UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DEL SOFTWARE

Sistema Web para mHealth: monitorización de actividad física y constantes vitales mediante Bluetooth

Web-based mHealth system for the monitoring of physical activity and vital signs via Bluetooth

Realizado por
Adrián Sevilla Arrabal

Tutorizado por
Eduardo Guzmán de los Riscos
María Mercedes Amor Pinilla

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE 2020

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

Sistema Web para mHealth: monitorización de actividad física y constantes vitales mediante Bluetooth

Web-based mHealth system for the monitoring of physical activity and vital signs via Bluetooth

Realizado por
Adrián Sevilla Arrabal

Tutorizado por
Eduardo Guzmán de los Riscos
María Mercedes Amor Pinilla

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2020

Fecha defensa: El Secretario del Tribunal

Resumen

La mHealth es un campo innovador que ofrece innumerables posibilidades de desarrollo para prestar servicios al usuario final y a proveedores de servicios: prevención y monitorización de enfermedades, servicios de telemedicina, y en definitiva, el control de la salud de forma más accesible. Con esta finalidad, existen una gran variedad de dispositivos electrónicos en el mercado, que toman mediciones de constantes vitales y que mediante una conexión Bluetooth envían los datos. Sin embargo, toda esta información se encuentra distribuida en diferentes aplicaciones móviles, dificultando al usuario el control de su salud. Este proyecto se centra en reunir las monitorizaciones de diferentes constantes vitales en una aplicación web multiplataforma y multidispositivo, y ofrecer alertas al usuario cuando las mediciones tomen valores anormales.

Palabras clave:

mHealth, Node.js, React, MongoDB, WebBluetoo

Abstract

MHealth is an innovative area that offers countless development opportunities for providing services to the user and service providers, such as, diseases prevention and monitorization, telemedicine and, ultimately, health control in a more accessible way. For this purpose, there is a wide variety of electronic devices on the market that measure vital signs and send the data through a Bluetooth connection. However, all this information is distributed in different mobile applications, making it difficult for the user to control their health. This project focuses on bringing together the monitoring of different vital signs in a multi-platform and multi-device web application, and offering alerts to the user when the vital signs measurements have outliners.

Keywords:

mHealth, Node.js, React, MongoDB, WebBluetooth

Índice

1

Introducción

1.1 Motivación

Son cada vez más numerosos los dispositivos que permiten la medición de constantes vitales y otros niveles e indicadores para determinar el estado fisiológico de un usuario. La lectura de estos parámetros es después trasladada al móvil del usuario mediante una aplicación móvil específica, que permitirá al usuario realizar un seguimiento de estos. La telemonitorización es sin duda un avance para la asistencia sanitaria útil, ya que convierte a los pacientes en agentes activo, más preocupado e involucrado en el cuidado de su salud.

La disponibilidad de dispositivos electrónicos o gadgets en el mercado es grande y aumenta constantemente. Es fácil encontrar en cualquier gran superficie y tienda de electrónica relojes inteligentes y pulseras de actividad que miden el ritmo cardiaco, la distancia recorrida, la actividad realizada e incluso las calorías consumidas. Es fácil también adquirir balanzas de peso corporal, que miden además el índice de masa muscular (o agua), las cuales envían toda esta información al móvil del usuario a través de una conexión Bluetooth.
En el ámbito de la salud hay dispositivos más específicos como tensiómetros, o glucómetros que miden la tensión arterial y la glucosa, y que además permiten almacenar las mediciones en aplicaciones (apps) de móvil a través de una conexión Bluetooth.

Según un informe de Ricoh (2015), el 74% de los hospitales que usan tabletas u otros dispositivos móviles para recopilar información de los pacientes son más eficientes que aquellos hospitales que no lo hacen. Sin embargo, ya no solo se trata de la eficiencia, es una cuestión de necesidad para evitar el colapso de los hospitales debido al crecimiento de la población en las ciudades masificadas y su progresivo envejecimiento. Se espera que para el 2050 se duplique el número de personas mayores de 60 años: de 962 millones en 2017 a 2100 millones en 2050 (Naciones Unidas, 2017). La utilización de tecnologías móviles (mHealth) se presenta como una solución pudiendo reducir el coste

sanitario per cápita en Europa un 18%, y hasta un 35% en el caso del tratamiento de pacientes crónicos (Quental, 2017).

Y es que una de las ventajas principales que ofrece la mHealth a la hora de atenuar el número de visitas a los hospitales consiste en la posibilidad de monitorizar el estado de salud de los pacientes telemáticamente, permitiendo a los médicos ofrecer diagnósticos y tomar decisiones más acertadas y precisas debido a la cantidad de datos recogidos.

Se proyecta que el mercado global de soluciones de mHealth alcanzará los 213,6 mil millones de dólares estadounidenses para 2025, situándose su valor en 50,8 mil millones de dólares en 2020. Este incremento se prevé a una alta tasa de crecimiento anual compuesto (TCAC) del 33,3%. (MarketsAndMarkets, 2020).

Figura 1.1. Pronóstico del valor del mercado global de soluciones de mHealth para 2025 (extraída de https://www.marketsandmarkets.com/Market-Reports/mhealth-apps-and-solutions-market-1232.html)

Con la capacidad existente de obtener esta cantidad de información de una persona, adquirida a través de diferentes sensores, se podría proporcionar un seguimiento completo de sus actividades físicas y su estado fisiológico, así como un análisis de sus rutinas, ayudando a seguir hábitos de vida saludable y detectando aquellos que no lo son. Sin embargo, toda esta información se encuentra distribuida en diferentes aplicaciones del móvil, lo que dificulta utilizar toda esta información relativa a un mismo usuario para poder extraer relaciones entre los hábitos, actividades y estado físico de un mismo usuario usando alguna sencilla técnica de minería de datos.

1.2 Objetivos

El objetivo principal de este trabajo es desarrollar un sistema web que monitorice y procese constantes vitales y actividades físicas de usuarios, que realice un análisis de los

hábitos diario, y que sea capaz de alertar ante posibles riesgos de la salud del usuario. Para ello se diseñará e implementará un sistema Web que se encargará de las siguientes tareas:

- Aplicación Web móvil que facilite la configuración de dispositivos de telemetría personal y/o gadget para la recepción de datos correspondientes a actividades físicas y constantes vitales. La comunicación entre el sistema y los dispositivos se realizará mediante Bluetooth. Inicialmente las posibles constantes vitales a controlar y registrar serán: temperatura, peso, presión arterial, frecuencia cardíaca, nivel de azúcar en sangre y nivel de oxígeno en sangre a través de diferentes dispositivos personales y sensores, reales o virtuales.

- API REST que permita el almacenamiento y recuperación de los datos de un usuario por otras aplicaciones.

- Aplicación Web que permite la búsqueda, recuperación y análisis de los datos. En la medida de lo posible, se intentará dotar a esta aplicación de ciertas funcionalidades que permita realizar recomendaciones a los usuarios sobre cuestiones de salud, en base a los datos analizados.

Existen muchas aplicaciones relacionadas con la salud, pero este TFG se centra en aunar varias de estas necesidades en una misma aplicación web. La aplicación web es por definición multiplataforma y multidispositivo, siendo una gran ventaja para una aplicación enfocada a la salud.

1.3 Estructura de la memoria

La estructura de esta memoria se divide en nueve capítulos; a continuación se explica de forma breve cada uno como objeto de resumen.

- Capítulo 1: Introducción
  Se explica la motivación de este trabajo, sus objetivos, y la estructura del mismo.

- Capítulo 2: Estudio del estado del arte
  Análisis de la situación actual de la mHealth, sus soluciones comerciales enfocándonos en aplicaciones web y móviles, y descripción de las tecnologías a utilizar.

- Capítulo 3: Metodología
  Especificación de la metodología empleada en el desarrollo de este trabajo, detallando sus diferentes fases.

- Capítulo 4: Especificación de requisitos
  Análisis y especificación de requisitos, describiendo de forma completa el sistema.

- Capítulo 5: Diseño y arquitectura

Descripción del sistema, su diseño y arquitectura, incorporando consideraciones de la implementación tecnológica.

- Capítulo 6: Implementación
  Desarrollo del sistema diseñado en la fase anterior. Desde la base de datos, la API REST y el registro e inicio de sesión de los usuarios, hasta la monitorización de cada una de las constantes vitales a controlar en este proyecto.

- Capítulo 7: Pruebas
  Comprobación de que el sistema funciona según lo especificado. Para ello se realizan pruebas software.

- Capítulo 8: Documentación
  Documentación del manual de usuario y el manual de instalación.

- Capítulo 9: Conclusiones y trabajo futuro
  Enumeración de las conclusiones extraídas del trabajo completo y discusión de posibles trabajos futuros que podrían realizarse.

2

Estudio del estado del arte

2.1 Análisis mHealth

Ya hemos visto la importancia de la mHealth en la actualidad y sobre todo en el futuro tan prometedor que ofrece, apostando por la prevención, ayudando a la sostenibilidad de los sistemas de salud y prestando servicios de telemedicina.

La mHealth ya ha tenido avances en el campo de la telemedicina, con proyectos como CardioMEMS (Telemedicina, Fundación Española del Corazón), donde la implantación de un chip al paciente vigila los cambios en la presión arterial pulmonar sin necesidad de acudir a un centro sanitario. En el ámbito español, nos encontramos con el programa iCOR (Telemedicina, Fundación Española del Corazón), tele atención sanitaria domiciliaria para pacientes con insuficiencia cardiaca crónica de alto riesgo. Los pacientes envían diariamente sus datos biométricos a una estación de trabajo ubicada en el hospital para que el equipo sanitario los analicen e informen al paciente en caso de cualquier descompensación.

Otro caso de éxito es BWell (García Esteller, David y González Ruiz, Mar), una plataforma que permite a los médicos supervisar el estado de salud de sus pacientes en tiempo real. Se pueden integrar múltiples dispositivos médicos equipados con Bluetooth Low Energy (BLE), que son los encargados de enviar los datos a las plataformas digitales de los distintos hospitales. Se ha llevado a cabo la monitorización de las oscilaciones de pacientes con insuficiencia cardíaca, a través de una báscula y un tensiómetro, ambos equipados con BLE. Esta plataforma consta de una herramienta de visualización online y de una aplicación móvil, promoviendo una comunicación fluida y colaborativa entre médicos y pacientes.

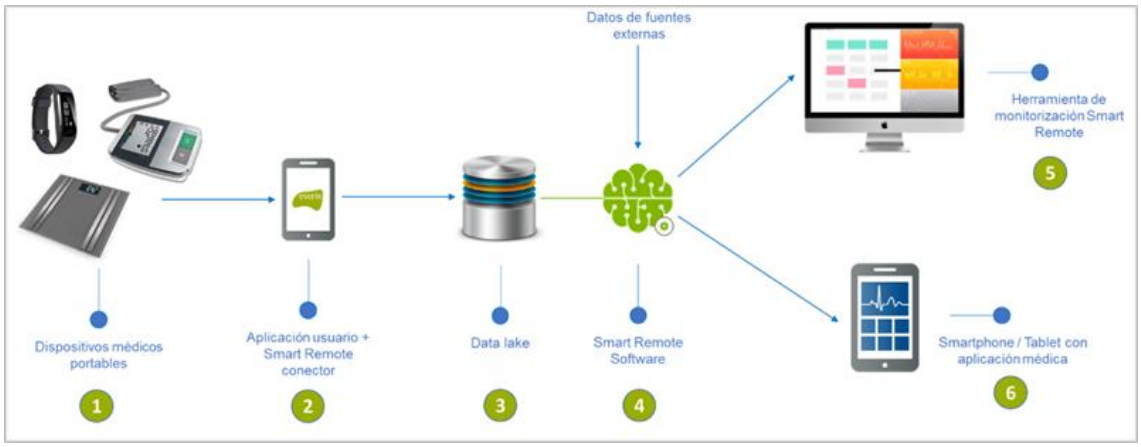


Figura 2.2. Características técnicas de BWell. (extraída de https://www.esmartcity.es/comunicaciones/comunicacion-bwell-sistema-integral-monitorizacion-pacientes-distancia)

Finalmente, la solución más avanzada disponible para cualquier usuario es MedM. Esta plataforma está disponible para dispositivos móviles, ya sean iOS o Android, y para dispositivos de escritorio, tanto Windows como Linux, además de disponer de una aplicación web. Es compatible con más de 170 dispositivos médicos equipados con BLE y puede monitorizar la presión arterial, nivel de oxígeno y glucosa en sangre, peso, temperatura, ritmo cardíaco, electrocardiogramas y frecuencia respiratoria.

Sin embargo, MedM está orientada a negocios de telemedicina a través de un plan mensual de suscripción que oscila entre los 100 USD para 10 licencias de usuario hasta 1000 USD para 500. Estos negocios son los encargados de supervisar la salud de los usuarios a los que le da una licencia. La aplicación no alerta sobre posibles riesgos, ni realiza un análisis de tus hábitos diarios.

Es aquí donde se quiere hacer hincapié en este trabajo: en aunar todas las aplicaciones que se encuentran en el mercado para monitorizar las constantes vitales de un usuario, con el fin de poder extraer relaciones entre ellas y conocer mejor su estado físico. Alertando siempre al usuario en caso de que sus mediciones se encuentren fuera del rango normal propio de cada constante vital. Sin la necesidad de estar afiliado a ningún negocio u entidad sanitaria responsable de supervisar esos datos.

2.2 Tecnologías empleadas

Diversos lenguajes de programación han sido útiles para la construcción de este proyecto, tales como JavaScript, HTML y CSS. Sin embargo, cabe destacar el uso mayoritario de los entornos de trabajo o marcos de trabajo (frameworks) para dichos lenguajes. El lenguaje de programación JavaScript ha sido el más utilizado, tanto para la parte del cliente como del servidor. Esto se debe al uso del framework Node.js, un entorno de ejecución de JavaScript. El segundo framework de JavaScript utilizado es Express.js, empleado para construir la infraestructura de la aplicación. La librería JavaScript React se ha utilizado para la parte gráfica de la aplicación. El sistema de base de datos elegido ha sido MongoDB, un sistema NoSQL que opera muy bien con Node.js. El uso de la API WebBluetooh ha hecho posible la comunicación de la aplicación web con los dispositivos Bluetooth.

2.2.1 Lenguajes de Programación

2.2.1.1 JavaScript

El lenguaje JavaScript, que se abrevia comúnmente como JS, es un lenguaje de programación interpretado. Esto significa que no requiere ser compilado, sino que es interpretado en tiempo real por un programa llamado intérprete. Por añadidura, esto aporta la capacidad de ser multiplataforma, ya que puede ser ejecutado en cualquier sistema operativo. Solo requiere que el programa intérprete esté presente en el sistema. JavaScript es dialecto del estándar ECMAScript, y desde 2012 todos los navegadores modernos soportan mínimo ECMAScript 5.1, una versión de JavaScript.

JavaScript fue desarrollado por Brendan Eich de la compañía Netscape en 1995, propietaria de un navegador con el mismo nombre. Este navegador está desaparecido a día de hoy, pero es el precursor del actual Firefox. JavaScript es una marca registrada de Oracle Corporation. La versión más actual del estándar es ECMAScript 7, publicada en Junio de 2016.

Se define también como un lenguaje orientado a objetos, basado en prototipos, imperativo y estructurado (es compatible con gran parte de la estructura de programación C). Como ocurre con la mayoría de los lenguajes de scripting es débilmente tipado y dinámico, el tipo está asociado al valor y no a la variable.

Se utiliza principalmente del lado del cliente, ejecutándose en contexto del cliente web (navegador). Es este último el encargado de soportar la carga de procesamiento y aportar los recursos necesarios para programar las aplicaciones. Gracias a este lenguaje podemos crear efectos especiales en las páginas (y dejen de ser estáticas) y definir interactividades con el usuario, además de poder acceder a recursos adicionales.

No obstante, JavaScript también puede ser usado del lado del servidor, formando parte como lenguaje de servidores tales como Node.js, Jaxer o RingoJS. De este modo, cuando un cliente solicita contenido, se ejecutan scripts en el servidor web. En este proyecto se utiliza JavaScript tanto en el lado del cliente como en el del servidor, y para ello utilizaremos el entorno de ejecución Node.js.

2.2.1.2 HTML

HTML, es un lenguaje de marcación que sirve para definir el contenido de las páginas web. Así lo indican sus siglas de Hypertext Markup Language o lenguaje de marcas de hipertexto. Estas marcas o etiquetas se utilizan para definir la estructura de un documento: cabecera, cuerpo, encabezados, párrafos, etc. Y mismamente se usan para definir su contenido: texto, imágenes, videos, juegos, etc.

Este lenguaje fue creado en 1991 por Tim Berners-Lee y es un estándar a cargo de la organización World Wide Web Consortium (W3C), la cual regula actualmente los estándares de casi todas las tecnologías relacionadas a la web. Todos los navegadores han adoptado este estándar y son los encargados de interpretar el código HTML y unir los elementos externos a la página (imagen, vídeo, script, etc.), cuya ubicación esta referenciada en el código HTML, para visualizar la página final.

La versión HTML 4.0 publicada en abril de 1998 introdujo la separación entre contenido y presentación, el contenido se mantiene definiendo en HTML mientras que la capa de presentación en el lenguaje CSS. Este lenguaje, por ende, es usado también en el proyecto. La versión más reciente de HTML, denominada HTML5, introdujo la estandarización de las API de JavaScript y facilitó el acceso a todo tipo de recursos adicionales: cámara, espacio para almacenamiento de datos, creación de gráficos basados en vectores y mapas de bits, flujos de datos con servidores, etc. Esta versión es la utilizada en el desarrollo de la aplicación web.

