Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Agente IoT de FIWARE para el control de un dispositivo de monitorización de actividad física

Autor: Rafael Díaz Fernández
Tutor: Jorge Calvillo Arbizu

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Agente IoT de FIWARE para el control de un dispositivo de monitorización de actividad física

Autor:
Rafael Díaz Fernández

Tutor:
Jorge Calvillo Arbizu
Departamento de Ingeniería Telemática

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Grado: Agente IoT de FIWARE para el control de un dispositivo de monitorización de actividad física

Autor: Rafael Díaz Fernández

Tutor: Jorge Calvillo Arbizu

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Estos cuatro años de grado han sido los más duros académicamente de mi vida, pero también los que mayor satisfacción me han dado. He podido conocer a personas maravillosas que, sin duda, me acompañarán toda mi vida, he adquirido una gran cantidad de conocimientos de mano de profesores muy cualificados y he sido muy bien preparado para mi futura vida laboral.

En primer lugar, quería agradecer a mis padres su apoyo incondicional durante todo el grado, sin presionarme en ningún momento y prestándome todo tipo de facilidades para que mi desarrollo en la carrera fuera el máximo posible. Estos años me han hecho madurar mucho, y he aprendido a valorar vuestro esfuerzo para que mi hermano y yo hayamos vivido sin preocupación alguna más que la de estudiar y formarnos. Me he abierto más a ustedes como persona y me encanta salir, viajar y hacer planes con vosotros. Desde pequeño me habéis inculcado la cultura del esfuerzo y sin ustedes no sería nada de lo que soy hoy. También agradecer a mi hermano Alberto, por tenerme como referencia y motivarme a seguir, y digo esto porque lo veo en tus ojos cada vez que cuento en casa algún pequeño éxito que haya conseguido, al igual que en los de papá y mamá. Es insuficiente este párrafo para agradeceros todo lo que habéis hecho por mi. Gracias a los tres.

Por otro lado, me gustaría agradecer a mis compañeros por el genial grupo de trabajo que hemos formado. El trabajo en grupo, en mi opinión, es uno de los aspectos más importantes para tener éxito en el grado, y nuestro grupo era un muy buen equipo. Además, habéis conseguido hacer el camino muy ameno. Me llevo muchos amigos de esta experiencia en Sevilla: Gonzalo, Antonio, David, Manu, Vicente…, pero en especial a Luis, Juan y Leo.

Desde el primer día que llegué a la residencia, Luis me ha acompañado en mi día a día y nos hemos vuelto inseparables. Desde su gran brillantez, nos ha ayudado siempre que ha podido, e incluso cuando tampoco podía, facilitándonos así muchas asignaturas de la carrera y ejerciendo de profesor particular en algunas ocasiones, véase Comunicaciones Digitales. Se ha convertido en un hermano para mí. A Leo y Juan los conocí uno de los primeros días, que iban a la cafetería y les dije que iba con ellos. Uno de los mayores aciertos de la carrera, pues son unas maravillosas personas y también hemos pasado estos cuatro años juntos, lo que nos ha hecho ser unos amigos de verdad. De ellos tres sé que, aunque acabemos en distintos lugares, nunca vamos a dejar de vernos, pues no querría dejar atrás a estas maravillosas personas, y sé que ellos tampoco querrán.

Por último, debo agradecer a mis profesores del grado por su enseñanza y exigencia, que nos ha formado como buenos ingenieros de telecomunicaciones, con vistas al mundo laboral, donde tendremos que lidiar con problemas para los que salimos muy bien preparados. En especial, a Jorge Calvillo, mi tutor del TFG, que me ha ayudado en todo momento con el proyecto, guiándome y ofreciéndome facilidades para que el desarrollo del mismo se produjera lo mejor posible.

Rafael Díaz Fernández

Sevilla, 2019

Resumen

La actividad física es conocida como una de las intervenciones más eficaces en la prevención de enfermedades y es usada como tratamiento principal de muchas de ellas, tales como la enfermedad pulmonar obstructiva crónica (EPOC). Sin embargo, una de las características de este tipo de tratamientos, es que se realizan fuera del entorno médico y dependen casi exclusivamente del paciente, luego puede llevar a una incorrecta realización de los ejercicios, además de una pobre adhesión a los mismos, y complicar así el seguimiento del paciente.

Es por ello que conviene disponer de una solución de monitorización del ejercicio físico de los pacientes, permitiendo que el personal médico pueda acceder sencillamente a los datos y llevar un control del tratamiento, facilitando así el seguimiento del paciente. Como solución a esto, el Grupo de Ingeniería Biomédica de la ETSI ha desarrollado una camiseta inteligente que recoge datos de la actividad física de los pacientes, aunque aún no permite que esta información sea fácilmente accedida por profesionales sanitarios.

Para solucionar esta carencia, en este trabajo se ha realizado un Agente IoT en una aplicación móvil que actúa de intermediario entre la camiseta inteligente y la plataforma de IoT FIWARE, permitiendo el acceso a los datos a cualquier usuario autorizado. Además, para completar la solución, se ha llevado a cabo un Servicio Web con Angular, que actúa de consumidor de la información almacenada en FIWARE y presenta una interfaz atractiva para el seguimiento de la actividad física de los pacientes, conectándose a su vez con una base de datos de Firebase.

Abstract

Physical activity is known as one of the most effective techniques in diseases prevention and it is used as the main treatment in some of them, as the chronic obstructive pulmonary disease (COPD). Nevertheless, one of the features in this type of treatments is that they are done outside the medical environment and they depend almost exclusively of the patient will, so it can arrange to a bad execution of the exercises, as well as a poor compliance to them, and make the patient tracking more difficult.

That is why it is needed a solution to exercise monitoring, allowing doctors to easily access to the data and follow the patient's treatment. The Biomedical Engineering Group at ETSI has developed an intelligent t-shirt that collects data from the patient's physical activity, but it can not let doctors access the information in an easy way.

As a solution to this problem, in this project an IoT Agent in a mobile app has been made that acts as an intermediary between the intelligent t-shirt and the IoT platform FIWARE, which allows access to the data to any authorized user. In addition, to provide a complete solution, it has been developed a Web Service with Angular, that consumes the stored information in FIWARE and offers an attractive interface to the patient's physical activity tracking, also being connected to a Firebase database.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Look up at the stars and not down at your feet. Try to make sense of what you see, and wonder about what makes the universe exist. Be curious.

- Stephen Hawking -

Como introducción a esta memoria se podrá conocer la motivación para la realización de este proyecto, así como los objetivos del mismo. Además, podrá encontrarse la metodología adoptada y el plan de trabajo que se planteó al inicio, junto con la estimación temporal realizada y el tiempo real dedicado finalmente.

1.1 Motivación y objetivos

1.1.1 Motivación

La inactividad física es, según la Organización Mundial de la Salud, el cuarto factor de riesgo de mortalidad mundial, así como la causa de entre el 21% y el 25% de los cánceres de mama y colon, el 27% de los casos de diabetes y aproximadamente el 30% de la carga de cardiopatía isquémica. Mientras, se ha demostrado que un nivel adecuado de actividad física de forma regular reduce el riesgo de hipertensión, diabetes, distintos tipos de cánceres como el de mama o colon, depresión, hipertensión… además de mejorar la salud ósea y funcional y ser determinante clave en el gasto energético, siendo fundamental para el control de peso. [1]

Los programas de ejercicio multicomponente y, en particular, el entrenamiento de fuerza, constituyen las intervenciones más eficaces para retrasar la discapacidad y otros eventos adversos [2]. Además, la actividad física no sólo tiene consecuencias en la prevención de enfermedades, sino que es usado como tratamiento de las mismas. Son muchas las patologías cuyo tratamiento principal consiste en la realización de ejercicio físico del paciente, siendo una de ellas la enfermedad pulmonar obstructiva crónica (EPOC). Sin embargo, una de las características de estos tratamientos es que su realización se lleva a cabo fuera del entorno médico y, por ello, dependen casi exclusivamente del paciente. Esto puede llevar a una incorrecta realización de los ejercicios y una pobre adhesión al tratamiento.

Es por ello que el seguimiento del tratamiento de los pacientes en este tipo de enfermedades se complica, pues no puede saberse exactamente cómo se ha llevado a cabo, ni siquiera si realmente se ha realizado, más que por la palabra del paciente. Así, la monitorización del ejercicio físico supone una gran ayuda al equipo médico encargado, obteniendo así los datos de los ejercicios realizados, aportando exactitud al tratamiento y mejorando su observación.

La camiseta inteligente desarrollada por el Grupo de Ingeniería Biomédica de la ETSI recoge datos de la actividad física del paciente, acompañándolo en la realización de actividad física y monitorizando sus ejercicios. Esto permite saber qué ejercicios se realizan y cómo. Sin embargo, en el estado actual de desarrollo de la camiseta, estos datos no pueden ser accedidos fácilmente por profesionales sanitarios que quieran conocer la información recogida, luego se necesitaría facilitar este acceso, como siguiente paso en el desarrollo del

dispositivo.

En la monitorización remota convendría disponer de los datos recogidos en la nube, ofreciendo la posibilidad de acceder a los datos del paciente en cualquier momento y en cualquier sitio, con exactitud y sin lugar a fallos, dando lugar a un control exhaustivo del tratamiento. Esto se puede llevar a cabo con una solución propietaria, pero en la actualidad existen interesantes opciones a considerar como el Internet of Things (IoT).

El uso del IoT para la disponibilidad de los datos en la nube conlleva una serie de ventajas, tales como la automatización de los procesos y una mayor flexibilidad, al poder incluir cualquier tipo de dispositivo en la red, entre otras. Además, permite un análisis de datos óptimo y profundo y un aprovechamiento de recursos disponibles a todo el mundo para la realización de soluciones específicas, lo cual puede llevar a un ahorro en el presupuesto dedicado.

Por este motivo, la motivación del proyecto es integrar la camiseta inteligente en una plataforma de IoT abierta, que permita almacenar los datos de manera automática para que otros sistemas puedan disponer de ella, así como el acceso del equipo médico a ellos fácilmente, en cualquier momento y en cualquier lugar. Además, otra de las motivaciones del trabajo es poner en práctica los conocimientos aprendidos en el Grado y profundizarlos, así como la investigación de nuevas tecnologías para el desarrollo de estas soluciones.

1.1.2 Objetivos

Como principal objetivo del proyecto, se encuentra el diseño y desarrollo de un Agente IoT para la plataforma FIWARE que actúe de intermediario entre una camiseta inteligente, que recoge datos de la actividad física del usuario, y FIWARE, permitiendo a cualquier usuario autorizado acceder a la información de una forma fácil y segura.

Este agente se deberá llevar a cabo en una aplicación móvil para Android, pues la camiseta inteligente está diseñada para que se comunique con ella, la cual debe tener conexión a internet para mandar los datos a la nube, y que permite al usuario visualizar la información recogida y enviada en la misma pantalla.

El objetivo principal se puede descomponer en los siguientes subobjetivos:

- Estudio de FIWARE y sus componentes. Es importante conocer bien la plataforma IoT a utilizar, pues queremos aprovechar sus ventajas para la solución.
- Mapeo de entidades para enviar a FIWARE, que contengan los datos recogidos por la camiseta.
- Diseño de la solución completa (aplicación móvil, servicio de almacenamiento de datos, interfaz de visualización…)
- Desarrollo de la aplicación móvil, que funcione como Agente IoT entre la camiseta y FIWARE.

Como objetivo secundario y con el fin de alcanzar una resolución más completa, se realizará un servicio web que actúe de consumidor de la información enviada a FIWARE y utilice una base de datos para guardarla y registrar el progreso realizado, permitiendo al usuario profesional ver los datos más recientes de la camiseta inteligente y conocer los recogidos en los días anteriores.

Este objetivo secundario también puede descomponerse en varios subobjetivos:

- Conocimiento y estudio de las tecnologías a aplicar en la creación del servicio.
- Desarrollo del servicio web, que consuma los datos de actividad física almacenados en FIWARE.
- Integración del servicio con una base de datos, dando lugar a una solución más completa.

1.2 Solución planteada

Se ha propuesto una aplicación móvil para Android que actúe como Agente IoT de FIWARE entre la camiseta inteligente y el Orion Context Broker de FIWARE, servidor que implementa una API de tipo REST mediante el protocolo de comunicación HTTP y que permite el acceso de los consumidores a esos datos, así como la suscripción a ellos. La solución diseñada usará FIWARE pues es una plataforma IoT abierta de fácil implementación y escalabilidad, que ofrece APIs estandarizadas para el desarrollo de aplicaciones y que está en continuo crecimiento.

La camiseta inteligente se comunica con el agente IoT mediante la tecnología MQTT, permitiendo que los datos lleguen a la aplicación móvil mediante mensajes JavaScript Object Notation (JSON).

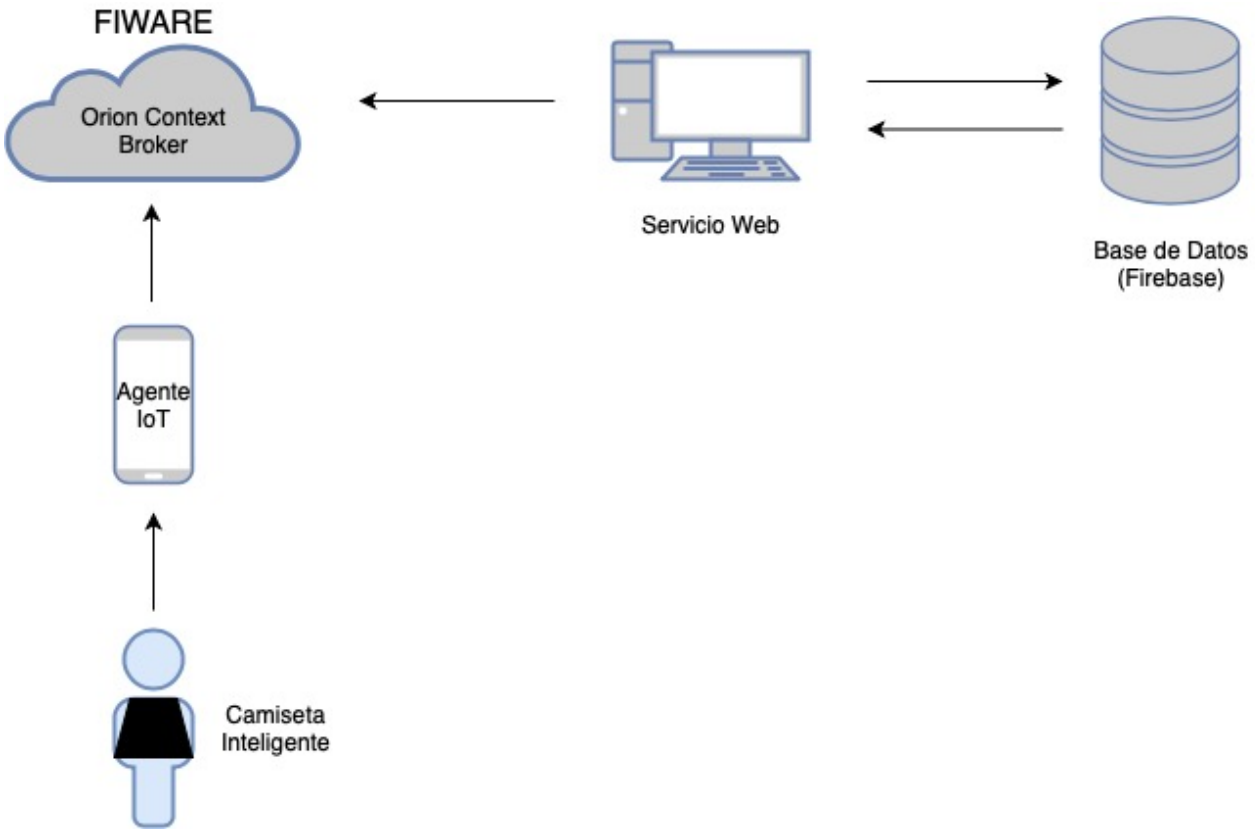

Figura 1. Solución planteada.

La aplicación móvil producirá mensajes para el Context Broker mediante peticiones POST. Además, queremos que la aplicación móvil muestre los mensajes que se mandan a FIWARE, para conocer así qué es exactamente lo que se está enviando.

En cuanto al servicio web a desarrollar, éste realizará peticiones GET a Orion y mostrará por pantalla los datos recibidos. También tendrá acceso a una base de datos de Google, Firebase, que guardará la información obtenida y permitirá crear una tabla para observar el progreso realizado en los últimos días. Se utilizarán varias tecnologías para su creación, destacando entre ellas Angular y Bootstrap, para el apartado visual.

1.3 Plan de trabajo

En este apartado se detallan las tareas que se realizarán para llevar a cabo el proyecto y una comparación del tiempo estimado y el tiempo real dedicado a cada una de ellas, incluyendo formación e investigación, así como la programación y pruebas de la aplicación de Android. Estas tareas son las siguientes:

- Actividad 1: Búsqueda y obtención de la información. En esta actividad inicial se estudiará la documentación de FIWARE, disponible en su web [3], con el objetivo de conocer la plataforma, haciendo hincapié en el uso de sus componentes y su funcionamiento, junto con las APIs que ofrece para el desarrollo de aplicaciones. Por otro lado, se va a estudiar la documentación de la camiseta inteligente utilizada en el proyecto, proporcionada por el equipo desarrollador.

Búsqueda y obtención de la información	**Tiempo estimado:**	**Tiempo real:**
Investigación y documentación FIWARE	40 horas	46 horas
Documentación Camiseta Inteligente	2 horas	2 horas
Total	42 horas	48 horas

Tabla 1. Búsqueda y obtención de información.

- Actividad 2: Curso de Angular. Se tomará un curso de Angular con el objetivo de conocer esta tecnología para la realización del servicio web y de aprender a programar en TypeScript. En esta actividad, también se llevará a cabo una investigación acerca de otras tecnologías que podrían incluirse junto a Angular, tales como Bootstrap, para el diseño, y Firebase, para la base de datos.

Curso de Angular	**Tiempo estimado:**	**Tiempo real:**
Realización de un curso de Angular	30 horas	40 horas
Documentación Bootstrap, Firebas	2 horas	2 horas
Total	32 horas	42 horas

Tabla 2. Curso de Angular.

- Actividad 3: Diseño de la solución. Se tendrá en cuenta el planteamiento de la solución descrita en el apartado anterior, y se ideará el diseño del funcionamiento del Agente IoT. Seguidamente, se realizará el diseño de la aplicación móvil que actúa como agente IoT, las vistas de las que dispondrá y la interfaz del usuario. Una vez acabada esta tarea, se llevará a cabo el diseño del servicio web de Angular, tanto de su funcionamiento como de su interfaz, además de la conexión del servicio con FIWARE, por un lado, y la base de datos, por otro.

Diseño	Tiempo estimado:	Tiempo real:
Planteamiento de la solución	2 horas	2 horas
Diseño del Agente IoT	4 horas	4 horas
Diseño de la Aplicación móvil	20 horas	20 horas
Diseño del Servicio Web	20 horas	20 horas
Total	46 horas	46 horas

Tabla 3. Diseño.

- Actividad 4: Implementación de la aplicación movil. En primer lugar, se tendrá que conectar la camiseta inteligente con la aplicación móvil, siguiendo la documentación aportada, e instalar la máquina virtual de CentOs, que alberga el Context Broker, el cual también tendrá que ser instalado, tal y como dice la documentación de FIWARE. Posteriormente, se seguirá con la implementación de la aplicación móvil, tal y como se haya diseñado, y se realizará la conexión con el Context Broker de FIWARE para las peticiones POST.

