obtenido datos sobre el suenio del usuario.")
    else:
        for series in sleep_sum_info:
            self.reset_datos() #resetear datos
            if 'id' in series:
                self.id_suenio = series['id']
            if 'date' in series:
                self.fecha = series['date']
            if 'model_id' in series:
                self.id_modelo = series['model_id']
            if 'hash_deviceid' in series:
                self.id_dispositivo = series['hash_deviceid']
            if 'startdate' in series:
                t1 = series['startdate']
                self.startdate = datetime.datetime.fromtimestamp(t1)
            if 'enddate' in series:
                t2 = series['enddate']
                self.enddate = datetime.datetime.fromtimestamp(t2)
            if 'created' in series:
                t3 = series['created']
                datas_time = datetime.datetime.fromtimestamp(t3)
            if 'data' in series:
                datas = series['data']

                if 'wakeupduration' in datas:
                    self.wakeup_duration = datas['wakeupduration']
                if 'wakeupcount' in datas:
                    self.wakeup_count = datas['wakeupcount']
                if 'durationtowakeup' in datas:
                    self.duration_to_wakeun = datas['durationtowakeup']

```

Figura 5.17 Captura del método de obtención de datos de sueño

```

# almacenamiento de datos en BD
# comprobamos si ya tenemos datos del mismo id en la base de datos
sql = "SELECT * FROM suenio WHERE id_suenio = '{0}'".format(self.id_suenio)
cur.execute(sql)
result = cur.fetchall()
if not result: # si no tenemos, guardamos directamente en la BD
    # almacenar datos en la base de datos
    # print("no tenemos datos en la BD, insertamos los datos")

    # 1. averiguamos el nombre del usuario mediante el_id modelo obtenido del json
    sql = "SELECT * FROM dispositivo WHERE device_id = '{0}'".format(self.id_dispositivo)
    cur.execute(sql)
    result = cur.fetchall()
    if result is None: # si no se ha encontrado el id del dispositivo en la base de datos, el nombre del usuario es None
        print('no se ha encontrado dispositivo')
    else:
        for id_d, nombre_disp, tipo, id_modelo, device_id, nombre_usu in result:
            self.nombre_usuario = nombre_usu

    # 2. guardamos datos general
    sql = "INSERT INTO suenio(id_suenio, date_suenio, model_id, device_id, startdate, enddate, nombre_usuario) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    val = (self.id_suenio, self.fecha, self.id_modelo, self.id_dispositivo, self.startdate, self.enddate, self.nombre_usuario)
    cur.execute(sql, val)

    # 3.insertamos datos resumen del suenio
    sql = "INSERT INTO datos_suenio(wakeup_duration, wakeup_count, duration_to_wakeup, total_sleep_time, sleep_efficiency, sleep_latency, waso, light_sleep_duration,
val = (self.wakeup_duration, self.wakeup_count, self.duration_to_wakeup, self.total_sleep_time,
       self.sleep_efficiency, self.sleep_latency, self.waso, self.light_sleep_duration,
       self.deep_sleep_duration, self.rem_sleep_duration, self.hr_average, self.hr_min, self.hr_max,
       self.sleep_score, self.id_suenio)
    cur.execute(sql, val)

    print('-----')
    print('Datos SUEÑO almacenados')
else: # si ya tenemos datos con mismo id, no hacemos nada
    # print('ya tenemos datos con este id, no hacemos nada')
    pass

```

Figura 5.18 Captura del método de inserción de datos de sueño

- El método de obtención de datos de ECG.

```

224      # -----obtencion de datos de ecg -----
225      ecg.get_datos(cur, url_ecg, url_health, headers, payload_ecg)
226

```

Figura 5.19 Llamada del método de obtención de datos de ECG en el main

```

if not ecg_info:
    print("no se ha obtenido datos sobre ecg del usuario.")
else:
    for series in ecg_info:
        self.reset_datos() #resetear datos
        if 'timestamp' in series:
            t1 = series['timestamp']
            time_ecg = datetime.datetime.fromtimestamp(t1)
        if 'model' in series:
            self.id_modelo = series['model']
        if 'deviceid' in series:
            self.id_dispositivo = series['deviceid']
        if 'ecg' in series:
            self.id_serial = series['ecg']['signalid']
        if 'heart_rate' in series:
            self.heart_rate = series['heart_rate']

# solicitamos datos de los qrs, qt, pr, qtc con getmeas
time_start = t1
time_end = time_start + 14400
payload_ini = 'action=getmeas&meastype=135%2C136%2C137%2C138&category=1&startdate=0&enddate=0'
payload_ecg2 = changepayload_time(payload_ini, time_start, time_end)
response_ecg2 = requests.request("POST", url2, headers=headers, data=payload_ecg2)

ecg2_data = response_ecg2.json()
ecg2_info = ecg2_data['body']['measuregrps']

if not ecg2_info:
    print("no se ha obtenido datos del usuario.")
else:
    for serie in ecg2_info:
        for series_data in serie['measures']:
            if series_data['type'] == 135:
                self.qrs = series_data['value']
            elif series_data['type'] == 136:
                self.pr = series_data['value']
            elif series_data['type'] == 137:
                self.qt = series_data['value']
            elif series_data['type'] == 138:

```

Figura 5.20 Captura del método de obtención de ECG

```

# almacenamos datos en BD
# comprobamos si ya tenemos datos con la misma señal en la base de datos
sql = "SELECT * FROM ecg WHERE signal_id = '{0}'".format(self.id_señal)
cur.execute(sql)
result = cur.fetchall()
if not result: # si no tenemos, guardamos directamente en la BD
    # almacenar datos en la base de dato
    # print("no tenemos datos en la BD, insertamos los datos")
    pass

    # 1
    # averiguamos el nombre del usuario mediante modelid obtenido del json
    sql = "SELECT * FROM dispositivo WHERE id_modelo = '{0}'".format(self.id_modelo)
    cur.execute(sql)
    result = cur.fetchall()
    if result is None: # si no se ha encontrado el id del dispositivo en la base de datos, el nombre del dispositivo es None
        print('no se ha encontrado dispositivo')
    else:
        for id_d, nombre_disp, tipo, id_modelo, device_id, nombre_usu in result:
            self.nombre_usuario = nombre_usu

    # 2. guardamos datos de ecg en la tabla
    sql = "INSERT INTO ecg(time_ecg, model_id, signal_id, qrs, qt, pr, qtc, nombre_usuario) VALUES ( %s, %s, %s, %s, %s, %s, %s, %s)"
    val = (time_ecg, self.id_modelo, self.id_señal, self.qrs, self.qt, self.pr, self.qtc, self.nombre_usuario)
    cur.execute(sql, val)

```

Figura 5.21 Captura del método de inserción de ECG

```

# guardamos datos de señal ecg
payload_signal_ini = 'action=get&signalid=0'

# modificamos el id_signal del payload
prop_signal = {}
new_payload_signal = payload_signal_ini.split('&')

for item in new_payload_signal:
    key, value = item.split('=')
    if key == 'signalid':
        value = self.id_señal
    prop_signal[key] = value

payload_signal = '&'.join([key + '=' + str(value) for key, value in prop_signal.items()])
response_signal = requests.request("POST", url, headers=headers, data=payload_signal)

signal_data = response_signal.json()
signal_info = signal_data['body']

if not signal_info:
    print("no se ha obtenido datos sobre señal ecg del usuario.")
else:
    señales = signal_info['signal']
    time1 = t1
    time_signal = datetime.datetime.fromtimestamp(time1)
    time_increment = 1 / 300
    for signal_data in señales:
        # guardamos datos en la bd
        sql = "INSERT INTO señal_ecg(time_señal, signal_id, valor_señal) VALUES ( %s, %s, %s)"
        val = (time_signal, self.id_señal, signal_data)
        cur.execute(sql, val)
        time1 = time1 + time_increment
        time_signal = datetime.datetime.fromtimestamp(time1)

```

Figura 5.22 Captura del método de obtención e inserción de señales de ECG

- El método de obtención de datos de entrenamiento.

```

218     # -----obtencion de datos de entrenamiento -----
219     entrenamiento.get_datos(cur, url_workouts, headers, payload_workouts)
220

```

Figura 5.23 Llamada del método de obtención de datos de entrenamiento en el main

```

def get_datos(self, cur, url, headers, payload):
    print()
    print("-----peticion de Entrenamiento-----")
    print()
    response_workouts = requests.request("POST", url, headers=headers, data=payload)

    workouts_data = response_workouts.json()
    workouts_info = workouts_data['body']['series']

    if not workouts_info:
        print("no se ha obtenido datos sobre actividad del usuario.")
    else:
        for series in workouts_info:
            self.reset_datos() #resetear datos
            if 'id' in series:
                self.id_entrenamiento = series['id']
            if 'date' in series:
                self.fecha = series['date']
            if 'startdate' in series:
                t1 = series['startdate']
                self.startdate = datetime.datetime.fromtimestamp(t1)
            if 'enddate' in series:
                t2 = series['enddate']
                self.enddate = datetime.datetime.fromtimestamp(t2)
            if 'deviceid' in series:
                self.id_dispositivo = series['deviceid']
            if 'modified' in series:
                t3 = series['modified']
                datas_time = datetime.datetime.fromtimestamp(t3)
            if 'model' in series:
                self.id_modelo = series['model']
            if 'data' in series:
                datas = series['data']

                if 'elevation' in datas:
                    self.elevacion = datas['elevation']
                if 'calories' in datas:
                    self.calorías = datas['calories']
                if 'hr_average' in datas:
                    self.hr_average = datas['hr_average']
                if 'hr_min' in datas:

```

Figura 5.24 Captura del método de obtención de datos de entrenamiento

```

# comprobamos si ya tenemos datos del mismo id en la base de datos
sql = "SELECT * FROM entrenamiento WHERE id_entrenamiento = '{0}'.format(self.id_entrenamiento)
cur.execute(sql)
result = cur.fetchall()
if not result: # si no tenemos, guardamos directamente en la BD
    # almacenar datos en la base de dato
    # print("no tenemos datos en la BD, insertamos los datos")

    # 1.verificamos el nombre del usuario mediante el id modelo obtenido del json
    sql = "SELECT * FROM dispositivo WHERE id_modelo = '{0}'.format(self.id_modelo)
    cur.execute(sql)
    result = cur.fetchall()
    if result is None: # si no se ha encontrado el id del dispositivo en la base de datos, el nombre del usuario es None
        print('no se ha encontrado dispositivo')
    else:
        for id_d, nombre_disp, tipo, id_modelo, device_id, nombre_usu in result:
            self.nombre_usuario = nombre_usu

    # 2.guardamos datos general
    sql = "INSERT INTO entrenamiento(id_entrenamiento, date_entrenamiento, startdate, enddate, device_id, modelo_disp, nombre_usuario) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    val = (self.id_entrenamiento, self.fecha, self.startdate, self.enddate, self.id_dispositivo, self.id_modelo, self.nombre_usuario)
    cur.execute(sql, val)

    # 3.insertamos datos de calorías
    sql = "INSERT INTO calorias_entrenamiento(time_calorias, calorias, id_entrenamiento) VALUES (%s, %s, %s)"
    val = (datas_time, self.calorias, self.id_entrenamiento)
    cur.execute(sql, val)

    # 4.insertamos datos de distancias
    sql = "INSERT INTO distancia_entrenamiento(time_distancia, distancia, id_entrenamiento) VALUES (%s, %s, %s)"
    val = (datas_time, self.distancia, self.id_entrenamiento)
    cur.execute(sql, val)

    # 5.insertamos datos de elevación
    sql = "INSERT INTO elevacion_entrenamiento(time_elevacion, elevacion, id_entrenamiento) VALUES (%s, %s, %s)"
    val = (datas_time, self.elevacion, self.id_entrenamiento)
    cur.execute(sql, val)

```

Figura 5.25 Captura del método de inserción de datos de entrenamiento

- El método de obtención de datos de actividad.

```

211
212     # -----obtencion de datos de actividad -----
213     actividad.get_datos(cur, url_activity, headers, payload_activity)
214

```

Figura 5.26 Llamada del método de obtención de datos de actividad en el main

```

43     def get_datos(self, cur, url, headers, payload):
44         print("-----peticion de actividad-----")
45         response_activity = requests.request("POST", url, headers=headers, data=payload)
46
47         activity_data = response_activity.json()
48         activity_info = activity_data['body']['activities']
49
50         # print (activity_info)
51
52         if not activity_info:
53             print("no se ha obtenido datos sobre actividad del usuario.")
54         else:
55             for series in activity_info:
56                 self.reset_datos() #resetear datos
57                 if 'elevation' in series:
58                     self.elevacion = series['elevation']
59                     if 'soft' in series:
60                         self.suave = series['soft']
61                     if 'moderate' in series:
62                         self.moderada = series['moderate']
63                     if 'intense' in series:
64                         self.intensa = series['intense']
65                     if 'active' in series:
66                         self.activa = series['active']
67                     if 'calories' in series:
68                         self.calorias = series['calories']
69                     if 'steps' in series:
70                         self.paso = series['steps']

```

Figura 5.27 Captura del método de obtención de datos de actividad

```

# comprobamos si tenemos datos para almacenar
if self.elevacion == 0 and self.suave == 0 and self.moderada == 0 and self.intensa == 0 and self.activa == 0 and self.calorias == 0 and self.paso == 0 and self.fecha == None:
    print('no tenemos datos de actividad para almacenar')
else:
    # comprobamos si ya tenemos datos de la misma fecha en la base de datos
    sql = "SELECT * FROM actividad WHERE date_actividad = '{0}'".format(self.fecha)
    cur.execute(sql)
    result = cur.fetchall()

    if not result: # si no tenemos, guardamos directamente en la BD
        # almacenar datos en la base de dato

        # 1.averiguamos el nombre del usuario mediante deviceid obtenido del json
        sql = "SELECT * FROM dispositivo WHERE device_id = '{0}'".format(self.id_dispositivo)
        cur.execute(sql)
        result = cur.fetchall()
        if result is None: # si no se ha encontrado el id del dispositivo en la base de datos, el nombre del usuario es None
            print('no se ha encontrado dispositivo')
        else:
            for id_d, nombre_disp, tipo, id_modelo, device_id, nombre_usu in result:
                self.nombre_usuario = nombre_usu