2.2.1.3 CSS

El lenguaje CSS, cuyas siglas en inglés provienen de Cascading Style Sheets o en español Hojas de estilo en cascada, es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado, tales como HTML o XML. Es por tanto muy usado para establecer el diseño visual de los documentos web e interfaces de usuario escritos en HTML, que junto a JavaScript (de forma opcional) otorgan una visualización muy atractiva.

Se define el cómo deben mostrarse los elementos de la página, su posición, forma, espaciados, bordes, colores, y en resumen, toda la parte estética. Esto se realiza a base de selectores, que seleccionan a qué elementos de la página nos referimos; atributos, que indican que propiedad del elemento seleccionado queremos cambiar; y valores, que indican el estilo a aplicar en unidades CSS (píxeles, puntos, etc.).

La especificación de CSS es también mantenida por el World Wide Web Consortium (W3C), la última versión disponible es CSS3. La especificación CSS describe además un sistema de prioridades para determinar que regla de estilo se aplica sobre un elemento si varias reglas coinciden sobre el mismo, el sistema de cascada (que da nombre al lenguaje). De este modo, las prioridades son calculadas y asignadas de una forma predecible para el desarrollador.

La separación entre el contenido (HTML) y presentación (CSS) implica muchas ventajas. A un mismo HTML podemos aplicarle diferentes hojas de estilo CSS, cambiando la visualización de la página sin necesidad de modificar el código HTML. Por lo que siempre que sea necesario hacer un cambio en el estilo de la página, alteraremos su código CSS sin variar la estructura y contenido de la página (código HTML).

2.2.2 Frameworks y API

2.2.2.1 Node.js

Node.js es un entorno de ejecución de JavaScript (runtime) de propósito general multiplataforma, de código abierto. Fue creado por Ralph Dalh en 2009, programador de la empresa Joyent la cual se convirtió en la propietaria. Está orientado a eventos asíncronos y fue diseñado para construir aplicaciones en red altamente escalables, tales como servidores web. Está basado en el motor V8 de Google, motor JavaScript open source del navegador Chrome.

Las operaciones computacionales que se realizan a través de las redes implementadas mediante hebras son relativamente ineficientes y difíciles de usar, por ello Node.js posee una única hebra de ejecución, usando entradas y salidas asíncronas que pueden ejecutarse concurrentemente sin costes en cambio de contexto. Esto convierte a este framework en una herramienta ideal para aplicaciones altamente concurrentes. Además, los procesos de bloqueo son inexistentes ya que casi ninguna función de Node.js realiza operaciones de Entrada/Salida.

Pero la ventaja principal que hemos aprovechado de este framework es la capacidad de hacer aplicaciones "*Full Stack*" con un solo lenguaje de programación (JavaScript), tanto en el lado del cliente como en el del servidor. Así como el desarrollo de la API REST, un servicio web que devuelve datos en formato JSON. También se puede construir aplicaciones de escritorio multiplataforma y programas de consola.

2.2.2.2 Express.js

Express es una infraestructura de aplicaciones web Node.js open source, mínima y flexible, que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Fue creada por TJ Holowaychuck en 2010, inspirada en el framework para Ruby, denominado Sinatra. Adquirida por StrongLoop y más tarde por IBM, actualmente se encuentra bajo la dirección de la fundación Node.js.

Las ventajas que aporta son enrutamientos de URL, facilidades para motores de plantillas (Jade, EJS, etc.), capa de middleware vía Connect, opciones para gestionar sesiones y cookies y una buena cobertura de pruebas.

2.2.2.3 React

React, o React.js, es una biblioteca JavaScript de código abierto que se centra en el desarrollo de interfaces de usuario. Es mantenido por Facebook y la comunidad de software libre. Y es que alrededor de React existe un completo ecosistema de módulos, herramientas y componentes que facilitan al desarrollador cumplir objetivos de dificultad avanzada con un esfuerzo relativamente pequeño.

React se presenta como una base sólida sobre la que construir cualquier idea usando JavaScript. Permite la asociación de la vista con los datos, de modo que si cambian los datos lo mismo ocurre con las vistas. Otra característica principal es el concepto "DOM Virtual". Se trata de una representación del DOM (Modelo de Objetos del Documento) del navegador en memoria, que aumenta el rendimiento de los componentes y aplicaciones de la interfaz. Cuando se actualiza la vista, se modifica el DOM Virtual, siendo este proceso más rápido que actualizar el DOM del navegador. React compara ambos DOM y se encarga de actualizar únicamente las partes del DOM que han sido modificadas en el virtual, evitando la necesidad de actualizar la vista entera.

2.2.2.4 Web Bluetooth API

Web Bluetooth API es una interfaz de programación de aplicaciones que proporciona la habilidad de conectarse e interaccionar con dispositivos Bluetooth Low Energy (BLE). Permite que aplicaciones web se comuniquen directamente con dispositivos Bluetooth

cercanos de una manera segura y preservando la privacidad. Hasta ahora, interactuar con dispositivos Bluetooth era solamente posible para aplicaciones nativas y no para navegadores. Con esta API, las páginas web pueden enviar y recibir datos directamente de un dispositivo emparejado. Es compatible con los navegadores Chrome, Edge y Opera, pero no lo es en cambio con Firefox.

Web Bluetooth API ha sido utilizada para establecer la comunicación entre los dispositivos de telemetría personal y la aplicación web desarrollada.

2.2.3 Herramientas

2.2.3.1 MongoDB

MongoDB es un sistema de bases de datos NoSQL, por lo que difiere del modelo clásico de Sistema de Gestión de Bases de Datos Relacionales (SGBDR). El aspecto más destacado es el no usar SQL como lenguaje principal de consultas. MongoDB es una base de datos orientada a documentos: en lugar de guardar los datos en registros, como se hace en una base de datos relacional, lo hace en documentos. Los documentos son archivos en formato BSON, una representación binaria de JSON.

Su esquema es dinámico, por lo que documentos de una misma colección pueden tener esquemas diferentes, no siendo necesario que sigan el mismo esquema como sí ocurría en una base de datos relacional. Su efecto es una integración de los datos más fácil y rápida. MongoDB está desarrollado en C++, no obstante las consultas se realizan usando como parámetros objetos de tipo JSON. Sin embargo, la consola desde la que se ejecutan los distintos comandos está construida con JavaScript, haciendo posible utilizar funciones propias de JavaScript además de las de MongoDB. Por añadidura, existen drivers para la gran mayoría de lenguajes. MongoDB dispone de un driver oficial para Node.js, el cual usaremos en este proyecto.

2.2.3.2 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft también disponible en Linux y macOS, lanzado en 2015. Sus características principales son depuración de programas, control integrado de Git, resaltado de sintaxis, finalización inteligente de código y refactorización del mismo. Es gratuito y de código abierto aunque la descarga oficial es bajo software privativo e incluye características personalizadas por Microsoft.

Se basa en Electron, un entorno de trabajo utilizado para desarrollar aplicaciones de escritorio implementadas con Node.js y Chromium. Tiene soporte nativo para JavaScript, TypeScript y Node.js. Se ha elegido este editor por su ligereza y potencia.

2.2.3.3 LightBlue

LightBlue es un producto de Punch Through, se trata de una aplicación desarrollada para iOS y Android enfocada a la depuración y prototipado de dispositivos Bluetooth Low Energy.

Permite crear dispositivos virtuales, y ha sido utilizado para simular el funcionamiento de dispositivos de telemetría personal Bluetooth usando el dispositivo móvil. Ofrece la posibilidad de crear dispositivos virtuales para la medición de la temperatura, frecuencia cardíaca, presión arterial y glucemia. Sin embargo, no ofrece esta funcionalidad para la medición del peso ni de la saturación de oxígeno en sangre.

3

Metodología

3.1 Metodología empleada

La metodología empleada en el desarrollo del TFG es Scrum, un marco de trabajo para desarrollo ágil de software. Se divide el trabajo en ciclos temporales denominados iteraciones (o sprints), donde en cada una de ellas, como resultado, se genera un producto software que integra nuevas funcionalidades con respecto a la anterior. Se mantiene una lista de objetivos/requisitos priorizada del producto y en cada sprint se seleccionan los más prioritarios que se prevé que se podrá completar en la iteración. La lista de objetivos/requisitos priorizada del producto es el Product Backlog, y el subconjunto de requisitos que serán desarrollados durante el siguiente sprint es el Sprint Backlog. En este último se subdividen los requisitos en tareas, a las cuales se le asignan ciertas horas de trabajo.

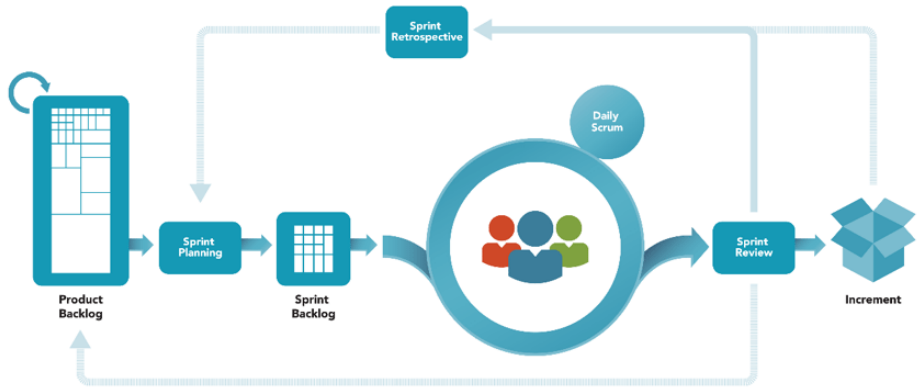


Figura 3.3. Procesos en la metodología ágil Scrum. (extraída de https://www.scrum.org/resources/blog/que-es-scrum)

Los beneficios de usar esta metodología son flexibilidad al cambio, reducción del Time to Market (se puede utilizar características del proyecto a medida que este avanza sin que sea necesario que esté completo), mayor calidad de software al requerir una versión

funcional después de cada iteración, mayor productividad al haber menos burocracia, predicciones de tiempos y reducción de riesgos.

3.2 Fases del trabajo

El trabajo se ha dividido en iteraciones (o sprints), de una duración de dos semanas cada uno, con una carga de alrededor a las 65 horas. Como resultado de cada sprint, se ha generado un producto software que integra las nuevas funcionalidades que dan nombre al sprint. El total de horas dedicado al proyecto es de 296 horas.

Sprint 1: Especificación de requisitos, diseño, arquitectura e implementación de la base de datos, registro e inicio de sesión.

- Especificación de requisitos: 20 horas.
- Diseño y arquitectura base de datos: 5 horas.
- Implementación base de datos: 20 horas.
- Diseño y arquitectura registro e inicio de sesión: 5 horas.
- Implementación registro e inicio de sesión: 15 horas.

Sprint 2: Diseño, arquitectura e implementación de la monitorización de la temperatura y del peso.

- Diseño y arquitectura monitorización de la temperatura: 5 horas.
- Implementación monitorización de la temperatura: 27 horas.
- Diseño y arquitectura monitorización del peso: 5 horas.
- Implementación monitorización del peso: 27 horas.

Sprint 3: Diseño, arquitectura e implementación de la monitorización del nivel de azúcar en sangre y del nivel de oxígeno en sangre.

- Diseño y arquitectura monitorización del nivel de azúcar en sangre: 5 horas.
- Implementación monitorización del nivel de azúcar en sangre: 27 horas.
- Diseño y arquitectura monitorización del nivel de oxígeno en sangre: 5 horas.
- Implementación monitorización del nivel de oxígeno en sangre: 27 horas.

Sprint 4: Diseño, arquitectura e implementación de la monitorización de la presión arterial, de la aplicación web para la consulta de datos y de la API REST.

- Diseño y arquitectura monitorización de la presión arterial: 5 horas.
- Implementación monitorización de la presión arterial: 27 horas.
- Diseño y arquitectura aplicación web: 5 horas.
- Implementación aplicación web: 25 horas.
- Diseño y arquitectura API REST: 5 horas.
- Implementación API REST: 5 horas.

Sprint 5: Pruebas del software y documentación.
- Pruebas: 15 horas.
- Documentación: 8 horas.

4

Análisis y especificación de requisitos

Esta fase se ha centrado en la obtención de información sobre el sistema, sus objetivos y alcance, especificando las funcionalidades que el sistema debe proporcionar. Esta especificación es un contrato en el que residen las características fundamentales del sistema. Los requisitos establecidos son funcionales o no funcionales, siendo funcionales si identifican los servicios que el sistema debe ofrecer y no funcionales si restringen el diseño e implementación del sistema. Los actores y participantes involucrados, directa o indirectamente, en el desarrollo y posterior uso del sistema también son identificados.

4.1 Actores y participantes

Se detallan a continuación los participantes (stakeholders) que son las partes interesadas e impactadas por la realización de este proyecto, y los actores que interactuarán directamente con el sistema.

4.1.1 Participantes

- Proveedores de servicios sanitarios. Cualquier sistema de asistencia sanitaria puede verse beneficiada del uso de este sistema web, dado que pueden ofrecer servicios de telemedicina a sus pacientes monitorizando su salud a través de este sistema.

- Personas que sufran de diabetes, hipoglucemia, hiperglucemia, hipertensión arterial, hipotensión, insuficiencia cardíaca, taquicardia, bradicardia, arritmia, asma, enfermedad pulmonar obstructiva crónica (EPOC) , infección respiratoria aguda y cualquier otra enfermedad relacionada con la temperatura, presión arterial, frecuencia cardíaca y nivel de azúcar en sangre.

- Personas que deseen monitorizar sus constantes vitales con el fin de obtener una mayor calidad de vida, alcanzar objetivos de bienestar personal, seguimiento de sus hábitos y actividades físicas obteniendo avisos cuando sus constantes alcanzan valores inadecuados.

4.1.2 Actores

- *Usuario*: persona que interactúa con la aplicación para monitorizar sus constantes vitales, configurando los dispositivos de telemetría personal para que envíen los datos a la aplicación web. El usuario podrá observar/obtener los datos mediante la aplicación web, donde se mostrarán de una forma significativa y útil para el usuario. También podrá acceder a ellos a través de la API REST.

4.2 Requisitos funcionales

Se detallan a continuación los requisitos funcionales, los cuales declaran los servicios que ofrecerá el sistema. Han sido divididos en grupos lógicos para agrupar funcionalidades relacionadas.

R.F.1 Registro de nuevos usuarios. El sistema debe permitir a un nuevo usuario registrarse en la aplicación.

R.F.2 Inicio de sesión de usuarios. El sistema debe autentificar a los usuarios y darle acceso únicamente al contenido en relación a su persona.

R.F.3 Cerrar sesión. El sistema debe permitir al usuario cerrar sesión.

R.F.4 Eliminar cuenta. El sistema debe permitir al usuario eliminar su cuenta, y por ende, todas las mediciones de constantes vitales tomadas.

R.F.5 Configuración de dispositivos de telemetría personal. El sistema debe permitir al usuario configurar los dispositivos de telemetría personal y permitirle:

- **R.F.5.1 Agregar un nuevo dispositivo.** Agregar un nuevo dispositivo para que se sincronice con el sistema.
- **R.F.5.2 Eliminar un dispositivo ya sincronizado.** Eliminar la sincronización con un dispositivo sin eliminar los datos que este ha registrado.

R.F.6 Gestionar las mediciones de constantes vitales. Las constantes vitales que monitoriza el sistema son: temperatura, peso, presión arterial, frecuencia cardíaca, nivel de azúcar en sangre y nivel de oxígeno en sangre. El sistema debe permitir al usuario:

- **R.F.6.1 Insertar** una medición de una constante vital tomada mediante un dispositivo de telemetría personal.
- **R.F.6.2 Mostrar** una medición de una constante vital que había sido registrada previamente.
- **R.F.6.3 Eliminar** una medición de una constante vital que había sido registrada previamente.

R.F.7 Mostrar las mediciones de una constante vital de forma significativa y útil para el usuario. El sistema debe permitir al usuario la lectura de todas las mediciones de una constante vital, siendo estas mostradas de una forma significativa y útil para el usuario.

- **R.F.7.1** Mostrar las mediciones de cada constante vital de forma aislada y agrupadas por día.
- **R.F.7.2** Mostrar las mediciones de cada constante vital de forma visual mediante una gráfica.

R.F.8 Obtener las mediciones de constantes vitales de un usuario mediante una API REST. El sistema dispondrá de una API que permitirá al usuario obtener las mediciones de sus constantes vitales según el estándar REST, haciendo dichos datos recuperables para su uso en otros servicios o aplicaciones web.

- **R.F.8.1** Para mayor facilidad, la API REST permitirá el registro de un nuevo usuario.

R.F.9 Alertar al usuario en caso de posible riesgo de su salud. El sistema monitorizará las mediciones de las constantes vitales del usuario y en caso de observar una anomalía en alguna de ellas notificará al usuario.

- **R.F.9.1** Las alertas estarán diferenciadas mediante una escala de gravedad de la anomalía.
- **R.F.9.2** Las alertas de días anteriores deberán ser accesibles.

4.3 Requisitos no funcionales

Los requisitos no funcionales que imponen restricciones en el diseño y la implementación del sistema son los siguientes.

R.N.F.1 Bluetooth Low Energy. Los dispositivos telemétricos deberán comunicarse usando el estándar Bluetooth Low Energy (BLE).

R.N.F.2 Conexión a internet. El sistema necesita de una conexión a internet para el acceso al sistema y a la base de datos, así como para modificaciones en esta última a la hora de añadir/eliminar una medición.

R.N.F.3 Sistema multiplataforma y multidispositivo. El sistema debe ser accesible desde cualquier plataforma y dispositivo.