Implementación Aplicación móvil	Tiempo estimado:	Tiempo real:
Conexión Camiseta Inteligente con la Aplicación de Android	6 horas	6 horas
Instalación Máquina Virtual y Orion Context Broker	6 horas	12 horas
Conexión Aplicación móvil con Orion Context Broker	12 horas	20 horas
Implementación Aplicación Móvil	20 horas	20 horas
Total	44 horas	58 horas

Tabla 4. Implementación de la aplicación móvil.

- Actividad 5: Implementación servicio web. Igualmente, una vez acabada la aplicación móvil, se comenzará la implementación del servicio web con las tecnologías estudiadas en la segunda actividad y se conectará este servicio con el Context Broker desplegado para las peticiones GET. Se instalará los módulos necesarios para el servicio, entre ellos los de Bootstrap y Firebase. Además, se realizará la conexión, por el otro lado, con la Realtime Database de Firebase.

Implementación Servicio Web	Tiempo estimado:	Tiempo real:
Conexión Servicio Web con Orion Context Broker	6 horas	10 horas
Conexión Servicio Web con Firebase	12 horas	24 horas
Implementación Servicio Web	20 horas	24 horas
Total	38 horas	58 horas

Tabla 5. Implementación del servicio web.

- Actividad 6: Pruebas y validación. Se realizarán las pruebas unitarias pertinentes y se validarán los requisitos tanto de la aplicación como del servicio web. Una vez realizadas, se terminará con la corrección de errores necesaria para el correcto funcionamiento del sistema.

Pruebas y validación	**Tiempo estimado:**	**Tiempo real:**
Pruebas unitarias	4 horas	4 horas
Corrección de errores	20 horas	30 horas
Total	24 horas	34 horas

Tabla 6. Pruebas y validación.

- Actividad 7: Documentación. Se trata de la escritura de la memoria del proyecto.

Documentación	**Tiempo estimado:**	**Tiempo real:**
Memoria del Proyecto	70 horas	80 horas
Total	70 horas	

Tabla 7. Documentación.

Resumen de tiempo total estimado y consumido:

Total	**296 horas**	**366 horas**

Tabla 8. Tiempo total.

2 ESTADO DEL ARTE

El esfuerzo continuo, incansable y persistente ganarán.

- James Whitcomb Riley -

EN este apartado se verá de dónde parte el proyecto, es decir, qué tecnologías existen actualmente y se van a utilizar en la realización de este trabajo, a partir de las cuales se va a construir el resultado final. Se dará una explicación de las mismas y se detallarán los componentes más importantes que se han necesitado de cada una de ellas. Además, se hablará del resto de materiales que se han usado en la realización del proyecto.

2.1 Internet of Things (IoT)

El Internet de las Cosas, traducción al español de IoT, se basa en la conexión de elementos cotidianos de las personas a la red. Gracias a esta tecnología, se crean redes de dispositivos conectados a Internet, o a redes privadas, que permiten mejorar la calidad de vida de las personas, a través de la recolección de datos, su análisis y su posterior uso. Entre las grandes ventajas que aporta el IoT podemos mencionar la automatización de los procesos, además de permitir un análisis de datos óptimo y profundo. Por otro lado, permite un aprovechamiento de dispositivos y plataformas ya desarrolladas, lo que puede provocar un ahorro de presupuesto en el despliegue de soluciones. Los elementos esenciales que componen una red IoT son los sensores, encargados de recoger los datos, el servidor que guarda los datos en la nube, el agente IoT, encargado de establecer la comunicación entre el dispositivo y el servidor, y, lo más importante, los datos.

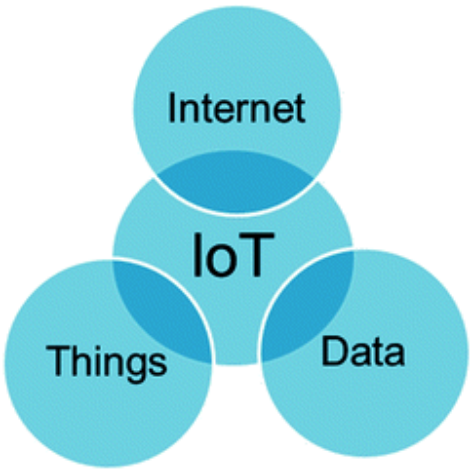

Figura 2. Definición de IoT en su forma más simple. Fuente: [4]

En un mundo en el que cada vez estamos más conectados, IoT representa la más reciente evolución de Internet y supone un avance enorme en la capacidad de recopilar datos y analizar información. El explosivo crecimiento de los smartphones y tablets elevó el número de dispositivos conectados a Internet a 12.500 millones en 2010, mientras que la población mundial llegó a los 6.800 millones, luego el ratio de dispositivos por persona subió a 1.84. [5] Actualmente se estima que para 2020 habrá 50.000 millones de dispositivos conectados, por una población mundial de 7.600 millones de personas, lo cual eleva el ratio anterior a 6.58 dispositivos conectados por persona, aproximadamente. Aún puede parecer que este número es algo bajo, pero esto es debido a que el cálculo está basado en toda la población mundial, parte de la cual aún no tiene conexión a Internet.

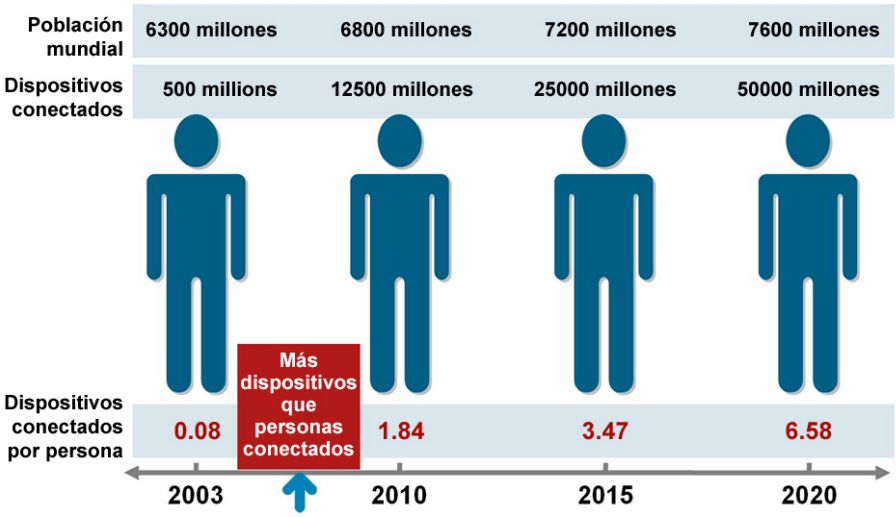

Figura 3. Evolución del número de dispositivos conectados por persona. Fuente: [5]

Para que el IoT alcance su pleno potencial, estos dispositivos deben ser autosuficientes y, por ello, gastar la mínima batería en la comunicación con los agentes IoT. Para ello utilizan distintos protocolos de baja energía, como MQTT o Bluetooth Low Energy, entre otros. Gracias a este bajo consumo, el IoT está extendido en todos los campos da la vida cotidiana, tanto en casas automatizadas, coches, pulseras electrónicas, smartwatches… Las empresas también son un ejemplo claro de usuarios, en la agricultura, el transporte… dando paso en muchos de estos casos al almacenamiento y análisis masivo de datos, o Big Data. Por supuesto, en la salud también se utiliza el IoT, generando una mayor calidad de vida en las personas, por ejemplo, en el rastreo remoto de pacientes o los sistemas de notificación de emergencias. Otros términos muy relacionados con IoT pueden ser "Smart Cities" y "Smart Buildings" dónde se utilizan dispositivos de IoT para mejorar el control del tráfico, el control de los suministros de agua y calefacción en un edificio, el control del transporte público, etc.

Aunque esta tecnología es muy completa y potente, existen algunos retos para el futuro. Uno de ellos es la seguridad. Los problemas más importantes y las áreas consideradas más vulnerables en relación con la seguridad de la red IoT son:

- El acceso a los datos en su transporte. La información está continuamente siendo enviada y puede ser intervenida en un ataque “man in the middle”, a no ser que los protocolos de transporte sean seguros y la información esté encriptada.
- La toma de control de los dispositivos IoT por parte de personas no autorizadas.
- El acceso a los datos una vez almacenados. Los datos son guardados en la nube y una persona no autorizada podría acceder a ellos o utilizar técnicas de suplantación de identidad.
- El robo de credenciales de la red.

2.2 FIWARE

FIWARE [3] es una iniciativa de código abierto que define un conjunto de estándares para la gestión de información de contexto y proporciona una serie de componentes con el objetivo de facilitar soluciones inteligentes para las SmartCities, además de empresas, salud… Estas soluciones se basan en recolectar información del entorno para la toma de decisiones automatizadas.

Figura 4. Logo de FIWARE.

El elemento principal que permite a FIWARE recopilar y administrar la información de contexto es el Orion Context Broker, un servidor web que permite peticiones de actualización o suscripción a los datos, utilizando una API de tipo REST sobre el protocolo de transporte HTTP. Este Context Broker puede acompañarse con otros elementos que proporcionan datos de distintas fuentes, tales como redes sociales, aplicaciones móviles o sensores IoT, dando soporte al procesamiento, análisis y virtualización de los datos. [6]

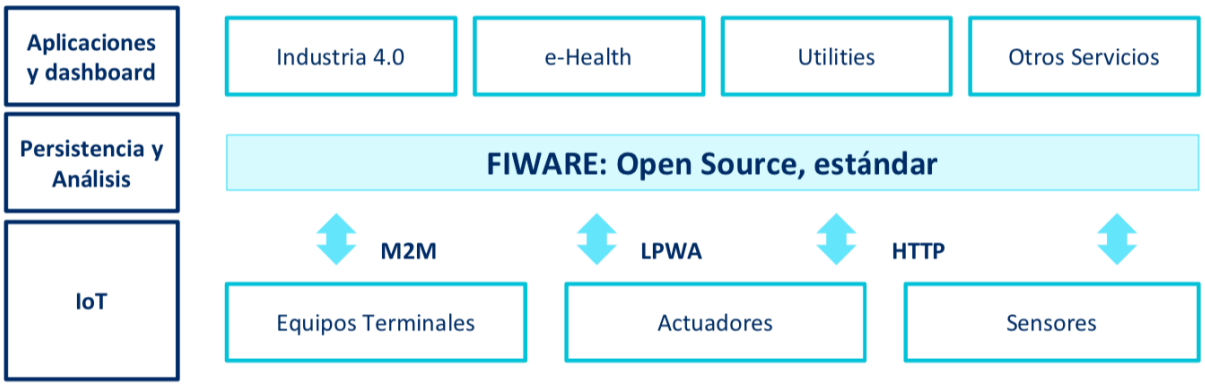

Figura 5. Elementos de un sistema IoT construido en torno a FIWARE. [7]

Los componentes de FIWARE están disponibles para su descarga y uso en la nube de FIWARE, un entorno donde los usuarios pueden probar la tecnología y sus aplicaciones, aprovechando los datos abiertos publicados por ciudades y otras organizaciones que ya usan esta plataforma. Al ser una plataforma basada en la nube, la capa de cloud hosting tiene una gran importancia. Es por ello que tanto la arquitectura como la implementación de la nube de FIWARE se basan en OpenStack, la plataforma de cloud de código libre líder actualmente. [8]

FIWARE permite a los desarrolladores reducir el tiempo dedicado a establecer la solución deseada y, a la vez, aumentar la modularidad, escalabilidad y flexibilidad del producto final. Esto es gracias a que se dispone de una API funcional que hace que los componentes de la red IoT sean fácilmente instanciados e interconectados.

2.2.1 Orion Context Broker (OCB)

El componente más importante de FIWARE y el único imprescindible y obligatorio en cualquier solución desarrollada con esta plataforma es el Orion Context Broker. Se trata de un servidor que permite administrar la información de contexto, consultarla y actualizarla, basándose en peticiones de una API tipo REST sobre el protocolo de transporte HTTP.

La información de contexto es publicada y actualizada por los productores de contexto, que son aquellos sensores, agentes IoT y demás dispositivos que registran y envían información a FIWARE para que otros puedan acceder a ella, los denominados consumidores de contexto, interesados en analizar y procesar estos datos.

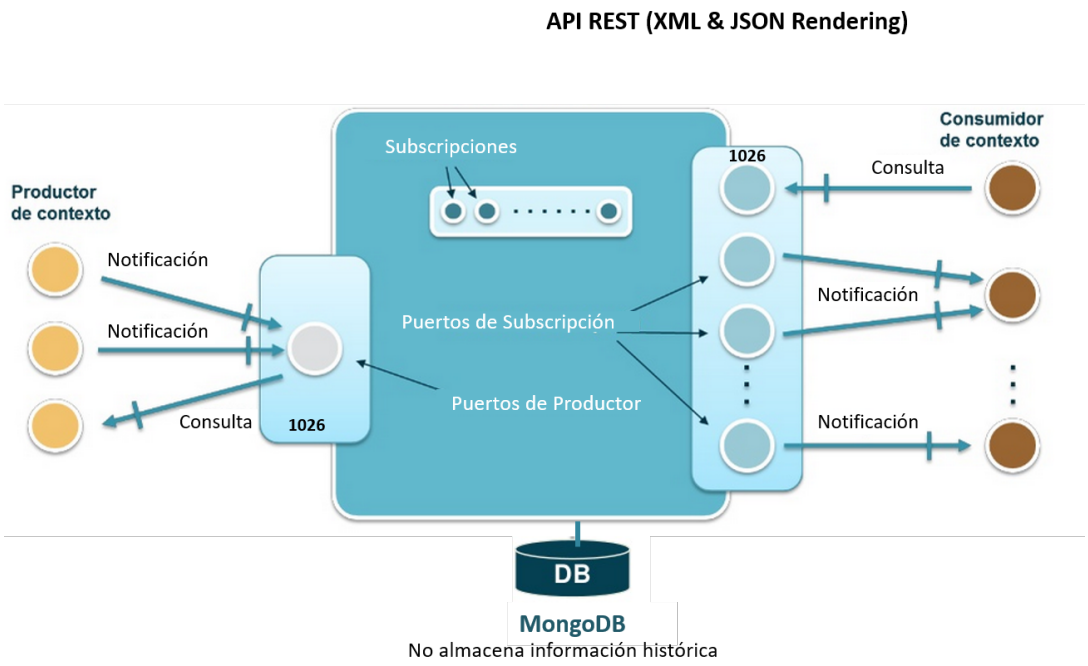

Figura 6. Estructura del Orion Context Broker. [9]

El OCB es un servidor que implementa una API que se basa en el modelo de información NGSI, por medio del cual se pueden realizar varias operaciones [9]:

- Registrar información de proveedores de contexto.
- Ser notificado ante una actualización de la información automáticamente, gracias a las suscripciones de los consumidores a los datos.
- Consultar información de contexto.

Orion se encuentra constantemente escuchando en un puerto, que por defecto es el 1026, y utiliza la base de datos MongoDB [10] para almacenar el estado actual de la información de las entidades, sin almacenar información histórica de sus cambios.

Un principio fundamental que soporta el Orion Context Broker es el de lograr una disociación total entre productores y consumidores de contexto. Esto quiere decir que los productores enviarán datos a la nube sin saber quién va a acceder a esa información, mientras que los consumidores no necesitan conocer quién ha subido la información que se ha publicado en un evento en concreto. Es por todo ello que los consumidores y los productores no necesitan estar conectados entre ellos, mas que tener acceso a la plataforma de FIWARE.

2.2.1 Modelo NGSI

La comunicación entre los distintos componentes de la red de FIWARE formada, con el Orion Context Broker en su núcleo, se lleva a cabo por el uso de una API RESTful NGSI. Es por ello que la información está representada en estructuras de datos genéricos referidos como elementos de contexto. Se tienen actualmente dos versiones de esta API que funcionan conjuntamente sin problemas: NGSIv1 y NGSIv2. La versión 2 de esta API incluye la posibilidad de incluir metadatos en los mensajes enviados a Orion. En este proyecto se ha utilizado la versión 1, pues no necesitamos la inclusión de metadatos, además de que es la más usada y extendida actualmente, en las peticiones de POST, mientras que en las de GET se ha usado la versión 2.

Un elemento de contexto se refiere a la información que es producida, recopilada o consultada y que puede ser relevante en su posterior procesamiento, análisis y extracción de nuevo conocimiento. Tiene asociado un valor definido que consiste en una secuencia de uno o más triplas que se refieren a atributos de un elemento de contexto. [6] Además, FIWARE permite crear entidades con estructuras de datos tanto básicas como complejas. Un elemento de contexto proporciona información relevante en torno a una entidad, cada una identificada exclusivamente por su id (EntityId) y su tipo (EntityType).

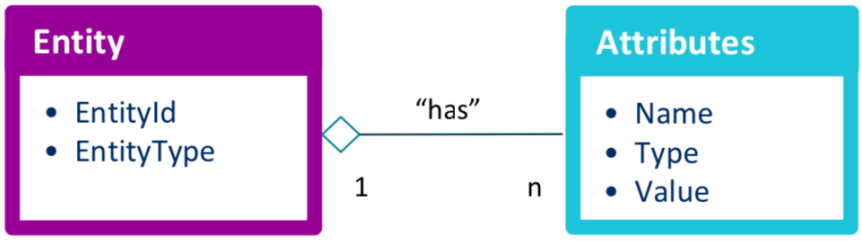

Figura 7. Diagrama de clases de una entidad de FIWARE. Fuente: [7]

La tecnología usada en los mensajes enviados al Context Broker es JSON, permitiendo la fácil creación de entidades con pares atributo-valor. Las respuestas ante las peticiones RESTful al servidor también son enviadas en código JSON con distintos valores según si la acción ha tenido éxito o ha ocurrido algún error.

Las operaciones estándar en NGSIv1 son las siguientes:

- UpdateContext: Se usa para mandar nueva información de contexto a Orion.
- QueryContext: Para solicitar datos al Context Broker.
- SubscribeContext: Para suscribirse a una fuente de información de contexto.
- UpdateContextSubscription: Para actualizar la suscripción a la fuente de información.
- UnsuscribeContext: Para desuscribirse de la misma.

En cuanto a NGSIv2, en este trabajo solo se usará para hacer las peticiones GET del servicio web a Orion, es decir, tan solo se usará la operación Entities y sus argumentos, que devuelve las entidades que coinciden con los argumentos dados.

2.3 e-Salud

La e-Salud es el término por el que es conocido el conjunto de tecnologías de la información y la comunicación que se emplean en el entorno sanitario a modo de herramientas en materia de prevención, diagnóstico, tratamiento, seguimiento, además de la gestión de la salud, permitiendo una mejora de la eficacia del mismo, así como un ahorro de costes al sistema sanitario.[11]

Como componentes que engloba este término encontramos diferentes modelos, productos y servicios tecnológicos aplicados al campo de la salud, tales como aplicaciones móviles, la telemedicina, el Big Data, los

sistemas de apoyo a la decisión clínica, el IoT y los dispositivos wearables, como la camiseta inteligente usada en este proyecto, entre otros.

Como principales características buscadas en el desarrollo de tecnologías de e-Salud encontramos:

- La consecución de una medicina personalizada y centrada en cada uno de los pacientes.
- La atención contínua a un paciente, permitiendo que pueda continuar con su vida en la normalidad.
- La extensión de las capacidades asistenciales más allá de las organizaciones sanitarias, siempre dentro de un ecosistema de elementos cooperantes de la salud.
- La búsqueda de nuevos escenarios asistenciales, que mejoren la asistencia al paciente.

2.4 Materiales

En este apartado se detallarán los elementos utilizados para la realización del proyecto, así como programas o dispositivos que hayan resultado de gran importancia en el trabajo.