        # 2.guardamos datos general
        sql = "INSERT INTO actividad(date_actividad, device_id, nombre_usuario, modified) VALUES ( %s, %s, %s, %s)"
        val = (self.fecha, self.id_dispositivo, self.nombre_usuario, modified)
        cur.execute(sql, val)

        # obtenemos el id del ultimo insertado
        sql = "SELECT * FROM actividad ORDER BY id_actividad DESC LIMIT 1"
        cur.execute(sql)
        result = cur.fetchall()
        if not result:
            print('error en la obtencion del id_actividad')
        else:
            for id_a, dia, id_d, nombre_u, modificacion in result:
                #print(id_a, dia, id_d, nombre_u, modificacion)
                id_act = id_a

        # 3.insertamos datos de elevacion
        if self.elevacion != 0:
            sql = "INSERT INTO elevacion_actividad(id_actividad, elevacion) VALUES ( %s, %s)"

    # 4.insertamos datos de distancia
    if self.distancia != 0:
        sql = "INSERT INTO distancia_actividad(id_actividad, distancia) VALUES ( %s, %s)"

    # 5.insertamos datos de pasos
    if self.paso != 0:
        sql = "INSERT INTO paso_actividad(id_actividad, pasos) VALUES ( %s, %s)"

    # 6.insertamos datos de calorias
    if self.calorias != 0 or self.total_calorias != 0:
        sql = "INSERT INTO calorias_actividad(id_actividad, calorias, total_calorias) VALUES ( %s, %s, %s)"

    # 7.insertamos datos de intensidad
    if self.suave != 0 or self.moderada != 0 or self.intensa != 0 or self.activa != 0:
        sql = "INSERT INTO intensidad_actividad(id_actividad, suave, moderada, intensa, activa) VALUES ( %s, %s, %s, %s, %s)"

    cur.execute(sql, val)

```

Figura 5.28 Captura del método de inserción de datos de actividad 1

```

# 3.insertamos datos de elevacion
if self.elevacion != 0:
    sql = "INSERT INTO elevacion_actividad(id_actividad, elevacion) VALUES ( %s, %s)"
    val = (id_act, self.elevacion)
    cur.execute(sql, val)

# 4.insertamos datos de distancia
if self.distancia != 0:
    sql = "INSERT INTO distancia_actividad(id_actividad, distancia) VALUES ( %s, %s)"
    val = (id_act, self.distancia)
    cur.execute(sql, val)

# 5.insertamos datos de pasos
if self.paso != 0:
    sql = "INSERT INTO paso_actividad(id_actividad, pasos) VALUES ( %s, %s)"
    val = (id_act, self.paso)
    cur.execute(sql, val)

# 6.insertamos datos de calorias
if self.calorias != 0 or self.total_calorias != 0:
    sql = "INSERT INTO calorias_actividad(id_actividad, calorias, total_calorias) VALUES ( %s, %s, %s)"
    val = (id_act, self.calorias, self.total_calorias)
    cur.execute(sql, val)

# 7.insertamos datos de intensidad
if self.suave != 0 or self.moderada != 0 or self.intensa != 0 or self.activa != 0:
    sql = "INSERT INTO intensidad_actividad(id_actividad, suave, moderada, intensa, activa) VALUES ( %s, %s, %s, %s, %s)"
    val = (id_act, self.suave, self.moderada, self.intensa, self.activa)
    cur.execute(sql, val)

```

Figura 5.29 Captura del método de inserción de datos de actividad 2

- El método de obtención de datos medidos con alta frecuencia.

```

226
227         # -----obtencion de datos de alta frecuencia -----
228         medidasAlta.get_datos(cur, url_high, headers, payload_high)
229

```

Figura 5.30 Llamada del método de obtención de datos de alta frecuencia en el main

```

30     def get_datos(self, cur, url, headers, payload):
31         response_high = requests.request("POST", url, headers=headers, data=payload)
32
33         high_data = response_high.json()
34         high_info = high_data['body']['series']
35
36         if not high_info:
37             print("no se ha obtenido datos de alta frecuencia del usuario.")
38         else:
39             for timestamps in high_info:
40                 self.reset_datos() #resetear datos
41                 tiempo = timestamps
42                 time_data = datetime.datetime.fromtimestamp(int(tiempo))
43                 datas = high_info[timestamps]
44
45                 if 'model' in datas:
46                     self.nombre_dispositivo = datas['model']
47                 if 'model_id' in datas:
48                     self.id_modelo = datas['model_id']
49                 if 'deviceid' in datas:
50                     self.id_dispositivo = datas['deviceid']
51                 if 'distance' in datas:
52                     self.distancia = datas['distance']
53                 if 'spo2_auto' in datas:
54                     self.spo2 = datas['spo2_auto']
55                 if 'heart_rate' in datas:
56                     self.heart_rate = datas['heart_rate']
57                 if 'elevation' in datas:
58                     self.elevacion = datas['elevation']

```

Figura 5.31 Captura del método de obtención de datos de alta frecuencia

```

# averiguamos el nombre del usuario mediante deviceid obtenido del json
sql = "SELECT * FROM dispositivo WHERE device_id = '{0}'".format(self.id_dispositivo)
cur.execute(sql)
result = cur.fetchall()
if result is None: # si no se ha encontrado el id del dispositivo en la base de datos, el nombre del dispositivo es None
    print("no se ha encontrado dispositivo")
else:
    for id_d, nombre_disp, tipo, id_modelo, device_id, nombre_usu in result:
        self.nombre_usuario = nombre_usu

# distancia
# comprobamos si tenemos datos
if self.distancia != 0:
    # comprobamos si ya tenemos datos de la misma fecha en la base de datos
    sql = "SELECT * FROM distancia WHERE timestamp_distancia = '{0}'".format(tiempo)
    cur.execute(sql)
    result = cur.fetchall()
    if not result: # si no tenemos, guardamos directamente en la BD
        # almacenar datos en la base de dato
        # print("no tenemos datos en la BD, insertamos los datos")
        # guardamos datos en la tabla
        sql = "INSERT INTO distancia(time_distancia, timestamp_distancia, nombre_dispositivo, model_id, device_id, distancia, nombre_usuario) VALUES ( %s, %s, %s, %s, %s, %s, %s)"
        val = (time_data, tiempo, self.nombre_dispositivo, self.id_modelo, self.id_dispositivo, self.distancia, self.nombre_usuario)
        cur.execute(sql, val)
    else: # si ya tenemos datos con mismo id, no hacemos nada
        # print("ya tenemos datos con este tiempo, no hacemos nada")
        pass
else:
    print('no hay datos distancia')

# heart_rate
# comprobamos si tenemos datos
if self.heart_rate != 0:
    # comprobamos si ya tenemos datos de la misma fecha en la base de datos
    sql = "SELECT * FROM heart_rate WHERE timestamp_heart_rate = '{0}'".format(tiempo)
    cur.execute(sql)
    result = cur.fetchall()
    if not result: # si no tenemos, guardamos directamente en la BD
        # almacenar datos en la base de dato
        # print("no tenemos datos en la BD, insertamos los datos")
        # guardamos datos en la tabla

```

Figura 5.32 Captura del método de inserción de datos de alta frecuencia

- Método para calcular el índice de masa corporal

```

# IMC = peso (kg)/ [estatura (m)]2
def calcularIMC(self, peso, estatura):
    altura = float(estatura) / 100.0
    imc = float(peso) / altura**2

    return imc


def obtenerIMC(self, cur, peso, nombre_u):
    imc = 0
    altura = 0

    sql = "SELECT * FROM usuario where nombre = '{0}'".format(nombre_u)
    cur.execute(sql)
    result = cur.fetchall()
    if not result:
        print('no existe el usuario', nombre_u)
    else:
        for id_u, ape, nomb, email, edad, alt, sexo, nombre_c in result:
            # print(id_u, ape, nomb, email, edad, alt, sexo, nombre_c)
            altura = alt

        if altura != 0 and peso != 0:
            imc = self.calcularIMC(peso, altura)

    return imc

```

Figura 5.33 Método para calcular IMC

- Método para calcular la tasa de metabolismo basal

```

def calcularTMB(self, weight, height, sex, age):
    tmb_general = (10 * float(weight)) + (6.25 * float(height)) - (5 * float(age))

    if sex == 'H':
        tmb = tmb_general + 5
    else:
        tmb = tmb_general - 161

    return tmb


def obtenerTMB(self, cur, peso, nombre_u):
    tmb = 0

    sql = "SELECT * FROM usuario where nombre = '{0}'".format(nombre_u)
    cur.execute(sql)
    result = cur.fetchall()
    if not result:
        print('no existe el usuario', nombre_u)
    else:
        for id_u, ape, nomb, email, edad, alt, sexo, nombre_c in result:
            # print(id_u, ape, nomb, email, edad, alt, sexo, nombre_c)
            altura = alt
            sex = sexo
            age = edad

        if altura != 0 and peso != 0 and age != 0 and sex != 'None':
            tmb = self.calcularTMB(peso, altura, sex, age)

    return tmb

```

Figura 5.34 Método para calcular TMB

- Método para calcular porcentaje de agua en el cuerpo

```

def obtenerAgua(self, cur, muscle_mass, nombre_u):
    pagua = 0
    # obtener el id del ultimo registro de la salud que pertenece al usuario
    sql = "SELECT * FROM salud where nombre_usuario = '{0}' ORDER BY id_salud DESC LIMIT 1".format(nombre_u)
    cur.execute(sql)
    result = cur.fetchall()
    if not result:
        print('no hay datos de este usuario')
    else:
        for id_s, fecha, id_d, nombre_u, creacion in result:
            #print (id_s, fecha, id_d, nombre_u, creacion)
            id_salud = id_s

        sql = "SELECT * FROM weight where id_salud = '{0}'".format(id_salud)
        cur.execute(sql)
        result = cur.fetchall()
        if not result:
            print('no hay datos de peso')
        else:
            for id_w, p, tiempo in result:
                peso = p

        porcentaje_muscular = muscle_mass / peso * 100
        pagua = porcentaje_muscular * 0.73

    return pagua

```

Figura 5.35 Método para calcular porcentaje de agua

- Método para calcular el porcentaje de proteína en el cuerpo

```

def obtenerProteina(self, cur, muscle_mass, nombre_u):
    pproteina = 0
    # obtener el id del ultimo registro de la salud que pertenece al usuario
    sql = "SELECT * FROM salud where nombre_usuario = '{0}' ORDER BY id_salud DESC LIMIT 1".format(nombre_u)
    cur.execute(sql)
    result = cur.fetchall()
    if not result:
        print('no hay datos de este usuario')
    else:
        for id_s, fecha, id_d, nombre_u, creacion in result:
            #print (id_s, fecha, id_d, nombre_u, creacion)
            id_salud = id_s

        sql = "SELECT * FROM weight where id_salud = '{0}'".format(id_salud)
        cur.execute(sql)
        result = cur.fetchall()
        if not result:
            print('no hay datos de peso')
        else:
            for id_w, p, tiempo in result:
                peso = p

        porcentaje_muscular = muscle_mass / peso * 100
        pproteina = porcentaje_muscular * 0.27