R.N.F.4 Seguridad de los datos. El sistema debe almacenar los datos de forma segura y restringir el acceso de modo que un usuario solo pueda acceder a sus propios datos.

5

Diseño y arquitectura

En esta sección se describe el sistema, su diseño y arquitectura, incorporando consideraciones de la implementación tecnológica. Se han producido varios modelos del sistema sirviendo de guía del diseño de la aplicación, además de seguir un conjunto de patrones y abstracciones para la arquitectura del sistema.

5.1 Modelo conceptual

En este apartado se muestra un diagrama de clases que representa a modo conceptual el dominio del sistema web. En primer lugar, se ha realizado un estudio de las variables que influyen en la medición de cada constante vital, con el objetivo de diseñar el sistema acorde a las necesidades de cada una. Es necesario saber si la medición de una constante resulta en un solo valor, dos o incluso más. La medición también puede verse afectada según ciertas condiciones, como si ha sido realizada en reposo, después de comer, haciendo ejercicio, etc.

La medición de la temperatura, peso y el nivel de oxígeno en sangre resulta en un solo valor y no hay variables a tener en cuenta. No ocurre lo mismo con las demás constantes, que requieren más características. Se explica a continuación con detalle la estructura del sistema y cuáles son estas características específicas de cada constante vital.

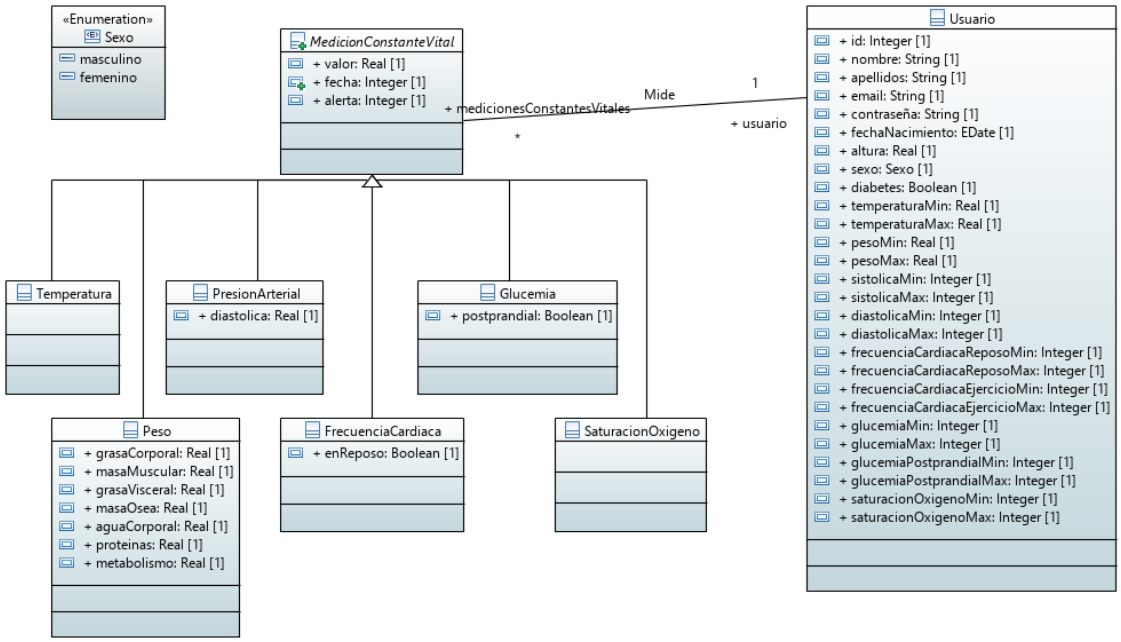


Figura 4.4. Modelo Conceptual del sistema.

Clase MedicionConstanteVital

Representa una medición de una constante vital y tiene los siguientes atributos:

- valor: propiedad de tipo Real que representa el resultado de la medición. Es de tipo Real dado que hay constantes vitales que se miden con precisión de decimales.
- fecha: propiedad de tipo Integer que representa la fecha en la que fue tomada la medición. Es de tipo Integer porque la fecha se va a almacenar como el número de milisegundos desde el 1 de Enero de 1970 hasta la fecha actual (JavaScript facilita las fechas en este formato).
- alerta: propiedad de tipo Integer que representa el nivel de alerta de la medición. El valor 0 indicará que la medición está dentro del rango normal, 1 que se encuentra fuera de este rango y 2 que se encuentra de forma alarmante fuera del rango normal.

Esta clase es abstracta debido a que para crear una medición debemos aclarar de que constante vital se trata, creando una instancia de una subclase. Todas las mediciones de constantes vitales son subclases de MedicionConstanteVital y heredan los atributos valor, fecha y alerta. Algunas de estas subclases adicionalmente añaden más atributos.

Clase Temperatura

Subclase de MedicionConstanteVital que no requiere de más atributos que los heredados (valor, fecha, alerta). El resultado de medir la temperatura en grados centígrados se almacena en el atributo heredado valor.

Clase Peso

Subclase de MedicionConstanteVital que además de los atributos heredados fecha, alerta y valor (donde se almacena el peso en kilogramos), consta de los siguientes atributos:

- grasaCorporal: propiedad de tipo Real que representa la proporción de grasa corporal respecto al peso corporal del usuario.
- masaMuscular: propiedad de tipo Real que representa la proporción de masa muscular respecto al peso corporal del usuario.
- grasaVisceral: propiedad de tipo Real que representa la grasa visceral del usuario en kilogramos.
- masaOsea: propiedad de tipo Real que representa la proporción de masa ósea respecto al peso corporal del usuario.
- aguaCorporal: propiedad de tipo Real que representa la proporción de agua corporal respecto al peso corporal del usuario.
- proteinas: propiedad de tipo Real que representa la proporción de proteínas respecto al peso corporal del usuario.
- metabolismo: propiedad de tipo Real que representa el metabolismo basal del usuario en calorías/m^2/hora.

Clase PresionArterial

La lectura de la presión arterial total se determina mediante la medición de las presiones arteriales sistólica y diastólica. La presión arterial sistólica, el valor superior, mide la fuerza que ejerce el corazón sobre las paredes de las arterias cada vez que late. La presión arterial diastólica, el valor inferior, mide la fuerza que ejerce el corazón sobre las paredes de las arterias entre cada latido. Por ello, el sistema almacena dos valores en cada medición de la presión arterial. Al ser una subclase de MedicionConstanteVital hereda el atributo valor, donde se almacena la presión arterial sistólica medida en milímetros de mercurio (mmHg). La clase PresionArterial añade el atributo diastolica donde se almacena la presión arterial diastólica en mmHg.

Clase FrecuenciaCardiaca

Subclase de MedicionConstanteVital que presenta el atributo heredado valor donde se almacena la frecuencia cardíaca medida en pulsaciones por minuto (ppm). No obstante, la frecuencia cardíaca debe diferenciarse si ha sido tomada en reposo o al realizar ejercicio físico, ya que esta aumenta durante el ejercicio y su rango de valores normales es diferente. Con este fin se añade el atributo enReposo de tipo Boolean, que en caso ser *true* indica que la medición fue realizada en reposo y en caso de ser *false* que la medición fue tomada durante la realización de ejercicio físico o justo después de este.

Clase Glucemia

Subclase de MedicionConstanteVital que presenta el atributo heredado valor donde se almacena el nivel de azúcar o glucosa en sangre, clínicamente denominado glucemia, medida en mg/dL. La glucemia varía a lo largo del día y en función de los alimentos ingeridos. Por la mañana en ayunas son más bajos y se elevan después de cada comida

y vuelven a descender dos horas más tarde. Es por ello necesaria la distinción en nuestro sistema de la glucemia y la glucemia postprandial, siendo esta última la medición de glucosa en sangre después de una comida. Esta distinción es posible con el atributo postprandial de tipo Boolean, que si tiene valor *true* indica que es postprandial.

Clase SaturacionOxigeno

Subclase de MedicionConstanteVital que no requiere de más atributos. El resultado de medir el nivel de oxígeno en sangre, o saturación de oxígeno, en SpO2 (Saturación porcentual de Oxígeno) se almacena en el atributo heredado valor.

Clase Usuario

Representa a cualquier usuario del sistema que desee monitorizar sus constantes vitales. Los atributos que la componen son:

- id: propiedad de tipo Integer que sirve de identificador para cada usuario.
- nombre: propiedad de tipo String que representa el nombre del usuario.
- apellidos: propiedad de tipo String que representa los apellidos del usuario.
- email: propiedad de tipo String que representa el correo electrónico del usuario. Sirve de acceso para el sistema por lo que debe de ser único.
- contraseña: propiedad de tipo String que guarda la contraseña del usuario para el acceso.
- fechaNacimiento: propiedad de tipo Date que almacena la fecha de nacimiento del usuario. De esta forma podemos saber la edad en cualquier momento futuro.
- altura: propiedad de tipo Real que representa la altura del usuario en metros.
- sexo: propiedad del tipo enumerado Sexo que puede tomar los valores masculino y femenino indicando el sexo del usuario.
- diabetes: propiedad de tipo Boolean que indica si el usuario padece diabetes (*true*) o no (*false*). Para una persona que padezca diabetes, su rango de valores adecuados de glucemia es diferente, por lo que el sistema debe ser conocedor de si el usuario lo padece o no, y ajustar los rangos apropiados para este caso.
- Para cada constante vital se requiere de un rango de valores aceptables, comprendidos entre un mínimo y un máximo. Como funcionalidad adicional del sistema, este almacena para cada usuario un rango de valores personalizado para cada constante vital, con el fin de que los rangos puedan ser modificados para satisfacer las necesidades específicas de cada usuario. Esta es la finalidad de los atributos restantes de la clase usuario, como ejemplo el rango entre pesoMin y pesoMax son los valores que el usuario considera adecuados para sus necesidades específicas (ya sea por una meta personal o por recomendación de su médico). Lo mismo se extrapola a las demás constantes vitales, que pueden requerir de un rango de valores aceptables por el sistema diferentes al establecido. Estos atributos son: temperaturaMin, temperaturaMax, pesoMin, pesoMax, sistolicaMin, sistolicaMax, diastolicaMin, diastolicaMax, frecuenciaCardiacaReposoMin, frecuenciaCardiacaReposoMax, frecuenciaCardiacaEjercicioMin, frecuenciaCardiacaEjercicioMax, glucemiaMin, glucemiaMax, glucemiaPosprandialMin, glucemiaPostprandialMax, saturacionOxigenoMin y saturacionOxigenoMax.

El usuario se relaciona con sus mediciones de constantes vitales mediante la relación "Mide", entre su clase Usuario y las subclases de MedicionConstanteVital.

5.2 Casos de uso

Los casos de uso son una descripción de las interacciones entre el sistema y un actor u otro sistema, con el fin de conseguir un objetivo específico. Se usa un lenguaje cercano al usuario final, evitando la jerga técnica. Son útiles para el diseño del sistema, además de para la captura de requisitos, puesto que se detallan en alto nivel los pasos que deben suceder para llevar a cabo una tarea. Cada caso de uso detalla un escenario principal y posibles escenarios alternativos que logran el mismo objetivo o terminan el caso de uso sin alcanzarlo.

Título	CU01 - Registro de un nuevo usuario (R.F.1)
Descripción	Un nuevo usuario se registra en el sistema, aportando sus datos de acceso y datos de su condición física.
Pre-condición	El sistema se encuentra en la página de Registro Usuario
Escenario principal	
1. El usuario introduce los datos pertinentes: • Nombre • Apellidos • Email • Contraseña • Repetir contraseña • Fecha Nacimiento • Altura • Sexo • Diabetes 2. El usuario hace clic en "Registrarse". 3. El sistema comprueba que los datos son correctos, esto es: ningún campo está vacío exceptuando diabetes que puede estarlo, el email no se encuentra en el sistema, los campos contraseña y repetir contraseña coinciden, fecha de nacimiento es una fecha válida y altura es un dato numérico comprendido entre un rango aceptable. 4. El sistema almacena los datos. 5. El sistema muestra la interfaz principal de usuarios.	
Escenario alternativo	
[1-2]A El usuario hace clic en el botón de retroceder. [1-2]A.1 El sistema muestra la interfaz Inicio de sesión. [1-2]A.2 El sistema se encuentra en el caso de uso CU02 - Inicio de sesión de usuario. 3A Algún campo está vacío (exceptuando diabetes que puede estarlo). 3A.1 El sistema muestra un mensaje con este error. 3A.2 El sistema vuelve al paso 1. 3B El email se encuentra en el sistema. 3B.1 El sistema muestra un mensaje con este error. 3B.2 El sistema vuelve al paso 1. 3C Los campos contraseña y repetir contraseña no coinciden. 3C.1 El sistema muestra un mensaje con este error.	

3C.2 El sistema vuelve al paso 1.

3D El campo de fecha de nacimiento y/o altura no es válido.

3D.1 El sistema muestra un mensaje con este error.

3D.2 El sistema vuelve al paso 1.

Maqueta de interfaz

Título	CU02 - Inicio de sesión de un usuario. (R.F.2)
Descripción	Un usuario inicia sesión en el sistema, escribiendo sus datos de acceso y accediendo a la interfaz principal del sistema.
Pre-condición	El usuario se encuentra en la página Inicio de Sesión

Escenario principal

1. El usuario introduce los datos pertinentes:
   - Email
   - Contraseña
2. El usuario hace clic en "Iniciar Sesión".
3. El sistema comprueba que los datos son correctos, esto es: ningún campo está vacío, el email se encuentra en el sistema y la contraseña corresponde con la almacenada en el sistema para ese email.
4. El sistema muestra la interfaz principal.

Escenario alternativo

[1-2]A El usuario hace clic en "Registro Usuario".

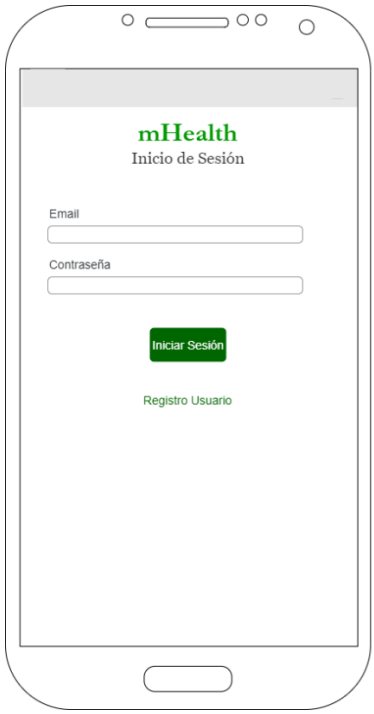
[1-2]A.1 El sistema muestra la interfaz Registro Usuario.

[1-2]A.2 El sistema se encuentra en el caso de uso CU01 - Registro de un nuevo usuario.

3A Algún campo está vacío.

3A.1 El sistema muestra un mensaje con este error.

3A.2 El sistema vuelve al paso 1.

3B El email no se encuentra en el sistema. 3B.1 El sistema muestra un mensaje con este error. 3B.2 El sistema vuelve al paso 1. 3C La contraseña no coincide con la almacenada en el sistema para el email introducido. 3C.1 El sistema muestra un mensaje con este error. 3C.2 El sistema vuelve al paso 1.
Maqueta de interfaz


Título	CU03 - Gestionar las mediciones de constantes vitales (R.F.6)
Descripción	Un usuario gestiona las mediciones de constantes vitales, observándolas, insertando nuevas conectando su dispositivo de telemetría personal Bluetooth o eliminándolas.
Pre-condición	El usuario ha iniciado sesión.
Escenario principal	
1. El usuario hace clic en una constante vital. 2. El sistema muestra las mediciones de la constante vital seleccionada para la fecha seleccionada (la fecha de hoy por defecto). 3. El usuario hace clic en "Conectar" (Botón solo visible si la fecha seleccionada es la actual y no una anterior/posterior) 4. El sistema muestra una lista de los dispositivos Bluetooth encontrados capacitados para enviar mediciones de la constante vital seleccionada. 5. El usuario hace clic en el nombre del dispositivo Bluetooth desde el que desee enviar los datos. 6. El sistema obtiene los datos de las mediciones del dispositivo Bluetooth y los muestra en la gráfica. 7. El usuario desconecta/apaga su dispositivo de telemetría personal Bluetooth.	

8. El sistema cesa de obtener datos, y por ende, de mostrar nuevas mediciones, aunque mantiene las ya registradas.

Escenario alternativo

[1-5]A El usuario hace clic en "Cerrar Sesión"

[1-5]A.1 El sistema se encuentra en el caso de uso CU02 - Inicio de sesión de un usuario.

[2-5]A El usuario hace clic en otra constante vital.

[2-5]A.1 El sistema vuelve al paso 1 para la nueva constante vital seleccionada.

[2-5]B El usuario cambia la fecha.

[2-5]B.1 El sistema vuelve al paso 2.

[2-5]C El usuario hace clic en "Eliminar"

[2-5]C.1 El sistema elimina las mediciones de la constante vital seleccionada para la fecha seleccionada.

[2-5]C.2 El sistema vuelve al paso 2.

[6-7]A El usuario hace clic en otra constante vital.

[6-7]A.1 El sistema vuelve al paso 2 para la nueva constante vital seleccionada.

[6-7]A.2 El sistema continúa obteniendo datos del dispositivo Bluetooth hasta que el usuario lo desconecte/apague o conecte otro dispositivo Bluetooth.

[6-7]B El usuario cambia la fecha.

[6-7]B.1 El sistema vuelve al paso 2.

[6-7]B.2 El sistema continúa obteniendo datos del dispositivo Bluetooth hasta que el usuario lo desconecte/apague o conecte otro dispositivo Bluetooth.

[6-7]C El usuario hace clic en "Eliminar".