2.4.1 Android Studio

Android Studio es el entorno de desarrollo oficial de aplicaciones de Android. La primera versión de este programa fue publicada por Google en 2014. Está disponible para las plataformas de Microsoft Windows, macOs y Linux. Ahora mismo, se encuentra en la versión 3.4.2 para macOs. Se trata del programa principal usado para el desarrollo de este proyecto, donde se ha realizado el agente IoT y la aplicación móvil que lo contiene.

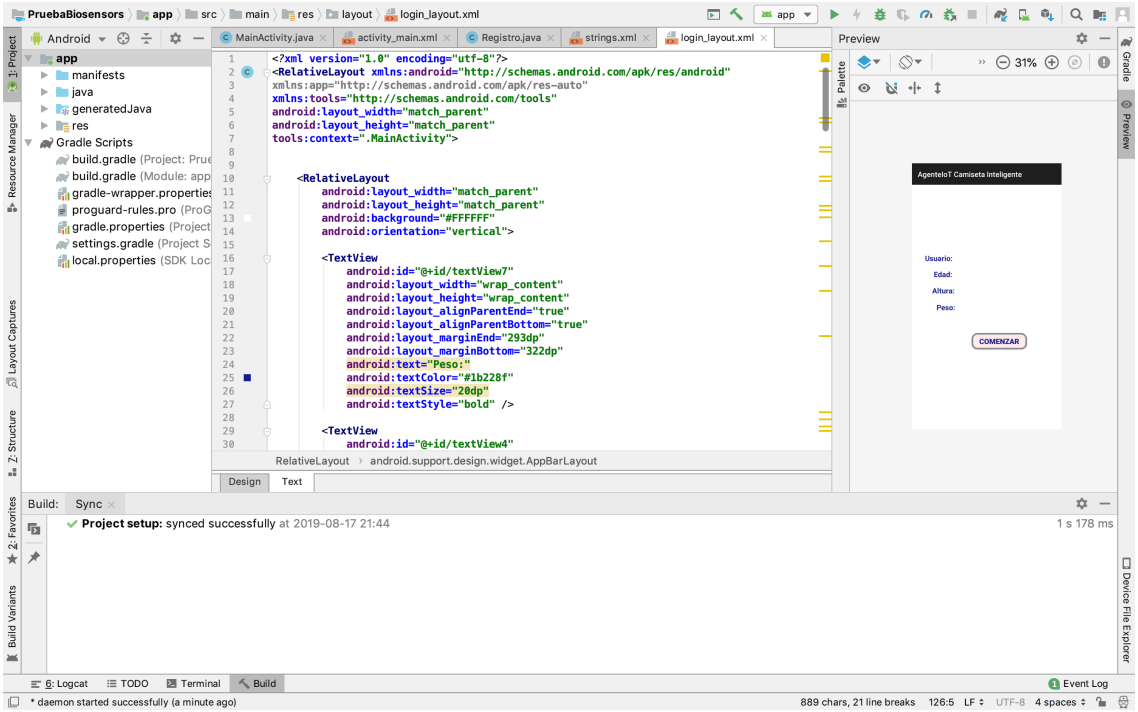

Figura 8. Android Studio en macOs.

Como características principales hasta la última versión tenemos las siguientes [12]:

- Integración de ProGuard y funciones de firma de aplicaciones.
- Mayor especificación a la hora de la programación.
- Renderizado en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción y estadísticas de uso.

- Soporte para construcción basada en Gradle.
- Refactorización específica de Android y arreglos rápidos.
- Editor de diseño enriquecido que permite arrastrar y soltar componentes en la interfaz de usuario.
- Herramientas Lint para detectar problemas de rendimiento y compatibilidad entre versiones, entre otros.
- Plantillas para crear diseños comunes de Android.
- Soporte para programar aplicaciones para Android Wear.
- Distintos dispositivos virtuales de Android con el objetivo de ejecutar y probar las aplicaciones desarrolladas.
- Herramienta de depuración integrada, para encontrar fallos en el programa.

Android Studio soporta todos los lenguajes de programación de IntelliJ, un entorno de desarrollo integrado para el desarrollo de programas informáticos basado en Java, tales como Java o C++. Además, a partir de la versión 3.0 de Android Studio, se soporta Kotlin, el lenguaje preferido por Google para el desarrollo de aplicaciones móviles, y todas las características de Java7 y Java8, así como algunas de Java9, aunque de momento no está claro la compatibilidad de Java12 en Android.

2.4.2 Camiseta Inteligente para el reconocimiento de actividad física

La camiseta inteligente utilizada en este trabajo es un dipositivo que recoge datos de la actividad física del usuario que la lleva puesta. Tiene como objetivo principal la monitorización del ejercicio del paciente, aplicando una tecnología precisa, simple y no obstructiva mediante un sensor capacitivo, integrado en la camiseta.

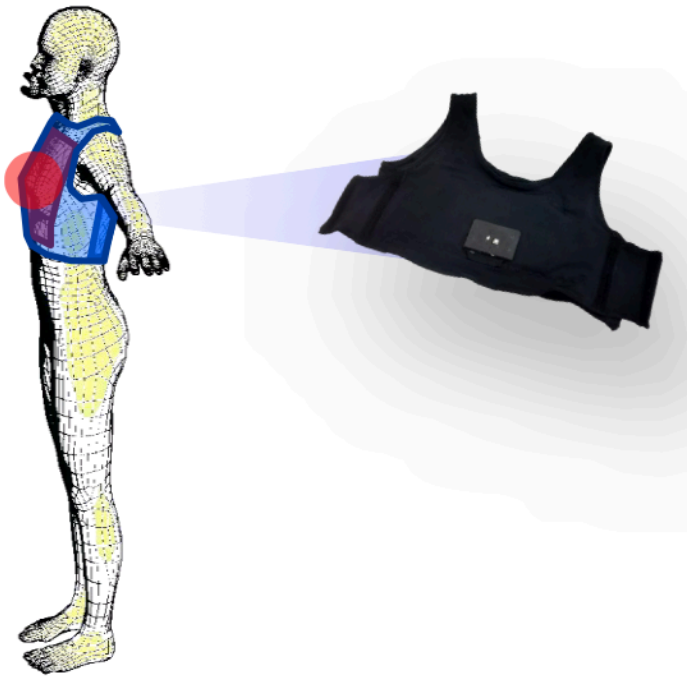

Figura 9. Camiseta inteligente.

Se compone de un sensor capacitivo, que actúa en la recolección y primer procesamiento de la información para la monitorización en tiempo real del ejercicio. Este sensor está integrado en un elástico ajustable colocado en el pecho del paciente, dando lugar a la camiseta. Ésta se comunica con una aplicación móvil, en principio, sin interfaz, mediante el protocolo ligero MQTT, estableciendo así una red de área corporal (WBSN) y permitiendo un bajo consumo de batería. [13]

La camiseta inteligente dispone de un diseño modular apoyado en los siguientes elementos:

- Unidad de sensor. Se trata del módulo que recoge la señal monitorizada. Está formada por un sistema de electrodos y de condicionamiento de la señal.
- Unidad de procesado. Se encarga del procesado de la señal para estimar la frecuencia respiratoria y los datos de actividad física.
- Unidad de comunicación. Es el módulo encargado de la comunicación con la aplicación móvil. Utiliza MQTT y Bluetooth.

La API de la que dispone el biosensor provee una capa de abstracción entre el terminal móvil y la camiseta inteligente, facilitando así la monitorización de la actividad física. La API está constituida por un conjunto de módulos funcionales que se encargan de la gestión de los eventos relacionados con la información recogida por el dispositivo. Estos módulos son los siguientes: [14]

- Módulo de monitorización respiratoria, que notifica periódicamente los datos asociados a la respiración del paciente.
- Módulo de actividad física, el cual notifica periódicamente la información asociada a la realización de actividad física del paciente en condiciones normales o realizando algún ejercicio en particular.

El proyecto se centra en el módulo de actividad física. Esta unidad tiene tres modos de funcionamiento, modo IDLE o inactivo, NORMAL y EXERCISE, en el cual se está ejecutando un ejercicio concreto. La periodicidad con la que se reciben los datos viene dada por un periodo de muestro.

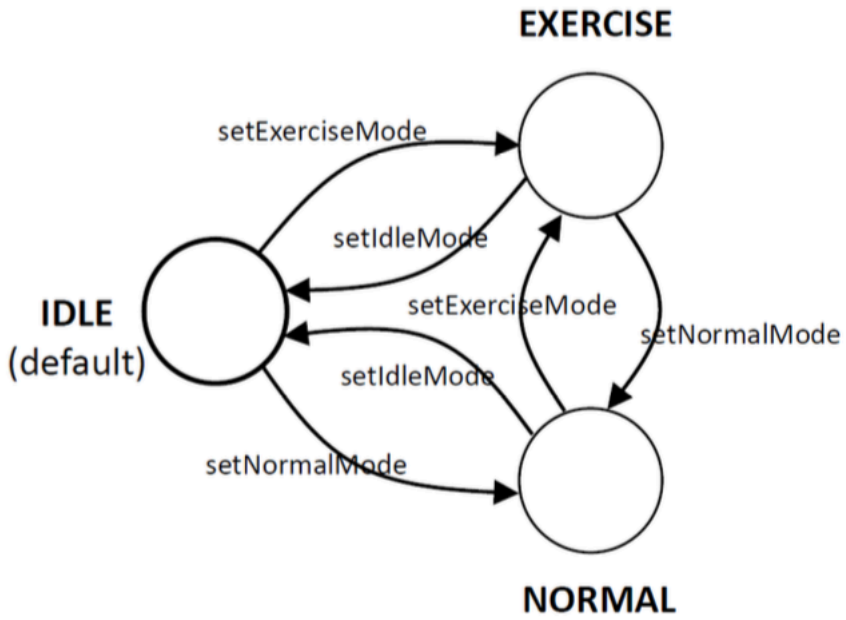

Figura 10. Modos de funcionamiento del módulo de actividad física de la camiseta y los respectivos métodos para pasar de uno a otro.

2.4.3 Postman Echo

Postman Echo es un servicio que permite probar un cliente REST realizando simples peticiones de la API, todo ello a través de un entorno sencillo y muy visual. Proporciona diferentes tipos de peticiones (GET, POST, PUT, entre otras), varios mecanismos de autenticación y otro tipo de utilidades. [15] Además, ofrece un apartado visual para las peticiones y las respuestas dadas por el servidor, así como las cabeceras que van junto con el mensaje.

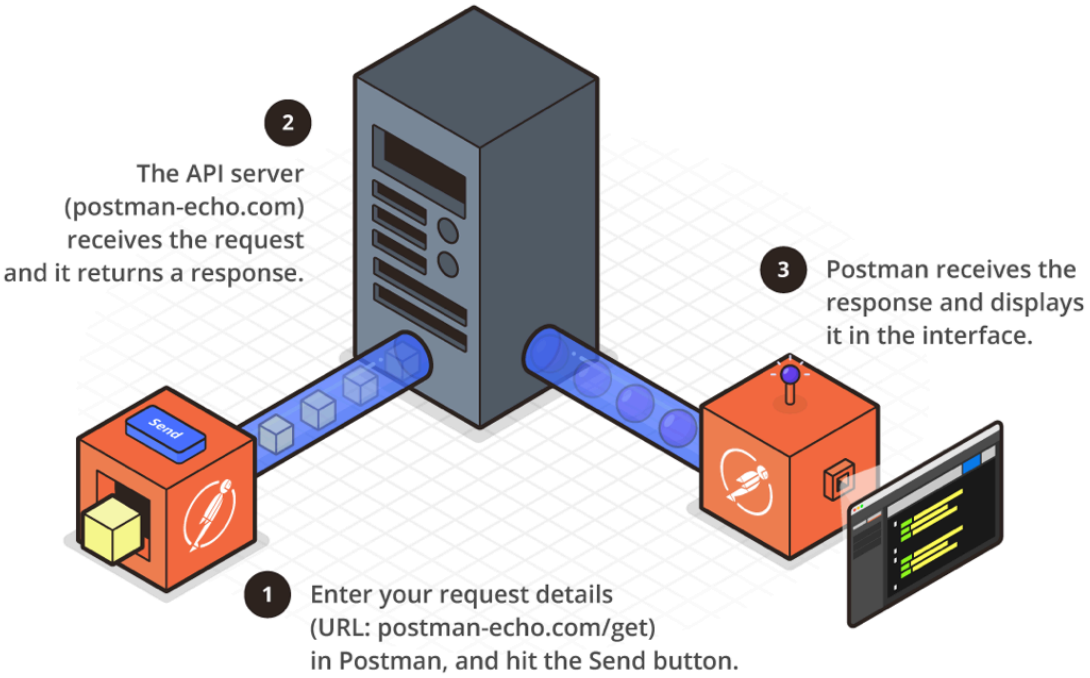

Figura 11. Funcionamiento de Postman Echo para una supuesta petición al servidor de postman-echo.com. Fuente: [15]

El funcionamiento de Postman Echo es el siguiente. Primero, se debe escribir la dirección web de la API a la que se desea acceder y el tipo de petición que se va a realizar. También, es importante incluir las cabeceras necesarias y el contenido del mensaje necesario para la petición de información. Una vez realizado esto, se envía la petición al servidor de la API, el cuál la recibe y devuelve una respuesta al usuario. Postman Echo recibe la respuesta y la muestra al usuario en su interfaz, permitiendo que la interacción sea simple y rápida.

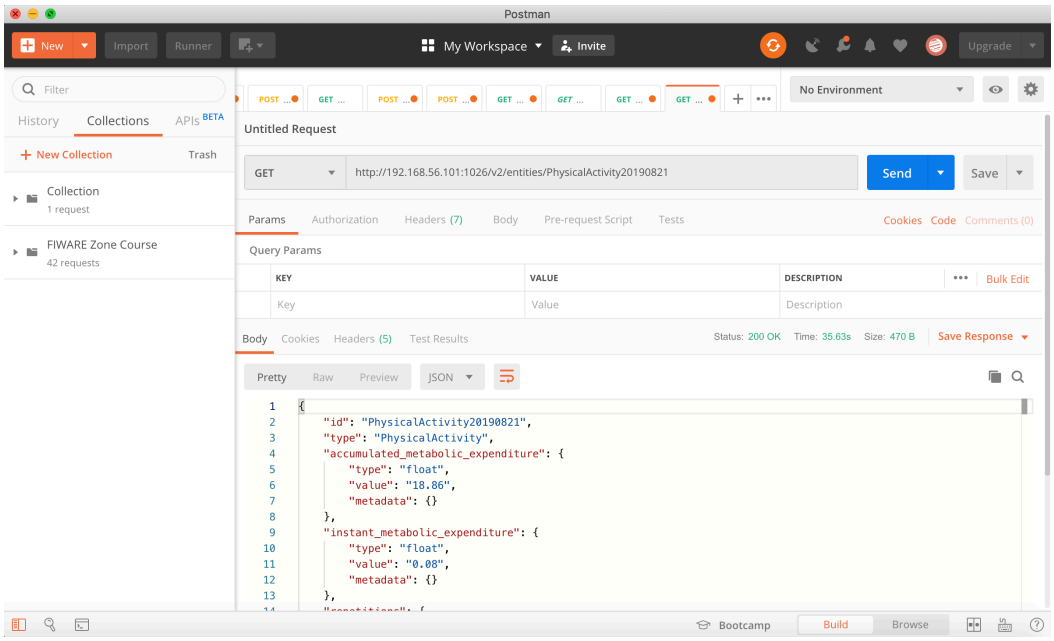

Figura 12. Interfaz de Postman Echo tras una petición GET hecha a Orion Context Broker.

2.4.4 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft y basado en Electron, un framework que se utiliza para implementar aplicaciones Node.js. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. [16] También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto.

Este editor es compatible con varios lenguajes de programación, pues incluye soporte para JavaScript, TypeScript y Node.js, además de tener un gran número de extensiones para otros lenguajes tales como C++, C#, Java, Python, PHP, Unity… También ofrece una interfaz al usuario bastante interesante, pues tenemos la posibilidad de abrir varias terminales en él y de ver a la izquierda de la pantalla todos los ficheros que componen el proyecto.

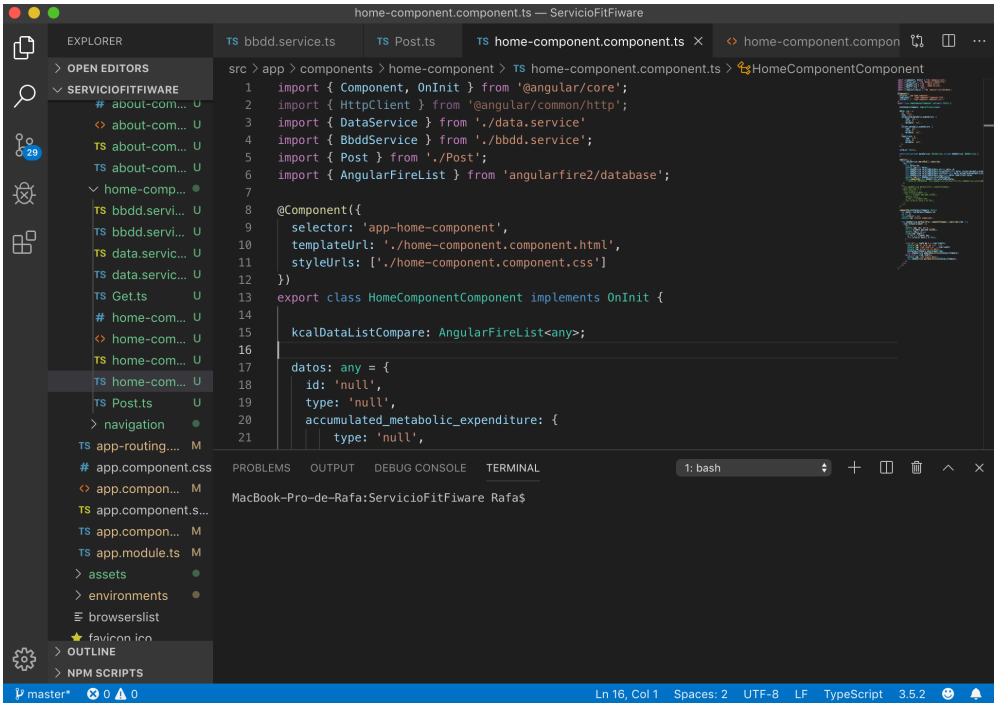

Figura 13. Interfaz de Visual Studio Code.

2.5 Tecnologías usadas para el Servicio Web

2.5.1 Angular

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Tiene como objetivo aumentar las aplicaciones basadas en navegador, implementando el Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.[17] La versión utilizada en este proyecto es la de Angular 7.

Figura 14. Logo de Angular.

La biblioteca obedece a las directivas de los atributos personalizados al leer HTML que los contiene y une las piezas de entrada o de salida de la página a un modelo representado por las variables estándar de JavaScript.

Está basado en clases de tipo Componentes, cuyas propiedades son las usadas para hacer el binding de los datos, permitiendo el routing, es decir, el acceso a un sistema de navegación en nuestra aplicación.