    return pproteina

```

Figura 5.36 Método para calcular porcentaje de proteína

5.5.3 Capa de visualización

Esta capa se ocupa de la presentación de los datos registrados en el sistema al usuario. Para conseguir esto, vamos a usar la herramienta de visualización Grafana ya mencionada en los capítulos anteriores.

Una vez instalada y configurada la Grafana en nuestro servidor, tenemos que conectarla al TimescaleDB y crear el dashboard con las informaciones personalizadas del usuario.

5.5.3.1 conexión al TimescaleDB

Los pasos que hay que seguir para conectar una base de datos son:

- Abrir el navegador e ir a la página de Grafana con la dirección <http://GRAFANAIP:puerto> o <http://localhost:puerto> y logueamos con nuestro nombre de usuario y contraseña.
- En el menú de barra situado en la parte izquierda clicamos en la opción *configuración*, y dentro seleccionamos *Add data source*. Y de todas las opciones que hay, elegimos PostgreSQL.

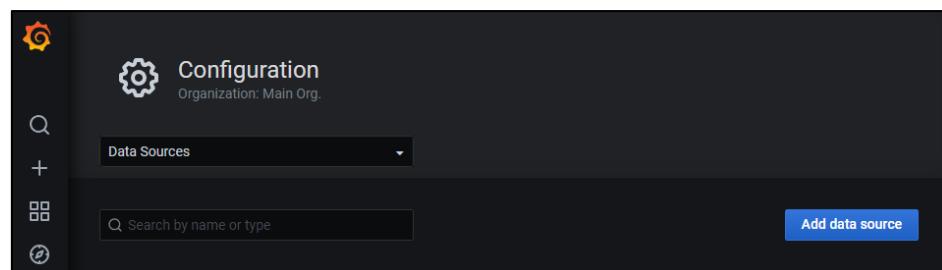


Figura 5.37 Captura del menú configuración de Grafana

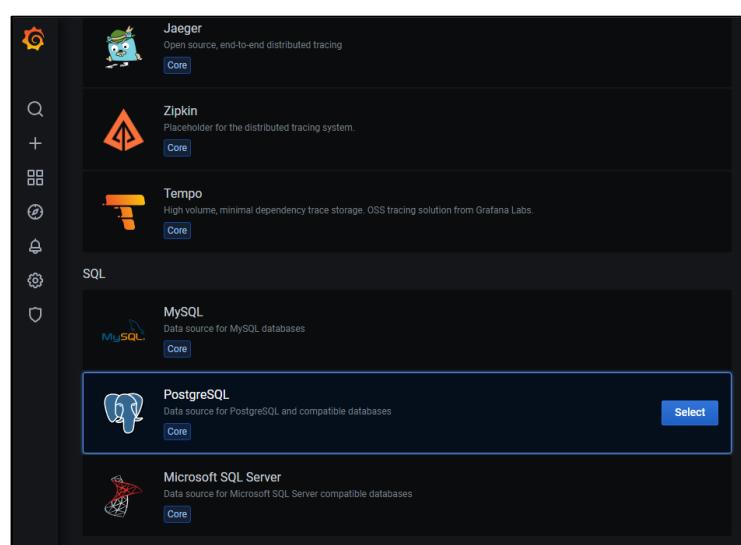


Figura 5.38 Opciones de fuente de datos

- Rellenar datos relacionados con el servidor para la conexión, tales como Host, nombre de base de datos, el usuario y su contraseña, etc. Rellenado los datos, clicamos en el botón *Save & Test* para comprobar la conexión con la base de datos, si no sale ningún error, hasta aquí ya tenemos conectada la base de datos y ya podemos consultar los datos.

The screenshot shows the 'Data Sources / PostgreSQL-withingsData' configuration page in Grafana. The 'Type' is set to 'PostgreSQL'. The 'Name' is 'PostgreSQL-withingsData' and it is marked as 'Default'. The 'Settings' tab is selected.

PostgreSQL Connection

Host	localhost:5432
Database	withings
User	within... <input type="button" value="Password"/> <input type="button" value="configured"/> <input type="button" value="Reset"/>
SSL Mode	require <input type="button" value=""/>
SSL Root Certificate	SSL/TLS root cert file <input type="button" value=""/>
SSL Client Certificate	SSL/TLS client cert file <input type="button" value=""/>
SSL Client Key	SSL/TLS client key file <input type="button" value=""/>

Connection limits

Max open	unlimited <input type="button" value=""/>
Max idle	2 <input type="button" value=""/>
Max lifetime	14400 <input type="button" value=""/>

PostgreSQL details

Version	9.3 <input type="button" value=""/> <input type="button" value="Help >"/>
TimescaleDB	<input checked="" type="checkbox"/> <input type="button" value="Help >"/>
Min time interval	1m <input type="button" value=""/>

User Permission

The database user should only be granted SELECT permissions on the specified database & tables you want to query. Grafana does not validate that queries are safe so queries can contain any SQL statement. For example, statements like `DELETE FROM user;` and `DROP TABLE user;` would be executed. To protect against this we Highly recommend you create a specific PostgreSQL user with restricted permissions.

Action Buttons

Figura 5.39 Configuración de base de datos en Grafana

5.5.3.2 Creación de Dashboard

Para la creación de un cuadro de mando elegimos en el menú de barra de la izquierda la opción *Dashboard* del *Create* (símbolo +). Nos mostrará un cuadro de mando vacío con opción de añadir un nuevo panel.

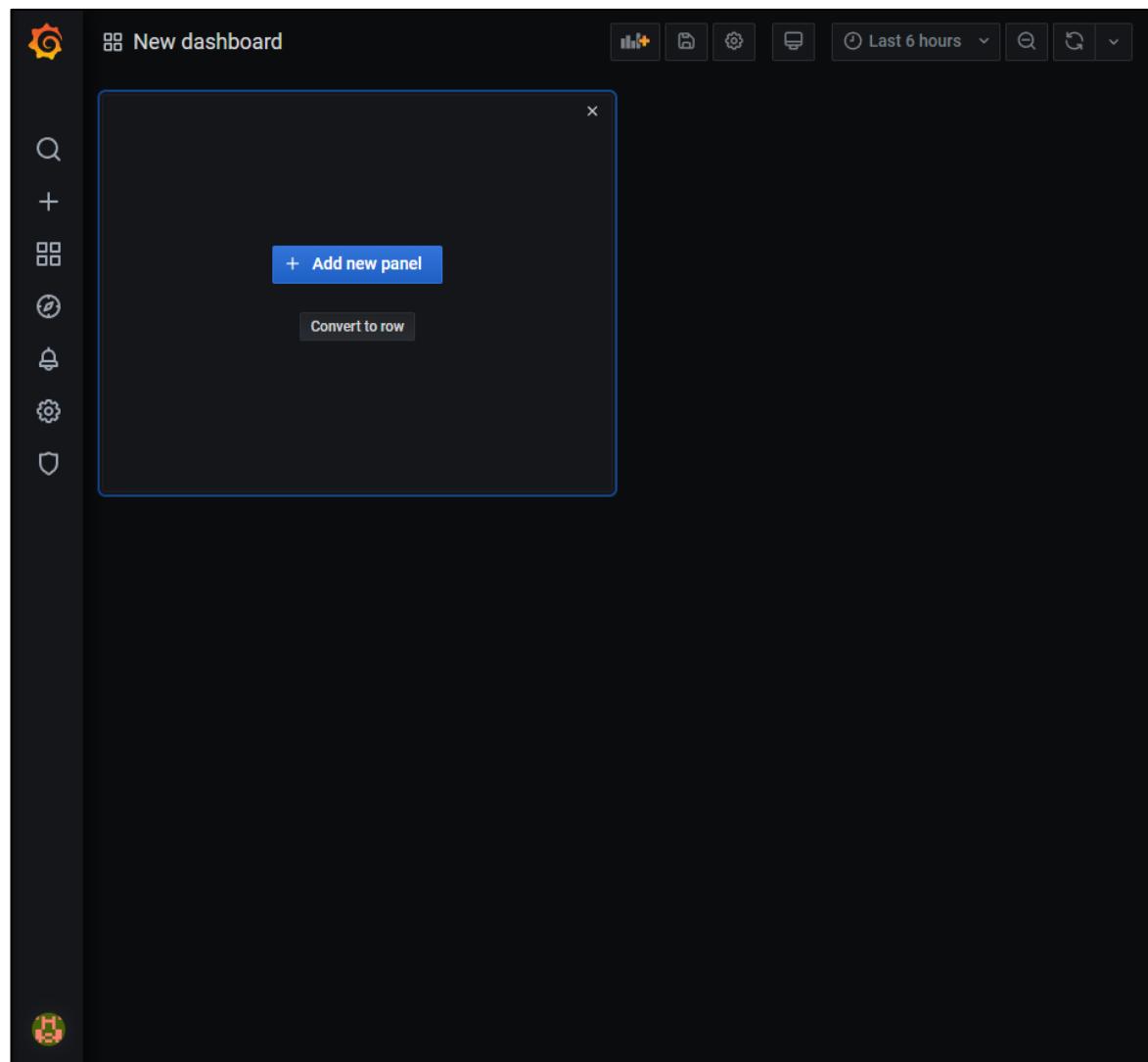


Figura 5.40 Dashboard vacío

Al pulsar añadir nuevo panel nos llevará a la ventana de configuración del panel. Debemos seleccionar la base de datos de la cual consultamos los datos, y la sentencia query de consulta de determinados datos.

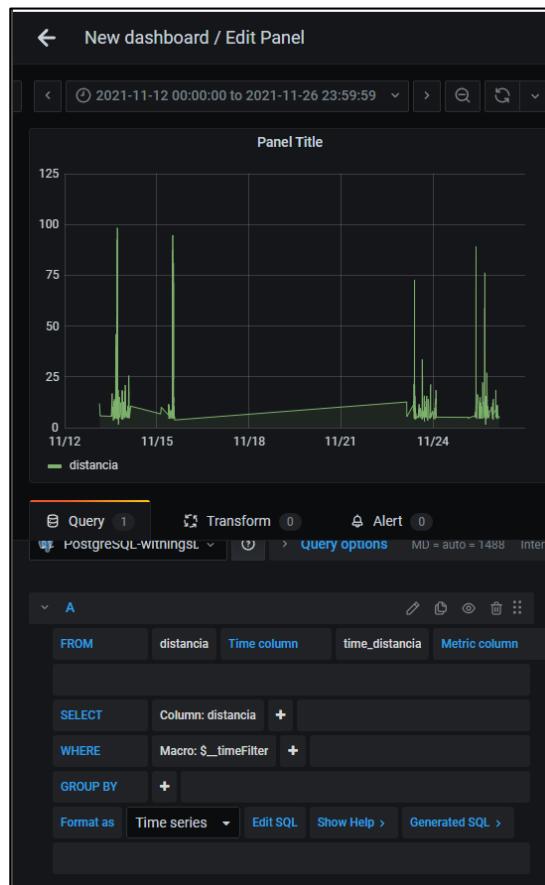


Figura 5.41 Configuración del panel

Nos da la posibilidad también de mostrar más de una consulta en el panel, basta con pulsar el botón + *Query*. En la parte izquierda, nos da varias opciones para personalizar el panel, como el tipo de visualización, el display, la leyenda, los colores etc. Una vez configurado el panel, clicamos en el botón *Apply*, el panel se mostrará en el cuadro de mando que hemos creado ante. Para el resto de los paneles el proceso es similar, podemos crear tantos paneles como queramos para el cuadro de mando.



Figura 5.42 Tipos de visualización

Creado todos los paneles, podemos personalizar el cuadro de mando moviendo los paneles de un lugar a otro, también podemos cambiar el tamaño de cada panel.



Figura 5.43 captura del dashboard del sistema

5.4 Dashboard de Grafana

En esta sección se va a describir un poco el cuadro de mando construido para el proyecto.

En la parte arriba encontramos la última medición de los datos generales de la salud del usuario, concretamente son el bone_mass, fat_mass, fat_free_mass, muscle_mass, hydration, weight y fat ratio.

Junto a estos datos se encuentran también el IMC, TMB, porcentaje de proteínas y agua.



Figura 5.44 Sección de datos generales

A continuación, se sitúan los datos del último entrenamiento, en el cual podemos ver la fecha de su realización, las calorías quemadas, la distancia recorrida, la elevación y los pasos realizados, la media de la frecuencia cardíaca y el spo2.

A su izquierda se encuentran la acumulación de los datos referentes a la elevación, los pasos, la distancia, las calorías quemadas y la media de frecuencia cardíaca medidos durante la actividad diaria del usuario en un día concreto.

Por debajo de estos datos, están la medición detallada, medida a alta frecuencia de cada uno de estos parámetros.



Figura 5.45 Datos de actividad y entrenamiento

En la parte más abajo tenemos los datos del sueño, en la que nos muestra datos referentes al tiempo inicio y fin del sueño, la duración total del sueño, la duración de cada fase de sueño, la latencia, eficiencia y la puntuación del sueño.

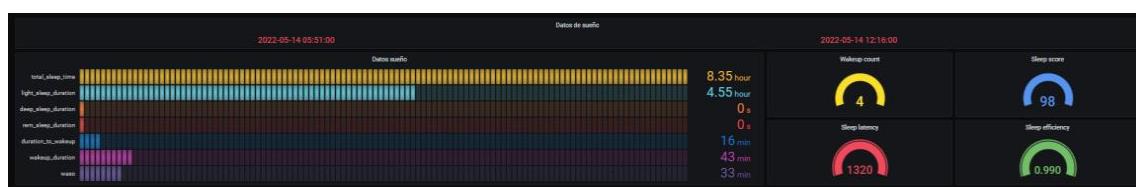


Figura 5.46 Datos de sueño

Y al final del cuadro tenemos los datos de ECG y de velocidad de onda de pulso.



Figura 5.47 Datos de ECG

Los datos mostrados en este cuadro de mando son datos generales, si queremos ver algunos datos con más detalles, podemos ir al panel correspondiente y seleccionar la opción explore. Dentro podemos explorar con detalle los datos y también podemos hacer distintas consultas a la base de datos.

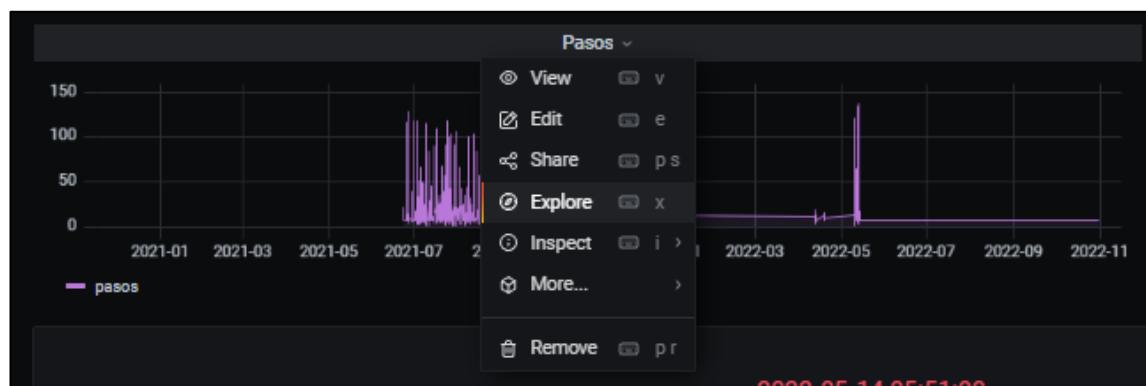


Figura 5.48 Opción explore

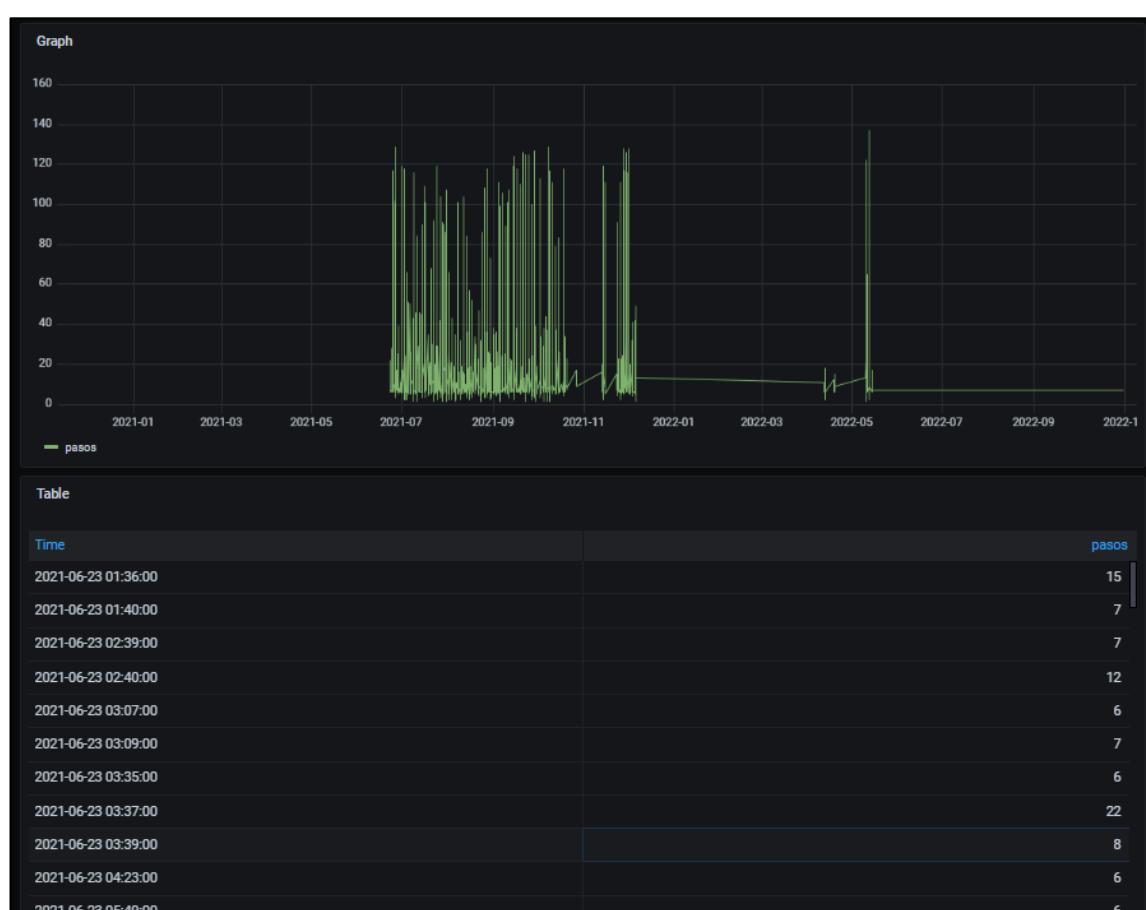


Figura 5.49 Detalle de los datos