[6-7]C.1 El sistema elimina las mediciones de la constante vital seleccionada para la fecha seleccionada.

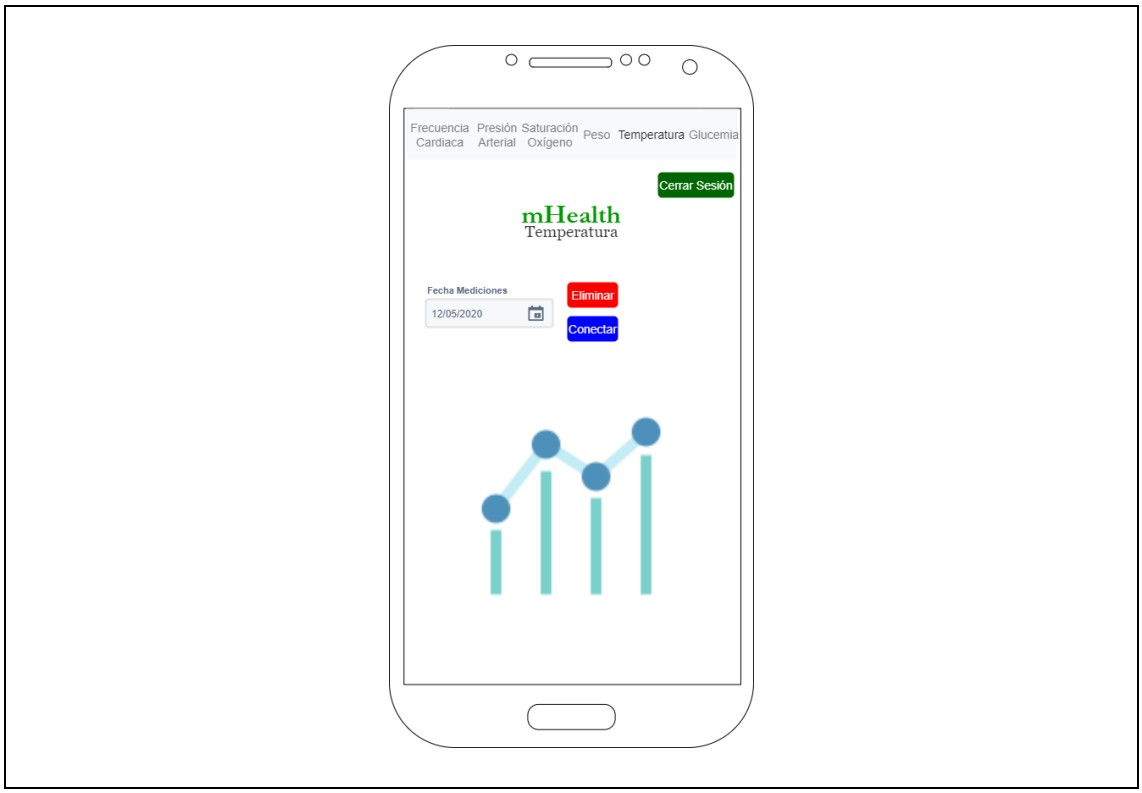
[6-7]C.2 El sistema vuelve al paso 6.

[6-7]D El usuario hace clic en "Cerrar Sesión"

[6-7]D.1 El sistema cesa de obtener datos.

[6-7]D.2 El sistema se encuentra en el caso de uso CU02 - Inicio de sesión de un usuario.

Maqueta de interfaz



5.3 Arquitectura

5.3.1 Modelo Cliente - Servidor

Es uno de los principales modelos usados en la programación de servicios de Internet. Está compuesto en dos grandes partes como indica su nombre, el cliente (o grupo de clientes) y el servidor.

El cliente requiere, para cualquier acción que implique manipulación de datos, ya sea obtenerlos, editarlos, almacenar nuevos datos, eliminarlos, etc., de una comunicación con el servidor; este último será el encargado de responder a estas acciones. Esto se debe a que el servidor es el responsable del depósito de los datos, funcionando como un sistema gestor de base de datos encargado de dar la respuesta demandada por el cliente.

Gracias a este modelo, la carga computacional de las operaciones es llevada por el servidor, siendo esta máquina de alto nivel con un hardware y software específico. Libera al usuario (cliente) de esta carga, quien principalmente se responsabiliza de hacer las peticiones y es el servidor el que responde a cada una de ellas realizando las operaciones necesarias.

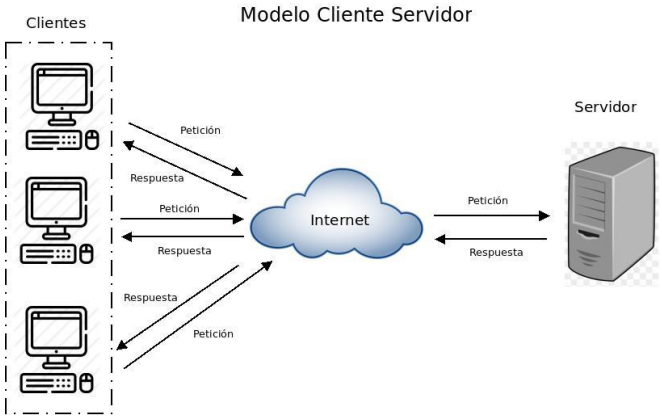


Figura 5.5 Comunicación entre Cliente y Servidor mediante peticiones-respuestas (extraída de https://blog.infranetworking.com/modelo-cliente-servidor/).

La comunicación entre el cliente y el servidor debe ser establecida bajo una interfaz conocida por ambos, de forma que puedan procesar adecuadamente las peticiones y respuestas. La interfaz usada en este proyecto es REST.

5.3.2 REST

REST (Representional State Transfer) es un término definido por Roy Fielding en el año 2000, quien también especificó HTTP. Se trata de un conjunto de restricciones con las que crear un estilo de arquitectura software, y usar este estilo para crear aplicaciones web respetando el estándar HTTP. Una API REST proporciona la posibilidad de crear servicios a partir de ese software, obteniendo crecimiento horizontal. Es un estándar lógico y eficiente para la creación de servicios web.

Las restricciones que definen REST incluyen el uso del protocolo cliente/servidor sin estado, donde cada petición HTTP contiene toda la información necesaria para ejecutarla, sin necesidad de guardar información de estados previos. Aunque se puede incorporar memoria caché con el objetivo de que el cliente en peticiones idénticas pueda ejecutar la misma respuesta. La interfaz debe ser uniforme, con el uso de las cuatro operaciones simples relacionadas con el manejo de los datos: POST, GET, PUT y DELETE y cada recurso está identificado con una URL. El sistema es por capas, una arquitectura jerárquica entre los componentes que facilita la escalabilidad, rendimiento y seguridad.

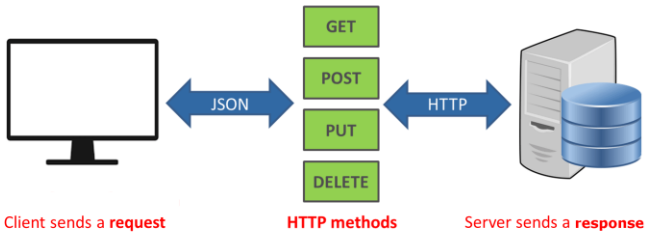


Figura 6.6 Flujo de datos entre Cliente y Servidor usando API REST (extraída de https://aprendiendoarduino.wordpress.com/2019/10/27/api-rest/).

La API REST creada en este proyecto con Node.js implica que está implementada con JavaScript, y las repuestas a las peticiones son en lenguaje JSON.

6

Implementación

6.1 Estudio y asignación de valores normales y críticos para cada constante vital.

Para cada constante vital se ha estudiado su rango de valores aceptables, así como unos valores críticos mínimo y máximo que indican que la medición de la constante vital es muy alarmante y debe acudir a un centro médico inmediatamente. Se muestra a continuación el estudio para cada una de las constantes vitales y los valores que han sido elegidos para ser asignados en el sistema.

La temperatura corporal considerada correcta varía según la persona, edad, actividades que realice y el momento del día, no obstante la temperatura promedio aceptada es de 37°C. Algunos estudios han mostrado que la temperatura corporal normal puede tener un amplio rango que va desde los 36.1°C hasta los 37.2°C, valores que han sido asignados como mínimo y máximo y en caso de no encontrase la temperatura del usuario entre esos valores será alertado. Una temperatura superior a 38°C suele indicar fiebre a causa de una infección o enfermedad. Cuando la temperatura es igual o superior a 41°C, conduce a la hipertermia, un fracaso del centro termorregulador que puede ocasionar importantes lesiones orgánicas. Se ha asignado 40°C como máximo crítico con el fin de alertar al usuario de forma crítica antes de alcanzar los 41°C y prevenir que llegue a ese estado. Por otro lado, cuando se sitúa por debajo de 35°C se considera patología, tratándose de hipotermia, y ha sido establecido como mínimo crítico del sistema.

El peso ideal se calcula con el Índice de Masa Corporal (IMC), una medida que nos ayuda a detectar si nos encontramos dentro de los valores saludables o no. El IMC se calcula dividiendo el peso en kilogramos por el cuadrado de la altura en metros (kg/m^2). La Organización Mundial de la Salud estima que un peso saludable es el que se sitúa entre valores de 18,5 y 24,9. Menor a 17 es delgadez moderada y superior a 30 es obesidad,

por lo que estos valores serán tomados como mínimo crítico y máximo crítico respectivamente. Estos rangos son solo válidos para adultos a partir de 20 años.

La frecuencia cardíaca normal en reposo oscila entre 60 y 100 pulsaciones por minuto (ppm) para personas mayores de 10 años, incluyendo a los adultos más mayores. Los atletas altamente entrenados pueden tener una frecuencia cardíaca inferior a 60 ppm, llegando incluso a los 40ppm. Es por esto, que se ha considerado 40ppm como mínimo crítico para el sistema.
La frecuencia cardíaca aumenta con el ejercicio y su nivel de intensidad, para el cálculo del rango de los valores normales durante el ejercicio se ha utilizado la frecuencia cardíaca máxima (FCM). Esta medida se calcula de forma aproximada restando la edad del usuario a 220, y es un límite teórico en el que no se compromete la salud durante una actividad física. Se considera que habitualmente una mujer tiene de media entre 5 y 15 ppm más que el hombre, por lo que la FCM para una mujer se considera más exacta restando su edad a 226, en vez de 220.
Para el ejercicio de intensidad moderada, la frecuencia cardíaca ideal debe ser de 50% a 70% de la frecuencia cardíaca máxima. Para el ejercicio vigoroso, la frecuencia cardíaca debe ser de 70% a 85% de la FCM.
Se ha establecido en nuestro sistema el rango de 50% de FCM a 85% de FCM como el adecuado mientras se realiza una actividad física, diferenciando el sexo del usuario para que sea más exacto el rango. El valor crítico máximo se ha fijado en 120 ppm en reposo, si supera esta cifra se debe consultar al médico ya que a partir de 130 ya se considera taquicardia. Durante la realización de ejercicio físico el valor fijado es de la FCM del usuario.

La lectura de la presión arterial total se determina mediante la medición de las presiones arteriales sistólica y diastólica. La presión arterial sistólica debe situarse entre los valores de 90 mmHg y 120 mmHg (milímetros de mercurios). Si el valor es superior a 130 se considera hipertensión, por lo que se ha establecido como valor crítico máximo. La presión arterial diastólica debe situarse entre los valores de 60 mmHg y 80mmHg. Si el valor es superior a 80 se considera hipertensión mismamente.

El nivel de azúcar o glucosa en sangre, clínicamente denominado glucemia, varía a lo largo del día y en función de los alimentos ingeridos. Nuestro sistema distingue entre glucemia (basal) y glucemia postprandial. Lo recomendable es que la glucemia se mida por la mañana antes del desayuno (glucemia basal), y se considera un valor aceptable si se encuentra entre los 70 y 100 mg/dL; y dos horas después de cada comida (glucemia postprandial), siendo un valor adecuado si se encuentra por debajo de 140 mg/dL. La voz de alarma debe saltar cuando los niveles de glucosa en sangre estando en ayunas se sitúen entre 100 y 125 mg/dL y después de comer entre los 140 y los 199 mg/dL. El sistema alertará en dichos casos, pero además se ha establecido 125 mg/dL como máximo crítico en ayunas y 200 mg/dL como máximo después de comer. Además, una glucemia menor a 54 mg/dL indica hipoglucemia, por lo que se ha establecido como mínimo crítico en ambas condiciones.

Si bien estos valores son apropiados, no lo son para una persona que padezca diabetes. La American Diabetes Association (ADA) recomienda que, en caso de diabetes, el nivel de glucosa antes de comer debería estar entre 80 y 130 mg/dL y después de comer

(postprandial) debería ser menor a 180 mg/dL. Una glucemia superior a 250 mg/dL indica hiperglucemia, por lo que este valor será tomado como máximo crítico para los usuarios diabéticos.

Finalmente, acerca el nivel de oxígeno en sangre o saturación de oxígeno, se considera que el porcentaje adecuado y saludable es de entre el 95% y el 100%. Cuando se encuentra por debajo del 90% se produce hipoxemia, siendo un síntoma característico la dificultad para respirar.

A continuación se muestra una tabla que esquematiza el rango de valores aceptables de cada constante vital, es decir, aquellos que se encuentran entre el mínimo y el máximo. Asimismo, un valor crítico mínimo y máximo que en caso de superarlo o rebajarlo el sistema mostrará una alerta que indique un nivel de amenaza mucho más alto a la salud del usuario, advirtiendo de la necesidad de acudir a un centro médico.

Constante vital	Crítico Mínimo	Mínimo	Máximo	Crítico Máximo
Temperatura	35°C	36.1°C	37.2°C	40°C
Peso	17*altura^2 kg	18.5*altura^2 kg	24.9*altura^2 kg	30*altura^2 kg
Presión arterial sistólica	---	90 mmHg	120 mmHg	130 mmHg
Presión arterial diastólica	---	60 mmHg	80 mmHg	81 mmHg
Frecuencia cardíaca (Reposo)	40 ppm	60 ppm	100 ppm	120 ppm
Frecuencia cardíaca (Ejercicio)	60 ppm	Hombre: (220-edad)*0.5 ppm Mujer: (226-edad)*0.5 ppm	Hombre: (220-edad)*0.85 ppm Mujer: (226-edad)*0.85 ppm	Hombre: 220-edad ppm Mujer: 226-edad ppm
Glucemia (Ayuno)	54 mg/dL	70 mg/dL	100 mg/dL	125 mg/L
Glucemia Postprandial	54 mg/dL	70 mg/dL	140 mg/dL	200 mg/L
Glucemia (Ayuno)(Diabetes)	54 mg/dL	80 mg/dL	130 mg/dL	180 mg/L
Glucemia Postprandial (Diabetes)	54 mg/dL	80 mg/dL	180 mg/dL	250 mg/L
Saturación oxígeno	90 SpO2	95 SpO2	100 SpO2	---

Estos valores establecidos serán usados para monitorizar las constantes vitales del usuario.

6.2 Estructura del proyecto

En esta sección se explica la estructura del proyecto, compuesta por dos aplicaciones: el backend desarrollado con Node.js y Express, y el frontend una aplicación React.

El backend es el servidor, y ha sido desarrollado para que sea en sí una API REST. No necesita del frontend para ser ejecutado, se pueden hacer peticiones al servidor desde una consola o desde cualquier otro servicio, siempre y cuando el URL y los parámetros de la petición sean los correctos. El servidor ha sido configurado para aceptar peticiones en el puerto 5000.

El frontend es el cliente, este cliente ha sido construido para que el usuario pueda comunicarse con el servidor de forma sencilla, visual e interactiva. Se trata de una aplicación React. El cliente es a la vez la aplicación web móvil (puede ser utilizada en cualquier dispositivo) que facilita la configuración de dispositivos de telemetría personal y/o gadget para la recepción de datos correspondientes a constantes vitales, como la aplicación web que permite la búsqueda, recuperación y análisis de los datos. El usuario puede acceder desde cualquier dispositivo a la aplicación web para simplemente visualizar los datos, por lo que tiene sentido aunar ambas funciones en una aplicación web.

Para redireccionar las peticiones del frontend al servidor creado, se ha añadido la siguiente línea en el fichero package.json:

"proxy": "http://localhost:5000"

Se ha instalado el módulo **concurrently** con el fin de que mediante un comando se ejecuten tanto servidor y cliente en la misma consola. El comando *npm start dev* que equivale a *concurrently --kill-others-on-fail "npm run server" "npm run client"* es el encargado de esta función.

También se ha instalado el módulo **nodemon**, cuya función es ayudar al desarrollo de código reiniciando la aplicación automáticamente cuando se detectan cambios en los ficheros del directorio. Evita el proceso manual de reiniciar la aplicación para cualquier pequeño cambio.

Desde el servidor podemos acceder al cuerpo del mensaje de la petición escribiendo *req.body* gracias a la presencia del módulo instalado **body-parser**, el cual actúa de middleware procesando los datos enviados en cada petición antes de que llegue a la función definida para cada ruta. Se han escrito las siguientes líneas de código para que los formatos de petición JSON y x-www-form-urlencoded sean aceptados (comprensibles) por el sistema:

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

En el directorio del proyecto podemos encontrar una carpeta “routes” con los métodos de ruta a la aplicación aceptados y la respuesta que da el servidor en cada caso. La carpeta “utils” contiene varias funciones que son útiles para los métodos de ruta. En la carpeta “models” nos encontramos con los modelos de datos de la base de datos. El fichero package.json contiene las dependencias del proyecto y el fichero app.js es el encargado de iniciar el servidor. La carpeta “tests” contiene varios ficheros con pruebas a los métodos de ruta.