Algunas de las principales características de Angular son las siguientes:

- Velocidad y rendimiento. Angular convierte las plantillas en código altamente optimizado para las máquinas virtuales de JavaScript actuales, es universal, es decir, ejecuta la primera vista de aplicación en Node.js, PHP y otros servidores para renderizado, obteniendo así de forma casi instantánea solo HTML y CSS; y permite la división automática de códigos, con el objetivo de que los usuarios solo carguen el código necesario para el procesamiento de la vista solicitada en cada momento.
- Productividad. Permite la fácil creación de vistas gracias a las plantillas simples y potentes que ofrece. También Angular CLI proporciona las herramientas de línea de comandos que ofrecen una forma fácil de crear componentes y servicios, así como obtener una previsualización del proyecto web y realizar tests de prueba. Por supuesto, también ofrece sugerencias de código inteligente, detección y corrección rápida de errores.
- Historia completa del desarrollo. Permite un intuitivo seguimiento de los proyectos gracias a Git, además de la posibilidad de realizar pruebas unitarias con Karma y pruebas end-to-end con Protractor de forma rápida y estable.

2.5.1.1 TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft, que cuenta con herramientas de programación orientada a objetos, lo cual facilita el desarrollo de grandes proyectos. [18] Puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js), pues extiende la sintaxis de JavaScript. Por lo tanto, cualquier código escrito en éste debe funcionar sin problemas.

Figura 15. Logo de TypeScript.

El compilador de TypeScript está escrito asimismo en TypeScript, compilado a JavaScript y con Licencia Apache 2.

Los ficheros de definición que contienen información sobre los tipos de librerías JavaScript existentes son también soportados por TypeScript. Esto permite a otros programas usar los valores definidos en los ficheros como si fueran entidades TypeScript de tipado estático. Poder definir los tipos durante el tiempo de diseño ayuda a evitar errores en tiempo de ejecución, como pasar el tipo de variable incorrecto a una función.

Los tipos básicos admitidos por TypeScript son:

- String: Define una cadena de caracteres.
- Number: Tipo de dato que representa un número.
- Boolean: Representa un tipo de dato de verdadero o falso.
- Array: Es un tipo de dato estructurado que almacena una colección de elementos.
- Tuple: Se trata de un tipo similar al Array, pero con un número fijo de elementos.
- Enum: Es una enumeración de elementos.
- Any: Se trata de una variable que puede ser de cualquier tipo. Es útil cuando no se sabe que variable va a devolver una librería externa.
- Void: Indica que la función no devuelve nada.
- Never: Representa el tipo de valores que nunca se producen.

2.5.1.2 HTML

HTML es el estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones. Define una estructura básica y un código HTML para la definición de contenido de una página web, como texto, imágenes, videos o juegos. Está estandarizado por el World Wide Web Consortium (W3C), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Además, se trata del estándar que se ha impuesto en la visualización de páginas web y el que todos los navegadores actuales han adoptado. [19]

Fue inventado en 1991 por Tim Berners-Lee, incluyendo en su diseño inicial la descripción de 18 elementos fundamentales de HTML, de los cuales 13 aun existen en HTML4, pero no fue hasta 1993 que la IETF lo reconoció formalmente, mediante la publicación de una proposición para una especificación de HTML. Actualmente, existe la versión HTML5 publicada en 2014.

El desarrollo de aplicaciones web mediante el lenguaje HTML se basa en la diferenciación. Para añadir un elemento externo a la página se debe hacer una referencia a la ubicación de dicho elemento mediante texto, no incrustarse directamente en ella. Así se consigue que la página web contenga solamente texto, mientras que la tarea de unir los elementos e interpretar el código para visualizar la página final recae sobre el navegador web utilizado.

Al ser un estándar, HTML tiene como objetivo ser un lenguaje que permita que cualquier web pueda ser interpretada por cualquier navegador de la misma forma, independientemente de su versión. Sin embargo, a lo largo de sus distintas versiones, se han incluido y suprimido multitud de características con el fin de hacer más eficiente su interpretación en los distintos dispositivos y plataformas. Esto obliga a los desarrolladores a producir parches que los navegadores deben incorporar en sus nuevas versiones. Es por ello que los navegadores deben estar actualizados para interpretar correctamente las páginas web, lo que obliga a los desarrolladores a tener que aplicar técnicas y cambios en la visualización de los mismos, así como a corregir problemas de visualización e incluso de interpretación del código. Por estos motivos, aún existen diferencias al interpretar una misma página web dependiendo del navegador que se esté utilizando.

HTML se basa en una estructura compuesta por elementos, que tienen dos propiedades principales: atributos y contenido, cada una de ellas con ciertas restricciones para que el documento HTML se muestre válido. Cada elemento tiene normalmente una etiqueta de inicio y otra de cierre, encontrándose entre ellas el contenido, aunque hay algunos elementos tales como
 (salto de línea), que no necesitan contenido ni etiqueta de cierre. También, este lenguaje propone un marcado estructural que, aunque no define cómo se mostrarán los elementos, la mayoría de navegadores web lo han estandarizado así.

En cuanto a los atributos, la mayoría de ellos son pares nombre-valor separados por un signo de igual "=" y escritos en la etiqueta de comienzo de un elemento, justo después del nombre del mismo. Normlamente, los valores se encuentran entre comillas dobles o simples, exceptuando algunos tipos.

Las etiquetas HTML básicas son:[20]

- <html>: Define el nicio del documento e indica al navegador que lo que sigue a esa etiqueta será código HTML.
- <script>: Permite la inclusión de un script en la web.
- <head>: Define la cabecera del documento HTML. Dentro de ella podemos encontrar: <title>, <style>, <link>, entre otras.
- <body>: Define el contenido principal o cuerpo del documento. Dentro de ella es posible encontrar numerosas etiquetas, tales como <h1> a <h6>, que representan encabezados de título con distinta relevancia, <table>, para usar tablas, , para hacer una división a la página, útil para usarla conjuntamente con CSS, o <img>, para incluir una imagen en la web, entre otras.

```
<br>
<h3> Este servicio web accede al <b>Orion Context Broker (OCB)</b> de la plataforma <b>FIWARE</b> para mostrar los datos
  de actividad física de un paciente almacenados. En primer lugar, se debe introducir el nombre del
  mismo en el formulario, para obtener sólo la información de esa persona. Una vez hecho esto, el servicio
  realiza peticiones GET al OCB, usando la API de FIWARE en su versión <b>NGSIv2</b>, para obtener los datos de la
  actividad física de ese paciente. <br><br>

  Esta información ha sido enviada al OCB por un <b>Agente IoT</b> desarrollado en una aplicación móvil de Android, que
  actúa de intermediario entre la <b>camiseta inteligente</b>, desarrollada por el Grupo de Ingeniería Biomédica de
  la ETSI, y FIWARE.<br><br>

  Además, el servicio web muestra los datos datos de la entidad Person del paciente y se comunica con una
  Realtime Database de <b>Firebase</b> que almacena los datos obtenidos del Context Broker y permite obtener
  una tabla del progreso reciente en la actividad física del paciente.<br><br>

  Se ha realizado con las tecnologías de <b>Angular7</b> y <b>Bootstrap4</b>.<br><br>
  ©2019, Rafael Díaz Fernández
</h3>
<div align= "center">
    <img src="../../assets/images/PoweredByFiware.png" width="200" height="200">
</div>
```

Figura 16. Código escrito en HTML.

Este lenguaje puede ser creado y editado con cualquier editor de texto, tanto Gedit, Emacs, Sublime Text, Visual Studio Code, como con aquellos básicos, como pueden ser el Bloc de Notas de Windows, Wordpad, Notepad++, entre otros.

2.5.2 Bootstrap

Se trata de una biblioteca multiplataforma que define un conjunto de herramientas de código abierto para el diseño de servicios y aplicaciones web, proporcionando un apartado visual atractivo al usuario. Contiene plantillas de diseño con tipografía, formularios, tablas, botones y otros elementos basados en HTML, CSS y, algunos, en JavaScript. Bootstrap hace que el diseño de la web sea compatible con distintos dispositivos, sin tener que hacer grandes cambios. La versión utilizada en este proyecto es Bootstrap4.[21]

Figura 17. Logo de Bootstrap.

Fue creado en 2011 por dos trabajadores de Twitter, Mark Otto y Jacob Thornton, como un marco de trabajo para el desarrollo de interfaces de usuario. En 2012, se convirtió en el proyecto de desarrollo más popular de Github.

Bootstrap es modular y consiste principalmente en una serie de hojas de estilo LESS, que implementan una gran variedad de componentes para la página web. Los desarrolladores de las aplicaciones web pueden utilizar el mismo archivo de Bootstrap seleccionando los componentes que deseen.

Estos componentes pueden ser modificados mediante cambios en las declaraciones LESS, de otra forma los ajustes están limitados por una hoja de estilo de configuración central. La configuración de Bootstrap posee también una opción de personalizar, mientras que los desarrolladores eligen los componentes deseados en un formulario y, seguidamente, el paquete generado ya incluirá la hoja de estilo CSS pre-compilada.

Una característica importante de Bootstrap es que estos componentes son reutilizables, incluyendo botones con características avanzadas, etiquetas, formatos de mensajes de alerta y barras de progreso, entre otros. Además, los componentes de JavaScript para Bootstrap están basados en la librería jQuery de JavaScript. Esto hace que sus plugins estén disponibles para proveer elementos adicionales a la interfaz del usuario, tales como diálogos, y que extiendan la funcionalidad de otros elementos ya existentes, como por ejemplo la función de autocompletar para los formularios.

Como añadido a esta herramienta, debe conocerse que existen plataformas en las que se proponen temas de código libre gratis para Bootstrap, tales como Bootswatch [22]. Esta plataforma ofrece temas para las aplicaciones web de fácil instalación, pues sólo tenemos que descargar el archivo CSS y reemplazarlo por el original de Bootstrap. Bootswatch permite una mayor personalización del diseño, asegurando una mejor compatibilidad con la página web.

2.5.3 Firebase

Se trata de una plataforma ubicada en la nube para el desarrollo de aplicaciones web y móviles, que ofrece un conjunto de herramientas para el desarrollo y sincronización de proyectos. Al estar integrada con Google Cloud Platform, permite que los proyectos realizados gocen de una mayor calidad, fomentando el número de usuarios de las aplicaciones creadas y la monetización de cada una de ellas. Fue creada en 2012 por James Tamplin y Andrew Lee y posteriormente adquirida por Google en 2014. [23]

Como ventajas obtenidas con el uso de Firebase tenemos:

- APIs intuitivas y de fácil uso para el usuario, junto con una gran documentación para la creación de aplicaciones, que mejoran la experiencia de los desarrolladores.[24] Además, ofrece soporte gratuito y sus desarrolladores participan de forma activa en Github o StackOverflow.
- Sincronización entre plataformas, haciendo uso de herramientas multiplataforma, permitiendo trabajar con iOS, Web o Android.
- Integración con la estructura de Google, escalando automáticamente para diferentes tipos de aplicaciones.
- No existe la necesidad del uso de un servidor para el proyecto, pues Firebase utiliza herramientas ya incluidas en los SDK de Android, iOS y demás.

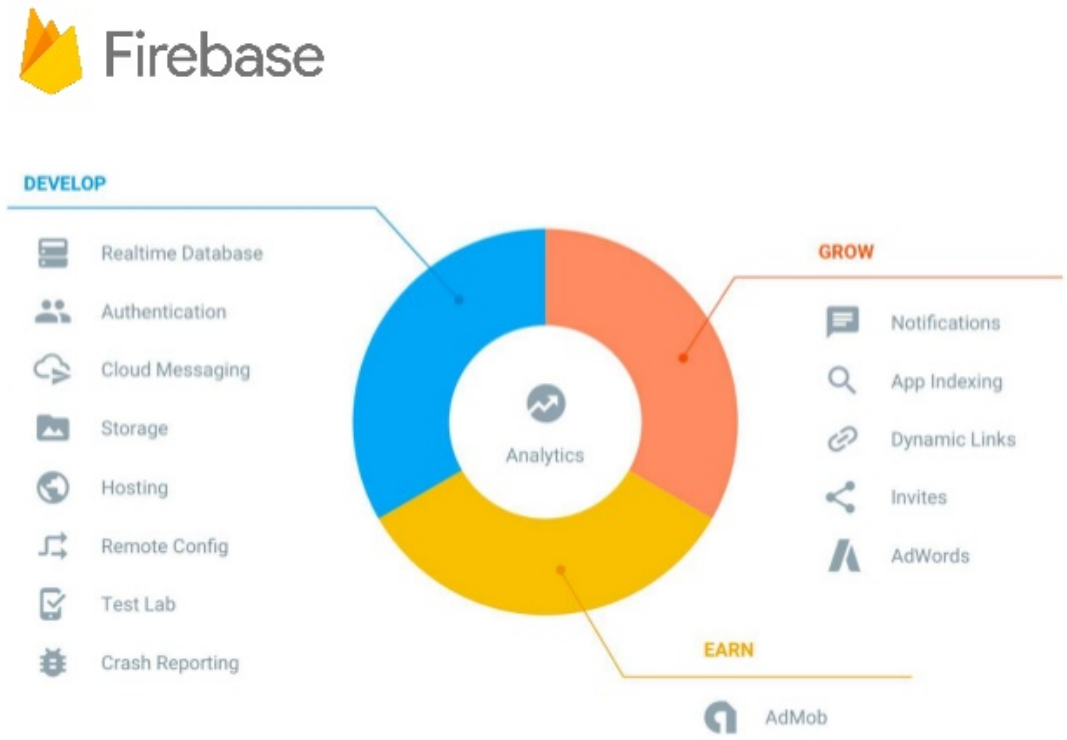

Figura 18. Servicios ofrecidos por Firebase. Fuente: [25]

Firebase ofrece multitud de servicios a los usuarios, tanto de análisis como de desarrollo. Además, dispone de funciones como la detección de errores y el testeo, aportando una reducción del tiempo de desarrollo. Entre ellas podemos encontrar las siguientes:

- Firebase Analytics. Se trata de una aplicación que aporta una visión más profunda del uso que hacen los usuarios del servicio o aplicación.
- Firebase Cloud Messaging. Es una plataforma para mensajes y notificaciones de Android, iOS y aplicaciones web.
- Firebase Auth. Se trata de un servicio para autenticar a los usuarios que hagan uso de las aplicaciones desarrolladas, utilizando únicamente código del lado del cliente. Ofrece autenticaciones mediante distintos métodos, Facebook, Google, Twitter o Github, entre ellos. También, dispone de un sistema de administración de usuarios y contraseñas. Gracias a este servicio, el desarrollador no tiene que preocuparse de preparar métodos de autenticación en sus aplicaciones, sino que tan solo debe incorporar este servicio de forma sencilla, asegurándose eficacia y seguridad.[26]
- Firebase Storage. Proporciona capacidad de almacenamiento, con la posibilidad de subir y bajar archivos de forma segura.
- Firebase Firestore. Es un servicio de base de datos NoSQL en la nube, que organiza los archivos almacenados en colecciones y subcolecciones.
- Firebase Realtime Database. Es el servicio de Firebase utilizado en este proyecto. Realtime Database es una base de datos NoSQL, de tiempo real, back-end y organizada en forma de árbol JSON. Proporciona una API que permite que la información sea almacenada y sincronizada en la base de datos y que los usuarios puedan también consumir esa información en sus aplicaciones. [27]

  Esta base de datos es también accesible desde una API de tipo REST, permitiendo una muy buena integración con varios sistemas de JavaScript, tales como Angular.

  Una ventaja del uso de este servicio es la sincronización en tiempo real de la información almacenada,

haciendo posible el acceso simultáneo de los usuarios en distintos dispositivos o plataformas, pudiendo modificar el contenido de ésta. Todo ello es posible compartiendo una instancia de la misma Realtime Database entre los usuarios, los cuales son notificados al momento de haberse realizado algún cambio en la misma.

Además, ofrece una solución muy interesante ante las pérdidas de conexión con la base de datos. Al perderse la conexión, el SDK de Realtime Database guarda en una caché local los cambios realizados por el usuario, realizando la subida de la información una vez se haya recuperado la conexión a Internet.

3 RESULTADOS

Los resultados que consigues serán directamente proporcionales al esfuerzo que aplicas.

- Denis Waitley -

EN esta sección del documento se van a conocer los requisitos iniciales del proyecto y el planteamiento inicial de las características a implementar. Se hablará del diseño de mapeo realizado, haciendo énfasis en los mensajes enviados al Orion Context Broker y en la creación y reutilización de entidades para el envío de información de contexto.

Además, se verá todo el proceso de desarrollo del proyecto, tanto de la aplicación móvil como del servicio web, explicando cómo se ha llevado a cabo cada uno y cuáles son las partes que lo componen. Se detallará también el funcionamiento del código y aquellos cambios realizados para solucionar los errores que han ocurrido durante la realización del trabajo. Por último, se hablará de las pruebas realizadas y la validación del proyecto, y ser verá un ejemplo de ejecución de la solución completa desarrollada.

3.1 Especificación de requisitos

Aquí se describirán los requisitos que se han tenido en cuenta para la elaboración del proyecto, en una división entre requisitos generales, funcionales y no funcionales, y sus subtipos.

3.1.1 Requisitos generales

Se trata de aquellos requisitos que describen las características u objetivos principales del proyecto.

ID: Req_Gen_001	Conexión del Agente IoT con la camiseta inteligente
Versión	1.0
Descripción	La aplicación de Android del Agente IoT debe de establecer una conexión con la camiseta inteligente y recibir los mensajes de los datos recolectados por ella.
Importancia	Muy alta
Prioridad	Muy alta
Estado	Analizado.
Comentarios	-

Tabla 9. Requisito general número 1.

ID: Req_Gen_002	Conexión del Agente IoT con FIWARE
Versión	1.0
Descripción	La aplicación del agente IoT debe de establecer una conexión con el Orion Context Broker de FIWARE y enviarle los mensajes construidos a partir de los datos recibidos.
Importancia	Muy alta
Prioridad	Muy alta
Estado	Analizado.
Comentarios	Es el principal requisito del sistema.

Tabla 10. Requisito general número 2.

ID: Req_Gen_003	Conexión del servicio web con FIWARE
Versión	1.0
Descripción	El servicio web debe establecer una conexión con el Orion Context Broker de FIWARE y disponer de acceso a la información que almacena.
Importancia	Alta
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 11. Requisito general número 3.

ID: Req_Gen_004	Conexión del servicio web con Firebase
Versión	1.0
Descripción	El servicio web debe establecer una conexión con la Realtime Database de Firebase con el objetivo de actualizarla y disponer de acceso a la información almacenada.
Importancia	Alta
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 12. Requisito general número 4.

ID: Req_Gen_005	Muestra de mensajes enviados a FIWARE
Versión	1.0
Descripción	La aplicación móvil del Agente IoT debe de mostrar por pantalla los mensajes enviados al Context Broker.
Importancia	Media
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 13. Requisito general número 5.

ID: Req_Gen_006	Muestra de progreso registrado en el Servicio Web
Versión	1.0
Descripción	El Servicio Web debe mostrar el progreso en los datos de actividad física recogidos.
Importancia	Media
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 14. Requisito general número 6.

3.1.2 Requisitos funcionales

Son aquellos requisitos que están relacionados con el comportamiento del sistema o con la función que se desempeña.

3.1.2.1 Requisitos de información

Apoyando la especificación de requisitos de información del sistema, se procede a especificar qué información debe almacenar el sistema.

ID: Req_Inf_001	Almacenamiento de información de actividad física
Versión	1.0
Descripción	El sistema debe guardar la información de actividad física en el Orion Context Broker.
Importancia	Muy alta
Prioridad	Muy alta

Estado	Analizado.
Comentarios	-

Tabla 15. Requisito de información número 1.

ID: Req_Inf_002	Almacenamiento de los datos de progreso de actividad física
Versión	1.0
Descripción	Al usar el Servicio Web se debe almacenar la información disponible en el Orion Context Broker en una base de datos.
Importancia	Media
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 16. Requisito de información número 2.