```

SELECT
    time_pasos AS "time",
    pasos
FROM pasos
WHERE
    $_timeFilter(time_pasos)
ORDER BY 1

```

Figura 5.50 Consulta a la base de datos

5.5 Pruebas de la plataforma

5.5.1 Pruebas unitarias

Pruebas	Resultado
Conectar a la API de Withings.	✓
Renovar el access_token.	✓
Conectar a base de datos.	✓
Obtener datos sobre objetivos del usuario.	✓
Obtener datos sobre los dispositivos del usuario.	✓
Obtener datos sobre la actividad del usuario.	✓
Obtener datos sobre el sueño del usuario.	✓
Obtener datos sobre la salud general del usuario.	✓
Cálculo de datos de salud adicionales.	✓

Tabla 5.3 tabla de las pruebas unitarias

```
yangchen@yangchen-VirtualBox:~/Escritorio/pruebas$ python3 pruebas_unitarias.py
access_token 60e139031b6f2c05efc805ffb81a2fab56d918ac
refresh_token 7d6bd7955c0db9e7b40c51ff5009405cb570d2ad

el tiempo del token pasado es 0.5205764770507812
no hay que refrescar el token
Respuesta del servidor:
{'weight': {'value': 82000, 'unit': -3}, 'steps': 10000}

el objetivo del usuario es :
Paso: 10000
Suenio: 0
```

Figura 5.51 Prueba de obtención de objetivos

```

yangchen@yangchen-VirtualBox:~/Escritorio/pruebas$ python3 pruebas_unitarias.py
el tiempo del token pasado es 0.012643575668334961
Respuesta del servidor:
[{'type': 'Scale', 'battery': 'medium', 'model': 'Body Cardio', 'model_id': 6, 'deviceid': '5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65'}, {'type': 'Activity Track', 'deviceid': '0901d7332a38349a4b5c483f2d86d3f12dd15ee5', 'hash_deviceid': '0901d7'}

Los datos del dispositivo son:
Nombre del dispositivo: Body Cardio
Tipo del dispositivo: Scale
ID del modelo: 6
ID del dispositivo: 5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65
Los datos del dispositivo son:
Nombre del dispositivo: ScanWatch
Tipo del dispositivo: Activity Tracker
ID del modelo: 93
ID del dispositivo: 0901d7332a38349a4b5c483f2d86d3f12dd15ee5

```

Figura 5.52 Prueba de obtención de dispositivos

```

yangchen@yangchen-VirtualBox:~/Escritorio/pruebas$ python3 pruebas_unitarias.py
el tiempo del token pasado es 0.012670278549194336
no hay que refrescar el token

-----peticion de actividad-----

Los datos de actividad del usuario son:
Elevacion: 0
Calorias: 0.57
Total calorias: 1691.621
Paso: 22
Distancia: 16.61
Hr media: 79
Hr min: 61
Hr max: 152
hr zone 0: 3864
hr zone 1: 467
hr zone 2: 467
hr zone 3: 42
Duracion intensidad suave: 120
Duracion intensidad moderada: 0
Duracion intensidad intensa: 0
Duracion en activa: 0

```

Figura 5.53 Prueba de obtención de datos actividad

```

yangchen@yangchen-VirtualBox:~/Escritorio/pruebas$ python3 pruebas_unitarias.py
el tiempo del token pasado es 0.012735843658447266
no hay que refrescar el token

-----peticion de datos de salud general-----

Datos de salud general del usuario son:
pulse_wave_velocity: 6.271
Datos de salud general del usuario son:
heart_pulse: 71
Datos de salud general del usuario son:
weight: 84.115
fat_mass: 20.67
muscle_mass: 60.28
hydration: 4.349
bone_mass: 3.16
fat_ratio: 24.574
fat_free_mass: 63.445

```

Figura 5.54 Prueba de obtención de datos de salud general

5.5.2 Pruebas de integración

En esta parte, se han probado los siguientes:

- Almacenar los datos en su tabla correspondiente en la base de datos.

- Consultar determinados datos de la base de datos.

id_salud	date_salud	device_id	nombre_usuario	time_created
1	2021-04-04	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1617528487
2	2021-04-06	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1617589230
3	2021-04-05	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1617602841
4	2021-04-07	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1617778530
5	2021-04-08	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618000509
6	2021-04-11	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618134835
7	2021-04-13	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618301881
8	2021-04-16	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618553342
9	2021-04-17	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618564464
10	2021-04-20	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618911099
11	2021-04-21	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1618994992
12	2021-04-24	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1619238953
13	2021-04-25	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1619251535
14	2021-04-28	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1619596321
15	2021-04-27	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1619516455
16	2021-05-02	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1619946141
17	2021-05-03	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620013540
18	2021-05-04	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620110819
19	2021-05-03	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620205367
20	2021-05-17	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620607006
21	2021-05-12	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620714885
22	2021-05-11	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620719082
23	2021-05-14	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1620976155
24	2021-05-15	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1621000000
25	2021-05-15	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1621060442
26	2021-05-18	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1621318238
27	2021-05-17	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1621235610
28	2021-05-19	Saa5cbcb7ad3c105343c4f08a0e40b5ab9bcc9f65	juan	1621415473

Figura 5.55 Datos de salud en la base de datos

id_suenio	date_suenio	model_id	device_id	startdate	enddate	nombre_usuario
2169455745	2021-06-24	93		2021-06-24 03:54:00	2021-06-24 08:32:00	None
2173828931	2021-06-26	93		2021-06-26 04:38:00	2021-06-26 09:27:00	None
2176518517	2021-06-28	93		2021-06-28 03:53:00	2021-06-28 08:08:00	None
2180211605	2021-06-30	93		2021-06-30 04:59:00	2021-06-30 08:49:00	None
2185889490	2021-07-02	93		2021-07-02 03:24:00	2021-07-02 08:31:00	None
2186000000	2021-07-04	93		2021-07-04 03:04:00	2021-07-04 08:11:00	None
2192759423	2021-07-05	93		2021-07-05 03:14:00	2021-07-06 08:51:00	None
2195669160	2021-07-08	93		2021-07-08 03:40:00	2021-07-08 07:48:00	None
2202343701	2021-07-10	93		2021-07-10 04:36:00	2021-07-10 10:57:00	None
2202783931	2021-07-12	93		2021-07-12 05:13:00	2021-07-12 10:18:00	None
2206865357	2021-07-14	93		2021-07-14 04:14:00	2021-07-14 08:42:00	None
2210887815	2021-07-16	93		2021-07-16 04:42:00	2021-07-16 08:58:00	None
2217791809	2021-07-20	93		2021-07-20 03:29:00	2021-07-20 08:39:00	None
2220000000	2021-07-22	93		2021-07-22 03:29:00	2021-07-22 08:44:00	None

Figura 5.56 Datos de sueño en la base de datos

id_entrenamiento	date_entrenamiento	startdate	enddate	device_id	modelo_disp	nombre_usuario
2173828942	2021-06-26	2021-06-26 10:45:00	2021-06-26 10:49:00		93	juan
2192759466	2021-07-04	2021-07-04 23:58:00	2021-07-04 23:57:00		93	juan
2244000000	2021-07-03	2021-07-03 01:00:00	2021-07-03 01:15:00	0901d7332a38349a4b5c483f2d86d3f12dd15ee5	93	juan
2291100858	2021-08-27	2021-08-27 18:17:00	2021-08-27 18:17:00		93	juan
2291100860	2021-08-27	2021-08-27 18:44:00	2021-08-27 18:55:00		93	juan
2319645941	2021-09-10	2021-09-10 17:30:00	2021-09-10 17:40:00		93	juan
2336749696	2021-09-20	2021-09-20 09:28:00	2021-09-20 09:33:00		93	juan
2340000000	2021-09-22	2021-09-22 01:00:00	2021-09-22 01:15:00		93	juan
2350398220	2021-09-28	2021-09-28 11:05:00	2021-09-28 11:13:00		93	juan
2449453342	2021-11-27	2021-11-27 18:36:00	2021-11-27 18:54:00	0901d7332a38349a4b5c483f2d86d3f12dd15ee5	93	juan
2449453346	2021-11-27	2021-11-27 19:09:00	2021-11-27 19:15:00	0901d7332a38349a4b5c483f2d86d3f12dd15ee5	93	juan
2468645445	2021-11-29	2021-11-29 12:01:00	2021-11-29 12:14:00	0901d7332a38349a4b5c483f2d86d3f12dd15ee5	93	juan

Figura 5.57 Datos de entrenamiento en la base de datos

- Conectar la base de datos con la herramienta de visualización.

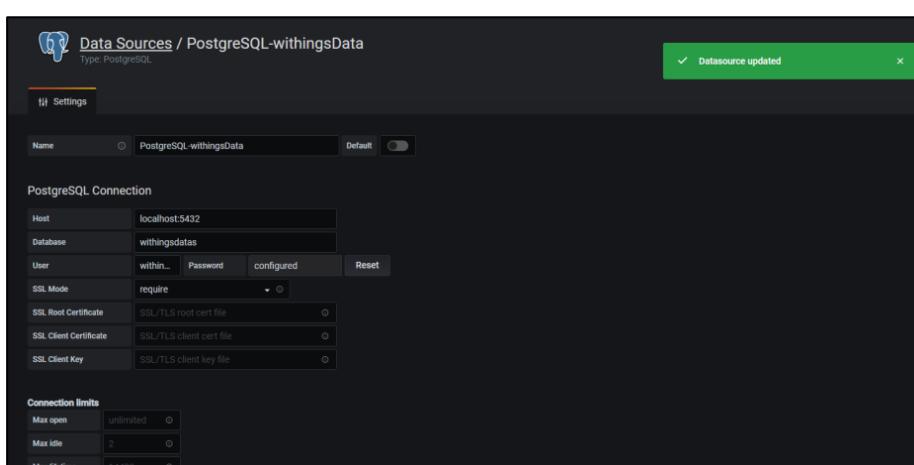


Figura 5.58 Conexión de Grafana con la base de datos

5.5.3 Pruebas de funcionamiento

Finalmente se ha probado el funcionamiento en general del sistema, que consiste en:

1. Conectar a la API de Withings.
 2. Obtener los datos de salud.
 3. Almacenar en la base de datos.
 4. Calcular nuevos datos de salud,
 5. Visualizar los datos a través de Grafana.
 6. Renovar el access token cuando se caduca.

No ha surgido algunos problemas durante la ejecución de programa, el programa ejecución del programa corresponde a lo esperado, no ha surgido algunos problemas. El programa es capaz de obtener los datos y almacenarlos en la base de datos correctamente, y permite también su visualización a través de Grafana.

A continuación, se puede ver algunas capturas sobre el funcionamiento del sistema

```
[...]
yangchen@yangchen-VirtualBox:~/Escritorio/claseV2
yangchen@yangchen-VirtualBox:~/Escritorio/pruebas
yangchen@yangchen-VirtualBox:~/Escritorio/claseV2$ python3 main_local.py 333ad77c411f52c90a9975c2bdb510b914ffcb8 078059a1fdd2fb59f85010bbd7728d3feezd98ac 12035246
access_token 333ad77c411f52c90a9975c2bdb510b914ffcb8
refresh_token 078059a1fdd2fb59f85010bbd7728d3feezd98ac

Ha pasado 0.245551108931396484 segundos desde que obtenemos el token de acceso
Cuando llega a 9000 segundos, tenemos que renovarlo

Todavia es valido el token, no hay que renovarlo

Empezamos la nueva peticion de datos
-----peticion de actividad-----
no tenemos datos de actividad para almacenar
    Datos de actividad almacenados

-----peticion de datos de salud general-----
no se ha obtenido datos sobre la salud del usuario.

-----peticion de Entrenamiento-----
no se ha obtenido datos sobre actividad del usuario.

-----peticion de Sueno resumen-----
no se ha obtenido datos sobre el sueño del usuario.

-----peticion de HR sueño-----
no se ha obtenido datos sobre el sueño del usuario.

-----peticion de ECG-----
no se ha obtenido datos sobre ecg del usuario.

-----peticion de Medidas de alta frecuencia-----
no se ha obtenido datos de alta frecuencia del usuario.