En el mismo directorio del proyecto, nos encontramos con la carpeta de “client”, donde está el código de la aplicación cliente. El fichero de inicio es index.html que se encuentra dentro de la carpeta “public”, aunque el fichero index.js de la carpeta “src” es ejecutado seguidamente renderizando los componentes de React que se encuentran en la misma carpeta.

6.3 Base de datos

La base de datos es, como ya se ha descrito, MongoDB: un sistema de bases de datos NoSQL, orientada a documentos. Los documentos se guardan dentro de colecciones de documentos. Su esquema es dinámico, por lo que documentos de una misma colección pueden tener esquemas diferentes.

Para restringir la dinamicidad de su esquema y facilitar la creación y gestión de esta base de datos, se ha instalado la biblioteca de JavaScript **Mongoose**. Mongoose es un Object Document Mapper (ODM) que asigna a un documento MongoDB un objeto con un esquema fuertemente tipado.

Utilizando esta biblioteca se han creado los modelos diseñados en la fase anterior. En Mongoose no hay clases abstractas y subclases, el mecanismo de herencia es el uso de Discriminators. Este mecanismo permite tener múltiples modelos con esquemas superpuestos bajo la misma colección de documentos. Al modelo *MedicionConstanteVital* se le ha asignado como *discriminatorKey* el atributo *tipoConstanteVital*, lo que significa que modelos diferentes (de constantes vitales diferentes) pueden almacenarse bajo la misma colección de *MedicionConstanteVital* indicando en este atributo que modelo es (que constante vital es).

```
const options = { discriminatorKey: 'tipoConstanteVital' };

const medicionConstanteVitalSchema = new Schema({
  valor: { type: Number, required: true },
  fecha: { type: Number, required: true },
  alerta: { type: Number },
  usuario: { type: String, required: true },
},
options);

const MedicionConstanteVital = mongoose.model('MedicionConstanteVital', medicionC
onstanteVitalSchema);


MedicionConstanteVital.discriminator('PresionArterial',
  new Schema({
    diastolica: Number,
  },
  options));
```

Para el enumerado Sexo, JavaScript no dispone de forma nativa de una implementación de tipo enumerado. Sin embargo, es fácil modelar una con un objeto estándar.

Combinando esto con Mongoose, que sí dispone de una notación para enumerados, se ha desarrollado el enumerado de Sexo.

```
const Sexo = Object.freeze({
  Masculino: 'masculino',
  Femenino: 'femenino',
});


const UsuarioSchema = new Schema({
  sexo: {
    type: String,
    enum: Object.values(Sexo),
  },
})
```

6.4 Registro/Inicio de sesión

El sistema web trata con datos personales de personas, y lo que es más, con datos relativos a su salud, que son especialmente sensibles legalmente. Es necesaria una protección de estos datos y una forma de autentificar al usuario para el acceso único a sus datos.

JSON Web Tokens (JWT) es un estándar para establecer una sesión de usuario entre dos partes de forma segura. Se trata de un objeto de tipo JSON que contiene, al menos, la identidad del usuario y la fecha de expiración del token. Este objeto es firmado digitalmente y codificado en un formato apto para URL.

Para saber si un JWT es válido, únicamente tenemos que verificar la firma digital, y comprobar que la fecha de expiración sigue siendo válida. De este modo, no necesitamos un servidor aparte para verificarlos ni mantener los tokens en memoria o en la base de datos entre peticiones.

Se ha instalado el módulo **jwt-simple** para llevar acabo la implementación de los JWT. Cuando un usuario inicia sesión o se registra, se le devuelve como respuesta a esa petición un JWT. El cliente desarrollado se encarga de almacenarlo en la sesión del navegador (sessionStorage), donde permanecerá hasta que el usuario cierre sesión en la aplicación web, cierre la pestaña del navegador o cierre el navegador. El cliente que ha sido desarrollado también comprueba la fecha de expiración del token, aunque esto ha sido un desarrollo opcional ya que el servidor lo comprueba en cada petición.

Para realizar peticiones se ha instalado el módulo **axios**, y con el fin de que todas las peticiones del cliente agreguen el JWT en la cabecera se ha añadido la línea de código:

```
axios.defaults.headers.common = { user_token: `${Common.getToken()}` };
```

Siendo *user_token* la cabecera elegida para mandar el token y *Common.getToken()* una función desarrollada encargada de obtener el token almacenado previamente en la sesión del navegador.

En el lado del servidor, para las rutas que requieren autentificación, antes de procesar la respuesta a la petición se hace uso del middleware de Express. Se ha desarrollado una función middleware que verifica el JWT y en caso de ser inválido o estar caducado, devuelve una respuesta de error al cliente y no continúa con el proceso. Si por el contrario, el token es válido, se añade un campo a la petición con el id del usuario (el cual se encontraba en el cuerpo del token) y se continúa con el proceso. El servidor ahora puede responder a la petición de ese usuario de forma personalizada puesto que conoce su id.

Otra medida de protección del sistema muy importante a desarrollar, es la encriptación de las contraseñas de los usuarios en la base de datos. De esta forma, ni el propio administrador del sistema o de la base de datos puede conocer la contraseña de ningún usuario. Para ello se ha instalado el módulo **bcrypt**, el cual incorpora un valor aleatorio a la contraseña (llamado *salt*) y obtiene un hash de ella. Así se evita que dos contraseñas iguales tengan el mismo hash y los problemas que ello conlleva, por ejemplo, un ataque de fuerza bruta. Este valor hash está asociado a la contraseña y es el valor almacenado en la base de datos en lugar de la contraseña.

6.5 Conexión de dispositivos Bluetooth de telemetría personal

La conexión con los dispositivos Bluetooth de telemetría personal se ha realizado integrando Web Bluetooth API en el cliente. Como ya se ha mencionado, es una interfaz de programación de aplicaciones que proporciona la habilidad de conectarse e interaccionar con dispositivos Bluetooth Low Energy (BLE).

WebBluetooth API requiere que la conexión sea segura mediante el protocolo HTTPS, pero para pruebas locales permite que sea HTTP el protocolo usando un navegador de desarrollo como Google Chrome Dev. Con esta API, las páginas web del navegador pueden enviar y recibir datos directamente de un dispositivo BLE, y así lo hace la aplicación web desarrollada.

Con el código mostrado abajo realizamos una búsqueda de dispositivos que ofrezcan el servicio Bluetooth indicado como parámetro y tras seleccionar uno desde el navegador, se conecta con dicho servicio.

```
const conectarBluetoothGetServicio = async (servicioNombre) => {
  const dispositivo = await navigator.bluetooth.requestDevice({
    filters: [{
      services: [servicioNombre],
    }],
  });
  const servidor = await dispositivo.gatt.connect();
  const servicio = await servidor.getPrimaryService(servicioNombre);
  return servicio;
};
```

Si a la función detallada arriba le pasamos como parámetro el servicio Bluetooth "*health_thermometer*", intentaría conectarse con un dispositivo que midiese la temperatura corporal. Si pasamos el resultado de esa función a la función mostrada abajo, obtenemos la característica "*temperature_measurement*" donde podremos leer el valor de la temperatura.

```
const getTemperatura = async (servicio) => {
  const caracteristica = await servicio.getCharacteristic(
    'temperature_measurement'
  );
  const temperatura = await handleCaracteristicaTemperatura(caracteristica);
  return temperatura;
};
```

Para obtener las mediciones de cada constante vital, al ser servicios y características Bluetooth diferentes, requieren un procesado de los datos diferente. Para el caso de la **temperatura**, del array de bytes original:

Ejemplo array de bytes original (hexadecimal):
06-68-01-00-FF-E2-07-03-0A-15-34-00-02
Equivale a:
[6,104,1,0,255,226,7,3,10,21,52,0,2]

- El byte 0 (6) es un bit-flag: 00000110. El bit 0 más a la derecha (con valor 0) indica que la temperatura está en grados Celsius (si fuera 1 sería en Fahrenheit), el bit 1 con valor 1 indica que el campo con la marca de tiempo está presente (si fuera 0 que no está presente), y el bit 2 con valor 1 indica que el campo con el tipo de temperatura está presente. El resto de bits no son relevantes dado que están reservados para uso futuro.

- El byte 1 (104) y el byte 2 (1) almacenan la temperatura multiplicada por 10 en formato FLOAT. Para obtener la temperatura hay que multiplicar el byte 2 (1) por 256 y sumar el byte 1 (104), para finalmente dividir el total entre 10:

  (1 * 256 + 104) / 10 = 36.0 grados Celsius.

- El resto de los bytes son usados para la marca de tiempo, el tipo de temperatura y reservados para uso futuro. El sistema no usa estos campos por lo que no los procesa.

Para el caso de la constante vital **presión arterial** y **glucemia**, los valores se encuentran en formato IEEE-11073 16-bit SFLOAT. SFLOAT se trata de un valor de 16 bits que contiene un exponente de 4 bits en base 10, seguido de una mantisa de 12 bits. Cada uno está en forma de complemento a dos. A continuación se muestra un ejemplo de como pasar un número decimal a formato SFLOAT:

Decimal: 120
Binario: 0000000001111000

Exponente (4 primeros bits):
Binario: 0000
Binario complemento a dos: 0000
Hexadecimal: 0

Mantissa (12 bits siguientes al exponente):
Binario: 000001111000
Binario complemento a dos: 000001111000
Hexadecimal: 078

Por lo que 120 en formato SFLOAT sería 0078. Para convertir un valor SFLOAT a decimal sería el proceso inverso.

Los dispositivos que miden la **presión arterial** se encuentran bajo el servicio Bluetooth "*blood_pressure*" y el cliente obtiene el valor de la característica "*blood_pressure_measurement*" con propiedad *Read*. El array de bytes tiene el siguiente formato:

- Byte 0: Flags. El bit 0 indica si la unidad es mmHg (bit igual a 0) o kPa (bit igual a 1).
- Byte 1-2: Presión arterial sistólica (SFLOAT).
- Byte 3-4: Presión arterial diastólica (SFLOAT).
- Los demás bytes se utilizan para la marca de tiempo, frecuencia cardíaca, id del usuario y estado de la medición.

Los dispositivos que miden la **glucemia** proveen el servicio Bluetooth "*glucose*" y el cliente obtiene el valor de la característica "*glucose_measurement*" con propiedad *Read*. El array de bytes tiene el siguiente formato:

- Byte 0: Flags. El bit 1 indica si los campos de concentración de glucosa, tipo y zona de medición de la muestra están presentes. El bit 2 indica si la unidad de medida es kg/L (bit igual a 0) o mol/L (bit igual a 1). Los demás bits no son importantes para nuestro sistema.
- Byte 1-2: Sequence Number
- Byte 3-9: Base Time
- Byte 10-11: Glucose Concentration (SFLOAT)
- Byte 12: Type
- Byte 13: Sample Location

Los bytes 10 y 11 son los útiles para nuestro sistema.

Para los dispositivos que miden la **frecuencia cardíaca** el servicio Bluetooh que proporcionan es "*heart_rate*" y el cliente obtiene los valores de la característica "*heart_rate_measurement*". Esta característica es manejada diferente ya que usamos la propiedad *Notify*. Esto crea una subscripción a la característica, la cual notifica al sistema cada vez que hay una nueva medición. Se debe a que la frecuencia cardíaca suele medirse durante un mayor período de tiempo puesto que cambia

constantemente. La temperatura, por ejemplo, no es necesaria que sea conocida cada segundo dado que la mayoría de las veces no va a haber variación. El array de bytes para la frecuencia cardíaca tiene el siguiente formato:

- Byte 0: Flags. El bit 0 indica el formato del campo del valor de la medición y por ende si ocupa un byte (uint8) o dos (uint16). El resto de los bits no son interesantes para nuestro sistema.
- Byte 1/1-2: Frecuencia Cardíaca en pulsaciones por minuto (uint8/uint16).

Dado que no era posible disponer de todos los dispositivos de telemetría personal Bluetooth para las diferentes constantes vitales, se ha usado la aplicación móvil **Light Blue de PunchThrough** para crear estos dispositivos de forma virtual. Con esta aplicación, el móvil simula el funcionamiento de los dispositivos de telemetría personal Bluetooth que seleccionemos.

La aplicación no ofrece la posibilidad de virtualizar un dispositivo que mida el peso, servicio Bluetooth "*weight_scale*". Lo mismo ocurre con la saturación de oxígeno, que sería el servicio "*pulse_oximeter*". Debido a esto, se ha decidido no implementar la monitorización de estas dos constantes vitales en el sistema, ya que no podrían ser probadas.

Para crear un dispositivo virtual BLE de las otras constantes vitales, se esquematiza a continuación las opciones a elegir para cada una de ellas. Debemos introducir el valor que el sistema va a leer en la aplicación, simulando el valor que el dispositivo hubiera medido al usuario. En el caso de la frecuencia cardíaca esto no es necesario ya que usa la propiedad *Notify.*

- Frecuencia Cardíaca:
    - Servicio: Heart Rate
    - Característica: Heart Rate Measurement
    - Propiedad: Notify

- Temperatura:
    - Servicio: Health Thermometer
    - Característica: Temperature Measurement
    - Propiedad: Read
    - Valor (Ejemplo): 06-68-01-00-FF-E2-07-03-0A-15-34-00-02 (equivale a 36 grados Celsius)

- Presión Arterial:
    - Servicio: Blood Pressure
    - Característica: Blood Pressure Measurement
    - Propiedad: Read
    - Valor (Ejemplo): 00-00-78-00-50-00-00-00-00-00-00-00-00 (equivale a 120/80mmHg)

- Glucemia:
    - Servicio: Glucose
    - Característica: 0x2A18 (Glucose Measurement)
    - Propiedad: Read
    - Valor (Ejemplo): 02-00-00-00-00-00-00-00-00-00-B0-53-00-00 (equivale a 83 mg/dL)

6.6 Interfaz: Componentes React

React, como ya se ha expuesto, es una biblioteca JavaScript de código abierto que se centra en el desarrollo de interfaces de usuario. Ofrece la gran ventaja de que existe un completo ecosistema de módulos, herramientas y componentes que facilitan al desarrollador cumplir objetivos.

Este es el caso del framework **Material-UI** instalado en el proyecto, tiene a su disposición una librería muy completa de componentes de React. La mayoría de los componentes de la interfaz de la aplicación web son de este framework, ya sea la interfaz de la página de inicio, la de registro o la principal de las constantes vitales. Ejemplos de componentes utilizados son: botón, campo de texto, menú, barra de herramientas, lista de elementos, etc.

Para la gráfica que muestra las mediciones de constantes vitales, se ha instalado **DevExtreme** y utilizado el componente *Chart*, que a su vez incluye componentes para las series de valores, ejes, márgenes, puntos de valores, cuadrícula, etc.

Con el objetivo de integrar WebBluetoothAPI se ha creado un componente para cada constante vital, componente que es renderizado en segundo plano únicamente al hacer clic en dicha constante vital.

React permite la asociación de la vista con los datos, de modo que si cambian los datos lo mismo ocurre con las vistas. Cuando se actualiza la vista, se modifica el DOM Virtual, siendo este proceso más rápido que actualizar el DOM del navegador. React compara ambos DOM y se encarga de actualizar únicamente las partes del DOM que han sido modificadas en el virtual, evitando la necesidad de actualizar la vista entera. Se ha utilizado esta funcionalidad para que la mayor parte de la lógica del cliente resida bajo un URL (‘*/dashboard*’) y según cambien los datos asociados a un componente, actualice únicamente ese componente y no la página entera. Y lo que es más, que no requiera una redirección a otra página con un URL diferente cuando el usuario realiza acciones en la interfaz.

Los estados principales del componente *Dashboard* que permiten que esto ocurra son:

- *constanteVital:* almacena la constante vital seleccionada.
- *fechaMediciones*: almacena la fecha seleccionada (hoy por defecto).
- *mediciones*: almacena las mediciones de la fecha seleccionada (obtenidas de la base de datos), también almacena las nuevas mediciones tomadas.

7

Pruebas

Las pruebas de software constituyen un proceso de la ingeniería del software muy importante; se comprueba que el sistema se comporta según lo esperado, descubriéndose fallos y solucionándolos en una etapa temprana. Permite además una automatización de las pruebas, que pueden ser ejecutadas en cualquier momento, verificando que a pesar de que haya habido cambios en el código el sistema se sigue comportando según lo esperado.

Nuestro sistema web tiene como base una API REST, por lo que tiene sentido enfocar las pruebas a verificar su correcto funcionamiento. Las pruebas ideales para una API son las pruebas de integración, donde se testea que todos los elementos que la componen funcionan juntos correctamente. Con este fin se ha instalado el framework de JavaScript **Jest**. Las pruebas de integración realizadas comprueban que para una petición específica (con sus respectivos parámetros) el servidor devuelve la respuesta esperada. Las pruebas han sido las siguientes:

- Método de ruta POST '*/iniciarSesion':*
  - Al enviar un email no registrado devuelve el mensaje de error 'Usuario no dado de alta en el sistema'.
  - Al enviar un email correcto y una contraseña incorrecta devuelve el mensaje de error 'Usuario o contraseña incorrectos'.
  - Al enviar un email y contraseña correctos devuelve los datos del usuario, fecha de expiración del token, etc.

- Método de ruta POST '*/registrarUsuario':*
  - Al no enviar ningún parámetro devuelve un mensaje de error indicando todos los campos que están vacíos y necesitan ser enviados.
  - Al enviar parámetros con formato incorrecto devuelve un mensaje de error indicando qué parámetros presentan un error de formato y cuál es el esperado por el sistema.
  - Al enviar los parámetros con el formato correcto devuelve los datos del usuario, fecha de expiración del token, etc.