3.1.2.2 Requisitos de reglas de negocio

En este subapartado se especifica qué reglas de negocio debe respetar el sistema, evitando que se incumplan durante su funcionamiento.

ID: Req_Neg_001	Uso de API NGSI de FIWARE
Versión	1.0
Descripción	Deberá de utilizarse la API ya proporcionada por FIWARE para las tareas de envío y recepción de datos en la comunicación con el Context Broker.
Importancia	Alta
Prioridad	Alta
Estado	Analizado.
Comentarios	Puede utilizarse indistintamente tanto NGSIv1 como NGSIv2.

Tabla 17. Requisito de reglas de negocio número 1.

ID: Req_Neg_002	Uso de API de Firebase
Versión	1.0
Descripción	Deberá de utilizarse la API ya proporcionada por Firebase para las tareas de envío y recepción de datos en la comunicación con la Realtime Database.
Importancia	Alta
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 18. Requisito de reglas de negocio número 2.

ID: Req_Neg_003	Uso de API propia de la camiseta inteligente
Versión	1.0
Descripción	Deberá de utilizarse la API ya proporcionada por el equipo de desarrollo de la camiseta inteligente para el envío de mensajes entr el Agente IoT y la camiseta.
Importancia	Alta
Prioridad	Alta
Estado	Analizado.
Comentarios	-

Tabla 19. Requisito de reglas de negocio número 3.

3.1.3 Requisitos no funcionales

A continuación, se procede a especificar las condiciones que se le imponen al sistema relacionadas principalmente con aspectos de calidad, algunas de las cuales influyen en la arquitectura del sistema.

3.1.3.1 Requisitos de fiabilidad

ID: Req_Fia_001	Máximo tiempo de inactividad continuada del agente IoT 10 minutos
Versión	1.0
Descripción	Las actualizaciones en la aplicación para mejorar los servicios ofrecidos a los usuarios no deben afectar a la calidad de servicio percibida por el usuario, por lo cual queda acotado el tiempo de inactividad, que no podrá superarse en el 99% de los casos.
Importancia	Alta

Prioridad	Media
Estado	Analizado.
Comentarios	-

Tabla 20. Requisito de fiabilidad número 1.

ID: Req_Fia_002	Máximo tiempo de inactividad continuada del servicio web 10 minutos
Versión	1.0
Descripción	Las actualizaciones en la aplicación para mejorar los servicios ofrecidos a los usuarios no deben afectar a la calidad de servicio percibida por el usuario, por lo cual queda acotado el tiempo de inactividad, que no podrá superarse en el 99% de los casos.
Importancia	Alta
Prioridad	Media
Estado	Analizado.
Comentarios	-

Tabla 21. Requisito de fiabilidad número 2.

3.1.3.2 Requisitos de eficiencia

ID: Req_Efi_001	Tiempo medio de respuesta inferior a 500ms.
Versión	1.0
Descripción	El usuario debe recibir por pantalla los mensajes enviados por la camiseta con un retraso medio inferior a 500ms.
Importancia	Alta
Prioridad	Media
Estado	Analizado.
Comentarios	-

Tabla 22. Requisito de eficiencia número 1.

ID: Req_Efi_002	Tiempo máximo de respuesta inferior a 1s.
Versión	1.0
Descripción	El usuario debe recibir por pantalla los mensajes enviados por la camiseta en un tiempo inferior a 1s.

Importancia	Alta
Prioridad	Media
Estado	Analizado.
Comentarios	-

Tabla 23. Requisito de eficiencia número 2.

3.1.3.3 Requisitos de portabilidad

ID: Req_Por_001	Compatibilidad con las versiones inferiores de Android
Versión	1.0
Descripción	La aplicación móvil del Agente IoT debe ser compatible con las versiones de Android 6.x (Marshmallow) y superiores.
Importancia	Media
Prioridad	Media
Estado	Analizado.
Comentarios	-

Tabla 24. Requisito de portabilidad número 1.

ID: Req_Por_002	Compatibilidad con las últimas versiones de los navegadores Google Chrome, Mozilla Firefox y Safari.
Versión	1.0
Descripción	El servicio web debe mantener la compatibilidad con las últimas versiones de los navegadores Google Chrome, Mozilla Firefox y Safari.
Importancia	Media
Prioridad	Media
Estado	Analizado.
Comentarios	Esto es conseguido al usar Angular, pues se encarga de que sea compatible con la última versión de los navegadores.

Tabla 25. Requisito de portabilidad número 2.

3.1.3.4 Requisitos de seguridad

ID: Req_Seg_001	Acceso por contraseña a base de datos
Versión	1.0
Descripción	Se pretende que la base de datos no sea accesible por nadie más que el desarrollador, es por ello que se propone el uso de un usuario y una contraseña para el acceso a la misma.
Importancia	Media
Prioridad	Media
Estado	Analizado.
Comentarios	Esto se consigue al usar Firebase, pues para acceder a la Realtime Database hace falta autenticarse mediante el usuario de Google.

Tabla 26. Requisito de seguridad número 1.

3.1.3.5 Requisitos de integración del sistema

ID: Req_Int_001	Integración de base de datos en servicio web
Versión	1.0
Descripción	La base de datos debe de estar correctamente integrada en el servicio web, con el objetivo de que se pueda guardar y consumir información de ella.
Importancia	Alta
Prioridad	Media
Estado	Analizado.
Comentarios	-

Tabla 27. Requisito de integración número 1.

3.2 Diseño del mapeo

Este apartado define el diseño realizado en cuanto a la definición y reutilización de entidades para su envío al Orion Context Broker de FIWARE, identificando los elementos de cada mensaje a enviar, así como la estructura de los mensajes enviados, en la versión NGSIv1 utilizada de la API.

3.2.1 Mensajes para Orion Context Broker

Los mensajes enviados al Context Broker son entidades de datos, cuya creación es una de las bases de las soluciones desarrolladas con FIWARE. Las entidades contienen la información de contexto a enviar a Orion, y

debe de seguir una estructura determinada dependiendo de la API escogida. En nuestro caso, para hacer actualizaciones en el Context Broker, se ha usado NGSIv1.

La estructura de las entidades a enviar de la versión 1 de NGSI se compone de tres campos iniciales:

- "type" es el modelo de entidad a enviar. Define qué tipo de información se va a describir después y no es único.
- "isPattern" describe si el usuario quiere que el ID escrito sea una expresión regular con la que luego poder obtener muchos elementos.
- "id", que identifica la entidad creada. Es único.

Seguidamente, se deben especificar los atributos que se le quiere dar a la entidad. Cada atributo debe tener un nombre y un valor. Además, debe especificarse el tipo de variable que es. Así, si se va a escribir un número entero como valor del atributo, se deberá especificar que es un integer; si el valor del atributo de trata de una cadena deberá identificarse como string; y lo mismo con las demás variables.

Por último, debe indicarse la acción de actualización que queremos que se realice. Tenemos varias posibilidades: [28]

- Append. Se usa para la creación de entidades y atributos y para la actualización de atributos en entidades existentes.
- AppendStrict. Se usa para la creación de entidades y atributos en entidades existentes, pero no modifica atributos ya existentes.
- Update. Modifica atributos existentes, pero no crea nuevas entidades ni atributos.
- Replace. Esta acción reemplaza atributos en entidades existentes.
- Delete. Es usada para eliminar atributos en entidades existentes o para eliminar entidades.

La opción escogida en todas las entidades utilizadas del proyecto es "append", pues se quiere crear las entidades en un primer momento y, posteriormente, estar actualizando sus atributos.

La respuesta dada por el Context Broker al realizar la actualización de la información correctamente vendrá dada por una entidad llamada "contextResponses", que indica el tipo de entidad, el valor del campo "isPattern" y su identificador, así como los atributos de la entidad, sin indicar el valor de cada uno. Además de la información de contexto, devuelve el estado del mensaje. En el caso de haberse realizado la actualización correctamente, contendrá el código "200". En la Figura 19 se puede ver un ejemplo de respuesta del Context Broker.

En cuanto a las peticiones de información almacenada en Orion, en este proyecto se ha utilizado NGSIv2, que permite hacer las peticiones GET de las entidades de una forma sencilla, accediendo a "/v2/entities/" y añadiendo el id de la entidad que se quiere obtener. Haciendo uso de esta versión de la API, no se requiere cuerpo del mensaje en la petición, facilitando así el proceso.

3.2.1 Definición y reutilización de entidades de FIWARE

La entidad principal que se ha creado es la que contiene datos de la actividad física, a partir de la información recogida y enviada a la aplicación móvil por la camiseta inteligente. El tipo de esta entidad es "PhysicalActivity" y su id "PhysicalActivity" más el nombre del paciente y la fecha en la que se han tomado los datos, esto es así para poder tener información actualizada de la actividad física de cada paciente en la fecha que se ha realizado, sin que se sobreescriban los datos. El campo "isPattern" se ha dejado en "false", pues no requiere utilizarse como expresión regular.

```
{
    "contextResponses": [
    {
        "contextElement":
            {
                "type":"Person",
                "isPattern":"false",
                "id":"Rafa",
                "attributes": [
                {
                    "name":"age",
                    "type":"integer",
                    "value":""
                },
                {
                    "name":"height",
                    "type":"integer",
                    "value":""
                },
                {
                    "name":"weight",
                    "type":"integer",
                    "value":""
                }
            ]
        },
    "statusCode":
        {
            "code":"200",
            "reasonPhrase":"OK"}
        }
    ]
}
```


Figura 19. Respuesta de Orion al updateRequest de la entidad "Person".

En cuanto a los atributos, se han mapeado los mismos existentes en los mensajes recibidos, adaptándolos a la estructura de las entidades de NGSIv1, y se ha añadido el nombre del usuario que utiliza la camiseta. Así, tenemos los siguientes:

- "accumulates_metabolic_expenditure". Se trata del gasto calórico total del paciente. Es una variable de tipo "float".
- "instant_metabolic_expenditure". Es el gasto calórico instantáneo en el momento en el que se envía el mensaje. También se trata de un "float".
- "repetitions". Es el número de repeticiones realizadas del ejercicio. Se trata de una variable de tipo "integer".
- "user". Es el usuario que utiliza la camiseta.

En la figura 20 se puede observar un ejemplo de la entidad de tipo "PhysicalActivity" enviada al OCB.

```
{
    "contextElements":[
    {
        "type":"PhysicalActivity",
        "isPattern":"false",
        "id":"PhysicalActivityRafa20190902",
        "attributes":[
            {
                "name":"accumulated_metabolic_expenditure",
                "type":"float",
                "value":"4.59"
            },
            {
                "name":"instant_metabolic_expenditure",
                "type":"float",
                "value":"0.07"
            },
            {
                "name":"repetitions",
                "type":"integer",
                "value":"6"
            }
            {
                "name":"user",
                "type":"string",
                "value":"Rafa"
            }
        ]
    }
],
    "updateAction":"APPEND"
}
```


Figura 20. Ejemplo de entidad de tipo “PhysicalActivity”.

Se ha creado también una entidad secundaria para identificar al paciente y tener almacenado información útil para el seguimiento del mismo. Ésta es de tipo “Person” y tiene como id el nombre del usuario de la camiseta. Como puede observarse en la figura 21, los atributos contienen los datos que se quieren almacenar de él: la edad, la altura (en centímetros) y el peso (en kilos).


```
{
    "contextElements": [
    {
        "type":"Person",
        "isPattern":"false",
        "id":"Rafa",
        "attributes": [
            {
                "name":"age",
                "type":"integer",
                "value":"21"
            },
            {
                "name":"height",
                "type":"integer",
                "value":"190"
            },
            {
                "name":"weight",
                "type":"integer",
                "value":"84"
            },
        ]
    }
],
    "updateAction":"APPEND"
}
```


Figura 21. Ejemplo de entidad de tipo “Person”.

Uno de los aspectos que se ha podido aprovechar en cuanto al almacenamiento de datos en el Context Broker es la reutilización de entidades ya creadas. En este caso, se ha decidido utilizar las entidades de "Device", para identificar la camiseta inteligente, y "Alert", para mostrar alertas.

La entidad "Device" identifica a la camiseta inteligente mediante los atributos "description", que realiza una breve descripción del dispositivo, "macAddress", que contiene la dirección MAC de la camiseta, y "name", el nombre del dispositivo, como se puede observar en la figura 22.

```
{
    "contextElements": [
    {
        "type":"Device",
        "isPattern":"false",
        "id":"intelligentTshirt",
        "attributes": [
            {
                "name":"description",
                "type":"string",
                "value":"Camiseta que recoge datos de la actividad física del paciente"
            },
            {
                "name":"macAddress",
                "type":"string",
                "value":"00:1E:C0:25:E6:99"
            },
            {
                "name":"name",
                "type":"string",
                "value":"Camiseta Inteligente"
            },
        ]
    }
],
    "updateAction":"APPEND"
}
```

Figura 22. Ejemplo de entidad de tipo "Device".

La otra entidad reutilizada es "Alert", la cual avisa al usuario de que ha gastado más de 500 kcal con el objetivo de que realice un descanso. Como atributos, se han escogido los siguientes: "category", que indica que la información es relacionada con la salud, "description", que contiene el mensaje de alerta para el paciente, y "severity", que indica que la intención del mensaje es informacional. En la figura 23 se puede ver un ejemplo de esta entidad.

3.3 Desarrollo del Agente IoT en la aplicación móvil

El elemento principal del trabajo es el Agente IoT que recoge información de la API de la camiseta inteligente y lo mapea a las entidades FIWARE definidas en el apartado 3.2.2. Este agente se ha llevado a cabo dentro de una aplicación móvil de Android, desarrollada en Android Studio. La estructura de la parte de la solución que se detalla en este apartado se encuentra esquematizada en la figura 24.

```
{
    "contextElements": [
    {
        "type":"Alert",
        "isPattern":"false",
        "id":"alert500",
        "attributes": [
            {
                "name":"category",
                "type":"string",
                "value":"health"
            },
            {
                "name":"description",
                "type":"string",
                "value":"Has gastado más de 500 kcal haciendo ejercicio.
                Quizás sea un buen momento para tomar un descanso."
            },
            {
                "name":"severity",
                "type":"string",
                "value":"informational"
            },
        ]
    }
],
    "updateAction":"APPEND"
}
```


Figura 23. Ejemplo de entidad de tipo “Alert”

Esta aplicación se ha dividido en dos vistas para el usuario. En primer lugar, debe aparecer una pantalla de registro, en la cual el paciente pueda registrar sus datos y mandarlos en la entidad “Person” al Context Broker. Una vez toque el botón de Comenzar, la aplicación debe mostrar la otra pantalla, en la cual se muestran los mensajes enviados en un ScrollView. Para ello, se ha dividido la aplicación en dos actividades, Registro y MainActivity, correspondiéndose con cada vista, respectivamente. Para la interfaz gráfica, cada actividad tiene su layout correspondiente, en el cuál se define el apartado visual de la aplicación.

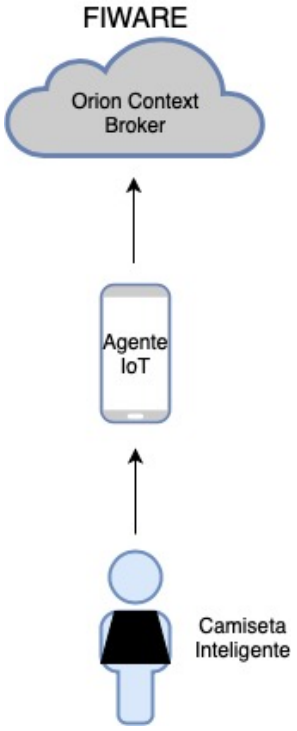

Figura 24. Solución parcial correspondiente al Agente IoT.

En el archivo AndroidManifest.xml, se dan varios permisos necesarios a la aplicación, entre ellos los de bluetooth, internet, y acceso al estado de la conexión wifi, para la correcta conexión con la camiseta inteligente. Esto permisos se pueden observar en la figura 25. Además, en él se declaran las actividades junto con sus correspondientes vistas. La versión del gradle utilizada en el desarrollo es la 3.4.2, definida en el archivo build.gradle.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Figura 25. Permisos en el archivo AndroidManifest.xml.

3.3.1 Actividad Registro

Al abrir la aplicación, debe aparecer la vista de Registro, la cual pide que el usuario se registre introduciendo en los cuatro EditText disponibles el nombre del paciente, su edad, su altura y su peso. Esta actividad tiene como objetivo el envío de una entidad con los datos del paciente a FIWARE, útiles para el seguimiento del mismo. Para completar el proceso, debe tocar el botón de comenzar, lo cual ejecutará la función "registrar", que recibe los datos introducidos y crea la entidad "Person" con ellos.

```
public String postRequest(final JSONObject entidad) throws IOException {
    final StringBuffer[] jsonString = new StringBuffer[1];
    AsyncTask.execute(() → {
            try {
                //Conexión con el Context Broker
                URL url = new URL( spec: "http://192.168.56.101:1026/v1/updateContext");
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setDoInput(true);
                conn.setDoOutput(true);
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Accept", "application/json");
                conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");
                OutputStreamWriter output = new OutputStreamWriter(conn.getOutputStream(), charsetName: "UTF-8");
                System.out.println("El JSONObject que se va a mandar es: " + entidad.toString());

                //Se realiza el envío
                output.write(entidad.toString());
                output.close();

                //Para recibir la respuesta del Context Broker y ver que se ha enviado la entidad correctamente
                BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                jsonString[0] = new StringBuffer();
                String line;
                while ((line = br.readLine()) != null) {
                    jsonString[0].append(line);
                }
                br.close();

                conn.disconnect();

                if (jsonString[0] == null)
                    System.out.println("null");
                else {
                    //Imprimo la respuesta del Context Broker
                    System.out.println(jsonString[0].toString());
                }


            }catch (IOException e) {
                e.printStackTrace();
            }
    });
    Log.d( tag: "Depuración", msg: "Ha llegado hasta el final del método PostRequest");
    String env = "Enviado correctamente";
    return env;
```

Figura 26. Método PostRequest de la actividad Registro.

Para realizar la petición POST al Context Broker se llama a la función postRequest, la cual se puede ver en la figura 26. Esta función recibe el JSON creado como argumento, y realiza la conexión HTTP con Orion para ejecutar la petición en una tarea asíncrona. Debe ser así pues el hilo principal de la aplicación no puede esperar hasta que se realice una tarea y bloquear así la ejecución. Además, recibe la respuesta del Context Broker y la imprime por consola. Una vez realizado esto, se realiza un Intent a la siguiente actividad, pasándole el nombre de usuario que se ha introducido. Este argumento es necesario para la actividad MainActivity para identificar las entidades de actividad física con el nombre del paciente al que pertenece la información recogida. La interfaz de la actividad Registro puede observarse en la figura 27.

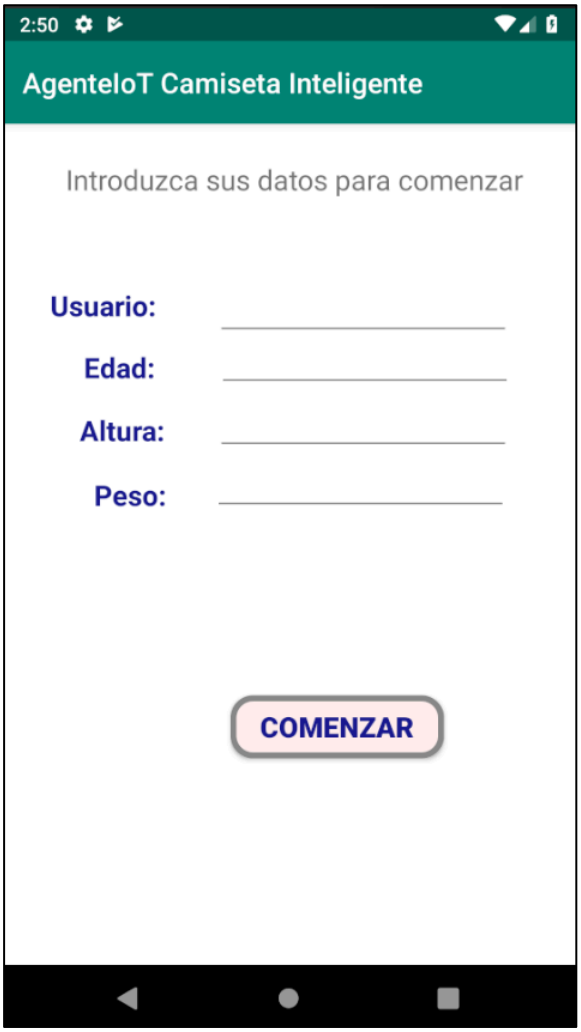

Figura 27. Vista de la actividad Registro.

3.3.2 Actividad MainActivity

Esta actividad principal se encarga de la conexión con la camiseta inteligente, del envío de mensajes a Orion y de la visualización de ellos por pantalla. Para la conexión con la camiseta se han incluido las librerías de es.us.gib.biosensors, proporcionadas por el Grupo de Ingeniería Biomédica de la ETSI. Se ha utilizado un módulo de simulación de actividad proporcionada en la librería, el cual se activa nada más cambiar de vista, en el método onCreate. Seguidamente, se crea la entidad "Device" y se manda a Orion usando el método postRequest.

```
Biosensors main = new Biosensors(getApplicationContext());
SmartShirt smartShirt = main.getSmartShirt( mac: "00:1E:C0:25:E6:99");

// Módulo de monitorización respiratoria (modo normal)
//smartShirt.getRespiratoryModule().setSimulation(true);
//smartShirt.getRespiratoryModule().setNormalMode(this, 1);

// Módulo de actividad física (modo normal)
//smartShirt.getActivityModule().setSimulation(true);
//smartShirt.getActivityModule().setNormalMode(this, 1);

// Módulo de actividad física (modo ejercicio)
smartShirt.getActivityModule().setSimulation(true);
```

Figura 28. Conexión con la camiseta inteligente y módulo de actividad física.

Una vez activado el módulo de actividad física, la camiseta comienza a mandar datos a la aplicación móvil, provocando continuas llamadas a la función onActivityExerciseMode. Esta función se encarga de la construcción del mensaje para el Context Broker, que contiene la entidad de tipo "PhysicalActivity", con los datos recibidos de la camiseta. Además, comprueba si el total de calorías gastadas por el paciente es mayor a 500, creando en ese caso la entidad "Alert".

Finalmente, muestra los mensajes enviados por pantalla en un ScrollView, cuyo scroll se mantiene constantemente hacia abajo para que así aparezcan a la vista los nuevos mensajes. Para ello, se ha examinado si es la primera vez que se llama a la función. Si lo es, se crea un StringBuilder vacío, que más tarde contendrá el primer mensaje enviado a Orion; si no, se crea un StringBuilder obteniendo lo que ya se ha impreso por pantalla, permitiendo así imprimir todos los mensajes enviados en el TextView. Esto es así porque se quiere mostrar todos los mensajes enviados, si no solo mostraría el último. El código relacionado con este aspecto se encuentra en las figuras 29 y 30.

```
//Para la impresión de mensajes enviados a ORION
TextView logsView = (TextView) findViewById(R.id.logsView);
StringBuilder stringBuilder = null;

//Para mantener el scroll abajo
ScrollView scrollView = findViewById(R.id.scrollview);
scrollView.fullScroll(scrollView.FOCUS_DOWN);
evitaBloqueoMensajes++;

if (isFirstTime || evitaBloqueoMensajes > 500) {
    //Para la impresión de los mensajes enviados a ORION
    stringBuilder = new StringBuilder();
    isFirstTime=false;
    evitaBloqueoMensajes = 0;
}else{
    stringBuilder = new StringBuilder(logsView.getText().toString());
}
```

Figura 29. Comprobación para la impresión de mensajes enviados a Orion.

```
// IMPRESIÓN LOGS EN LA PANTALLA
stringBuilder.append(entidadActFisica.toString() + "\n\n");
logsView.setText(stringBuilder.toString());
```

Figura 30. Impresión de mensajes enviados a Orion.

Además, se ha añadido una variable "evitaBloqueoMensajes", que se trata de un contador que evita que se bloquee la aplicación en la impresión de mensajes por pantalla. Esto ocurría porque al ir acumulando contenido en el ScrollView, cuando la aplicación llevaba mucho tiempo funcionando, se bloqueaba. Es por ello que una vez el contador llega a más de 500, se crea un nuevo stringBuilder, borrando los anteriores mensajes impresos y volviendo a empezar desde arriba del ScrollView.

Para añadir la fecha al campo "id" de la entidad de tipo "PhysicalActivity", se ha llamado a la función dateFormatter creada, que devuelve la fecha actual en formato "yyyyMMdd" (año, mes y día, sin espaciar), y se ha usado el nombre del paciente introducido en la vista de Registro, el cual es pasado a esta actividad como argumento del Intent.

La vista completa de la actividad MainActivity puede observarse en la figura 31.

Figura 31. Vista de la actividad MainActivity.

Para la conexión con el Context Broker, se encontró el problema de que al optarse por HTTP, en lugar de HTTPS, para la transmisión, no dejaba conectarse, pues daba error de "java.io.IOException: Cleartext HTTP traffic to 192.168.56.101 not permitted". Para solucionarlo, se creó el archivo network_security_config.xml, en el cual se deja explícito la permisión de tráfico sin encriptar para la IP 192.168.56.101, correspondiente a la dirección del Context Broker. El contenido de este archivo se encuentra explícito en la figura 32.

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">192.168.56.101</domain>
    </domain-config>
</network-security-config>
```

Figura 32. Código del archivo network_security_config.xml.

3.4 Desarrollo del Servicio Web

Como elemento secundario del proyecto se ha desarrollado, haciendo uso de Visual Studio Code, este servicio web, encargado de mostrar los datos actualizados de la actividad física almacenados en el Context Broker. Además, muestra la información de la entidad Person registrada por el usuario en la aplicación móvil y, al estar conectado por otro lado con la Realtime Database de Firebase, permite mostrar un historial de progreso en el ejercicio realizado por el paciente. La figura 33 describe la estructura de la parte de la solución detallada en este subapartado.

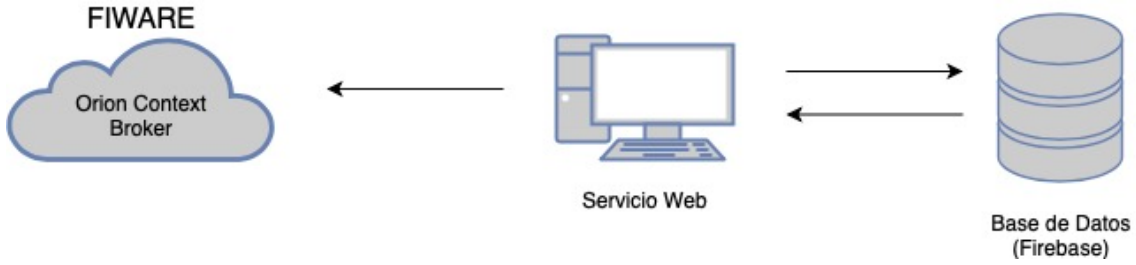

Figura 33. Solución parcial correspondiente al Servicio Web.

El servicio web está dividido en componentes, cada uno correspondiente con una vista o un apartado de la vista, excepto el componente principal app.component, que tiene como única función permitir la navegación entre los otros tres componentes. Estos últimos son navigation.component, que implementa la barra de navegación, home-component.component, encargado de la primera vista del servicio y que ejerce la actividad principal para cumplir el objetivo de la aplicación web, y el componente about.component, en el cual se describe la función de la página web con un fin informativo al usuario. Cada uno se dispone en un archivo de TypeScript, donde se definen las funciones que dan lugar al servicio, y uno HTML, donde se trabaja con el apartado visual de la aplicación.

En el archivo app.module.ts se ha incluido todos los módulos necesarios para llevar a cabo el servicio. Entre ellos podemos encontrar los módulos para las peticiones HTTP, aquellos que permiten la navegación entre componentes y los módulos necesarios para la implementación de Firebase en la aplicación. Además, en este archivo se han importado los componentes y servicios que se han usado en el proyecto.

El componente principal app.component, antes mencionado, tan solo contiene la etiqueta de <router-outlet> para permitir la navegación a los componentes Home y About. Las rutas se encuentran definidas en el archivo app-routing.module.ts, especificando Home como la ruta por defecto al abrir la aplicación web. Para la navegación, hace falta la barra de navegación, implementada en el componente Navigation, en la que se encuentran los nombres de los apartados visuales a los que puede desplazarse el usuario y de los que se encargan los otros dos componentes restantes.

Para la base de datos se ha creado un proyecto en la web de Firebase, en la cual he iniciado con mi cuenta personal de Google. En el archivo enviroment.ts se ha implementado la API y la clave del proyecto creado. Esta plataforma de Google ofrece una interfaz de usuario sencilla, como se puede observar en la figura 34, donde los elementos se encuentran ordenados por su clave.

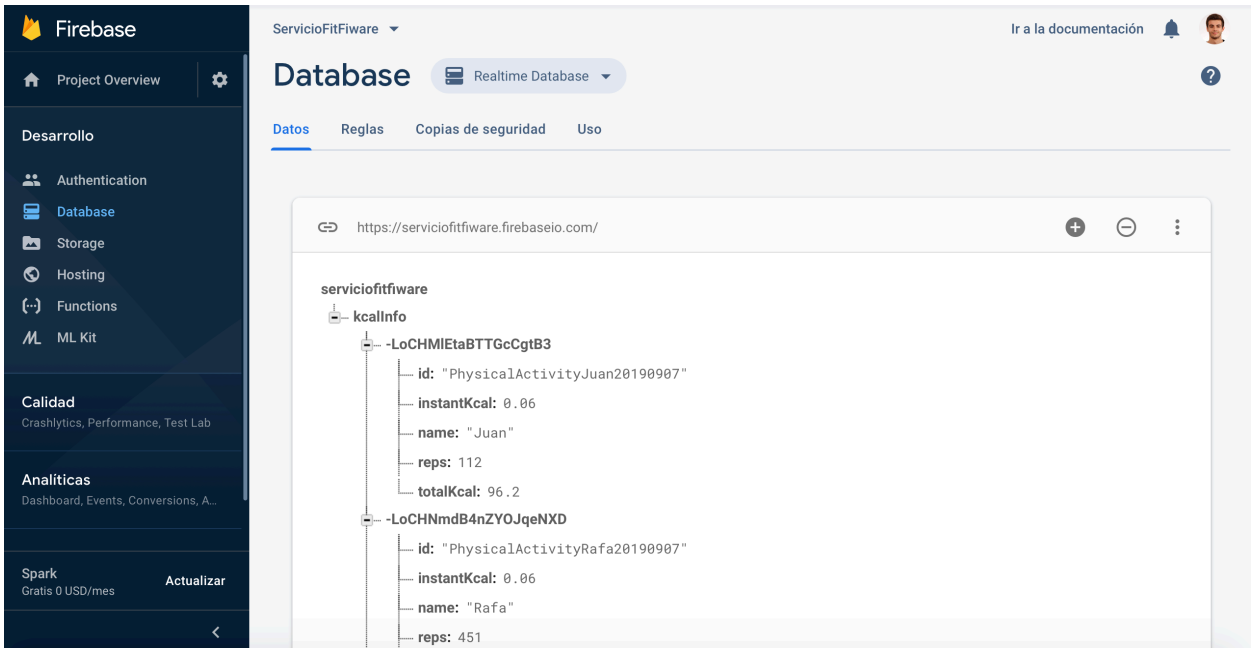

Figura 34. Interfaz de usuario de la Realtime Database de Firebase.

Por último, para el diseño se ha implementado Bootstrap4, concretamente la plantilla Flatly, disponible en Bootswatch. Esto se ha realizado incluyendo los scripts necesarios en el archivo index.html.

3.4.1 Componente Home

Se trata del componente donde se realizan las funciones principales del servicio web. En primer lugar, se muestra un formulario para introducir el nombre del paciente del que se quiere recibir la información. Una vez introducido, se muestran los datos de actividad física recogidos por la camiseta inteligente y almacenados en el Context Broker de FIWARE, así como sus datos de registro, que incluyen el nombre del paciente, su edad, su altura y su peso. La información que se muestra es la de la última actualización que se recibió al entrar en la aplicación web y se dispone en tres columnas, quedando así el gasto calórico instantáneo a la izquierda, el gasto calórico total en el centro y el número de repeticiones en la derecha, todo ello bajo una imagen de FIWARE (Figura 35).

Figura 35. Servicio web en su vista principal.

Debajo de estos datos, justo después de la información de registro, se encuentra la tabla de Progresión Reciente, que muestra un historial del progreso realizado en la actividad física del paciente que se ha introducido en el formulario. La información de esta tabla se actualiza cada vez que se ha usado el servicio web y podemos observar un ejemplo de ella en la figura 36.

Figura 36. Tabla de progresión reciente del servicio web y datos de registro del paciente.

Este componente utiliza las funciones de los servicios DataService, para la conexión con FIWARE, y BbddService, para la comunicación con Firebase. En primer lugar, al iniciarse la página todos los datos se encuentran inicializados a 0. Esto se hace así para evitar el error de 'undefined is not an object', que ocurre cuando un elemento no se ha inicializado correctamente. Una vez introducido el nombre del paciente en el formulario, éste es pasado como argumento a la función mostrarInfo, la cual se suscribe a dos funciones de DataService: searchUserInfo, para obtener los datos de registro del paciente, y searchKcal, para mostrar la información de actividad física del mismo. La información devuelta por searchKcal es usada como contenido de una variable de tipo Post, la cual sirve como argumento de las funciones que se conectan con Firebase. Este tipo de variable ha sido creado en el archivo Post.js con el objetivo de mandar la información deseada a la base de datos del servicio. Se compone de una id, los datos de actividad física y el nombre del paciente. Además, tiene una clave que sirve para su almacenamiento en la base de datos y que obtiene un valor automáticamente al enviar la información a Firebase. Este tipo de dato creado se muestra en la figura 37.

```
export class Post {
    $key: string;
    id: string;
    instantKcal: number;
    totalKcal: number;
    reps: number;
    name: string;
}
```

Figura 37. Tipo de dato Post.

Seguidamente, con la variable de tipo Post completa, se llama a la función postKcalInfo de BbddService para su almacenamiento en la Realtime Database de Firebase. Aquí se barajó la posibilidad de realizar una comparación entre los campos "id" de los elementos y comprobar si ya existía información de actividad física del paciente en esa misma fecha, actualizándola en ese caso o creando un elemento nuevo en el contrario. Esto no se ha hecho así pues para ello habría que suscribirse al método snapshotChanges al llamar a la función getKcalInfo de DataService. Este método se encarga de devolver un objeto con los datos de la tabla cada vez que se actualiza su información. Entonces, al hacer la comprobación dentro de esta suscripción y, una vez comprobado, una creación de un elemento nuevo para la base de datos, el método se volvía a ejecutar, pues se había actualizado la información de la base de datos. Por ello, volvían a almacenarse elementos ya guardados, incluso sobreescribiendo otros anteriores. Otra limitación es que no se podían añadir varios registros de actividad física del mismo día y el mismo paciente, pues se sobreescribiría el anterior, al tomar ésta como una actualización. Es por estas dos limitaciones que se ha decidido almacenar la información recibida del Context

Broker siempre como un nuevo elemento. Se ha dejado como trabajo futuro una actualización que permita sobreescribir esa información en Firebase.

Por último, se llama a la función mostrarProgreso (Figura 38), encargada de mostrar la tabla de progreso de actividad física del paciente, registrado en Firebase. Para ello, se suscribe al método snapshotChanges, explicado en el párrafo anterior, de la función getKcalInfo de BbddService, que devuelve los elementos almacenados en la base de datos. Estos elementos son recorridos mediante un bucle forEach, completando la tabla de progreso reciente con los datos del paciente. Para que solo se muestren los datos de ese paciente, esta función recibe como argumento el nombre del mismo y, antes de actualizar la tabla de progreso, se compara este nombre con el del elemento de Firebase recorrido en ese momento. Si coinciden, se trata del paciente del que se desea obtener la información, actualizándose así el registro.

```
//Función que muestra la tabla de progreso registrado en Firebase
mostrarProgreso(newName : string){
  console.log('Funcion mostrarProgreso')

  //Me suscribo a la función que devuelve los elementos almacenados en Firebase
  this.bbddService.getKcalInfo().snapshotChanges().subscribe(item => {
    this.infoList = [];

    //Recorro todos los elementos devueltos
    item.forEach(element =>{
      let i = element.payload.toJSON();

      //Con este if relleno la tabla de progreso reciente sólo del paciente introducido
      //comparando el nombre de cada usuario cuyos datos están almacenados en Firebase con el del paciente
      if(i["name"] == newName){
        i["key"] = element.key;
        this.infoList.push(i as Post);
      }
    });
  });
}
```

Figura 38. Función mostrarProgreso.

3.4.1.1 Servicio DataService

El servicio DataService es el encargado de la comunicación con el Orion Context Broker, importando para ello HttpClient. Se compone de tres funciones:

- searchKcal: Realiza la petición GET al Context Broker con la versión de la API NGSIv2 para obtener los últimos datos de actividad física del paciente registrados en ese día. Recibe como argumento el nombre del paciente introducido en el formulario para así sólo obtener sus datos.
- searchUserInfo: Similiar a la anterior. Obtiene los datos de registro del paciente almacenados en FIWARE. También recibe como argumento el nombre del mismo.
- getDate: Es una función auxiliar que devuelve la fecha. Es útil para la id buscada en la petición GET de searchKcal.

En el intercambio de mensajes con Orion, CORS (Cross-Origin Resource Sharing) bloqueaba las peticiones GET a este destino. Esto se intentó solucionar habilitando CORS al iniciar el Context Broker, pero no terminaba de funcionar. Se cree que este apartado de FIWARE está aún en desarrollo y que en próximas versiones estará disponible. Mientras tanto, se ha habilitado un proxy, que permite el intercambio de mensajes entre el servicio web y el Context Broker, encontrándose su configuración en el archivo proxy.conf.json.

3.4.1.2 Servicio BbddService

Este servicio se encarga del intercambio de información entre la aplicación web y la Realtime Database de Firebase, incorporando para ello los elementos necesarios del módulo de AngularFire2. Está compuesto por dos

funciones:

- getKcalInfo: Devuelve la lista de la información de actividad física almacenada en la base de datos de Firebase.
- postKcalInfo: Se encarga de crear una nueva instancia en la base de datos con la información obtenida del Context Broker, la cual se pasa en un objeto de tipo Post como argumento de la función. Hace uso del método push de AngularFirelist para el envío. Este método es un ejemplo de la fácil integración de Firebase en Angular, pues el envío se realiza de una forma sencilla y rápida. Se puede observar el código de la función en la figura 39.

```
//Función encargada de crear un nuevo elemento en la bbdd de Firebase
postKcalInfo( post: Post){
  this.kcalDataList = this.firebase.list('/kcalInfo');
  console.log(post);
  this.kcalDataList.push({
    id: post.id,
    instantKcal: post.instantKcal,
    totalKcal: post.totalKcal,
    reps: post.reps,
    name: post.name,
  });
  console.log(this.kcalDataList);
}
```

Figura 39. Función postKcalInfo.

3.4.2 Componente About

Este componente secundario se corresponde con una nueva vista a la principal, accesible en la pestaña About desde el menú de navegación. Se encarga de imprimir un resumen de la funcionalidad del servicio web con objeto de informar al usuario, luego solamente está desarrollado en código HTML. Bajo la descripción realizada se encuentra la imagen de "Powered by FIWARE", como se puede ver en la figura 40.

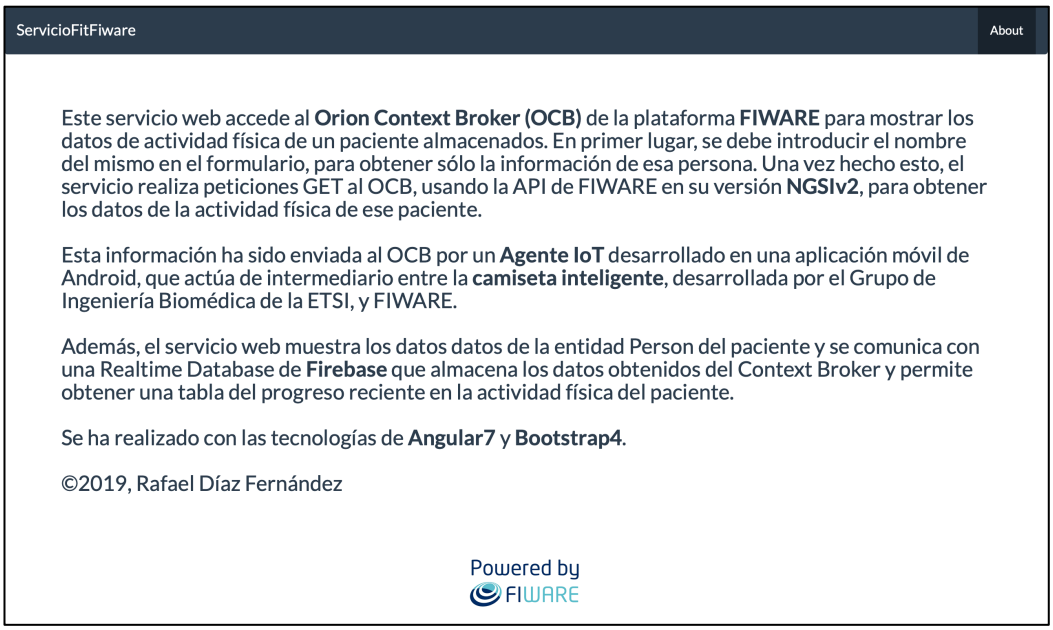

Figura 40. Servicio web en su vista secundaria, correspondiente al componente About.

3.5 Pruebas y validación

Para la comprobación de las funcionalidades de la solución desarrollada y su posterior validación se han llevado a cabo varias pruebas, entre ellas las que se detallan a continuación.

Para la conexión del Agente IoT con FIWARE, se comprobó el correcto intercambio de mensajes imprimiendo la respuesta del Context Broker por consola, ante un mensaje con contenido de actividad física enviado por la aplicación móvil. Se utilizó Postman como complemento a esto, haciendo peticiones GET a las entidades almacenadas en Orion. En un principio, no había respuesta, luego no se estaba enviando correctamente. Tras comparar el mensaje enviado con los ejemplos disponibles en la documentación de FIWARE [5], se llegó a la conclusión de que el formato de este mensaje no coincidía exactamente con el de los ejemplos y que necesitaba unos muy leves cambios para que fuera aceptado por el Context Broker. Seguidamente, se solucionó el problema cambiando esos detalles que no permitían el correcto envío, produciéndose así la respuesta ContextResponses y el almacenamiento de la información en el Context Broker.

Otro aspecto que se ha observado es la actuación de la aplicación de Android a largo tiempo. Se vio que la aplicación fallaba cuando llegaba a un número de mensajes muy alto en el ScrollView. Es por ello que se ha introducido una variable contadora "evitaBloqueoMensajes" a la función onActivityExerciseMode. Esta variable cuenta el número total de mensajes que aparecen en el ScrollView y, cuando llega a más de 500 mensajes, lo vacía borrando los mensajes que aparecen en ese momento y volviendo a mostrar los nuevos mensajes desde arriba. Así, la aplicación no se sobrecarga de mensajes y puede seguir ejecutándose indefinidamente.

En cuanto al servicio web, se comprobó el correcto almacenamiento de la información en la Realtime Database ejecutando el servicio y observando en la consola de Firebase los datos almacenados. Al principio, no se conseguía el almacenamiento de la información. Se estuvo observando la ejecución del código y se comprobó que el método Push nunca se ejecutaba, sino que la ejecución se paraba justo antes. Se observó también en la consola del navegador, que daba error en algunos objetos de tipo any, diciendo que eran "undefined", aunque ya tenían contenido pues se podía imprimir sus datos por pantalla correctamente. Este error no se solucionó tras varios intentos, luego se exploró la opción de cambiar la base de datos a MySQL, aunque la integración de Angular con Firebase es mejor y permite una interacción más sencilla. Tras bastante tiempo investigando, se inicializaron las variables que, según la consola, eran undefined, con los formatos que recibirían de las funciones una vez llamadas estas, evitando así que la consola las diera por "undefined" y las reconociera como el tipo que realmente eran. Una vez realizados estos cambios, la Realtime Database almacenaba correctamente toda la información y se imprimía por pantalla el progreso realizado en actividad física por el paciente.

3.6 Ejecución de la solución completa

Para establecer la solución completa desarrollada en el sistema, en primer lugar hay que tener desplegado el OCB en CentOs, en nuestro caso una máquina virtual. Una vez está funcionando, se iniciará la aplicación en su primera vista, en la cual el paciente debe ingresar sus datos de registro para comenzar a enviar mensajes. Esta vista se puede ver en la figura 41.

Al tocar el botón de "Comenzar", si se han introducido los datos correctamente, se pasará a la segunda vista de la aplicación (Figura 42), la cual muestra los mensajes enviados a FIWARE en un ScrollView. En caso contrario, se mostrará un mensaje Toast avisando al usuario de que debe introducir sus datos correctamente.

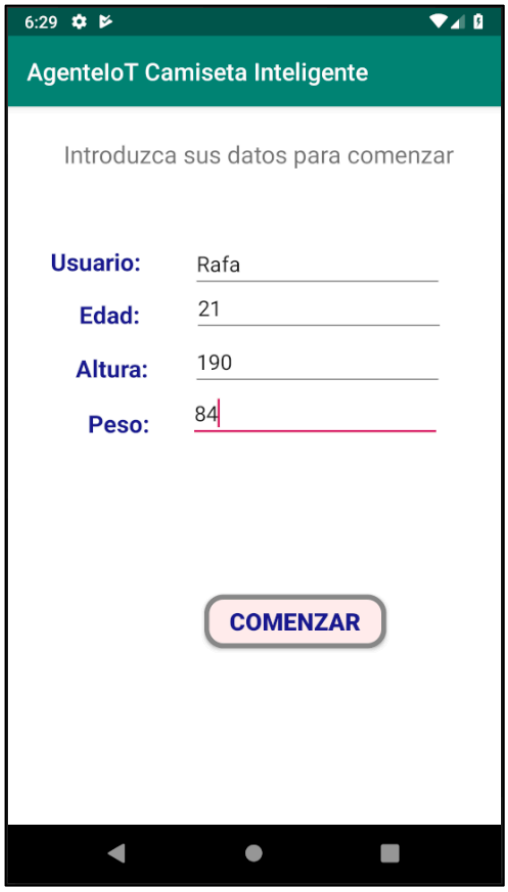

Figura 41. Ejecución del Agente IoT: Vista de la actividad Registro.

Figura 42. Ejecución del Agente IoT: Vista de la actividad MainActivity.

Ahora ya se tendría información almacenada en el Context Broker, que puede ser accedida por el servicio web. Así, se puede acceder a "localhost:4200" para usar el servicio en el navegador. La primera vista mostrada al usuario se observa en la figura 43.

Figura 43. Ejecución del Servicio Web: Vista principal al acceder al servicio.

Se pide que se introduzca en el formulario el nombre del paciente para comenzar a recibir los datos del OCB. Una vez introducido y presionado el botón de "Mostrar datos", se realizan las peticiones anteriormente explicadas y se muestran los datos de actividad física del paciente, así como sus datos de registro y la tabla de progreso reciente, como se observa en la figura 44.

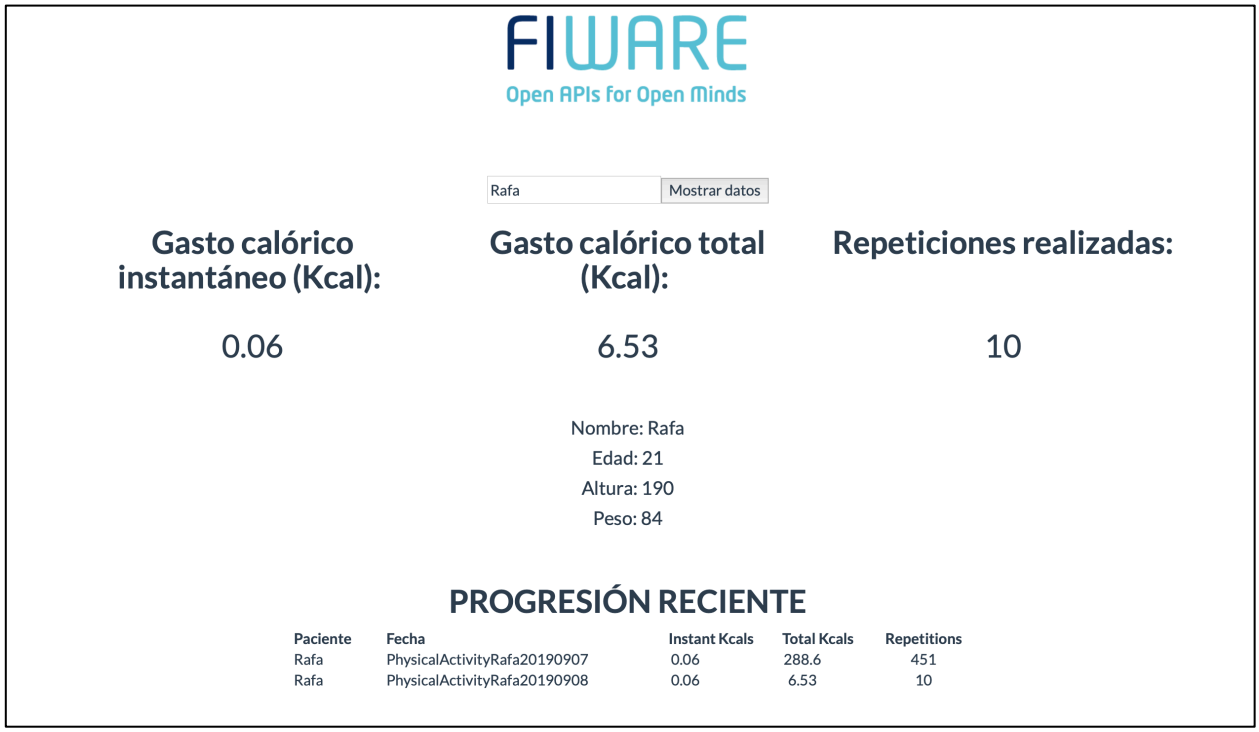

Figura 44. Ejecución del Servicio Web: Vista principal una vez introducido el nombre del paciente.

Así, los nuevos datos obtenidos de actividad física se habrán actualizado en la Realtime Database de Firebase,

como podemos ver en la figura 45.

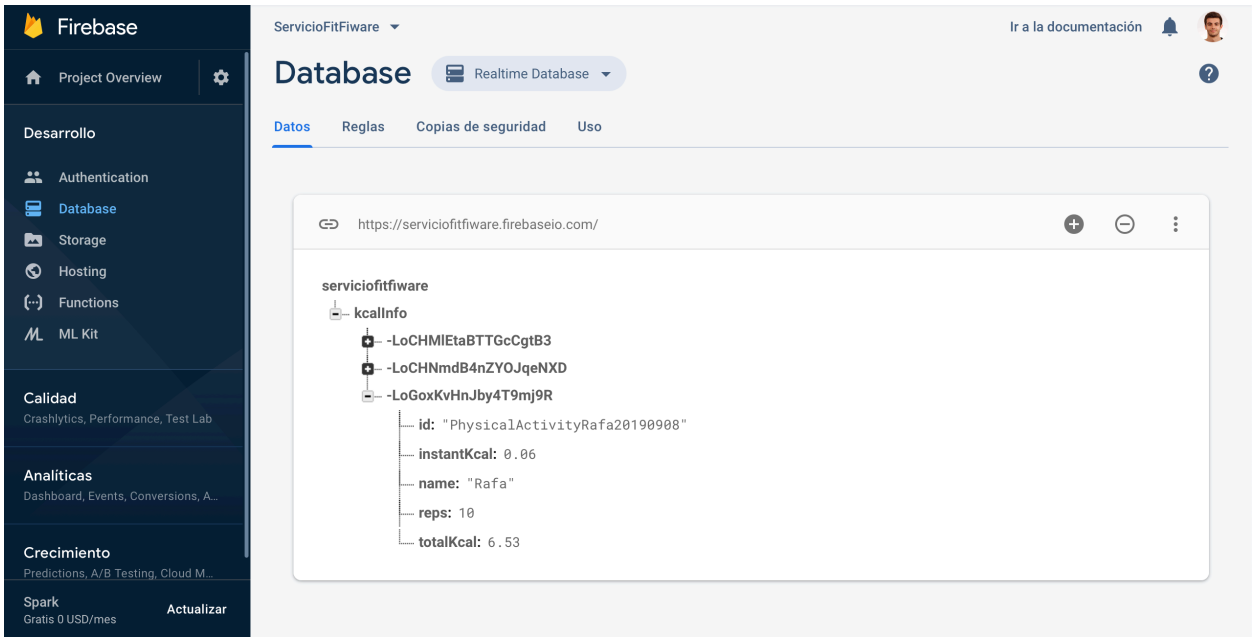

Figura 45. Ejecución del Servicio Web: Base de datos de Firebase actualizada.

Además, en la barra de navegación, puede seleccionarse la pestaña de "About" para obtener la vista informativa descrita en el apartado 3.4.2. Ésta se puede observar en la figura 46.

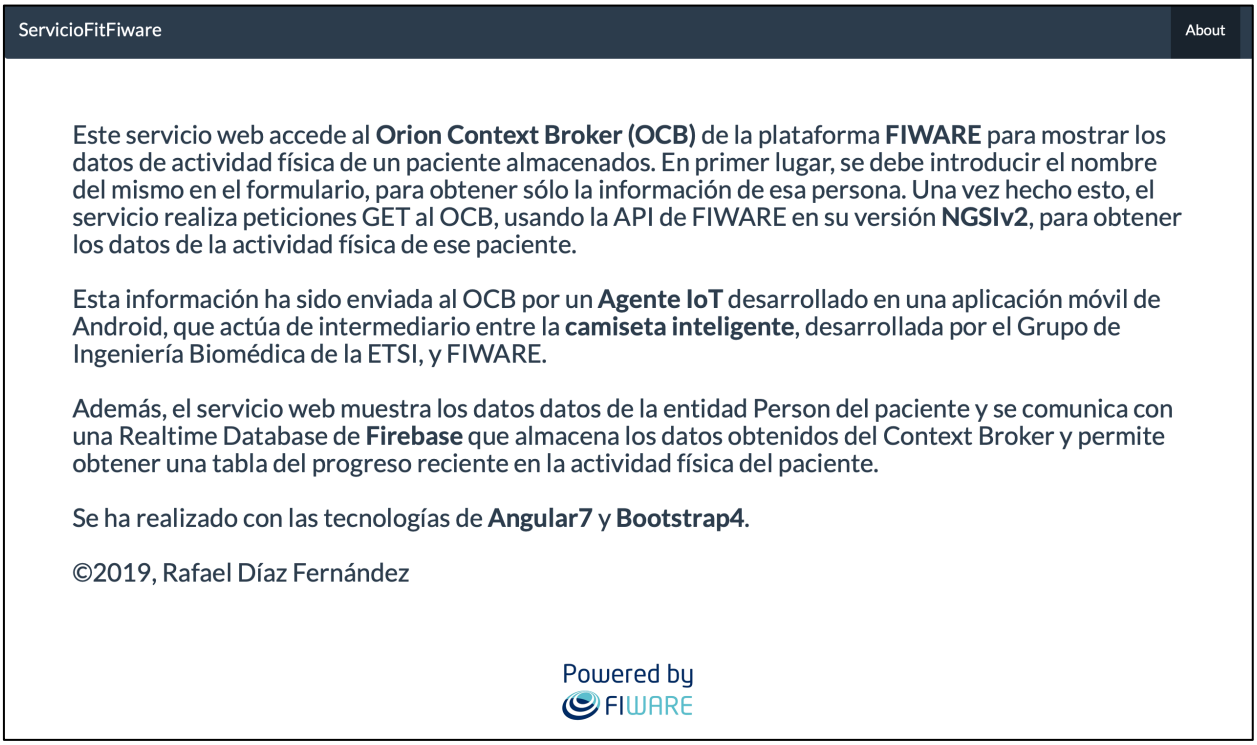

Figura 46. Ejecución del Servicio Web: Vista secundaria informativa.

4 CONCLUSIONES Y LÍNEAS FUTURAS

El 90% del éxito se basa simplemente en insistir.

- Woody Allen -

En la realización de este proyecto se ha conseguido integrar la camiseta inteligente desarrollada por el Grupo de Ingeniería Biomédica de la ETSI en la plataforma de IoT abierta FIWARE mediante el agente desarrollado, permitiendo almacenar los datos de forma automática para que así otros sistemas puedan disponer de ella. Como añadido, se ha desplegado un servicio web que acompaña al agente para proporcionar una solución completa a los objetivos del trabajo. Así, se ofrece una propuesta atractiva para mejorar el seguimiento de los pacientes en sus tratamientos, monitorizando la actividad física realizada y permitiendo el fácil acceso a profesionales sanitarios que necesiten conocer esta información, en cualquier momento y lugar.

Además, se ha investigado a fondo el Internet of Things y la plataforma FIWARE, junto con las numerosas posibilidades que ofrece para el desarrollo de soluciones de IoT, así como otras nuevas tecnologías tales como Angular para el desarrollo de servicios web o la base de datos Firebase, que permite una sencilla interacción con los elementos almacenados y una buena integración con las aplicaciones web desarrolladas, en este caso, en Angular.

Una vez se ha finalizado el proyecto, se ha observado que FIWARE es una plataforma idónea para el desarrollo de soluciones de IoT, permitiendo un fácil despliegue de sus componentes y proporcionando una API sencilla de usar. Angular es una tecnología que no conocía y que volvería a utilizar en el desarrollo de servicios web, ya que no me ha resultado de demasiada complicación y, junto con Bootstrap, ofrece un apartado visual muy bueno al usuario. Sin embargo, intentaría usar otra base de datos en lugar de Firebase pues, aunque tiene una buena integración con Angular, también he observado algunas carencias que no tienen las bases de datos con las que he trabajado durante el grado.

No obstante, existen algunas ideas relacionadas con el proyecto que no se han podido realizar y que seguirían mejorando la solución desarrollada. Estos detalles finalmente no se realizaron debido a la falta de tiempo disponible, teniendo que acotar el trabajo por la extensión del mismo. Entre ellas, destacamos las siguientes:

- Vista para la muestra de información de actividad física en la aplicación móvil. La aplicación móvil muestra los mensajes enviados al Context Broker. Esta idea se trata de implementar una nueva vista que muestre la información de actividad física al usuario que utiliza la camiseta inteligente también en su móvil. En este proyecto se ha optado por hacerlo en un servicio web para completar la solución planteada, permitiendo al personal médico acceder a los datos. Además, de esta forma se ha conseguido investigar más tecnologías, aparte del diseño de aplicaciones móviles.
- Actualización de los elementos en Firebase que tengan la misma id y sus datos sean diferentes, en lugar de la creación de uno nuevo. Como ya se ha comentado, se obtuvo un comportamiento inusual y descontrolado de la base de datos al intentar proporcionar una funcionalidad de actualización de datos de actividad física que se hubieran realizado el mismo día con información diferente, en lugar de la creación de nuevos elementos. Esto ocurría debido a la creación o actualización de entidades dentro de

la suscripción al método snapshotChanges, el cual devuelve una vista de los elementos almacenados en la base de datos de Firebase cada vez que cambian. Entonces, una vez creado o actualizado un elemento, terminaba volviendo a entrar en la función, realizando lo mismo un número aleatorio de veces. Finalmente, se optó por crear entidades con los datos obtenidos del Context Broker, sin consultar si ya había datos de actividad física del mismo día realizada por el mismo paciente. Esto también tiene la ventaja de que se puede registrar información de ejercicio físico realizado en dos momentos distintos del día, sin sobreescribirse. No obstante, se cree interesante que, como actualización, se pueda realizar la función explicada sin perder esta ventaja.

- Impresión de una gráfica de progreso reciente en la aplicación web. Se trata de imprimir los datos de progreso del paciente, almacenados en la Realtime Database de Firebase, en una gráfica, en lugar de una tabla, dando así un aspecto más visual al usuario y permitiendo que sea más sencillo observar el progreso del paciente. Para llevar a cabo esta idea, primero es necesario realizar la anterior propuesta, pues sólo tiene sentido mostrar una gráfica cuando se tiene un único dato por día de cada paciente.

ANEXO A: MANUAL DE INSTALACIÓN Y DESPLIEGUE DEL OCB

En la documentación de FIWARE [29], se recomienda la instalación del Orion Context Broker mediante el sistema operativo en CentOs 7.x. Existen dos modos recomendados de instalación:

- Modo RPM. Se descarga el paquete directamente desde el repositorio Yum de FIWARE, que se encuentra en la documentación. Seguidamente, se instala el paquete usando el siguiente comando con usuario root.

```
rpm -i contextBroker-X.Y.Z-1.x86_64.rpm
```

- Modo Yum. Se descarga la configuración deseada del repositorio público de FIWARE mediante el comando:

```
sudo wget -P /etc/yum.repos.d/
https://nexus.lab.fiware.org/repository/raw/public/repositories/el/7/x86_64/fiware-release.repo
```

  Luego se instala como root:

```
yum install contextBroker
```

El OCB utiliza MongoDB, luego hay que instalarlo también:

- Se crea el archivo /etc/yum.repos.d/mongodb-org.repo con el siguiente contenido:

```
[mongodb-org-4.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc
```

- Se instala con el comando:

```
yum install -y mongodb-org
```

- Finalmente, se inicia el servicio:

```
service mongod start
```

Una vez realizados estos pasos, sólo tendríamos que iniciar el Context Broker. Para ello tenemos dos alternativas:

```
/etc/init.d/contextBroker
```
ó
```
contextBroker
```

Para comprobar su funcionamiento puede realizarse la siguiente petición, que devolverá la información de la versión del OCB desplegado.

```
curl -X GET 'http://localhost:1026/version'
```

ANEXO B: MANUAL DE INSTALACIÓN Y DESPLIEGUE DE LA SOLUCIÓN DESARROLLADA

Ya con el Context Broker desplegado y funcionando, para establecer la solución desarrollada se debe seguir los pasos que se detallan a continuación.

En la ventana inicial de Android Studio se debe de usar la opción de "Abrir un proyecto existente de Android Studio" y seleccionar el proyecto de "PruebaBiosensors".

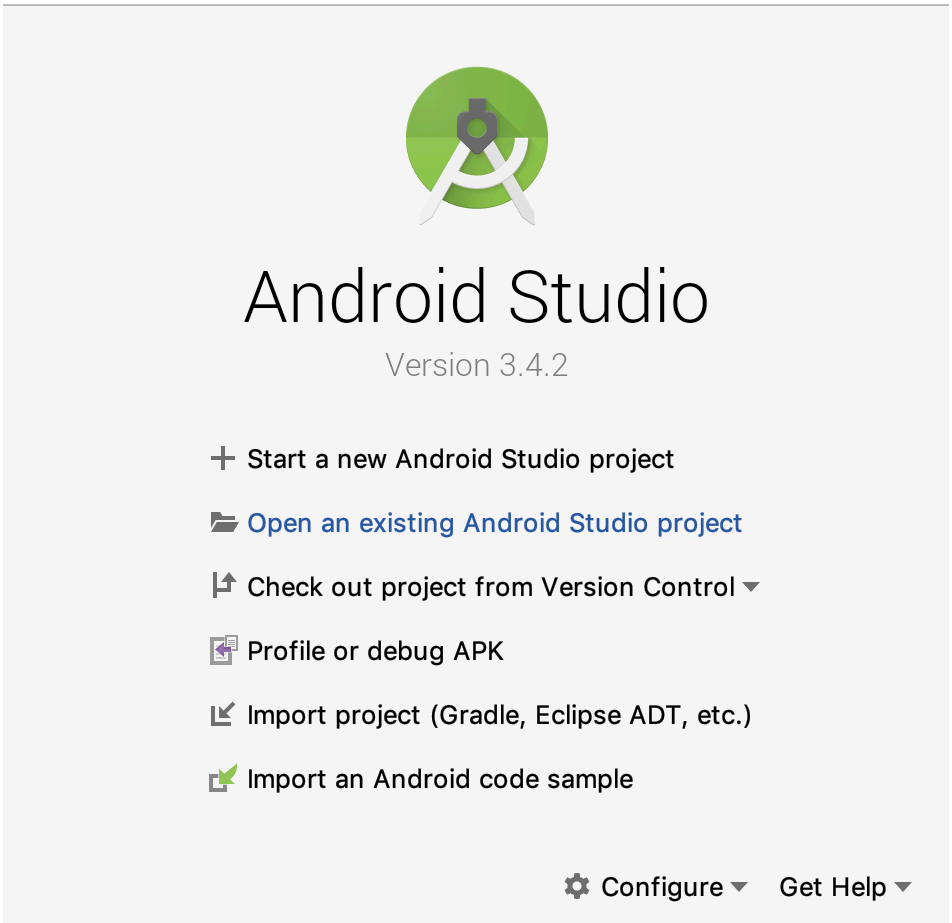

Figura 47. Ventana inicial de Android Studio.

Una vez dentro, se debe tocar la opción de "Build and Run" y escoger el simulador deseado para su despliegue. En este trabajo se ha utilizado el simulador de un Nexus 5X con API 28.

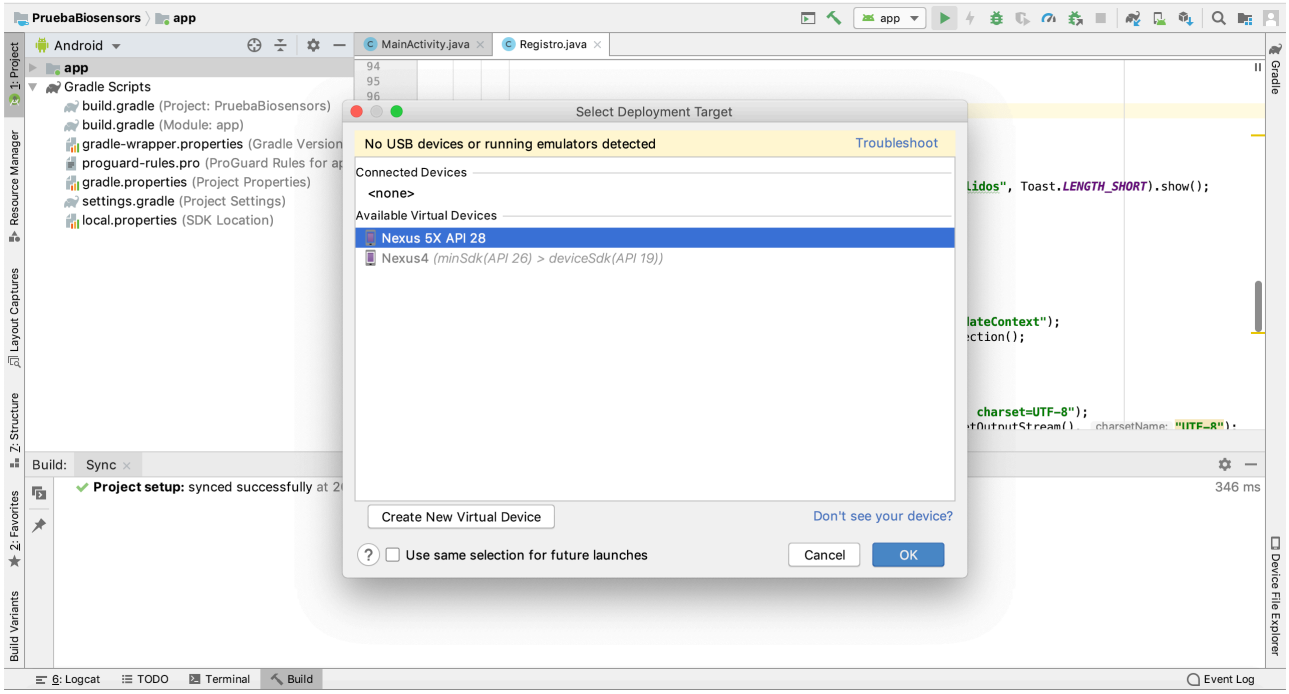

Figura 48. Opción "Build and Run" de Android Studio.

Ahora, la aplicación deberá haberse iniciado en el simulador del dispositivo escogido. Tan solo habría que registrarse en la app y tocar el botón de comenzar para mandar mensajes al OCB.

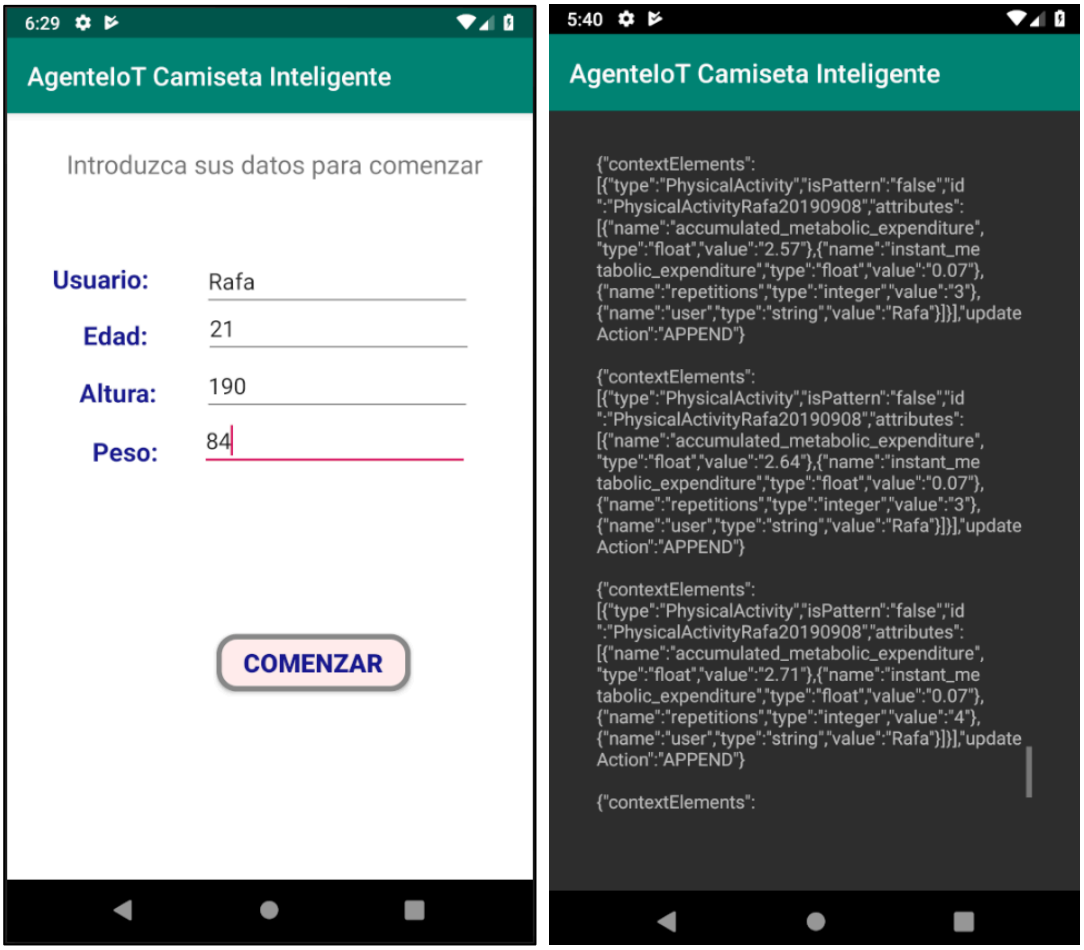

Figura 49. Aplicación móvil desplegada y en uso.

Ahora ya se dispondría de contenido almacenado en el Context Broker, al cuál podría accederse a con el servicio web. Para desplegarlo, en la terminal, una vez situados en la carpeta del servicio, usamos el comando siguiente:

```
npm start
```

Una vez hecho esto, ya podría accederse al servicio web en la dirección "localhost:4200", disponiendo de conexión con el OCB, por un lado, y la base de datos de Firebase, por otro.

Figura 50. Servicio web desplegado y en uso.

REFERENCIAS

[1] Estrategia mundial sobre régimen alimentario, actividad física y salud. Organización Mundial de la Salud. [Consultado el 4 de julio de 2019]. Disponible en: https://www.who.int/dietphysicalactivity/pa/es/

[2] Izquierdo IM, Ibáñez J, Antón M, Cebollero P, Cadore E, Casas Herrero A, López Chicharro J, Vicente-Campos D, Vicente-Rodríguez G, Gómez-Cabello A, Casajus J, Pérez Ruiz M, Herrero F, San Juan Ferrer A, Zamorano Cauto R. Ejercicio Físico es Salud. Prevención y tratamiento de enfermedades mediante la prescripcion de ejercicio físico. 2014. Disponible en: https://www.researchgate.net/publication/267752299_Ejercicio_Fisico_es_Salud_Prevencion_y_tratamiento_de_enfermedades_mediante_la_prescripcion_de_ejercicio_fisico

[3] Página web oficial de FIWARE. [Consultado el 12 de agosto de 2019] Disponible en: https://www.fiware.org

[4] Rayes A, Salam S. Internet of Things From Hype to Reality The Road to Digitization. Cham: Springer; 2017. Disponible en: https://fama.us.es/discovery/fulldisplay?docid=alma991013067530004987&context=L&vid=34CBUA_US:VU1&search_scope=all_data_not_idus&tab=all_data_not_idus&lang=es

[5] Evans D. Internet of Things. La próxima gran evolución de Internet lo está cambiando todo. Cisco IBSG. Abril 2011. [Consultado el 24 de julio de 2019] Disponible en: https://www.cisco.com/c/dam/global/es_es/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf

[6] What is FIWARE?. FIWARE Foundation. 2019. [Consultado el 25 de julio de 2019] Disponible en: https://www.fiware.org/developers/

[7] Presentación "Usando FIWARE y datos de contexto". Fiware Zone. 2019. /

[8] Celesti A, Fazio M, Galán Márquez F, Glikson A, Mauwa H, Bagula A, Celesti F, Villari M. How to Develop IoT Cloud e-Health Systems Based on FIWARE: A Lesson Learnt. 2011.

[9] Orion Context Broker (OCB). Laboratorio Nacional de Internet del Futuro. FIWARE. 2018. [Consultado el 14 de agosto de 2019] Disponible en: https://fiware-training.readthedocs.io/es_MX/latest/ecosistemaFIWARE/ocb

[10] Página web de MongoDB. [Consultado el 14 de agosto de 2019] Disponible en https://www.mongodb.com/es

[11] La eSalud, referencia en eHealth en español. COM SALUD. 2016. [Consultado el 15 de agosto de 2019] Disponible en: https://laesalud.com/que-es-esalud/

[12] Android Studio. Wikipedia, la enciclopedia libre. [Consultado el 15 de agosto de 2019] Disponible en: https://es.wikipedia.org/wiki/Android_Studio

[13] Naranjo-Hernández D, Talaminos-Barroso A, Reina-Tosina J, Roa LM, Barbarov-Rostan G, Cejudo-

Ramos P, Márquez-Martín E, Ortega-Ruiz F. Smart Vest for Respiratory Rate Monitoring of COPD Patients Based on Non-Contact Capacitive Sensing. 2018.

[14] Manual de usuario de la camiseta inteligente. Grupo de Ingeniería Biomédica de la ETSI. 2018.

[15] Documentación de Postman. [Consultado el 16 de agosto de 2019] Disponible en: https://docs.postman-echo.com/?version=latest

[16] Visual Studio Code. Wikipedia, la enciclopedia libre. [Consultado el 16 de agosto de 2019] Disponible en: https://es.wikipedia.org/wiki/Visual_Studio_Code

[17] Angular. Wikipedia, la enciclopedia libre. [Consultado el 17 de agosto de 2019] Disponible en: https://es.wikipedia.org/wiki/Angular_(framework)

[18] TypeScript. Wikipedia, la enciclopedia libre. [Consultado el 17 de agosto de 2019] Disponible en: https://es.wikipedia.org/wiki/TypeScript

[19] HTML. Wikipedia, la enciclopedia libre. [Consultado el 17 de agosto de 2019] Disponible en: https://es.wikipedia.org/wiki/HTML

[20] Lista de etiquetas HTML 5.2. Páginas web HTML y hojas de estilo CSS. [Consultado el 17 de agosto de 2019] Disponible en: http://www.mclibre.org/consultar/htmlcss/html/html-etiquetas.html

[21] Bootstrap. Wikipedia, la enciclopedia libre. [Consultado el 18 de agosto de 2019] Disponible en: https://es.wikipedia.org/wiki/Bootstrap_(framework)

[22] Página web de Bootswatch. [Consultado el 18 de agosto de 2019] Disponible en: https://bootswatch.com

[23] Tamplin J. Firebase expands to become unified app platform. Google. 2016. [Consultado el 19 de agosto de 2019] Disponible en: https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html

[24] Documentación. Firebase. Google. [Consultado el 19 de agosto de 2019] Disponible en: https://firebase.google.com/docs Último acceso: 19 Agosto 2019.

[25] Cetiner Y. Presentación "Firebase Services". 2016. [Consultado el 19 de agosto de 2019] Disponible en: https://www.slideshare.net/Yasinetiner/firebase-services

[26] Firebase Authentication. Firebase. Google. [Consultado el 19 de agosto de 2019] Disponible en: https://firebase.google.com/docs/auth

[27] Firebase Realtime Database. Firebase. Google. [Consultado el 19 de agosto de 2019] Disponible en: https://firebase.google.com/docs/database

[28] Update action types. FIWARE. [Consultado el 22 de agosto de 2019] Disponible en: https://fiware-orion.readthedocs.io/en/master/user/update_action_types/index.html

[29] Instalación Context Broker. FIWARE. [Consultado el 30 de agosto de 2019] Disponible en: https://fiware-orion.readthedocs.io/en/latest/admin/install/index.html