El tiempo de inicio para el siguiente peticion es 1617411600
2021-04-03 03:00:00
```

Figura 5.59 Programa en ejecución

```
Ha caducado el access token, vamos a renovarlo
-----
Actualizacion del token
-----
Tokens actualizados

El nuevo token de acceso es 6b2af17630a9947a7c77bc24de78b60b977de34f
El nuevo refresh token es 078059a1fdd2fb59f85010bbd7728d3fee2d98ac

Empezamos la nueva peticion de datos
-----
peticion de actividad
-----
no tenemos datos de actividad para almacenar
```

Figura 5.60 Actualización del token

	withingsdatas=# select * from agua;		
	id_agua	agua	id_salud
1	44.30949408477498	37	
2	44.67816450525128	38	
3	44.251361704674686	39	
4	44.31793113727827	40	
5	43.60335528472243	42	
6	43.903521263997355	44	
7	51.72844801906464	45	
8	44.10097035281491	46	
9	44.12648497554158	54	
10	44.200296242556156	56	
11	43.920978619167144	59	
12	43.42538003364606	62	
13	43.518024333981636	66	
14	43.992924314353964	67	
15	44.22762249466142	68	
16	43.57762228771135	70	
17	45.66062870354775	72	
18	44.39959574924203	75	
19	44.02695459076373	76	
20	43.37351422372964	77	
21	43.72745675377254	78	
22	43.687001036016824	79	
23	43.88320362767327	80	
24	44.47880467231059	84	
25	44.18629533955948	85	
26	44.35513732690689	86	
27	43.93259288804344	88	
28	44.315550545365501	89	

Figura 5.61 Datos de porcentaje de agua en base de datos

	withingsdatas=# select * from proteina;		
	id_proteina	proteina	id_salud
1	16.388443017656503	37	
2	16.524800570435406	38	
3	16.36694200035913	39	
4	16.391563571322102	40	
5	16.12726839297953	42	
6	16.238288686683955	44	
7	19.132439678284186	45	
8	16.311317801726066	46	
9	16.320754716981135	54	
10	16.34805477464406	56	
11	16.24474551667826	59	
12	16.061441930252652	62	
13	16.095707630376772	66	
14	16.2713555683227	67	
15	16.35816174457861	68	
16	16.117750679564473	70	
17	16.888177739668347	72	
18	16.42176829081555	75	
19	16.28394210889042	76	
20	16.042258685489045	77	
21	16.17316893632683	78	
22	16.15820586263636	79	
23	16.230773944481896	80	
24	16.451064741813507	84	
25	16.34287635846721	85	
26	16.405324764746386	86	
27	16.249041205166755	88	
28	16.39068307842301	89	
29	16.17875350204382	94	

Figura 5.62 Datos de porcentaje de proteína en base de datos

time_ecg	id_ecg	model_id	signal_id	qrs	qt	pr	qtc	nombre_usuario
2021-06-23 19:03:54	1	93	48267247	0	0	0	0	juan
2021-06-22 22:29:34	2	93	47992539	48	340	136	361	juan
2021-06-24 15:57:16	3	93	48448093	0	0	0	0	juan
2021-06-30 21:00:00	4	93	49537548	70	320	128	340	juan
2021-07-02 16:38:31	5	93	49935822	0	0	0	0	juan
2021-07-10 04:16:15	6	93	51462452	0	0	0	0	juan
2021-07-16 15:45:41	7	93	52369006	73	323	146	361	juan
2021-07-18 23:18:37	8	93	52779740	70	350	123	428	juan
2021-07-24 04:30:31	9	93	53803927	70	346	140	389	juan
2021-07-29 11:00:06	10	93	54755964	53	470	143	492	juan
2021-08-07 03:24:24	11	93	56347860	53	350	0	375	juan
2021-08-10 03:28:53	12	93	56970543	0	0	0	0	juan
2021-08-13 03:38:13	13	93	57445737	0	0	0	0	juan
2021-08-20 02:54:57	14	93	58850697	76	356	143	370	juan
2021-09-05 16:06:45	15	93	62127009	53	350	153	401	juan
2021-09-07 21:46:56	16	93	62482140	0	0	0	0	juan
2021-09-15 10:55:13	17	93	63991858	0	0	0	0	juan
2021-09-26 09:46:41	18	93	66316004	0	0	0	0	juan
2021-10-03 10:09:33	19	93	67829020	53	353	140	355	juan

Figura 5.63 Datos de ECG en base de datos

id_imc	imc	id_salud
1	26.25216049382716	3
2	26.237037037037034	4
3	26.0858024691358	7
4	26.116049382716046	8
5	26.084567901234564	10
6	26.125925925925923	11
7	26.19351851851852	12
8	26.123148148148147	13
9	26.139506172839504	14
10	26.098765432098766	15
11	26.08703703703704	16
12	25.98364197530864	17
13	26.162654320987652	19
14	26.061419753086415	20
15	26.116049382716046	21

Figura 5.64 Datos de imc en base de datos



Figura 5.65 Visualización de datos a través de Grafana

Capítulo 6. Conclusiones

6.1 Conclusión general

El trabajo se puede dividir en dos grandes partes, la primera parte se centra fundamentalmente en la investigación y estudio de los dispositivos, infraestructuras y herramientas que pueden servir para el desarrollo del proyecto. Mientras que la segunda parte consiste en la implementación y la prueba del sistema.

Concretamente se ha llevado a cabo un estudio de las variables fisiológicas más representativas que pueden afectar al estado de salud de las personas. Una investigación de los dispositivos wearables y las básculas inteligentes que existen en el mercado, luego se ha hecho un estudio de sus características y las variables fisiológicas que pueden medir. Tras el estudio se ha escogido dos dispositivos de la marca Withings, el reloj inteligente ScanWatch y la báscula inteligente Body Cardio.

También se ha hecho un análisis general de las arquitecturas que podemos encontrar en los sistemas de monitorización continua tradicional y de los que están basados en Internet de las cosas. Además de las arquitecturas, se ha estudiado y comparado también las distintas bases de datos para el almacenamiento de datos de series temporales, y herramientas para la visualización de dichos datos. Luego se ha llevado un estudio detallado del ecosistema Withings. En la que se ha analizado sus dispositivos, la infraestructura y los parámetros medibles según el producto.

Tras los estudios y análisis realizados, se ha tomado la propuesta de desarrollar el sistema usando la arquitectura en capas, formada por la capa de percepción, la capa de

red, la capa de procesamiento, y almacenamiento y la capa de visualización.

Para el desarrollo del sistema, se ha realizado un repaso tanto de las técnicas de realización de diagramas de casos de uso como de modelo entidad relación.

En relación a la implementación de capa de percepción y de red, se ha aprovechado la infraestructura que proporciona la empresa Withings. Usando el ScanWatch y el Body Cardio, los datos registrados son transferidos a la nube de Withings a través del dispositivo móvil o el router.

La mayor dificultad de la implementación del sistema se encuentra en la capa de procesamiento y almacenamiento. Antes de la implementación, se ha hecho un estudio del lenguaje de programación Python y se ha realizado un estudio detallado de las medidas que pueden obtener el reloj y la báscula. Más concretamente, de cada variable se ha analizado su tipo, el método y los parámetros que necesitan para su obtención. Luego se ha probado también el envío de las peticiones de dichos datos usando la herramienta Postman para el estudio del formato de json de la respuesta devuelta por el servidor de recursos de Withings. De todas estas informaciones del Json, se han separado las informaciones comunes (fecha, id del dispositivo, etc) de las informaciones específicas referente a las variables (el valor de los pasos, el valor de las calorías, etc.) de cada método. Tras el análisis, se ha clasificado las medidas en medidas instantáneas, que son aquellas que se han medido en un tiempo concreto o durante un periodo de tiempo muy corto. Y medidas de tipo resumen, son aquellos datos que no se miden en un tiempo concreto, sino nos muestra su valor acumulado durante un periodo de tiempo determinado o en un día concreto, por ejemplo, los pasos realizados en un día, las calorías quemadas en un día, etc. Clasificada las medidas, se ha hecho una selección de una serie de datos que nos resulten más interesante recogerlo para nuestro proyecto.

Para el almacenamiento de los datos, se ha usado la base de datos timescaleDB, basado en PostgreSQL. A través de un programa escrito en Python, se va extrayendo los datos de la nube de Withings a través de su API, y se almacenan en timescaleDB. Con los datos obtenidos es posible calcular nuevas medidas como el índice de masa corporal, la tasa de metabolismo basal, etc.

Y finalmente para la visualización se ha optado la creación del dashboard personalizado mediante la herramienta Grafana. Se han hecho bocetos de dashboard antes de su creación en la herramienta.

6.2 Análisis de los objetivos del proyecto

Objetivos específicos	Nivel de logro	Observaciones
Identificar los parámetros fisiológicos más significativos que determinan el estado de salud de una persona.	Se ha realizado el objetivo al 100%.	Se ha identificado y estudiado los parámetros fisiológicos más significativos que pueden medir los dispositivos inteligentes del mercado.
Analizar los dispositivos wearables que hay en el mercado que puedan tener impacto en el conocimiento de parámetros fisiológicos.	Se ha realizado el objetivo al 100%.	Se ha analizado una gran cantidad de dispositivos wearables y básculas inteligentes capaces de registrar parámetros fisiológicos.
Estudiar los ecosistemas o plataformas que se utilizan para el registro de datos de dispositivos wearables que hay en el mercado.	Se ha realizado el objetivo al 100%.	Se ha realizado un estudio general de algunos sistemas de monitorización continua y un estudio detallado de ecosistema de Withings.
Desarrollar una plataforma para el registro de datos y poder supervisar la evolución de dichos datos.	Se ha realizado el objetivo, pero no por completo.	Se ha desarrollado la plataforma capaz de registrar los datos y poder supervisar la evolución de dichos datos, pero no se ha desarrollado toda la parte de obtención de datos adicionales que están relacionados con la salud a partir de los datos

		registrados.
Diseñar un cuadro de mandos o dashboard que facilite la visualización de los datos fisiológicos.	Se ha realizado el objetivo al 100%.	Se ha diseñado un cuadro de mando con la herramienta Grafana.

Tabla 6.1 Logro de los objetivos específicos

6.3 Conclusiones personales

Como conclusión personal, me gustaría en primer lugar, agradecer a mi tutor Juan Antonio Holgado Terriza, que me ha enseñado y ayudado mucho tanto en la realización del proyecto como la memoria. Sin su ayuda creo que sería bastante difícil terminar el proyecto.

Durante el desarrollo del proyecto, he encontrado varios problemas.

- El primero es la necesidad de aprender lenguajes de programación nuevas.
- El segundo problema consiste en la conexión a la API del Withings debido a que a la mitad del proyecto Withings hicieron algunas modificaciones sobre la forma de acceder a su API, por lo que la manera con la que se accedíamos antes deja de funcionar. Pero esto se resolvió gracias a mi tutor.
- Y el último problema es la búsqueda de las fórmulas para el cálculo de nuevas variables relacionados con la salud. La mayoría de las fórmulas encontradas corresponden a las medidas que nos dan directamente los dispositivos, o que nos faltan datos para poder calcularlos.

Finalmente, como trabajo futuro, se plantean:

- Mejorar el sistema para poder integrar dispositivos de distintas marcas de un mismo usuario.
- Añadir más fórmulas para el cálculo de los nuevos datos de la salud.
- Mejorar el diseño de dashboard.

Capítulo 7. Bibliografía

- [oms21a] Organización Mundial de la Salud. URL: <https://www.who.int/es/news-room/fact-sheets/detail/physical-activity> Último acceso: 01/11/22
- [oms21b] Organización Mundial de la Salud. URL: <https://www.who.int/es/about/frequently-asked-questions> Último acceso: 01/11/22
- [Med21] MedlinePlus – Pulso. URL: <https://medlineplus.gov/spanish/ency/article/003399.htm> Último acceso: 01/11/22
- [Axa21] Axa health keeper. URL: <https://www.axahealthkeeper.com/blog/que-es-y-como-calcular-la-tasa-metabolica-basal/> Último acceso: 01/11/22
- [Wik21] Wikipedia – frecuencia respiratoria. URL: https://es.wikipedia.org/wiki/Frecuencia_respiratoria#cite_ref-Rodr%C3%ADguez_2-1 Último acceso: 01/11/22
- [Npu22] Alonso Rubio, Pablo. Npunto Volumen III. Número 29. Agosto 2020. Utilización de sistemas de monitorización continua de glucosa en edad pediátrica en España. (2020)
- [Sed22] Sociedad Española de Diabetes. Monitorización continua de glucosa y monitorización flash de glucosa, PDF, pag.1-23 (2018)
- [Med22] MedlinePlus – Monitor Holter (24 horas). URL:

- <https://medlineplus.gov/spanish/ency/article/003877.htm> Último acceso: 05/11/22
- [Stan22] Stanford Medicine – Monitor Holter. URL: <https://www.stanfordchildrens.org/es/topic/default?id=holter-monitor-92-P09316> Último acceso: 05/11/22
- [Pat22] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Patrones de Diseño. Pag 3-26. Editorial Pearson Educación(2003)
- [Arq22] Arquitectura multicapa y observadores. URL: <http://www.inf-cr.uclm.es/www/mpolo/asig/0304/0102/arquitecturamulticapayobservadores.PDF> Último acceso: 02/11/22
- [Wik22] Wikipedia – programación por capas. URL: https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas Último acceso: 02/11/22
- [Wit22a] Withings Developer – Scopes. URL: <https://developer.withings.com/developer-guide/v3/get-access/oauth-web-flow#scopes> Último acceso: 02/11/22
- [Inf22] Departamento de Informática Universidad de Valladolid. El modelo cliente/servidor. URL: https://www.infor.uva.es/~fdiaz/sd/2005_06/doc/SD_TE02_20060305.pdf Último acceso: 03/11/22
- [Que22] QuestDB URL: <https://questdb.io/> Último acceso: 02/11/22
- [Inf22] InfluxDB URL: <https://www.influxdata.com/> Último acceso: 02/11/22
- [Tim22] TimescaleDB URL: <https://www.timescale.com/> Último acceso: 02/11/22
- [Pro22] Prometheus URL: <https://prometheus.io/> Último acceso: 02/11/22
- [Gra22] Graphite URL: <https://graphiteapp.org/> Último acceso: 02/11/22
- [Ope22] OpenTSDB URL: <http://opentsdb.net/> Último acceso: 02/11/22
- [Gra22] Grafana URL: <https://grafana.com/> Último acceso: 02/11/22
- [Wit22b] Whitings URL: <https://www.withings.com/es/es/> Último acceso: 02/11/22
- [Wit22c] Withings API reference. URL: <https://developer.withings.com/api-reference> Último acceso: 02/11/22

Anexo 1. Gestión y planificación del proyecto

Anexo1.1 Metodología para el desarrollo del proyecto

El desarrollo del proyecto ha seguido una metodología scrum en las que se realiza una reunión cada una o dos semanas.

La metodología scrum se conoce como un marco de trabajo ágil para la creación de softwares que consiste en la entrega parcial y regular del producto final. El proceso se compone de varios ciclos de duración cortas y fijas, normalmente 2 semanas cada ciclo. Al principio de cada iteración de ciclo se seleccionan unos requisitos y se elabora una lista de las tareas a realizar. Y al final del ciclo, se reúnen el equipo para realizar una revisión del ciclo, en esa reunión, se deben presentar un resultado sobre el proyecto.

Para la gestión del proyecto, se ha usado también como apoyo la herramienta Redmine, que nos permite realizar un mejor seguimiento y organización del proyecto.

Anexo1.2 Gestión del proyecto

A continuación, se presenta una tabla de las tareas que se han realizado en cada ciclo junto con sus horas dedicadas.

- Estudiar variables fisiológicas para el proyecto. En esta tarea se pretende estudiar y analizar los diferentes tipos de variables fisiológicas que podemos medir utilizando un dispositivo wearable.
- Estudio de dispositivos wearables para clínica. Su objetivo es estudiar dispositivos wearables con los que se puedan medir con precisión las variables fisiológicas, de actividad física y de sueño.
- Ampliación de estudio de dispositivos wearables. Consiste en ampliar el estudio de dispositivos realizados en la tarea anterior.
- Análisis de bases de datos o infraestructuras de almacenamiento de datos. Se intenta estudiar las bases de datos o infraestructuras software que nos permiten almacenar datos de series temporales.
- Estudio del dispositivo wearable Withings ScanWatch. En esta tarea se ha realizado un estudio detallado del dispositivo ScanWatch.
- Estudio de InfluxDB y TimescaleDB. Se ha realizado un estudio detallado de estas dos bases de datos.
- Preparación de plantilla para la memoria. Se ha realizado una plantilla en Word para la realización de la memoria.
- Elaboración capítulo 2 de la memoria. Se empieza a escribir el capítulo 2 de la memoria.
- Estudio de la infraestructura de Withings. Se pretende estudiar y entender la infraestructura del proyecto Withings.
- Extraer datos del usuario demo usando API. Se intenta extraer los datos de la salud del usuario demo proporcionado por el Withings,
- Realizar el capítulo 5 de la memoria. Se empieza a escribir el capítulo 5 de la memoria.
- Extracción de datos de usuario real con Postman. Se pretende extraer los datos de un usuario real desde la nube de Withings usando Postman.
- Extracción de datos de usuario real con Python, Se intenta extraer datos de un usuario real desde la nube de Withings usando un programa Python.
- Automatización de la tarea de extracción de datos. Automatizar las tareas de

obtención de datos de la salud del usuario.

- Configuración del servidor. Se instala y configura las herramientas necesarias en el servidor.
- Análisis de los distintos parámetros. Analizar los distintos parámetros que se pueden obtener con las diferentes formas y enlazar aquellos que estén relacionados.
- Identificar las tablas de base de datos que se van a almacenar los datos.
- Realizar diagrama de modelo entidad relación.
- Creación de base de datos. Creación de base de datos en el servidor con las tablas definidos en el modelo entidad relación.
- Lectura de datos desde la nube de Withings y su almacenamiento. Se pretende almacenar los datos obtenidos de la nube de Withings en la base de datos local.
- Modificación y mejora de la base de datos. Se modifica la estructura de la base de datos para adaptar mejor a los datos obtenidos del Withings
- .
- Realización de mockups de dashboard y diagrama de clase del sistema.
- Implementación del sistema. Se empieza a implementar el sistema.
- Realización de dashboard. Se realiza el dashboard siguiendo el mockup realizado anteriormente.
- Pruebas de funcionamiento. Se realizan distintas pruebas sobre el funcionamiento del sistema
- Modificación de la implementación del sistema. Se realizan modificaciones y correcciones sobre el código del sistema.
- Nuevas pruebas de funcionamiento. Se realizan las distintas pruebas sobre el sistema.
- Agregar funcionalidad de calcular datos de salud adicionales. Se añade la funcionalidad de poder calcular algunos datos de salud que no son medidos directamente por los dispositivos wearables, como tmb, imc.
- Probar las nuevas funciones.
- Completar memoria.

Tarea	Tiempo estimado (horas)	Tiempo dedicado (horas)
Estudiar variables fisiológicas para el proyecto	10	5.5
Estudio de dispositivos wearables para clínica	15	10
Ampliar el estudio de dispositivos wearables para clínica	5	8
Análisis de bases de datos o infraestructuras de almacenamiento de datos	20	14
Estudio del dispositivo wearable Withings ScanWatch	15	10
Estudio de InfluxDB y TimescaleDB	10	7
Preparación de plantilla para la memoria	3	1.5
Elaboración capítulo 2 de la memoria	15	8
Estudio de la infraestructura de Withings	15	13
Extraer datos del usuario demo usando API	10	14
Realización del capítulo 5 de la memoria	20	15
Extracción de datos de usuario real con Postman	10	8
Automatización de la tarea de extracción de datos	20	20
Configuración del servidor	15	8
Ánalisis de los distintos parámetros de salud	10	15
Modificar y completar el	6	4

documento de la clasificación de las medidas		
Identificar las tablas que se van a almacenar	5	3
Lectura de datos en la nube de Withings	20	15
Realizacion de mockups de dashboard de Grafana	5	3
Realizacion de diagrama de entidad relación	5	3
Creación de base de datos	5	4
Almacenar datos en la base de datos	10	6
modificación y mejora de base de datos	5	3
Realizacion de diagrama de clase	5	3
Implementación del sistema	30	20
Realizacion de dashboard	5	2.5
Pruebas de funcionamiento	10	5
Modificación de la implementación del sistema	10	5
Nuevas pruebas de funcionamiento	10	4
Agregar nuevas funcionalidades	20	10
Probar las nuevas funcionalidades	8	4
Modificación en implementación del sistema	50	30
Completar memoria	50	30

Tabla A1.1 Lista de las tareas

Anexo1.3 Planificación del proyecto.

En el diagrama de Gantt se puede observar cómo se ha desarrollado todas las tareas del proyecto.

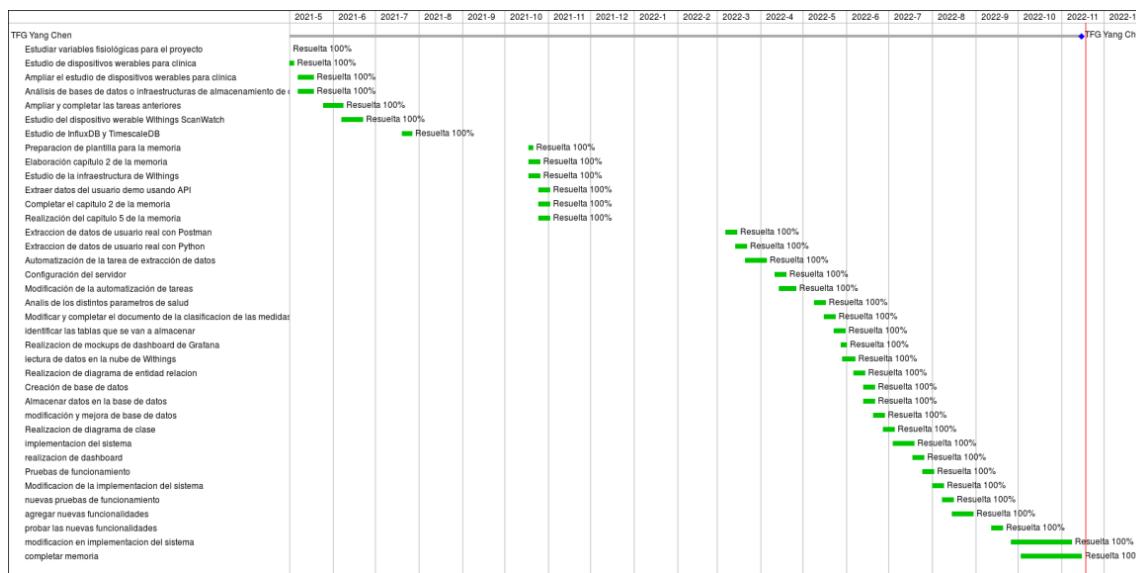


Figura A1.1 Diagrama de Gantt

Anexo1.4 Costes del proyecto

En esta sección vamos a mostrar una tabla sobre el coste del proyecto, que incluye el coste de recursos humanos y los materiales utilizados.

Descripción	Precio por unidad	Cantidad	Total
Reloj inteligente Withings ScanWatch	299.90 €	1	299.90 €
Báscula inteligente Withings ScanWatch	179.95 €	1	179.95 €
TOTAL			479.85 €

Tabla A1.2 Coste de los materiales

Descripción	Numero de horas	Coste por horas	Total
Ingeniero	311.5	30	9345 €

Tabla A1.3 Coste de recurso humano

Por lo tanto, el coste final del proyecto es 9824.85 €.

Anexo 2. Manual de uso

Para la ejecución de este proyecto es necesario tener instalado en el sistema los siguientes programas: Python V3, PostgreSQL con la extensión de TimescaleDB y la herramienta de visualización Grafana. Para la instalación del TimescaleDB y Grafana se puede recurrir al capítulo 3 de la memoria.

Anexo 2.1 Obtención de los tokens

Antes de ejecutar el programa Python, tenemos que obtener el `access_token` y `refresh_token`, los cuales se obtienen con el proyecto desarrollado basado en el ejemplo del Withings.

Este proyecto tenemos que ejecutarlo en un servidor. Si es la primera vez que ejecutamos el proyecto, tenemos que crear primero un entorno virtual de Python. Para ello situamos en la carpeta raíz del proyecto y ejecutamos los siguientes comandos:

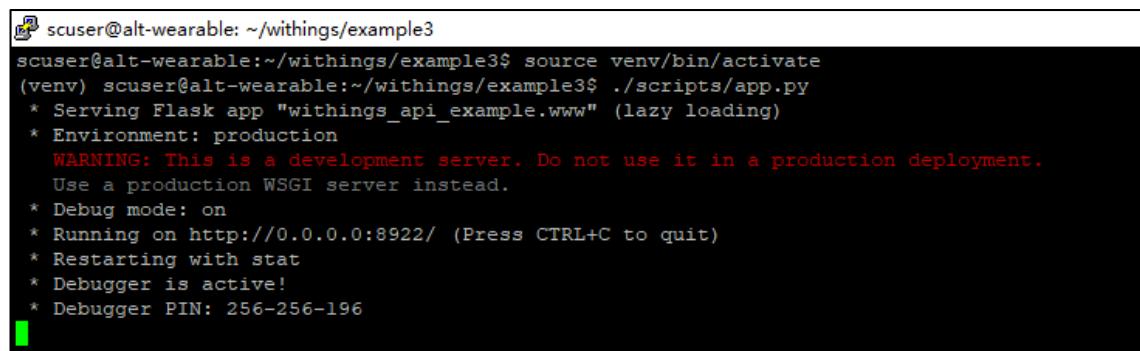
```
virtualenv venv  
source venv/bin/activate  
pip install -e .
```

Luego hay que crear una cuenta de desarrollador en la API de Withings en la URL: https://account.withings.com/partner/add_oauth2

Creado la cuenta, debemos configurar el archivo project.conf rellenando el client_id, client_secret y los demás datos referentes a nuestra cuenta de Withings.

Creado el entorno y configurado el archivo, vamos a lanzar el proyecto con los comandos:

```
source venv/bin/activate  
./scripts/app.py
```



```
scuser@alt-wearable: ~/withings/example3  
scuser@alt-wearable:~/withings/example3$ source venv/bin/activate  
(venv) scuser@alt-wearable:~/withings/example3$ ./scripts/app.py  
* Serving Flask app "withings_api_example.www" (lazy loading)  
* Environment: production  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
* Debug mode: on  
* Running on http://0.0.0.0:8922/ (Press CTRL+C to quit)  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 256-256-196
```

Figura A2.1 Ejecución del proyecto Withings

Ahora abrimos cualquier navegador y vamos a la URL: <http://localhost:puerto>. Nos tenemos que loguear y elegir a que usuario queremos dar el permiso.

```
{  
  "body": {  
    "access_token": "42fb53616f18971ed796f3803a6259def6ab15ee",  
    "expires_in": 10800,  
    "refresh_token": "8b95beccf24d7850f1fe3e8d5beda04bc952a93c",  
    "scope": "user.info,user.metrics,user.activity",  
    "token_type": "Bearer",  
    "userid": "12035246"  
  },  
  "status": 0  
}
```

Figura A2.2 Obtención de los tokens

Anexo 2.2 Ejecución del programa

Solicitado el permiso, vamos a ejecutar el programa Python para la recuperación de datos de salud y su almacenamiento en la base de datos.

Antes de ejecutar el programa, si no tenemos creado la base de datos correspondiente, debemos crearlo dentro del PostgreSQL y luego podemos usar los archivos *.sql de la

carpeta para crear las tablas e insertar los datos iniciales necesarios.

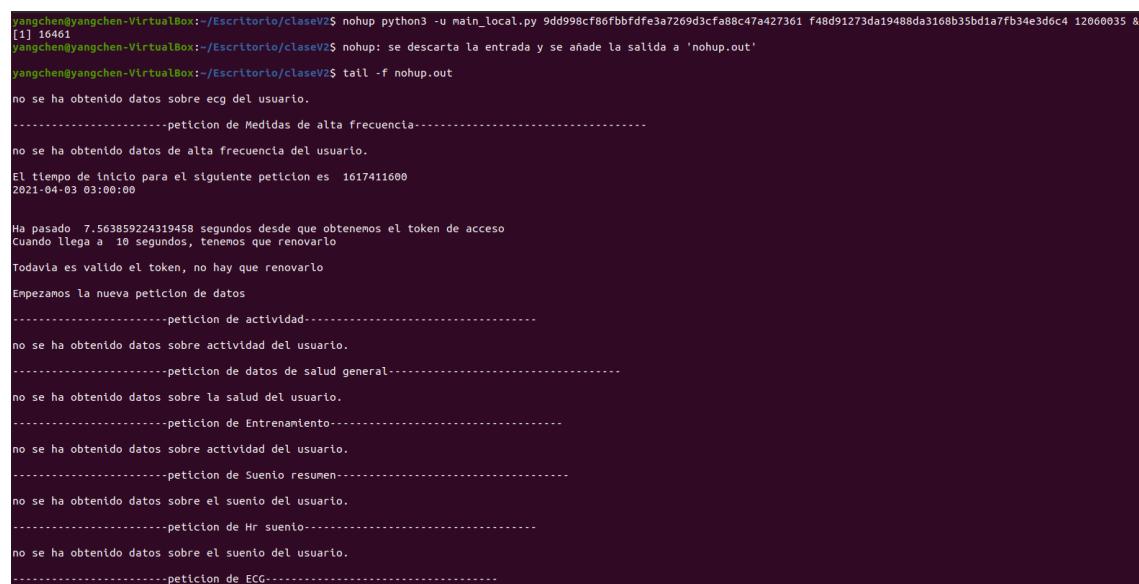
Para los datos personales de los usuarios de la cuenta tenemos que introducirlos manualmente debido a que Withings no autoriza a las cuentas normales la obtención de dichos datos. Se puede modificar la parte de los códigos del archivo main.py que corresponde a los datos de usuarios. Ó insertarlo directamente a través de base de datos.

La ejecución del programa es como cualquier programa Python, pero tenemos que pasarlo como parámetros desde la línea de comando el access_token, el refresh_token y el userid obtenidos.

```
python3 main.py access_token refresh_token userid
```

Como el programa necesita estar ejecutando continuamente en el servidor, es preferible ejecutarlo en segundo plano con &. Una forma de dejarlo ejecutando en el segundo plano y poder ver en tiempo real la salida del programa es usar la orden nohup, que redirige la salida hacia un archivo llamado nohup.out. y después usar la orden tail para ver su contenido.

```
nohup python3 -u main.py access_token refresh_token userid  
tail -f nohup.out
```



```
yangchen@yangchen-VirtualBox:~/Escritorio/claseV2$ nohup python3 -u main_local.py 9dd998cf86fbffdf3a7269d3cfa88c47a427361 f48d91273da19488da3168b35bd1a7fb34e3d6c4 12060035 &  
[1] 16461  
yangchen@yangchen-VirtualBox:~/Escritorio/claseV2$ nohup: se descarta la entrada y se añade la salida a 'nohup.out'  
yangchen@yangchen-VirtualBox:~/Escritorio/claseV2$ tail -f nohup.out  
no se ha obtenido datos sobre ecg del usuario.  
-----peticion de Medidas de alta frecuencia-----  
no se ha obtenido datos de alta frecuencia del usuario.  
El tiempo de inicio para el siguiente petición es 1617411600  
2021-04-03 03:00:00  
  
Ha pasado 7.563859224319458 segundos desde que obtenemos el token de acceso  
Cuando llega a 10 segundos, tenemos que renovarlo  
Todavía es válido el token, no hay que renovarlo  
Empezamos la nueva petición de datos  
-----peticion de actividad-----  
no se ha obtenido datos sobre actividad del usuario.  
-----peticion de datos de salud general-----  
no se ha obtenido datos sobre la salud del usuario.  
-----peticion de Entrenamiento-----  
no se ha obtenido datos sobre actividad del usuario.  
-----peticion de Suenio resumen-----  
no se ha obtenido datos sobre el sueño del usuario.  
-----peticion de Hr sueño-----  
no se ha obtenido datos sobre el sueño del usuario.  
-----peticion de ECG-----
```

Figura A3.3 Ejecución en segundo plano y visualización de contenido con tail

Así ya tenemos el programa en ejecución que recogerá los datos y almacenará en la base de datos de forma continua y automática.

Anexo 2.3 Visualización de datos

Para la visualización de datos, abrimos el navegador y vamos a la URL del Grafana. Logueamos y dirigimos a la opción dashboard que hay en el menú izquierda, y ahí dentro seleccionamos el dashboard que queremos visualizar.

Si no tenemos ningún dashboard, podemos crear uno personalizada.



Figura A.4 Dashboard de grafana

Anexo 3. Continuación de la sección 4.5.2 y sección 4.5.3

● Distance

Es la distancia en metro que recorremos durante un día o un periodo de tiempo según el método que usamos.

➤ Measure v2 – Getactivity

El valor para el parámetro data_fields es distance.

La respuesta del servidor tiene la siguiente estructura:

```
{  
    "status": 0,  
    "body": {  
        "activities": [  
            {  
                "distance": 719.37,  
                "deviceid": null,  
                "hash_deviceid": null,  
                "timezone": "Europe/Madrid",  
                "date": "2021-09-09",  
                "brand": 18,  
                "is_tracker": true  
            },  
            {  
                "distance": 1764.52,  
                "deviceid": null,  
                "hash_deviceid": null,  
                "timezone": "Europe/Madrid",  
                "date": "2021-09-10",  
                "brand": 18,  
                "is_tracker": true  
            }  
        ],  
        "more": false,  
        "offset": 200  
    }  
}
```

Figura 4.17 Respuesta de la petición de distance Getactivity

- Measure v2 – Getintradayactivity
El valor para el parámetro data_fields es distance.
La respuesta del servidor tiene la siguiente estructura:

```
{
  "status": 0,
  "body": {
    "series": {
      "1631146200": {
        "distance": 4.79,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631146260": {
        "distance": 4.79,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631150340": {
        "distance": 5.94,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      . . .
      . . .
      . . .
      "1631225700": {
        "distance": 5.48,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631225940": {
        "distance": 9.92,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631226480": {
        "distance": 6.31,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      }
    }
  }
}
```

Figura 4.18 Respuesta de la petición de distance getintradayactivity

- **Calories**

Es la estimación de las calorías activas quemadas en un día o durante un periodo de tiempo, expresado en Kcal.

- Measure v2 – Getactivity

El valor para el parámetro data_fields es calories.

La respuesta del servidor tiene el siguiente formato:

```
{  
    "status": 0,  
    "body": {  
        "activities": [  
            {  
                "calories": 25.289,  
                "deviceid": null,  
                "hash_deviceid": null,  
                "timezone": "Europe/Madrid",  
                "date": "2021-09-09",  
                "brand": 18,  
                "is_tracker": true  
            },  
            {  
                "calories": 77.951,  
                "deviceid": null,  
                "hash_deviceid": null,  
                "timezone": "Europe/Madrid",  
                "date": "2021-09-10",  
                "brand": 18,  
                "is_tracker": true  
            }  
        ],  
        "more": false,  
        "offset": 200  
    }  
}
```

Figura 4.19 Respuesta de la petición de calories Getactivity

- Measure v2 – Getintradayactivity

El valor para el parámetro data_fields es calories.

La respuesta del servidor tiene el siguiente formato:

```
{
  "status": 0,
  "body": {
    "series": {
      "1631146200": {
        "calories": 0.16,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631146260": {
        "calories": 0.16,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631150340": {
        "calories": 0.2,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      . . .
      . . .
      . . .
      "1631225700": {
        "calories": 0.19,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631225940": {
        "calories": 0.34,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      },
      "1631226480": {
        "calories": 0.22,
        "model": "ScanWatch",
        "model_id": 93,
        "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"
      }
    }
  }
}
```

Figura 4.20 Respuesta de la petición de calories Getintradayactivity

● Intensity

Es la intensidad del trabajo que realizamos, puede ser suave, moderado o intenso. Se puede obtener el tiempo que dura cada intensidad de trabajo usando el método getactivity, cambiando el valor del data_fields a soft, moderate, intense respectivamente. También es posible obtener la suma de tiempo de trabajo moderado e intenso usando el valor active para el parámetro data_fields.

● Elevation

Es el número de piso que subimos en un día o en un periodo de tiempo.

➤ Measure v2 – Getactivity

El valor para el parámetro data_fields es elevation.

El formato de la respuesta es:

```
{
    "status": 0,
    "body": {
        "activities": [
            {
                "elevation": 5.94,
                "deviceid": null,
                "hash_deviceid": null,
                "timezone": "Europe/Madrid",
                "date": "2021-09-09",
                "brand": 18,
                "is_tracker": true
            },
            {
                "elevation": 2.53,
                "deviceid": null,
                "hash_deviceid": null,
                "timezone": "Europe/Madrid",
                "date": "2021-09-10",
                "brand": 18,
                "is_tracker": true
            }
        ],
        "more": false,
        "offset": 200
    }
}
```

Figura 4.21 Respuesta de la petición de elevation Getactivity

➤ Measure v2 – Getintradayactivity

El valor para el parámetro data_fields es elevation.

La estructura de la respuesta es:

```
{  
    "status": 0,  
    "body": {  
        "series": {  
            "1631146200": {  
                "elevation": 0,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631146260": {  
                "elevation": 0,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631150340": {  
                "elevation": 0,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            . . .  
            . . .  
            . . .  
            "1631225700": {  
                "elevation": 0,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631225940": {  
                "elevation": 0,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
        }  
    }  
}
```

Figura 4.22 Respuesta de la petición de elevation Getintradayactivity

4.5.2.2 Medidas relacionadas con la salud

- **Continuous heart rate**

Es la medición del ritmo cardiaco durante la realización de las actividades en un periodo de tiempo.

Se obtiene con el método Measure v2 – Getintradayactivity. El valor para el parámetro data_fields es heart_rate.

La estructura de la respuesta es:

```
{  
    "status": 0,  
    "body": {  
        "series": {  
            "1631145867": {  
                "heart_rate": 73,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631146468": {  
                "heart_rate": 75,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631147098": {  
                "heart_rate": 75,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            . . .  
            . . .  
            . . .  
            "1631229858": {  
                "heart_rate": 72,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631231075": {  
                "heart_rate": 68,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
        }  
    }  
}
```

Figura 4.23 Respuesta de la petición de heart rate

- **SpO2 auto**

Es la medición automática de SpO2 por el dispositivo durante la etapa de sueño del usuario.

Se obtiene con el método Measure v2 – Getintradayactivity. El valor para el parámetro data_fields es spo2_auto.

La respuesta del servidor tiene el siguiente formato:

```
{  
    "status": 0,  
    "body": {  
        "series": {  
            "1631153020": {  
                "spo2_auto": 91,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631153620": {  
                "spo2_auto": 93,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            . . .  
            . . .  
            . . .  
            "1631169820": {  
                "spo2_auto": 93,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
            "1631171620": {  
                "spo2_auto": 100,  
                "model": "ScanWatch",  
                "model_id": 93,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5"  
            },  
        }  
    }  
}
```

Figura 4.24 Respuesta de la petición de Spo2

- Workouts

Devuelve el resumen de entrenamiento que realizamos durante el día, es una agregación de los datos que se capturan durante la realización del entrenamiento, este resumen se crea a partir de los datos que se obtiene con el método Measure v2 – Getintradayactivity.

Hay que usar el método Measure v2 – Getworkouts para la obtención del dicho resumen. Los parámetros para este método son:

- Action, es un string con valor getworkouts.
- Startdateymd, es la fecha inicio del entrenamiento
- Enddateymd, es la fecha fin del entrenamiento
- Data_fields, es una lista de los datos de los parámetros que queremos obtener, solo ponemos en caso de que queremos obtener algunos parámetros en concretos.

Las medidas que se pueden medir durante la realización de un entrenamiento son:

- Calories
- Intensity
- Hr_average, hr_min, hr_max, hr_zone_0, hr_zone_1, hr_zone_2, hr_zone_3
- Pause_duration
- Algo_pause_duration
- Spo2_average
- Steps
- Distance
- Elevation
- Pool_laps
- Strokes
- Pool_length

La estructura de la respuesta devuelta por el servidor es:

```
{  
    "status": 0,  
    "body": {  
        "series": [  
            {  
                "id": 2319645941,  
                "category": 1,  
                "timezone": "Europe/Madrid",  
                "model": 93,  
                "attrib": 0,  
                "startdate": 1631287800,  
                "enddate": 1631288400,  
                "date": "2021-09-10",  
                "deviceid": null,  
                "data": {  
                    "calories": 41,  
                    "intensity": 30,  
                    "manual_distance": 0,  
                    "manual_calories": 0,  
                    "steps": 902,  
                    "distance": 664,  
                    "elevation": 38  
                },  
                "modified": 1631304318  
            }  
        ],  
        "more": false,  
        "offset": 0  
    }  
}
```

Figura 4.25 Respuesta de la petición de workouts

4.5.2.3 Medidas individuales de salud utilizando getmeas

Son medidas que se obtienen usando el método Measure - Getmeas. La mayor parte de estas medidas son datos de salud recopilados por el usuario. Para que sean visibles estos datos se requiere una sincronización del dispositivo con el servidor Withings.

● VO2max

Es la cantidad máxima de oxígeno que nuestro cuerpo es capaz de absorber, consumir y transportar durante un período de tiempo, expresada en ml/min/kg.

Se obtiene con el método Measure – Getmeas, los parámetros que necesitamos son:

- Action, es un string con el valor getmeas.
- Meastype, es un entero que representa el dato del parámetro que queremos obtener, en este caso sería el entero 123.
- Meastypes, es una lista de enteros que representan los distintos parámetros que queremos consultar.
- Startdate, entero que indica el tiempo de inicio.
- Enddate, entero que indica el tiempo de fin.

● Manual SpO2

Es la medición de saturación de oxígeno (cantidad de oxígeno que transporta nuestra sangre) iniciado manualmente por el usuario en un tiempo determinado. Empieza a medir cuando el usuario entra en la función de la medición del spo2 del reloj.

Se obtiene con el método Measure – Getmeas. El valor para el entero meastype es 54.

La respuesta de la petición tiene siguiente formato:

```
{  
    "status": 0,  
    "body": {  
        "updatetime": 1652025511,  
        "timezone": "Europe/Paris",  
        "measuregrps": [  
            {  
                "grpId": 3018688475,  
                "attrib": 0,  
                "date": 1631310835,  
                "created": 1631356434,  
                "category": 1,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "measures": [  
                    {  
                        "value": 98,  
                        "type": 54,  
                        "unit": 0,  
                        "algo": 33619969,  
                        "fm": 131,  
                        "appffmid": 9,  
                        "appliver": 2141  
                    }  
                ],  
                "comment": null  
            },  
        ]  
    }  
}
```

Figura 4.26 Respuesta de la petición de manual spo2

4.5.2.4 Medidas de sueño

- **Sleep duration**

Es el tiempo total del sueño de un día, se calcula sumando el tiempo de sueño de cada fase.

se obtiene usando el método Sleep v2 – Getsummary. Los parámetros necesarios son:

- Action, es un string que toma el valor getsummary.
- Startdateymd, indica la fecha de inicio
- Enddateymd, indica la fecha fin
- Data_fields, una lista de los datos que queremos obtener, en este caso total_sleep_time.

La estructura de la respuesta es:

```
{  
    "status": 0,  
    "body": {  
        "series": [  
            {  
                "id": 2318025916,  
                "timezone": "Europe/Madrid",  
                "model": 16,  
                "model_id": 93,  
                "hash_deviceid": null,  
                "startdate": 1631152740,  
                "enddate": 1631170620,  
                "date": "2021-09-09",  
                "data": {  
                    "total_sleep_time": 17160  
                },  
                "created": 1631224574,  
                "modified": 1631224573  
            },  
        ]  
    }  
}
```

Figura 4.27 Respuesta de la petición de sleep duration

- **Sleep state duration**

Es el tiempo de cada fase de sueño (awake, light, deep, rem).

Se obtiene usando el método Sleep v2 – Getsummary, el valor para el parámetro data_fields es wakeupduration, lightsleepduration, deepsleepduration, remsleepduration.

La respuesta tiene el siguiente formato:

```
{  
    "status": 0,  
    "body": {  
        "series": [  
            {  
                "id": 2318025916,  
                "timezone": "Europe/Madrid",  
                "model": 16,  
                "model_id": 93,  
                "hash_deviceid": null,  
                "startdate": 1631152740,  
                "enddate": 1631170620,  
                "date": "2021-09-09",  
                "data": {  
                    "wakeupduration": 720,  
                    "lightsleepduration": 6540,  
                    "deepsleepduration": 10620  
                },  
                "created": 1631224574,  
                "modified": 1631224573  
            }  
        ],  
        "more": false,  
        "offset": 0  
    }  
}
```

Figura 2.28 Respuesta de la petición de sleep state duration

● Sleep wakeup counts

Es el número de veces que el usuario se despierta mientras estaba en la cama. No incluye número de veces en el que el usuario se levantó de la cama.

Se obtiene usando el método Sleep v2 – Getsummary, el valor para el parámetro data_fields es wakeupcount

La respuesta tiene el siguiente formato:

```
{  
    "status": 0,  
    "body": {  
        "series": [  
            {  
                "id": 2318025916,  
                "timezone": "Europe/Madrid",  
                "model": 16,  
                "model_id": 93,  
                "hash_deviceid": null,  
                "startdate": 1631152740,  
                "enddate": 1631170620,  
                "date": "2021-09-09",  
                "data": {  
                    "wakeupcount": 0  
                },  
                "created": 1631224574,  
                "modified": 1631224573  
            }  
        ],  
        "more": false,  
        "offset": 0  
    }  
}
```

Figura 4.29 Respuesta de la petición de sleep wakeup count

Además de las medidas anteriores, podemos obtener muchos más datos con el método Sleep v2 – Getsummary, tales como:

- Nb_rem_episodes
- Sleep_efficiency
- Sleep_latency
- Total_sleep_time
- Total_timeinbed
- Wakeup_latency
- Waso
- Apnea_hypopnea_index
- Breathing_disturbances_intensity
- Asleepduratin
- Deepsleepduration
- Hr_average, hr_max, hr_min,
- Lightsleepduration

Usando el método getsummary y poniendo la lista anterior como parámetro data_fields, podemos obtener un resumen de todos estos datos.

● Sleep heart rate

Es la frecuencia cardiaca medido mientras el usuario está en el sueño.

Se obtiene con el método Sleep v2 – Get, los parámetros son:

- Action, es un string con valor get
- Startdate, un entero que indica el tiempo de inicio
- Enddate, un entero que indica el tiempo de fin
- Data_fields, es una lista de string que indica los parámetros que queremos obtener, en este caso hr.

La respuesta tiene el siguiente formato:

```
{  
    "status": 0,  
    "body": {  
        "series": [  
            {  
                "startdate": 1631152740,  
                "state": 0,  
                "enddate": 1631152860,  
                "model": "ScanWatch",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "model_id": 93  
            },  
            {  
                "startdate": 1631152860,  
                "state": 1,  
                "enddate": 1631153040,  
                "model": "ScanWatch",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "model_id": 93  
            },  
            . . .  
            . . .  
            . . .  
            {  
                "startdate": 1631162220,  
                "state": 2,  
                "enddate": 1631168640,  
                "model": "ScanWatch",  
                "hr": {  
                    "1631162663": 58,  
                    "1631163262": 63,  
                    . . .  
                }  
            }  
        ]  
    }  
}
```

Figura 4.30 Respuesta de la petición de sleep heart rate

4.5.2.5 ECG signal y Atrial fibrillation result from ECG

Para obtener un ECG con detalle tenemos que seguir dos pasos. El primer paso consiste en usar Heart v2 – List para obtener una lista de registros de ECG con la clasificación de Afib, donde nos proporciona también el id de la señal ECG, que nos servirá luego para la petición de los detalles de un ECG concreto.

Los parámetros de esta petición son:

- Action, un string con valor list
- Startdate, tiempo de inicio
- Enddate, tiempo fin

La respuesta tiene la siguiente estructura:

```
{  
    "status": 0,  
    "body": {  
        "series": [  
            {  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "model": 93,  
                "ecg": {  
                    "signalid": 110038499,  
                    "afib": 0  
                },  
                "heart_rate": 85,  
                "timestamp": 1646620501  
            }  
        ],  
        "more": false,  
        "offset": 0  
    }  
}
```

Figura 4.31 Respuesta de la petición de ecg list

Una vez obtenido el id de la señal, podemos usarlo con la petición Heart v2 – Get, que nos devuelve los detalles de un ECG.

Los parámetros son:

- Action, un string con valor get
- Signalid, el id que obtenemos con la petición anterior.

La estructura de la respuesta es:

```
{  
    "status": 0,  
    "body": {  
        "signal": [  
            0,  
            0,  
            -10,  
            -16,  
            -17,  
            -8,  
            ...  
            ...  
            ...  
            71,  
            61,  
            36,  
            11,  
            -3,  
            -6  
        ],  
        "sampling_frequency": 300,  
        "wearposition": 0,  
        "model": 93,  
        "heart_rate": {  
            "grpid": 3514918513,  
            "value": 62,  
            "date": 1649703520  
        }  
    }  
}
```

Figura 4.32 Respuesta de la petición de `ecg get`

Además de usar el método Heart, podemos obtener otras medidas generales complementarias relacionadas con ECG mediante el método Getmeas, las cuales son:

- **QRS Interval duration**

Es la duración del intervalo QRS basado en la señal de un ECG. Se obtiene con el método Measure – Getmeas, el valor del entero meastype es 135.

La respuesta tiene la siguiente forma:

```
{  
    "status": 0,  
    "body": {  
        "updatetime": 1652026319,  
        "timezone": "Europe/Paris",  
        "measuregrps": [  
            {  
                "groupid": 3068274108,  
                "attrib": 0,  
                "date": 1631696113,  
                "created": 1633166180,  
                "category": 1,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "measures": [  
                    {  
                        "value": 60,  
                        "type": 135,  
                        "unit": 0,  
                        "algo": 0,  
                        "fm": 131,  
                        "apppfmid": 9,  
                        "appliver": 2141  
                    }  
                ],  
                "comment": null  
            }  
        ]  
    }  
}
```

Figura 4.33 Respuesta de la petición de QRS

- **PR Interval duration**

Es la duración del intervalo PR basado en la señal de un ECG. Se obtiene con el método Measure – Getmeas, el valor del entero meastype es 136.

La respuesta tiene la siguiente forma:

```
{  
    "status": 0,  
    "body": {  
        "updatetime": 1652026420,  
        "timezone": "Europe/Paris",  
        "measuregrps": [  
            {  
                "groupid": 3068274108,  
                "attrib": 0,  
                "date": 1631696113,  
                "created": 1633166180,  
                "category": 1,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "measures": [  
                    {  
                        "value": 150,  
                        "type": 136,  
                        "unit": 0,  
                        "algo": 0,  
                        "fm": 131,  
                        "apppfmid": 9,  
                        "appliver": 2141  
                    }  
                ],  
                "comment": null  
            }  
        ]  
    }  
}
```

Figura 4.34 Respuesta de la petición de PR

- **QT Interval duration**

Es la duración del intervalo QT basado en la señal de un ECG. Se obtiene con el método Measure – Getmeas, el valor del entero meastype es 137.

La respuesta tiene la siguiente forma:

```
{  
    "status": 0,  
    "body": {  
        "updatetime": 1652026460,  
        "timezone": "Europe/Paris",  
        "measuregrps": [  
            {  
                "grpid": 3068274108,  
                "attrib": 0,  
                "date": 1631696113,  
                "created": 1633166180,  
                "category": 1,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "measures": [  
                    {  
                        "value": 370,  
                        "type": 137,  
                        "unit": 0,  
                        "algo": 0,  
                        "fm": 131,  
                        "apppfmid": 9,  
                        "appliver": 2141  
                    }  
                ],  
                "comment": null  
            }  
        ]  
    }  
}
```

Figura 4.35 Respuesta de la petición de QT

- **QTC Interval duration**

Es la duración del intervalo QTC basado en la señal de un ECG. Se obtiene con el método Measure – Getmeas, el valor del entero meastype es 138.

La respuesta tiene la siguiente forma:

```
{  
    "status": 0,  
    "body": {  
        "updatetime": 1652026527,  
        "timezone": "Europe/Paris",  
        "measuregrps": [  
            {  
                "grpid": 3068274108,  
                "attrib": 0,  
                "date": 1631696113,  
                "created": 1633166180,  
                "category": 1,  
                "deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "hash_deviceid": "0901d7332a38349a4b5c483f2d86d3f12dd15ee5",  
                "measures": [  
                    {  
                        "value": 379,  
                        "type": 138,  
                        "unit": 0,  
                        "algo": 0,  
                        "fm": 131,  
                        "apppfmid": 9,  
                        "appliver": 2141  
                    }  
                ],  
                "comment": null  
            }  
        ]  
    }  
}
```

Figura 4.36 Respuesta de la petición de QTC

4.5.3 Medidas relacionadas con la balanza

Ahora vamos a ver los datos que se pueden medir mediante la balanza, son:

- Weight, es el peso del usuario

- Muscle mass, la masa muscular es el volumen de los tejidos totales del cuerpo que corresponden al músculo.
- Bone mass, la masa ósea es la cantidad de minerales que contiene los huesos.
- Fat mass, la grasa corporal es la cantidad de grasa que tiene una persona en el cuerpo.
- Fat free mass, la masa libre de grasa es todo el peso del cuerpo de una persona que no sea grasa.
- Fat ratio, es el índice de masa corporal
- Heart rate, es la frecuencia cardíaca
- Pulse wave velocity, la velocidad de onda de pulso, es el patrón que se usa para determinar la rigidez arterial.

Todas estas medidas se pueden obtener con el método Measure – Getmeas, en este caso, el valor del parámetro data_fields sería 1,76,88,8,5,6,77,91

La respuesta tiene la siguiente estructura:

```
{
  "status": 0,
  "body": {
    "updatetime": 1652044896,
    "timezone": "Europe/Paris",
    "measuregrps": [
      {
        "grpid": 3033500830,
        "attrib": 8,
        "date": 1631904272,
        "created": 1631904324,
        "category": 1,
        "deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
        "hash_deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
        "measures": [
          {
            "value": 6422,
            "type": 91,
            "unit": -3,
            "algo": 0,
            "fm": 131
          }
        ],
        "comment": null
      },
      {
        "grpid": 3033500823,
        "attrib": 10,
        "date": 1631904272,
        "created": 1631904324,
        "category": 1,
        "deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
        "hash_deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
        "measures": [
          {
            "value": 6422,
            "type": 91,
            "unit": -3,
            "algo": 0,
            "fm": 131
          }
        ],
        "comment": null
      }
    ]
  }
}
```

```

    "date": 1631904272,
    "created": 1631904324,
    "category": 1,
    "deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
    "hash_deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
    "measures": [
        {
            "value": 6315,
            "type": 91,
            "unit": -3,
            "algo": 0,
            "fm": 131
        }
    ],
    "comment": null
},
{
    "grpid": 3033500806,
    "attrib": 0,
    "date": 1631904272,
    "created": 1631904323,
    "category": 1,
    "deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
    "hash_deviceid": "5aa5bcb7ad3c105343c4f68a0e40b54b09cc9f65",
    "measures": [
        {
            "value": 83859,
            "type": 1,
            "unit": -3,
            "algo": 3,
            "fm": 131
        },
        {
            "value": 2129,
            "type": 8,
            "unit": -2,
            "algo": 3,
            "fm": 131
        },
        {
            "value": 5943,
            "type": 76,
            "unit": -2,
            "algo": 3,
            "fm": 131
        }
    ]
}

```

```
        "fm": 131
    },
    {
        "value": 4291,
        "type": 77,
        "unit": -2,
        "algo": 3,
        "fm": 131
    },
    {
        "value": 312,
        "type": 88,
        "unit": -2,
        "algo": 3,
        "fm": 131
    },
    {
        "value": 25388,
        "type": 6,
        "unit": -3
    },
    {
        "value": 62569,
        "type": 5,
        "unit": -3
    }
],
"comment": null
}
]
}
```