- Método de ruta POST '*/mediciónConstanteVital/frecuenciaCardiaca*'
- Método de ruta POST '*/mediciónConstanteVital/temperatura*'
- Método de ruta POST '*/mediciónConstanteVital/presionArterial*'
- Método de ruta POST '*/mediciónConstanteVital/glucemia*'

A cada uno de estos métodos de ruta llamados para enviar mediciones de constantes vitales, se le ha realizado dos test:

  - Uno enviándole un array de mediciones con formato incorrecto (ya sea que le faltan campos o que estos no son del tipo que deberían), comprobando que devuelve un mensaje de error indicando el formato que debería de tener este array para la constante vital enviada.
  - Otro enviándole un array de mediciones con formato correcto comprobando que devuelve estas mediciones como respuesta.

Un ejemplo de una prueba para mostrar cómo han sido implementadas con Jest es el siguiente, que comprueba el método de ruta POST '*/registrarUsuario*' enviándole parámetros con formato incorrecto.

```
test('POST incorrecto con varios campos con formato incorrecto devuelve los errores específicos a solventa
r', async () => {
    const res = await request(app)
      .post('/registroUsuario')
      .send({
        nombre: 'test',
        apellidos: 'test',
        email: 'estonoesunemail',
        contraseña: '1234',
        fechaNacimiento: '45/03/1993',
        altura: '0.2',
        sexo: 'niño',
        diabetes: 'negativo',
      });
    expect(JSON.parse(res.text)).toEqual({
      ok: false,
      err: {
        altura: 'El valor introducido en el campo Altura no es válido',
        diabetes: 'El valor introducido en el campo Diabetes no tiene un formato válido (si/no)',
        email: 'El valor introducido en el campo Email no tiene un formato válido',
        fechaNacimiento: 'El valor introducido en el campo Fecha Nacimiento no tiene un formato válido (DD
/MM/AAAA)',
        sexo: 'El valor introducido en el campo Sexo debe ser masculino o femenino',
      },
    });
  });
```

8

Conclusiones y líneas futuras

8.1 Conclusiones

La mHealth es sin duda un campo que ofrece innumerables posibilidades de desarrollo, servicios al usuario final y a los proveedores de servicios y que vela por un futuro prometedor donde la salud obtiene un papel de mayor importancia. Apuesta por la prevención, ayuda a la sostenibilidad de los sistemas de salud y presta servicios de telemedicina.

Actualmente existen una gran cantidad de dispositivos Bluetooth que miden las constantes vitales de los usuarios en el mercado, y son fácilmente accesibles. El inconveniente principal que se presenta es la cantidad de aplicaciones móviles que el usuario debe instalar para monitorizar cada una de ellas, conllevando además una dificultad para visualizar las diferentes mediciones. Este proyecto ha aunado la monitorización de la frecuencia cardíaca, temperatura, presión arterial y glucemia en una sola aplicación, la cual además es web. Esto implica que es una aplicación multiplataforma y multidispositivo, con las ventajas que ello conlleva.

La comunicación entre la aplicación web y los dispositivos se realiza mediante Bluetooth, gracias a la interfaz de programación Web Bluetooth API, la cual es compatible con múltiples navegadores. La documentación de esta interfaz y de Bluetooth en sí es escasa, poco detallada y presentada en un formato complicado, dificultando el desarrollo dado que hay una enorme cantidad de servicios y características Bluetooth.

Como tecnología de backend Node.js es un framework de JavaScript muy sencillo y potente con el que construir una API REST, ayudado de la infraestructura de aplicaciones Express. El sistema de base de datos MongoDB se integra perfectamente con este framework. La aplicación web se ha estructurado en componentes actualizables de

forma aislada y con datos asociados con la biblioteca JavaScript React, acelerando el desarrollo y mejorando la eficiencia de la aplicación web.

El resultado final ha sido un sistema web capaz de monitorizar cuatro importantes constantes vitales, considerando la situación del usuario en cada momento en el que se realizan las mediciones (si se encuentra haciendo ejercicio, si la medición ha sido tomada después de comer) y lo alerta mediante notificaciones con un nivel asignado de gravedad acerca de las que se encuentran fuera del rango normal para esa constante vital. Todos los datos son almacenados en una base de datos, accesibles para el usuario en cualquier momento, pudiendo ver el histórico de sus mediciones. Las mediciones son siempre mostradas visualmente en una gráfica, diferenciando con una escala de colores aquellas que se encuentran fuera del rango normal. Las mediciones pueden ser impresas o exportadas a diferentes formatos de archivo. Estas funciones, menos las dos últimas, pueden ser consumidas también a través de la API REST.

8.2 Líneas futuras

Hay una serie de elementos que se habrían podido añadir o ampliar en el sistema web y que no han tenido cabida en esta versión debido a la envergadura del mismo. Se explican a continuación algunas líneas futuras que se podrían realizar para ampliar la funcionalidad del sistema:

- **Monitorización de la saturación oxígeno en sangre**. El estudio de esta constante vital ya ha sido realizado.

- **Monitorización del peso.** Esta constante vital ofrece además la posibilidad de monitorizar muchas propiedades asociadas a él, como son la grasa corporal, masa muscular, grasa visceral, masa ósea, agua corporal, proteínas y metabolismo basal.

- **Personalización del rango de valores normal**. El rango de valores adecuado para cada constante vital es establecido por el sistema y actualmente no es personalizable por el usuario. Los atributos para almacenar estas preferencias ya se encuentran implementados en la base de datos. No lo están para personalizar el valor mínimo crítico y máximo crítico de cada constante, lo que añadiría un plus de personalización de las alertas.

- **Registro de personal sanitario.** Permitir el registro de personal sanitario, el cual añada a sus pacientes (bajo el consentimiento de ellos) y visualice sus constantes vitales. Esta funcionalidad posibilita que el personal sanitario supervise de forma telemática las constantes vitales de sus pacientes, sin que tengan la necesidad de acudir al centro médico. Permite también a los médicos ofrecer diagnósticos y tomar decisiones más acertadas y precisas debido a la cantidad de datos recogidos.

Referencias

Market Research Firm. MHealth Solutions Market Report. https://www.marketsandmarkets.com/Market-Reports/mhealth-apps-and-solutions-market-1232.html (accedido en abril 2020).

Ricoh. Technology and the Patient Experience. https://www.ricoh-usa.com/en/insights/library/articles/technology-and-the-patient-experience (accedido en abril 2020).

García Esteller, David y González Ruiz, Mar. BWell: sistema integral de monitorización de pacientes a distancia, publicado en Comunicación al V Congreso Ciudades Inteligentes. https://www.esmartcity.es/comunicaciones/comunicacion-bwell-sistema-integral-monitorizacion-pacientes-distancia (accedido en abril 2020).

Naciones Unidas. Envejecimiento. https://www.un.org/es/sections/issues-depth/ageing/index.html (accedido en abril 2020).

Quental. Tecnología IoT en el Sector Hospitalario. https://quental.com/media/files/Informe-Tecnologia-IoT-en-el-Sector-Hospitalario.pdf (accedido en abril 2020).

Fundación Española del Corazón. Telemedicina: control de los pacientes a distancia. https://fundaciondelcorazon.com/corazon-facil/blog-impulso-vital/2151-telemedicina-control-delos-pacientes-a-distancia.html (accedido en abril 2020).

MedM. https://www.medm.com/ (accedido en mayo 2020).

Web Bluetooth. Draft Community Group Report, 1 July 2020. https://webbluetoothcg.github.io/web-bluetooth/ (accedido en mayo 2020).

Beaufort, François. Interact with Bluetooth devices on the Web. https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web (accedido en mayo 2020).

Ordóñez, Jonathan. ¿Qué es una API REST?. https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/ (accedido en mayo 2020).

Rubenfa. MongoDB: qué es, cómo funciona y cuándo podemos usarlo (o no). https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no (accedido en mayo 2020).

Schiaffarino, Andrés. Modelo Cliente Servidor. https://blog.infranetworking.com/modelo-cliente-servidor/ (accedido en mayo 2020).

DesarrolloWeb. JavaScript. https://desarrolloweb.com/home/javascript (accedido en mayo 2020).

DesarrolloWeb. HTML. https://desarrolloweb.com/home/html (accedido en mayo 2020).

DesarrolloWeb. CSS.
https://desarrolloweb.com/home/css (accedido en mayo 2020).
DesarrolloWeb. NodeJS.
https://desarrolloweb.com/home/nodejs (accedido en mayo 2020).
Fundación OpenJS. Node.js.
https://nodejs.org/es/ (accedido en mayo 2020).
Fundación OpenJS. Express
https://expressjs.com/es/ (accedido en mayo 2020).
Facebook Open Source. React.
https://es.reactjs.org/ (accedido en mayo 2020).
DesarrolloWeb. Qué es React. Por qué usar React.
https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html (accedido en mayo 2020).
MedlinePlus. Temperatura corporal normal.
https://medlineplus.gov/spanish/ency/article/001982.htm (accedido en junio 2020).
Gómez Ayala, Adela-Emilia. Trastornos de la temperatura corporal.
https://www.elsevier.es/es-revista-offarm-4-articulo-trastornos-temperatura-corporal-13108301 (accedido en junio 2020).
Fernández Muñoz, Miriam. Aprende a calcular tu peso “más saludable”.
https://www.efesalud.com/aprende-calcular-peso-saludable/ (accedido en junio 2020).
Grupo Las Mimosas. ¿Qué es la saturación de oxígeno y cuáles son los niveles normales?.
https://grupolasmimosas.com/mimoonline/saturacion-de-oxigeno/ (accedido en junio 2020).
MacGill, Markus. ¿Cuál debería ser mi frecuencia cardíaca?.
https://www.medicalnewstoday.com/articles/291182 (accedido en junio 2020).
MedlinePlus. Ejercite su corazón.
https://medlineplus.gov/spanish/ency/patientinstructions/000763.htm (accedido en junio 2020).
Carramiñana, Francisco. Al hacer deporte ¿cuál es el límite de mi corazón?.
http://blogs.hoy.es/salud-para-todos/2013/03/11/a-cuanto-debe-ir-el-corazon-cuando-hago-deporte/?ref=https:%2F%2Fwww.google.com%2F (accedido en junio 2020).
Sanitas. Niveles de glucosa en sangre.
https://www.sanitas.es/sanitas/seguros/es/particulares/biblioteca-de-salud/diabetes/niveles-glucosa-sangre.html (accedido en junio 2020).
Social Diabetes. ¿Sabes cuáles son los niveles normales de azúcar o glucosa en la sangre?
https://blog.socialdiabetes.com/sabes-cuales-son-los-niveles-normales-de-azucar-en-la-sangre/ (accedido en junio 2020).
Marín, Lara, Gómez, Ana y López, Rocío. Hipoglucemia e hiperglucemia, ¿cómo se presenta y qué debo hacer?.
https://diabetesmadrid.org/hipoglucemia-e-hiperglucemia-como-se-presenta-y-que-debo-hacer/ (accedido en junio 2020).
Mayo Clinic. Expediente médico de presión: Significado de tus resultados.

https://www.mayoclinic.org/es-es/diseases-conditions/high-blood-pressure/in-depth/blood-pressure/art-20050982 (accedido en junio 2020).
Pacheco, Thiago. FullStack setup (Node.js, React.js and MongoDB). https://dev.to/pacheco/my-fullstack-setup-node-js-react-js-and-mongodb-2a4k (accedido en agosto 2020).
Brunfeldt, Kimmo. Concurrently. https://www.npmjs.com/package/concurrently (accedido en agosto 2020).
Sharp, Remy. Nodemon. https://www.npmjs.com/package/nodemon (accedido en agosto 2020).
Axios. Axios. https://www.npmjs.com/package/axios (accedido en agosto 2020).
Fundación OpenJS. Body-parser. https://www.npmjs.com/package/body-parser (accedido en agosto 2020).
Campbell, Nick. Bcrypt. https://www.npmjs.com/package/bcrypt (accedido en agosto 2020).
Hokamura, Kazuhito. Jwt-simple. https://www.npmjs.com/package/jwt-simple (accedido en agosto 2020).
Munro, Jaime. Una introducción a Mongoose para MongoDB y Node.js https://code.tutsplus.com/es/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527 (accedido en agosto 2020).
Automattic. Mongoose. https://mongoosejs.com/docs/guide.html (accedido en agosto 2020).
Angular University. Angular Security - Authentication With JSON Web Tokens (JWT): The Complete Guide. https://blog.angular-university.io/angular-jwt-authentication/ (accedido en agosto 2020).
Vicente, David. Encriptación de password en NodeJS y MongoDB: bcrypt. https://solidgeargroup.com/password-nodejs-mongodb-bcrypt/ (accedido en agosto 2020).
Material-UI. Material-UI. https://material-ui.com/es/ (accedido en agosto 2020).
DevExpress. DevExtreme, UI Widgets. https://js.devexpress.com/Documentation/ApiReference/UI_Widgets/ (accedido en septiembre 2020).
PunchThrough. LightBlue. https://punchthrough.com/lightblue/ (accedido en septiembre 2020)
Bluetooth SIG. Temperature Measurement. https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Characteristics/org.bluetooth.characteristic.temperature_measurement.xml (accedido en septiembre 2020).
Bluetooth SIG. Blood Pressure Measurement characteristic. https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Characteristics/org.bluetooth.characteristic.blood_pressure_measurement.xml (accedido en septiembre 2020).
Bluetooth SIG. Glucose Measurement characteristic. https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Characteristics/org.bluetooth.characteristic.glucose_measurement.xml (accedido en septiembre 2020).

Bluetooth SIG. Heart Rate Measurement characteristic.
https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Characteristics/org.bluetooth.characteristic.heart_rate_measurement.xml (accedido en septiembre 2020).

Apéndice A

Manual de Usuario

El sistema web ofrece la posibilidad de ser utilizado de dos formas: mediante la aplicación web, la cual ha sido desarrollada de cara al usuario o mediante el envío de peticiones a la API REST, pensando en que otros servicios puedan obtener los datos del sistema. A continuación se detallan ambos manuales de usuario.

A.1. Aplicación Web

Al acceder a la aplicación web se le requerirá que se autentifique para acceder al sistema.

A.1.1 Iniciar sesión

Si ya se encuentra registrado en el sistema, introduzca su email y contraseña en los campos respectivos y haga clic en el botón "Iniciar Sesión".

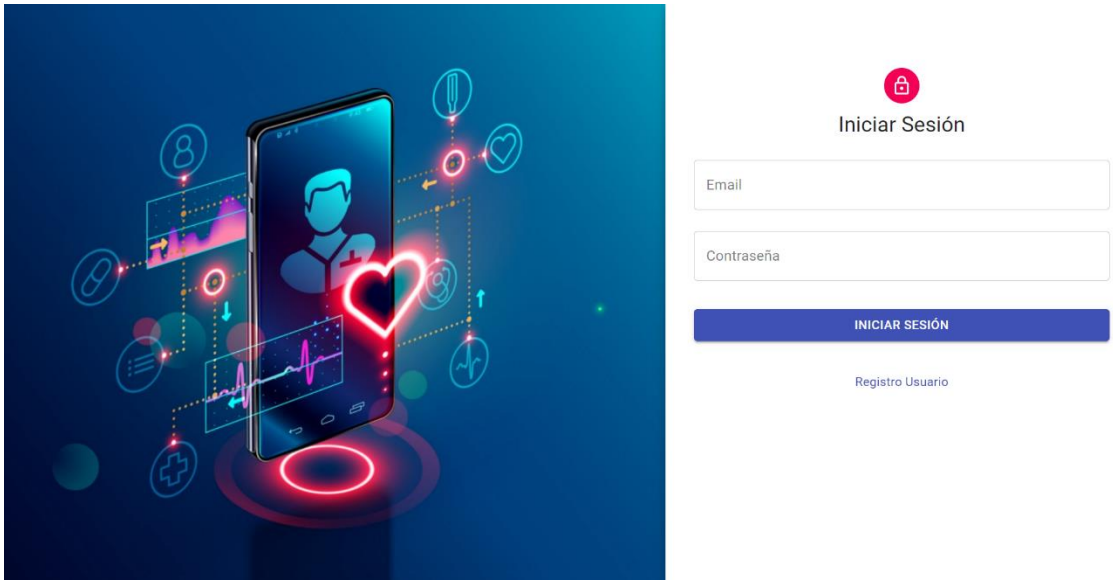


Figura A.1 Iniciar sesión.

En caso de error en la autentificación se le mostrará un mensaje.

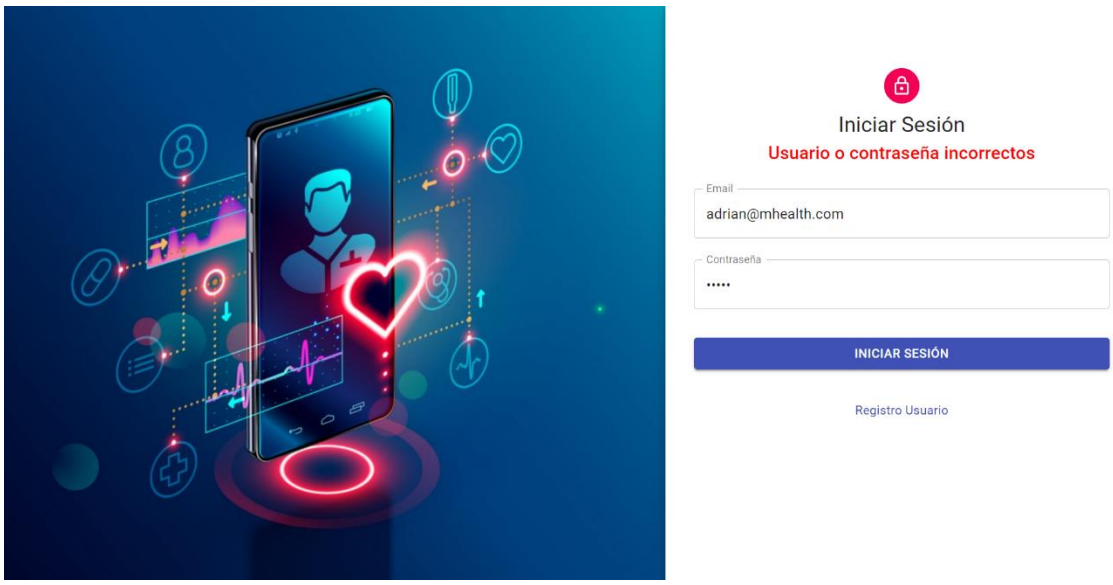


Figura A.2 Error de autentificación al iniciar sesión.

Introduzca los datos de nuevo asegurándose de que son correctos y vuelva a hacer clic en "Iniciar Sesión".

A.1.2 Registrarse

Si no se ha registrado anteriormente en el sistema, deberá hacerlo por primera vez. Para ello haga clic en el enlace de "Registro Usuario" que se encuentra en la página de Iniciar Sesión, y será redirigido a la siguiente página.

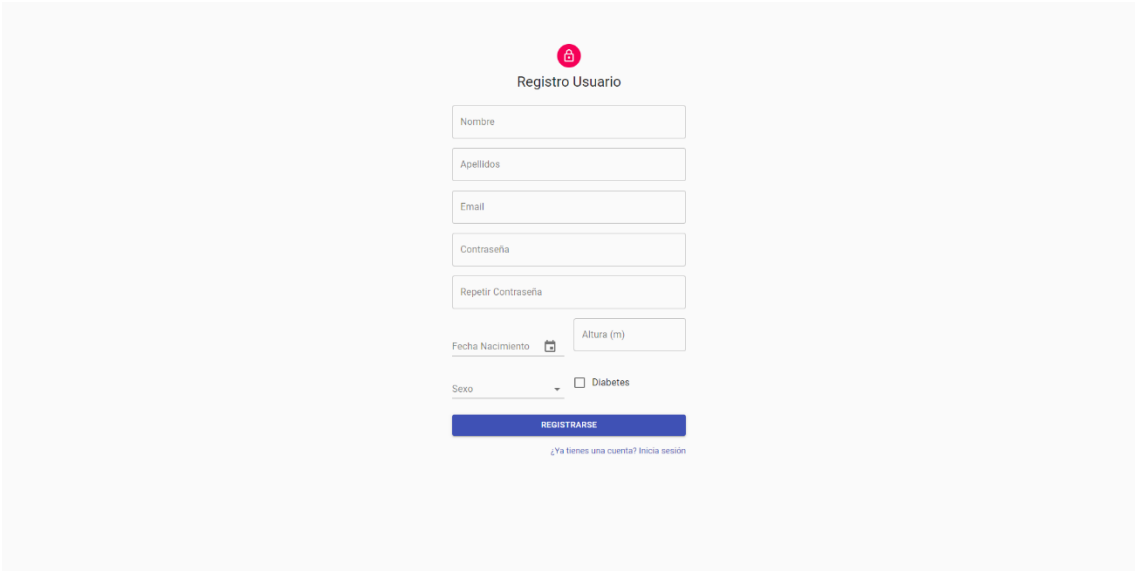


Figura A.3 Registrarse

En el caso de que se encuentre en esta página y sí tenga una cuenta registrada, haga clic en el enlace "Ya tienes una cuenta? Inicia sesión".

Para registrarse, introduzca su información en todos los campos, todos ellos son obligatorios, y haga clic en el botón "Registrarse". La información introducida en cada campo es validada y, en caso de encontrar algún error, se mostrará junto al campo la causa de este.

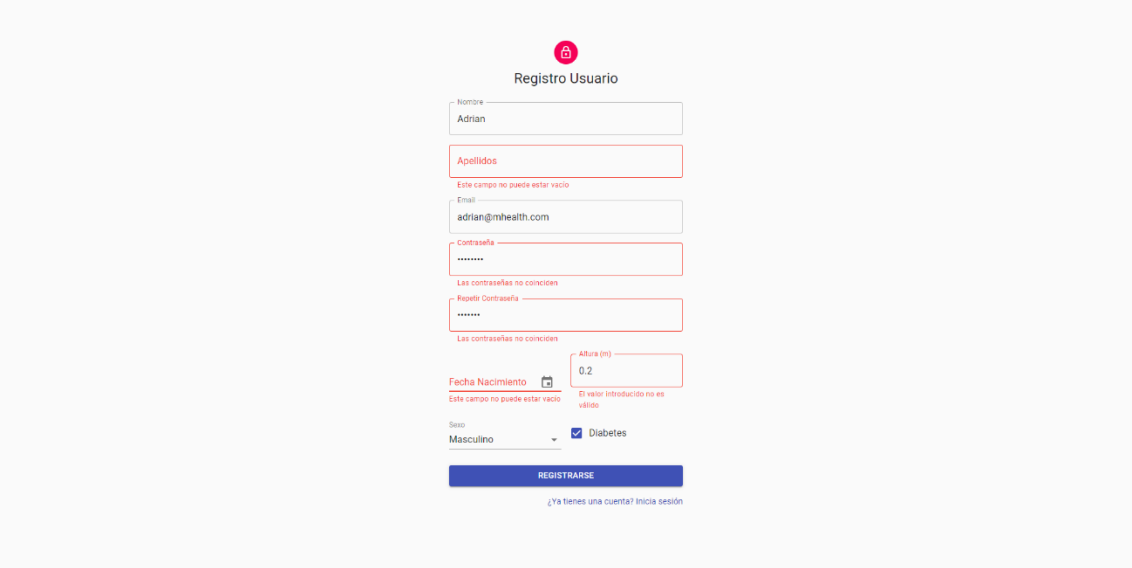


Figura A.4 Errores de formato en los campos al registrarse.

Si este es su caso, solucione los errores y haga clic de nuevo en el botón "Registrarse".

A.1.3 Seleccionar una constante vital

Se encuentra en la página de inicio, donde puede seleccionar la constante vital que desee gestionar en el menú lateral de la página.

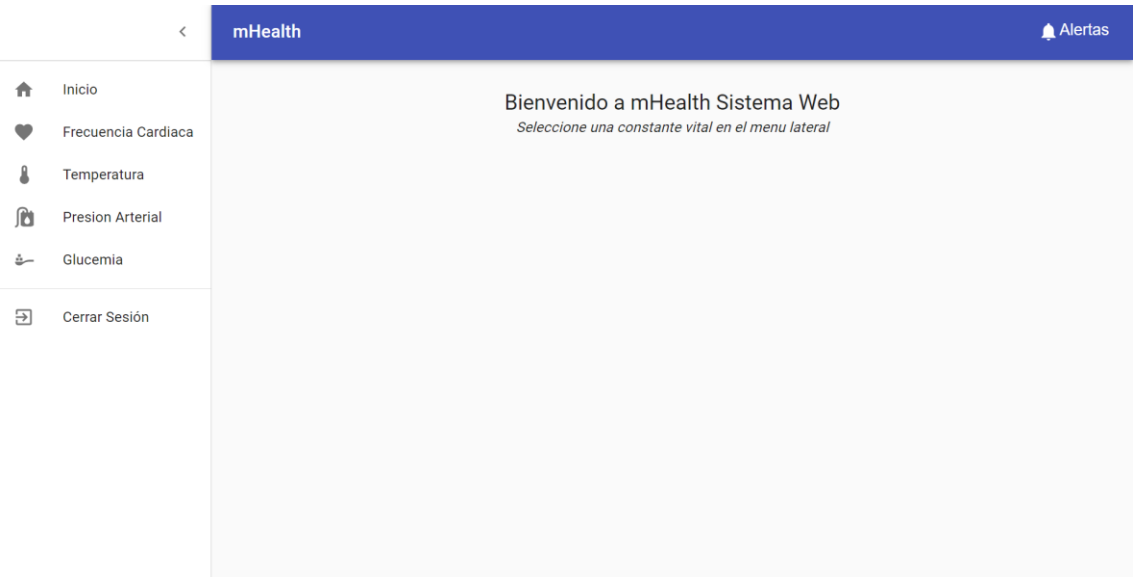


Figura A.5 Seleccionar una constante vital.

Una vez seleccionada una constante vital, se selecciona por defecto la fecha actual en el calendario y se muestran las mediciones tomadas para esa constante en el día actual.

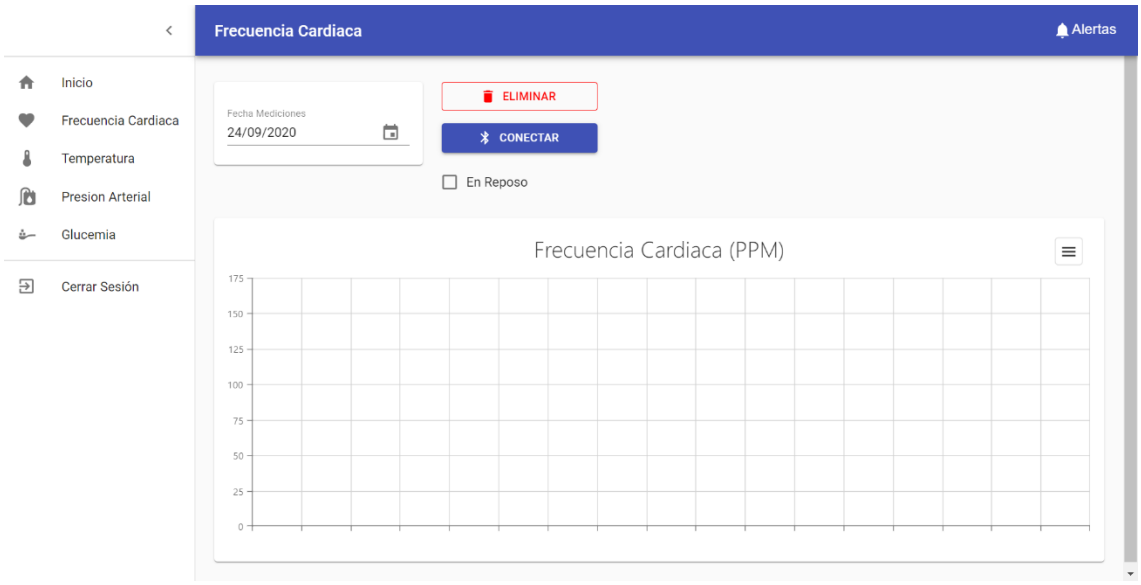


Figura A.6 Mediciones fecha actual

A.1.4 Añadir mediciones

Las mediciones son añadidas a través de dispositivos Bluetooth de telemetría personal. Haga clic en el botón de "Conectar", el navegador mostrará un desplegable con la lista de dispositivos Bluetooth Low Energy que ofrecen el servicio de medición para la constante vital seleccionada. Seleccione el dispositivo que desee conectar y haga clic en el botón "Vincular" del desplegable.

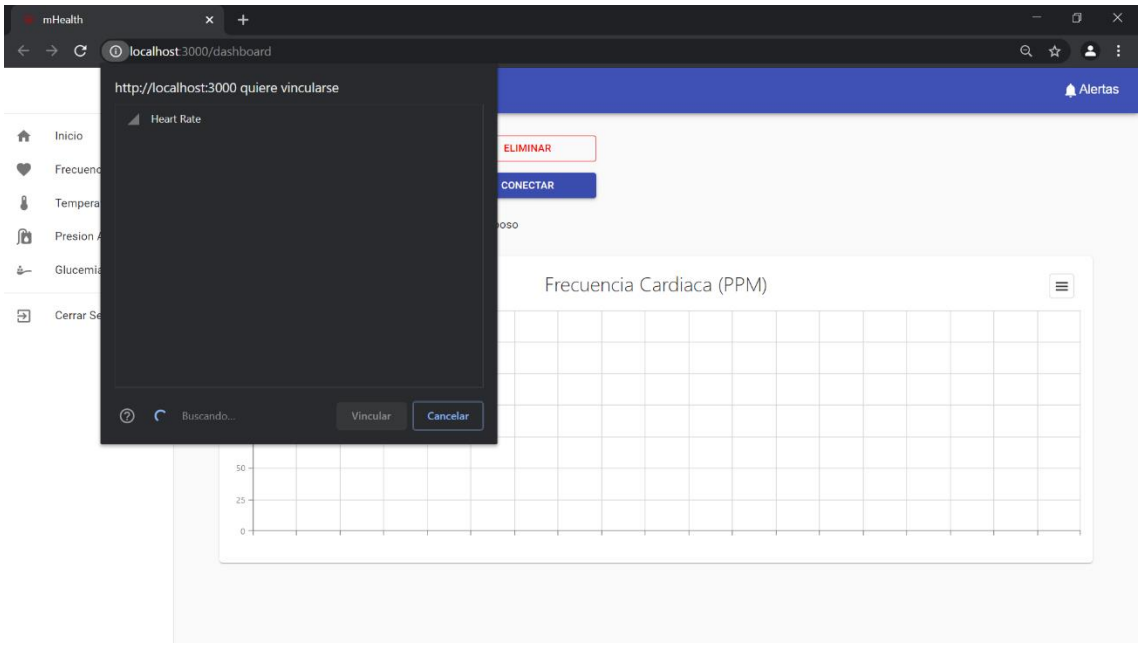


Figura A.7 Desplegable lista dispositivos Bluetooth.

La conexión Bluetooth tarda unos segundos en establecerse, pero una vez establecida el sistema empieza a leer las mediciones que el dispositivo está tomando y las muestra en la gráfica en tiempo real. Estas mediciones son también almacenadas en la base de datos del sistema.

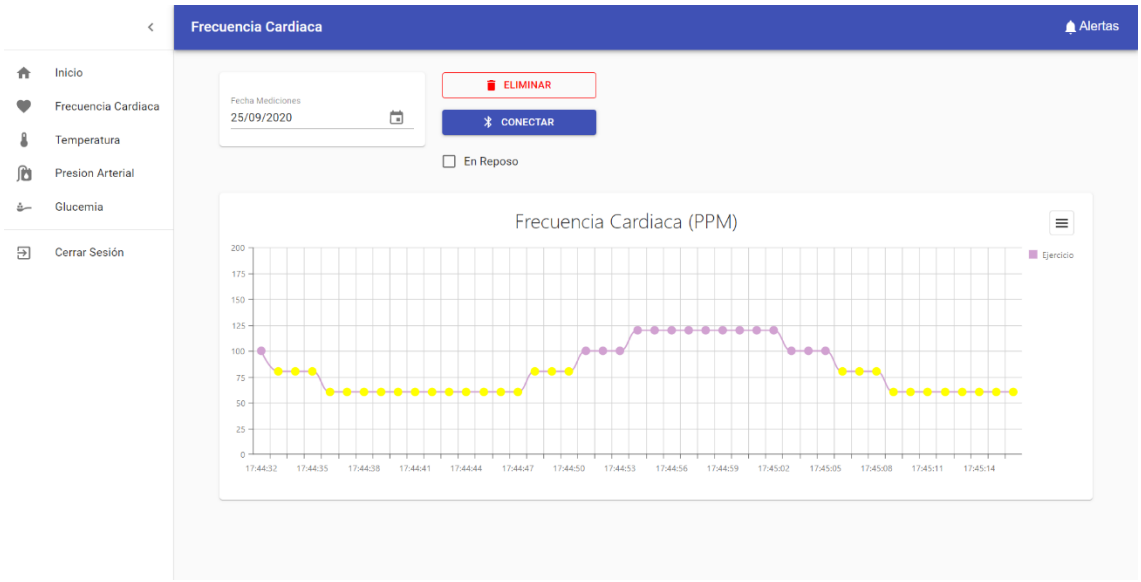


Figura A.8 Monitorización de mediciones en tiempo real.

Los colores de la gráfica indican el grado de alerta de cada medición. El color amarillo indica que la medición se encuentra fuera del rango de valores normal para esa constante vital; el color rojo indica también que el valor está fuera del rango, pero con una diferencia alarmante que requiere de la atención de un médico de forma urgente. El color morado y rosa pálido únicamente se utilizan para diferenciar diferentes tipos de mediciones para la misma constante vital, por ejemplo, que ha sido tomada en reposo o haciendo ejercicio. En tal caso, se muestra en la leyenda situada junto a la gráfica que representa el color morado y que representa el rosa pálido.

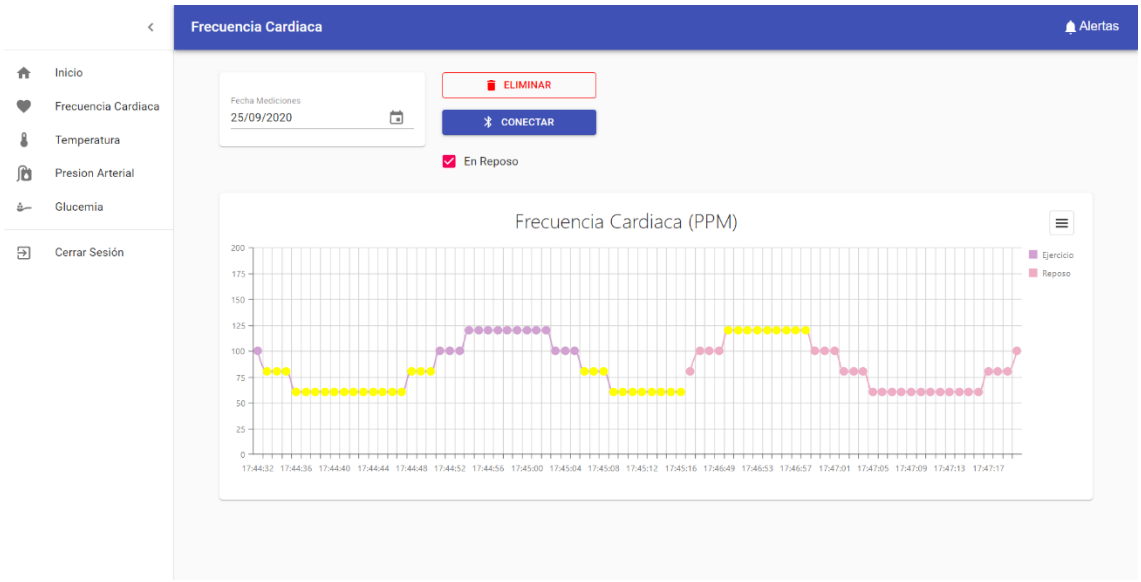


Figura A.9 Leyenda de la gráfica.

Para finalizar la toma de mediciones debe terminar la conexión desde el dispositivo Bluetooth que ha sido conectado.

Para ciertas constantes vitales se dispone de una casilla a marcar en caso de que se cumpla esa condición. Para la frecuencia cardíaca, si marca la casilla "En Reposo" le indica al sistema que la medición está siendo tomada en reposo; y si no la marca, que se encuentra haciendo ejercicio o justo después de haberlo realizado. Para el caso de la Glucemia, si marca la casilla "Postprandial" le indica al sistema que la medición está siendo tomada después de haber comido; y en caso de no marcarla, que se encuentra en ayuno. La casilla debe ser marcada, si procede, antes de conectar el dispositivo y que empiece a enviar las mediciones. Estos datos son importantes para el sistema dado que el rango de valores adecuados para estas constantes vitales es diferente según la condición en la que fueron tomadas las mediciones.

Solo puede añadir mediciones para el día actual, si selecciona otro día en el recuadro "Fecha Mediciones" el botón "Conectar" no será visible.

A.1.5 Ver las alertas

Haga clic en el botón "Alertas" que se sitúa en la barra de navegación superior. Se mostrará un desplegable con las alertas de todas las constantes vitales para el día seleccionado en el calendario. Las alertas incluyen la constante vital a la que se refiere, su logo, la condición en la que fue tomada (reposo/ejercicio, postprandial/ayuno) si corresponde, el valor de la medición y la hora. El valor puede encontrarse en color amarillo o rojo, indicando el nivel de gravedad, explicado anteriormente, de la alerta.

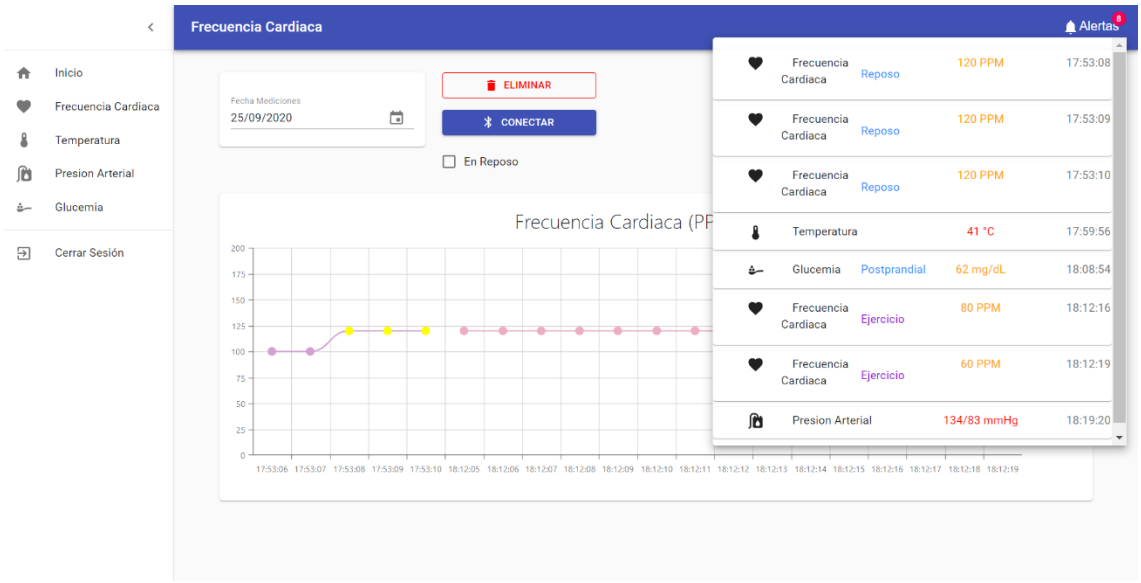


Figura A.10 Alertas de todas las constantes vitales.

A.1.6 Ver las mediciones/alertas de una fecha anterior

Para ver las mediciones o las alertas de una fecha anterior, haga clic en el icono del calendario en el recuadro "Fecha Mediciones" y seleccione la fecha sobre el calendario, o haga clic en la fecha directamente y escríbala usando el teclado.

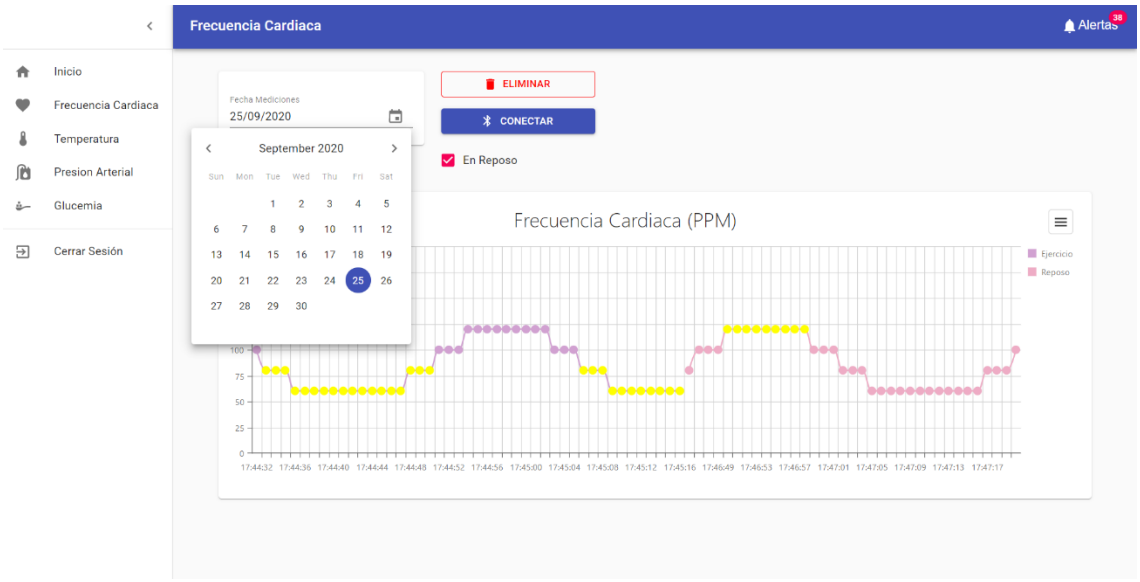


Figura A.11 Seleccionar otra fecha usando el calendario.

La grafica mostrará ahora las mediciones de la fecha seleccionada y al hacer clic en "Alertas" mostrará las alertas de aquel día.

A.1.7 Imprimir/Exportar las mediciones

Haga clic en el botón con tres líneas horizontales de la gráfica, y aparecerá un desplegable con la opción de imprimir y los posibles formatos a exportar. Seleccione la opción que desee para completar el proceso.

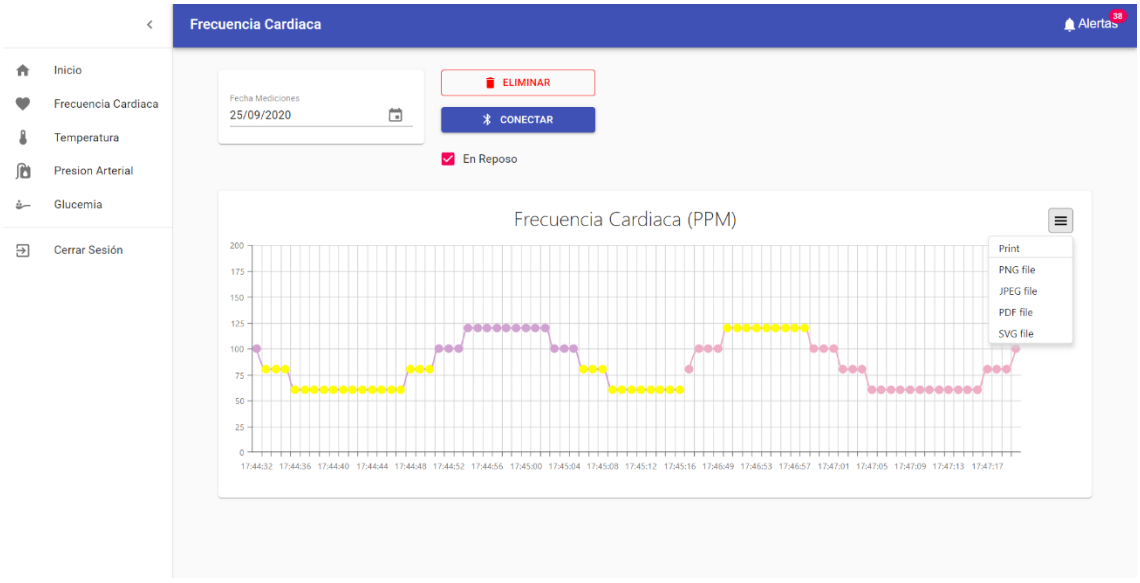


Figura A.12 Imprimir o exportar las mediciones en diferentes formatos.

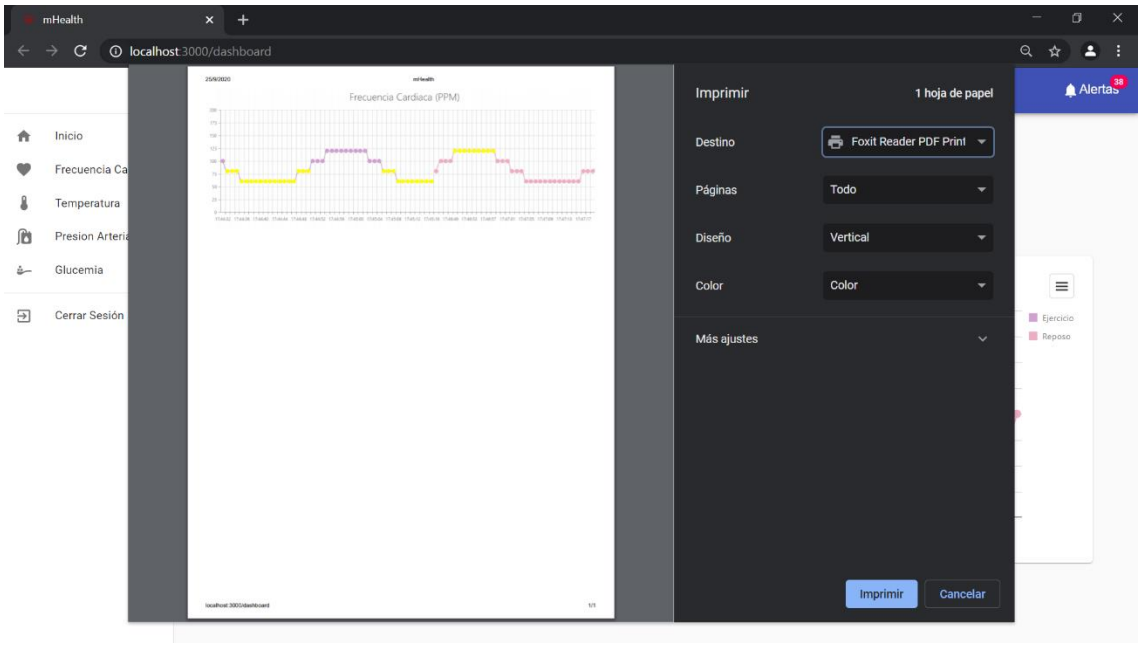


Figura A.13 Ventana para seleccionar las opciones de impresión.

A.1.8 Eliminar mediciones

Haga clic en el botón "Eliminar" para eliminar las mediciones de la constante vital seleccionada para el día seleccionado. Puede cambiar este día como se ha explicado en el apartado 8.1.1.6 y eliminar las mediciones de la constante vital seleccionada para la fecha que desee.

A.1.9 Cerrar sesión

Haga clic en el botón "Cerrar sesión" que se encuentra en el menú lateral de la aplicación web con el fin de cerrar la sesión de usuario. Será redireccionado a la página de iniciar sesión, signo de que su sesión fue cerrada correctamente.

A.2 API REST

La API REST idealmente se encontraría accesible bajo un dominio en Internet, 'mhealthsistemaweb.com', por ejemplo. Actualmente, la forma de acceder es instalando y ejecutando el sistema web para hacer la petición al URL 'localhost:3000'. En los siguientes URL se han omitido el dominio y puerto por simplicidad, aunque deben ser añadidos.

A.2.1 Iniciar sesión

- Método HTTP: POST
- URL: '/iniciarSesion'
- Parámetros:
    - email (String)
    - contraseña (String)

A.2.2 Registrarse

- Método HTTP: POST
- URL: '/registroUsuario'
- Parámetros:
    - nombre (String)
    - apellidos (String)
    - email (String)
    - contraseña (String)
    - fechaNacimiento (String, 'DD/MM/YYYY')
    - Altura (Number)
    - Sexo (String, 'masculino/femenino')
    - Diabetes (String, 'si/no')

A.2.3 Obtener mediciones frecuencia cardíaca de una fecha

- Método HTTP: GET
- URL: '/medicionConstanteVital/frecuenciaCardiaca'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - fecha (String, 'DD/MM/YYYY')

A.2.4 Obtener mediciones temperatura de una fecha

- Método HTTP: GET
- URL: '/medicionConstanteVital/temperatura'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - fecha (String, 'DD/MM/YYYY')

A.2.5 Obtener mediciones presión arterial de una fecha

- Método HTTP: GET
- URL: '/medicionConstanteVital/presionArterial'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - fecha (String, 'DD/MM/YYYY')

A.2.6 Obtener mediciones glucemia de una fecha

- Método HTTP: GET
- URL: '/medicionConstanteVital/glucemia'
- Cabeceras:

    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - fecha (String, 'DD/MM/YYYY')

A.2.7 Enviar mediciones frecuencia cardíaca

- Método HTTP: POST
- URL: '/medicionConstanteVital/frecuenciaCardiaca'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - valores (String, array de objetos con las siguientes claves y valores)
        - valor (Number, P.P.M.)
        - fecha (Number, milisegundos)
        - enReposo (Boolean)

A.2.8 Enviar mediciones temperatura

- Método HTTP: POST
- URL: '/medicionConstanteVital/temperatura'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - valores (String, array de objetos con las siguientes claves y valores)
        - valor (Number, grados Celsius)
        - fecha (Number, milisegundos)

A.2.9 Enviar mediciones presión arterial

- Método HTTP: POST
- URL: '/medicionConstanteVital/presionArterial'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - valores (String, array de objetos con las siguientes claves y valores)
        - valor (Number, sistolica en mmHg)
        - fecha (Number, milisegundos)
        - diastolica (Number, diastolica en mmHg)

A.2.10 Enviar mediciones glucemia

- Método HTTP: POST
- URL: '/medicionConstanteVital/glucemia'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)

- Parámetros:
    - valores (String, array de objetos con las siguientes claves y valores)
        - valor (Number, grados celsius)
        - fecha (Number, milisegundos)
        - postprandial (Boolean)

A.2.11 Eliminar mediciones de una constante vital para una fecha

- Método HTTP: DELETE
- URL: '/medicionConstanteVital'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - constanteVital (String,'frecuenciaCardiaca/temperatura/presionArterial/glucemia)
    - fecha (String, 'DD/MM/YYYY')

A.2.12 Obtener alertas de todas las constantes vitales para una fecha

- Método HTTP: GET
- URL: '/medicionConstanteVital/alertas'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)
- Parámetros:
    - fecha (String, 'DD/MM/YYYY')

A.2.13 Obtener mis datos personales

- Método HTTP: GET
- URL: '/usuario'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)

A.2.14 Eliminar la cuenta

- Método HTTP: DELETE
- URL: '/usuario'
- Cabeceras:
    - user_token: token devuelto como respuesta en los métodos de ruta '/iniciarSesion' y '/registroUsuario' (String)

Apéndice B

Manual de Instalación

El sistema web idealmente sería ejecutado de forma ininterrumpida en un servidor, y la aplicación web sería accesible desde un URL en Internet, de forma que el usuario no necesitaría instalar nada para utilizar los servicios. Es una aplicación web multidispositivo y multiplataforma, que además no requiere de instalación.

No obstante, dado que la infraestructura mencionada requiere un coste, esta no ha sido establecida. Este manual de instalación es útil para instalar el sistema web en el servidor y para probar el sistema web en una computadora de forma local.

Pasos:

1. Instalar Node.js

2. Instalar Google Chrome Dev.

3. Abrir una consola/terminal.

4. Ejecutar en la carpeta raíz el siguiente comando para instalar las dependencias del servidor:
   npm install

5. Navegar a la carpeta "client" con el comando:
   cd client

6. Ejecutar siguiente comando para instalar las dependencias del cliente:
   npm install

7. Volver a la carpeta raíz con el comando:
   cd ..

8. Ejecutar el siguiente comando para lanzar el servidor y cliente a la vez:

npm run dev

9. Seleccionar Google Chrome Dev para abrir la aplicación web o escriba en la barra de navegación de Google Chrome Dev:
   http://localhost:3000/

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA