Universidad Pública de Navarra

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

Máster en Ingeniería Biomédica

Trabajo Final de Máster

TENSIONAPP:

SISTEMA DE DETECCIÓN DE LA FRECUENCIA CARDIACA MEDIANTE EL SENSOR FOTOGRÁFICO DE UN SMARTPHONE Y MEDIANTE UN MICROCONTROLADOR.

Autor: Miguel Fuertes Fernández

Tutor: Javier Rodríguez Falces (Universidad Pública de Navarra)

Director: Javier Rodríguez Falces (Universidad Pública de Navarra)

Tabla de Contenidos

1 Resumen 6

2 Contexto y objetivos 7

2.1 Contexto 7

2.2 Objetivos 8

3 Estado del arte 8

4 Implementación 9

4.1 Requisitos 9

4.2 Arquitectura 9

4.3 Modelo de datos 10

4.4 Cliente 11

4.4.1 Diseño y funcionalidades 11

4.5 Servidor 12

4.6 Medición de frecuencia cardiaca 13

4.6.1 Smartphone: Cámara y flash 13

4.6.2 Microcontrolador: WemosD1 + MAX30100 16

5 Resultados de la implementación 20

5.1 ¿Se ha conseguido desarrollar/implementar completamente cada una de las técnicas? 20

5.2 ¿Funcionan cada una de las técnicas correctamente, se han comparado los valores de frecuencia cardiaca? 20

5.3 Comparación de ambas técnicas desde el punto de vista computacional y desde el punto de vista del programador. 21

5.4 Comparar las dos técnicas desde un punto de vista de operatividad para el médico o del usuario 21

5.5 Comparar las dos técnicas desde un punto de vista económico 21

5.6 ¿En qué escenarios/entornos recomiendas una técnica y la otra? 22

5.7 Tabla de fortalezas y debilidades 22

5.7.1 Arduino (WeMosD1) + Sensor MAX3010 22

5.7.2 Smartphone (Cámara + Flash) 22

6 Conclusiones 22

7 Líneas futuras 23

8 Referencias 25

9 Anexos 26

9.1 **Anexo 1**: Documentación de la API (Servidor) 26

9.1.1 Endpoints 26
9.1.2 Respuestas 28
9.1.3 Modelo de Datos 29
9.2 **Anexo 2**: Aplicación móvil 33
9.2.1 Pantallas Generales 33
9.2.2 Pantallas del paciente 35

Tabla de Figuras

Figura 1.-Descripción de los elementos integrantes del sistema 10
Figura 2.- Modelo de datos presente en todo el sistema, información almacenada del paciente con sus mediciones y la relación entre ellas. 10
Figura 3.- Pulsioxímetro comercial 13
Figura 4.- Diagrama de funcionamiento de un pulsioxímetro 13
Figura 5.- Posicionamiento del dedo en el Smartphone 13
Figura 6.- Representación matricial en capas de una imagen. Cada capa es un canal, rojo, verde y azul, con las intensidades de cada canal. 14
Figura 7.- Codigo del cálculo de un latido mediante la aplicación en el smartphone: cámara y flash. 15
Figura 8.- Agregación o diferencia de intensidades en un tiempo concreto para la medición ahora en latidos por minutos. 15
Figura 9.- La instalación del microcontrolador (en la parte derecha) y el sensor de pulso (en la parte izquierda) y los 4 cables necesarios para su correcto funcionamiento. 16
Figura 10.- Código de inicialización del microcontrolador, inicialización de la red WIFI y del sensor de pulso. 17
Figura 11.- Código que se repite indefinidamente. Se realiza aquí la medición en sí, la detección de latido y la agregación de varios latidos para calcular los bpm. 18
Figura 12.- Página web servida por el microcontrolador para ver lo que está midiendo. (Complementario al visor en la aplicación) 18
Figura 13.- Pieza de código encargada de recoger el dato de pulso desde el microcontrolador en la aplicación Flutter (Android) 19
Figura 14.- Futuro del sistema (Aplicaciones y Servidores) 23

Tabla de Tablas

Tabla 1.- Comparación de medidas entre ambos sistemas. 20
Tabla 2.- Fortalezas y debilidades del microcontrolador 22
Tabla 3.- Fortalezas y debilidades del smartphone (Cámara + Flash) 22

GLOSARIO

A

API
Application Programming Interface
Interfaz de un servidor para la obtencion de sus datos, hoy dia el estadar es REST 12, 20, 26, 27, 28, 33
Arduino
Arduino
Microcontrolador de hardware libre que utiliza C para su programación 6, 9, 13, 16, 17, 20, 21, 22

C

Cordova/PhoneGap
Framework web para creación de aplicaciones hibridas. 11
CSV
Comma Separated Values
Archivo de datos separados por comas 12

D

Dart
Dart
Lenguaje de programación orientado a objetos desarrollado por Google (https
//dart.dev) 11

E

ESP8266
Microcontrolador libre, economico y con Wifi, con arquitectura compatible con Arduino 16

F

Flask
Flask
Framework de modelado de APIs para Python (https
//flask.palletsprojects.com) 9, 12
Flutter
Flutter
Framework para creación de aplicaciones moviles (https
//flutter.dev) 9, 11, 12, 19, 20

J

JSON
Javascript Object Notation
Sistema de representacion de objetos en lenguaje Javascript. Estandar para transmision de datos. 12, 18
JWT
JSON Web Tokens
Tokens encriptados en formato JSON utilizados para la autenticación en servicios y APIs 12, 26, 27, 28

M

Machine Learning
  Machine Learning o ML
    Aprendizaje automatico, inteligencia artificial. 7
MAX30100
  Sensor de frecuencia cardiaca por IC de la casa Maximintegrated. 13, 16

O

OTG
  On The Go
    Forma de trabajar del USB que permite a los dispositivos moviles actuar como si fueran PC, permitiendo conectar perifericos. 8

P

pixel
  Picture Element
    La unidad mas pequeña de informacion para la representacion de colores en una pantalla. 14
Python
  Python
    Leguaje de programacion interpretado. 9

R

ReactNative
  ReactNative
    Framework para creación de aplicaciones moviles (https //reactnative.dev) 11

S

Sketch
  Forma de llamar a los programas para arduino. 17

U

USB
  Universal Serial Bus
    Sistema estandar de transmision de datos en un ordenador. 8, 9, 16, 17, 19, 21, 37, 39

W

websockets
  Tecnologia de comunicación entre programas, portada al ambiente web. 18
WemosD1
  Microcontrolador de la casa Wemos con arquitectura compatible al Arduino 16

1 Resumen

Introducción: La detección de la frecuencia cardiaca tradicionalmente se realiza mediante los dispositivos médicos conocidos como pulsioxímetros, los cuales realizan la medición en base a la cantidad de veces que cambia la intensidad de rojo al hacer pasar la luz por la punta del dedo.

Objetivos: Este proyecto tiene como **principal objetivo** implementar una técnica de detección de la frecuencia cardiaca utilizando el flash de un teléfono inteligente, así como la cámara para, a la vez, iluminar el dedo y captar la diferencia de rojo del mismo. Adicionalmente se plantea como **segundo objetivo** la utilización de un microcontrolador y un sensor de pulso para capturar la frecuencia cardiaca.

Por otro lado, como **tercer objetivo** se propone un sistema de gestión de mediciones de paciente, como la frecuencia cardiaca, la presión sanguínea (sistólica y diastólica) y el peso, además de varios indicadores del paciente. La idea es que esta plataforma sirva al médico como punto de recogida de datos para posibles análisis o estudios. De esta manera, se enseña a modo de ejemplo unas simples graficas de frecuencia, presión y peso frente a la edad, así como la posibilidad de generar y exportar las mediciones a un fichero anónimo para su posterior estudio en un paquete estadístico.

Metodología: Se definió la arquitectura de software, el diagrama y documentación de los distintos elementos que intervienen de manera física en el sistema; como son bases de datos, servidores, clientes y canales de comunicación. En base a artículos médicos de medición y a los conocimientos adquiridos durante mi formación, se han desarrollado los distintos componentes; los componentes de medición (smartphone y microcontrolador) y los componentes de gestión de datos (historia médica del paciente)

Resultados: Se implementaron con éxito las dos técnicas de detección de la frecuencia cardiaca (sensor del Arduino y el sensor del Smartphone). Se hizo la comparación de la frecuencia cardiaca detectada con el sensor del Arduino y el sensor del smartphone (la cámara y el flash) tomando como referencia un equipo profesional como gold standard y se encontró que el sensor del Arduino da mejores resultados.

En cuanto a la parte de gestión, habría que verla en funcionamiento, esto es una prueba de concepto de lo que se podría hacer. A futuro sería bueno una reunión con los médicos que vayan a utilizar la aplicación para ver cómo adaptarla mejor a sus necesidades. En líneas futuras al final del documento se incorporan algunas mejoras.

Conclusiones: Como sistema barato y sencillo y fácil de fabricar, el smartphone y las tecnologías móviles aportan mucho potencial. Si bien la medición de smartphone tiene que mejorar con algoritmos mejores de detección, la detección mediante el microcontrolador está mucho más avanzada, es barato de producir y fácil de implementar.

2 Contexto y objetivos

2.1 Contexto

La idea surge de un artículo publicado en un medio de noticias, donde destacaban la labor de Ainara Garde en el campo del diagnóstico precoz mediante las medidas de saturación de oxígeno y frecuencia cardiaca.

> Ainara Garde (natural de Pamplona y licenciada por la UPNA en Telecomunicaciones y con mater en Biomedicina), al frente de un equipo de diferentes disciplinas y cuatro países, desarrolló un instrumento que permite a los trabajadores de salud de primera línea detectar rápidamente la necesidad de los niños de ser hospitalizados y agregan que una característica común de la mayoría de las enfermedades infantiles tratables es la falta de oxígeno. Este trabajo le llevo a ganar el premio "Winter Prize" en 2017 bajo el título de "La información derivada de la frecuencia respiratoria y la oximetría de pulso como predictores de ingreso hospitalario en niños pequeños en Bangladesh" publicado en la revista BMJ Abierta.
>
> Para medir este factor de riesgo, el proyecto utilizó un sensor de dedo, el oxímetro de teléfono. De hecho, recopila datos en una aplicación de teléfono inteligente para controlar la saturación de oxígeno en la sangre y la frecuencia cardíaca de una persona. Esta información se combina con una medición de la frecuencia respiratoria.
>
> Garde desarrolló un modelo predictivo que identifica datos anormales de forma fácil y automática.
>
> Este procedimiento fue probado en un hospital en Bangladesh.
>
> Ainara Garde – Diario de Noticias – 27/11/2017

Adicionalmente se propuso por iniciativa del que realiza el proyecto, almacenar estas medidas en un sistema, como control de tensión (hipertenso) y peso. Dicho sistema diferencia como ya veremos entre médico y pacientes, y entre presión sanguínea, peso y pulso.

Posteriormente y aunando ambas iniciativas, surge la idea de exportar todas las medidas para la elaboración de estadísticas posteriores o para enriquecer modelos de Machine Learning que faciliten la predicción en futuras investigaciones.

2.2 Objetivos

Los principales objetivos del proyecto son:

- **Medición del pulso mediante la cámara y el flash del smartphone**: Todo el mundo dispone hoy en día de un smartphone, y este tiene la capacidad de captar y trabajar con la luz usando la cámara. aprovechándonos de eso podemos medir diferencia de color y reconocer latidos, pudiendo al cabo de un periodo de tiempo, calcular la frecuencia cardiaca.
- **Medición del pulso mediante un microcontrolador y un sensor**: Así mismo, los smartphones, a la vez cuenta con tecnología USB-OTG, que permite conectar periféricos al aparato a través del puerto USB. De esta manera se ha desarrollado un pequeño programita en un microcontrolador unido a un sensor más fiable, para realizar las mediciones en la aplicación.
- **Sistema (cliente-servidor) de gestión de medidas e historia médica**: Por último y para unir todo esto, se ha desarrollado un sistema de gestión de medidas por médico y paciente para el control en el tiempo, a la vez de poder realizar un seguimiento. Esta información siempre podrá ser exportada anónimamente para cualquier investigación en un paquete estadístico externo.

3 Estado del arte

La técnica del pulsioxímetro es la más utilizada hasta la fecha para la medición de la frecuencia cardiaca. Al tratarse de una técnica no invasiva es fácil su uso y su medición, a la par de ser muy precisa.

Como se expone varias veces en el trabajo, el pulsioxímetro u oxímetro de pulso, mide frecuencia cardiaca en base a la variación de luz recogida por un sensor. Con cada latido la cantidad de luz recibida por el sensor disminuye. De esta manera con un simple recuento se puede obtener la frecuencia cardiaca (Jubran et al 2015).

Existen muchos tipos de pulsioxímetro, de dedo, de muñeca, de mesa, de mano, pero la técnica es similar (https://www.healthline.com/health/pulse-oximetry). Este trabajo lo que pretende es extender el pulsioxímetro de dedo mediante las nuevas tecnologías y hacerlo accesible a todo el mundo (o al menos abaratar mucho su coste y aumentar su disponibilidad).

La tecnología ha avanzado a pasos agigantados, y lo que antes suponía un problema, como es la gestión de datos o la medición de los mismos ahora se puede resolver con mucha facilidad y desde la distancia si hiciera falta. Hoy día los móviles disponen de la tecnología suficiente como para realizar gran cantidad de seguimiento médico, inmediatez de diagnóstico, y posibilidad de hacer todo esto desde la comodidad de un aparato que siempre llevas en el bolsillo (https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007).

Es por eso que haciendo uso de un móvil o smartphone podemos realizar la medición del pulso (ya que cuentan con emisor de luz y sensor de luz) facilitando un pulsioxímetro en cada bolsillo.

4 Implementación

A continuación, se describen los requisitos del sistema (Servidor y Aplicación). La funcionalidad de medir el pulso es una de las que ofrece la aplicación. Aquí cabria distinguir dos cosas, la medición del pulso y la aplicación en sí. La aplicación se utiliza para gestionar medidas (pulso, tensión, peso, etc), así como para facilitar los datos anónimos de estas medidas para posibles estudios. Dentro de la aplicación, cuando queremos realizar una medición de frecuencia cardiaca, es donde decidiremos si se realiza mediante el smartphone o mediante el microcontrolador, pero la aplicación (y el servidor) sigue siendo la misma.

4.1 Requisitos

Los requisitos (o casos de uso, en su forma más generalizada) es la forma que se emplea en el desarrollo de software para ver que necesidades sugiere el cliente y exponerlos de manera clara en una tabla. Los requisitos recogidos para este proyecto se exponen a continuación:

- Como Médico, quiero poder registrar la edad y el género de un paciente
- Para un paciente, quiero poder introducir variables físicas observadas durante la exploración de dicho paciente, esto es además de la presión sanguínea, el pulso o el peso, los siguientes indicadores: “Enfermedad Renal Crónica”, “Asma”, “Enfermedad Pulmonar Obstructiva Crónica”, “Diabetes”, “Dislipemia”, “Cardiopatía Isquémica”, “Insuficiencia cardiaca previa”.
- Para un paciente, quiero poder realizar una medida de pulso (frecuencia cardiaca) utilizando la cámara y el flash del smartphone.
- Para un paciente, quiero poder realizar una medida de pulso (frecuencia cardiaca) utilizando un pequeño sensor conectado por USB (Arduino).
- Como médico, quiero poder guardar los datos de esta exploración en un servidor para alimentar un modelo predictivo.
- Como Médico, quiero poder hacer uso de ese modelo predictivo para mostrar una valoración automática por pantalla.
- Como Médico, quiero poder exportar los datos de la base de datos, anonimizados para el posterior análisis en un ordenador.
- Como Medico, quiero poder acceder al mismo sistema (base de datos) desde cualquier smartphone utilizando una cuenta personal (email y contraseña).
- Como Medico, me gustaría poder utilizar la huella y demás sensores biométricos para utilizar la aplicación, sin depender de meter la contraseña cada vez.

4.2 Arquitectura

Se define arquitectura de software, al diagrama y documentación de los distintos elementos que intervienen de manera física en el sistema; como son bases de datos, servidores, clientes y canales de comunicación. En este caso son tres principalmente, la base de datos (MariaDB en este caso), el servidor (Python + Flask en este caso) y el cliente móvil (Flutter en este caso), como se muestra en la figura 1.

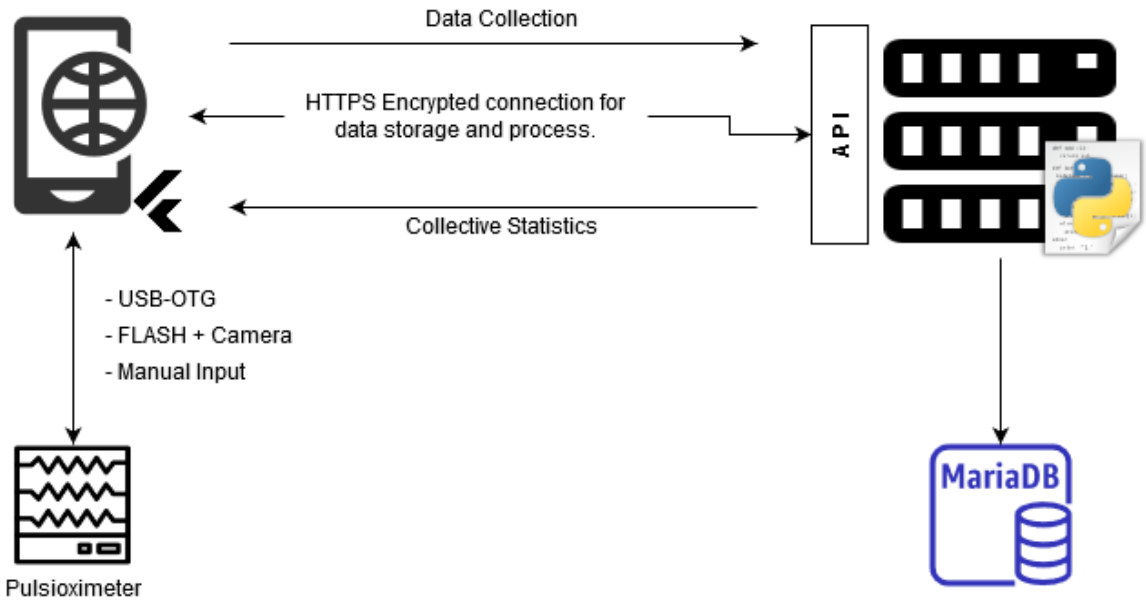


Figura 1.-Descripción de los elementos integrantes del sistema

4.3 Modelo de datos

La aplicación consta de dos partes diferenciadas. Una es la toma de medidas y su posterior análisis, exportación de datos, etc. Otra es la gestión de pacientes. Por lo tanto, el modelo que se propone es el siguiente (Figura 2).

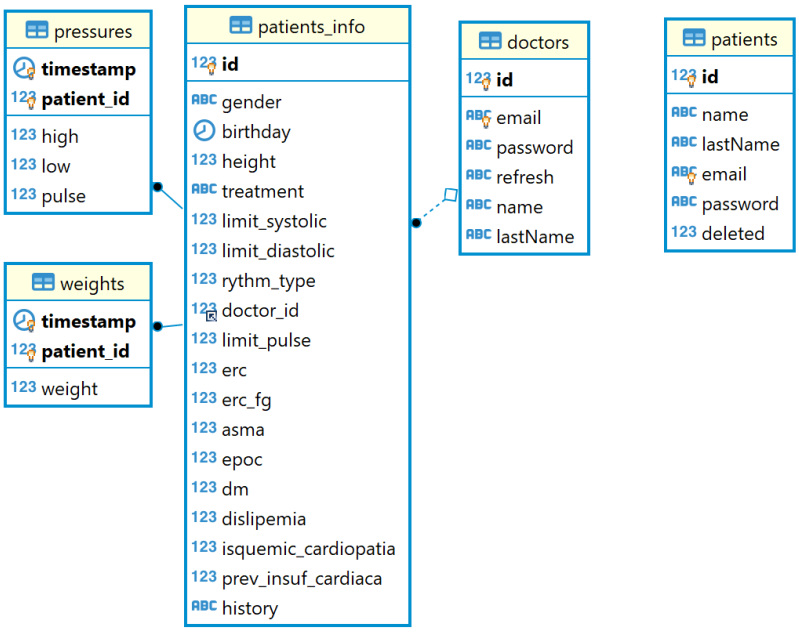


Figura 2.- Modelo de datos presente en todo el sistema, información almacenada del paciente con sus mediciones y la relación entre ellas.

Podemos diferenciar cuatro tablas principales, y una quinta separada y sin relación que es la información sensible del paciente. La idea detrás de esto es poder externalizar en un futuro la tabla de pacientes, la información sensible, de manera que nuestro sistema no albergue información sensible y sea más fácil de mantener e institucionalizar.

Las relaciones son de "muchos a muchos" entre la presión sanguínea y la información anónima de paciente, y entre el peso y la información anónima del paciente, ya que son datos. El resto de los datos, como se pueden ver solo se almacenará una copia, la última.

4.4 Cliente

La programación del cliente (o teléfono) de este sistema lo único que requiere es acceso a bajo nivel al hardware del teléfono, con lo que tendremos que ir a lenguajes de programación nativos si queremos obtener datos fiables de la cámara, puesto que necesitaremos diferencias de color en tiempo real para poder extrapolar la gráfica del latido del corazón.

Quedan descartadas entonces las soluciones hibridas, Cordova/PhoneGap así como las soluciones web PWA o aplicaciones progresivas en HTML5, ya que por tema de rendimiento y accesibilidad a las APIs básicas del teléfono, sería muy poco eficiente y muy lento, si es que al final puede realizarse.

Para la programación nativa lo ideal será realizarla en las dos plataformas más extendidas como son Android e iOS. Sin embargo, últimamente han surgido soluciones muy interesantes por parte de Google y Facebook para facilitar la compilación a código nativo en estas dos grandes plataformas, partiendo de un único código común. Estas soluciones son Flutter (https://flutter.dev/) y ReactNative (https://reactnative.dev/) respectivamente.

Mi propuesta inicial, y siempre y cuando estos frameworks no supongan una limitación, es utilizar Flutter para el desarrollo de la aplicación de cliente, que como ya he dicho antes se traducirá en Android/Java cuando compilemos para Android y en ObjectiveC/Swift cuando compilemos para iOS.

Para las partes donde hace falta acceso a más bajo nivel, se han tenido que programar piezas especificas en Android (no se dispone de un ordenador Mac para la programación en iOS), por la ya comentada carencia en el framework de Flutter.

Flutter utiliza un lenguaje de programación de alto nivel desarrollado por Google, llamado Dart (https://dart.dev/). Emplea la orientación a Objetos y es muy flexible.

4.4.1 Diseño y funcionalidades

Esta es la aplicación móvil que se encarga de hablar con el servidor. La parte que el usuario/medico va a ver. Entre sus funcionalidades encontramos:

- Consultar el histórico de pacientes
- Consultar estadísticas de todas las mediciones
- Crear nueva medición para un paciente concreto
- Medición de peso *(Entrada manual)*
- Medición de presión sanguínea *(Entrada manual)*
- Medición de pulso *(Entrada manual, y usando Cámara y flash del teléfono Android)*

4.5 Servidor

Para la implementación en servidor se plantea el uso de Python como lenguaje de programación web. No es el más versátil, pero si es el que mejor va a manejar todo el procesamiento de datos y ofrece la posibilidad si hiciera falta de interactuar con R en el servidor en el que decidamos desplegarlo.

Para la parte de la API, con el framework Flask, estaríamos cubiertos, puesto que no queremos mostrar nada vía web, ni mostrar ningún contenido html, no necesitamos dJango (https://www.djangoproject.com/) para nada, ya que al ser un framework completo vista-controlador, incorpora muchas librerías de ayuda para la gestión de páginas web, para frontend, Flask (https://flask.palletsprojects.com/en/1.1.x/) por otro lado es un framework simple que nos permitirá devolver en formato JSON la información para que el cliente la reciba y pinte la información en el teléfono. No queremos más que mostrar una página web de inicio donde bajarse la aplicación, el resto de la funcionalidad del servidor se expondrá mediante una API, un conjunto de direcciones de internet que el cliente (la aplicación Flutter) llamara y utilizara para recuperar los datos.

Adicionalmente se propone el formato CSV (Comma Separated Value, o archivo de valores separados por comas) para descargar el listado de las mediciones, como se plantea en uno de los requisitos.

La manera que tienen las aplicaciones cliente de conectarse con el servidor es mediante el uso de API. La API o “Application Programing Interface” es la interfaz que un servidor expone a internet para que se le pueda preguntar por información. Anexo, expongo los diferentes “endpoints” disponibles y para que se utilizan, así como el modelo de datos (DTO).

La autenticación se realiza mediante tokens JWT (https://tools.ietf.org/html/rfc7519) o “Json Web Tokens”. Estos tokens se envían en la cabecera de la llamada, y contienen el identificador del médico logueado, la fecha hasta que ese token es válido, y todo ello firmado con una clave privada que solo conoce el servidor. Si un atacante quisiera suplantar a un médico, la firma fallaría y se vería que el token no es válido. Es el estándar hoy día para esta clase de servicios “solo” API, son sencillos de entender, implementar y usar, y se integran muy bien con la aplicación que se ha desarrollado. El servidor devolverá un 401-403 (código HTTP para no autorizado o no autenticado, https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) indicando que no está autorizado.

4.6 Medición de frecuencia cardiaca

Como se ha indicado en la introducción, se presentan dos métodos para la obtención de la frecuencia cardiaca, el teléfono con el flash y la Cámara, y un pequeño dispositivo Arduino (un microcontrolador) con un sensor de frecuencia MAX30100.

4.6.1 Smartphone: Cámara y flash

La manera en que un pulsioxímetro realiza la medición de la frecuencia cardiaca es mediante la intensidad de luz que llega a un receptor tras pasar a través del dedo del paciente (Figura 3 y 4).

Figura 3.- Pulsioxímetro comercial

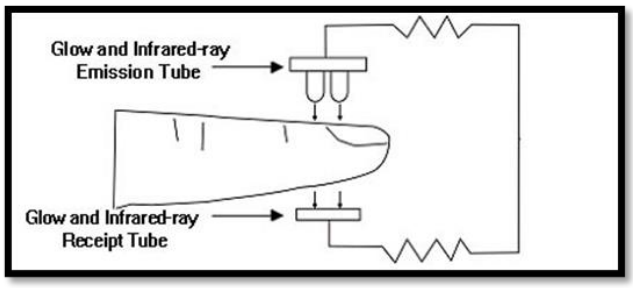


Figura 4.- Diagrama de funcionamiento de un pulsioxímetro

Al oxigenarse el dedo con cada latido, hay más sangre circulando por los vasos del dedo, lo que impide que más cantidad de luz llegue definitivamente al sensor. Midiendo la cantidad de esas variaciones en un tiempo concreto, podemos determinar el número de latidos por minuto. Vamos a replicar este mecanismo con el flash y la Cámara de un teléfono inteligente.

En el caso del smartphone la idea es similar, solo que en vez de tener el emisor de luz y el receptor alineados en la misma vertical, usaremos el flash del smartphone como la fuente de luz, y la cámara como el sensor de luz.

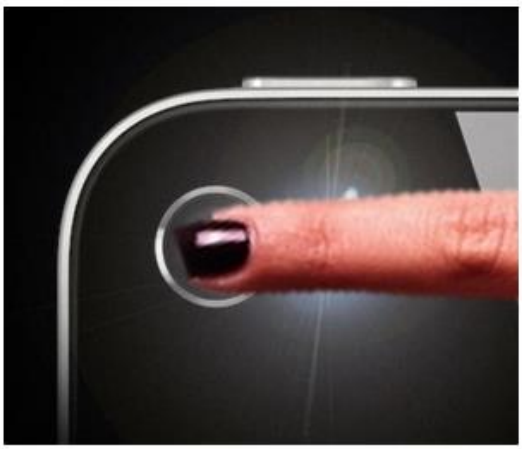


Figura 5.- Posicionamiento del dedo en el Smartphone

Mediremos diferencia de color (en este caso el rojo) para ver cuando hay un flujo de sangre (debido al latido) y cuando no.

Ahora que sabemos cómo identificar un latido, simplemente tenemos hacer el recuento de latidos en un periodo de tiempo para obtener las pulsaciones por minuto.

Esta diferencia de color la mediremos solamente en el canal rojo (Siddiqui et al, 2016), ya que la sangre es de ese color, y sería inútil contemplar los otros canales ya que añadirían coste computacional para no obtener ningún beneficio.

Antes de seguir con los algoritmos implementados, es interesante esbozar cómo se representan en memoria y en un ordenador las imágenes, para entender el proceso aplicado al cálculo de la variación de la intensidad.

Esencialmente en un ordenador la forma en que se representan los diferentes colores en una pantalla es mediante la combinación con distinta intensidad de los tres colores principales, Rojo, Verde y Azul. Así un pixel (o Picture Element) es la unidad mínima representable en un sistema lógico, y tendrá como si de un sistema de coordenadas se tratara, tres valores asociados, una intensidad para el rojo, una intensidad para el verde y una intensidad para el azul (Figura 6).

Podemos entender entonces que una imagen es una repetición de estos pixeles ordenados en dos dimensiones, y un video es un array en el tiempo de estas imágenes. Así podemos entender una imagen como una matriz de tantas columnas como el ancho de la imagen, tantas filas como el alto de la imagen y con tanta profundidad como el número de canales, normalmente tres, rojo, verde y azul.

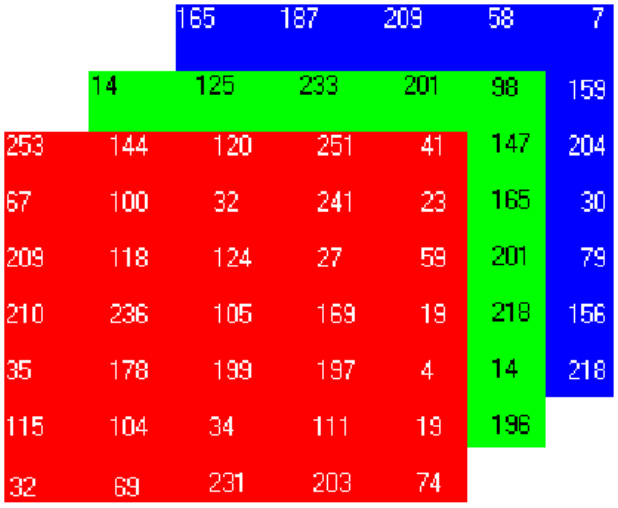


Figura 6.- Representación matricial en capas de una imagen. Cada capa es un canal, rojo, verde y azul, con las intensidades de cada canal.

De esta manera y siguiendo la figura 6 en la posición 1,1 de la imagen, tendremos el valor (253,14,165) lo que equivale a este color: COLOR. La forma en la que se almacenan los colores puede variar, dando más o menos definición a la intensidad que se puede alcanzar, pero el estándar común aceptado por la red es de 32bits de información por canal o lo que es lo mismo 256 valores de intensidad (o FF valores si lo representamos en hexadecimal, que es muy común).

Al capturar una imagen de un objeto eminentemente rojo, los otros dos canales (verde y azul) no aportarán nada a la imagen, con lo que tendrán valores próximos a 0 y desechables. Convirtiendo una matriz de AxAx3 en una matriz de AxA (donde A y A son alto y ancho respectivamente), reduciendo enormemente la complejidad (Siddiqui et al, 2016).

El cálculo ahora se limita a agregar la matriz de colores como la media de las intensidades, a repetirlo en un tiempo establecido y a tomar la variación de dicha agregación (en este caso la media) como un latido del corazón.

Se adjunta el código completo (Figura 7). Sin embargo, quiero resaltar aquí las partes más importantes. Una primera parte en la que calculamos la media de intensidades de la imagen y lo guardamos en un array de medias. Con ese array podemos ver cuando estamos subiendo y cuando bajando para detectar un latido.

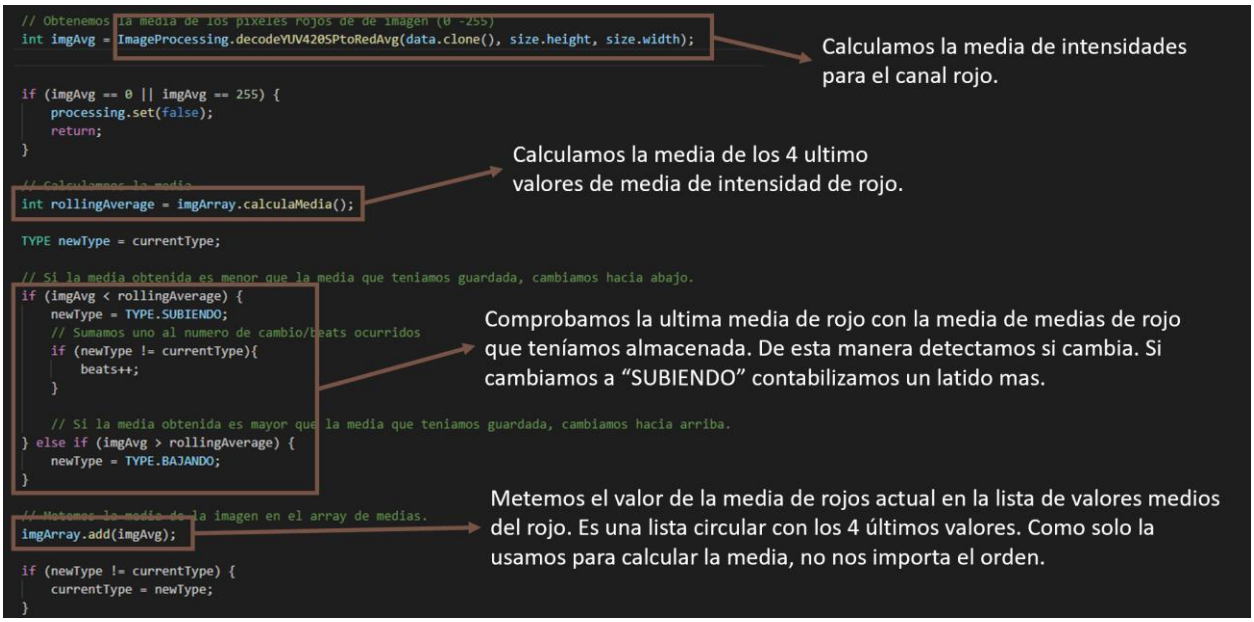


Figura 7.- Codigo del cálculo de un latido mediante la aplicación en el smartphone: cámara y flash.

En el siguiente paso (Figura 8), cronometramos cuantos latidos ha habido en un periodo de tiempo y devolvemos el número de pulsaciones por minuto o bpm.

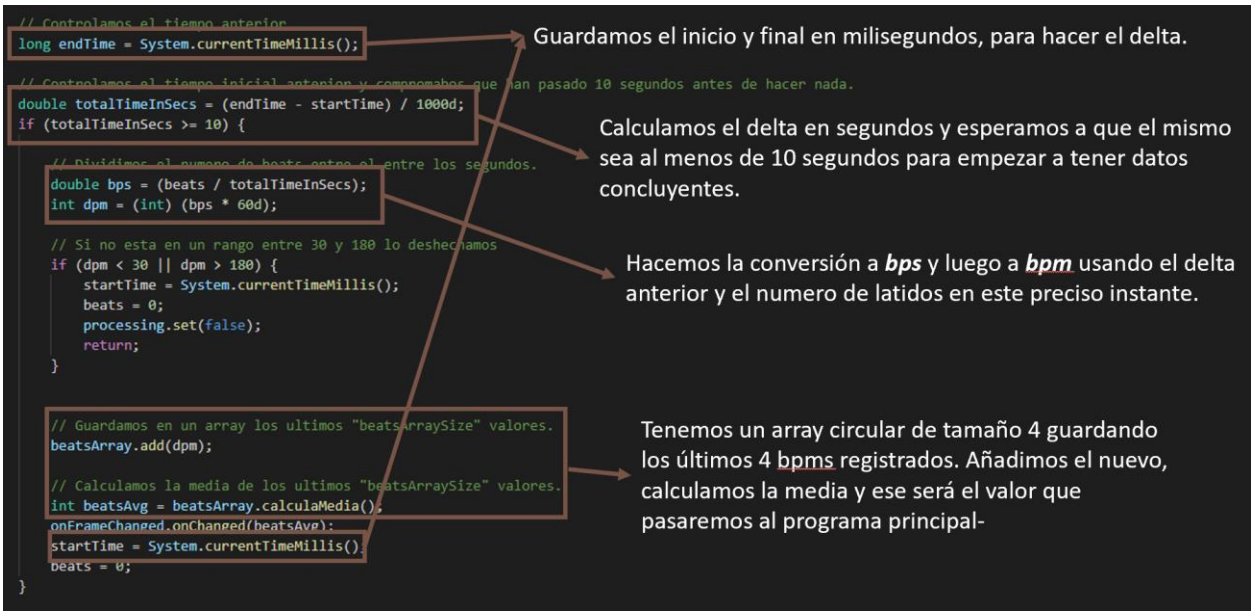


Figura 8.- Agregación o diferencia de intensidades en un tiempo concreto para la medición ahora en latidos por minutos.

4.6.2 Microcontrolador: WemosD1 + MAX30100

Adicionalmente y como alternativa a la cámara del teléfono, y puesto que este puede no tener flash o estar en una posición imposible la cámara con respecto del flash, se añade la posibilidad de utilizar un microcontrolador y un sensor de frecuencia para realizar la medición (Figura 9). Este controlador se conectará al teléfono mediante el puerto USB del mismo, haciendo uso de la tecnología OTG existente en todos los teléfonos actuales (Figura 1).

Para este proyecto el microcontrolador elegido es un WemosD1 (de la marca Wemos), una implementación del conocido ESP8266 con más ROM y conexión serial mediante USB. El ESP8266 es un microprocesador que surgió hace ya unos años con la peculiaridad de implementar toda la pila de red y conexión inalámbrica 802.11b/g/n. Los drivers permiten la conexión con el PC de la misma manera que lo haría un Arduino, la programación es C, similar al Arduino y se puede programar desde el mismo entorno de desarrollo de Arduino. A todos los efectos y con la configuración adecuada es un Arduino con conexión inalámbrica. Por ello en adelante se referirá a él como Arduino.

Como sensor de frecuencia cardiaca se eligió el MAX 30100 (Maxim Integrated), un sensor económico que se conecta de manera sencilla con cualquier microcontrolador mediante el protocolo I2C. Trabaja a 3.3v y solo necesita de dos entradas digitales en el Arduino para funcionar.

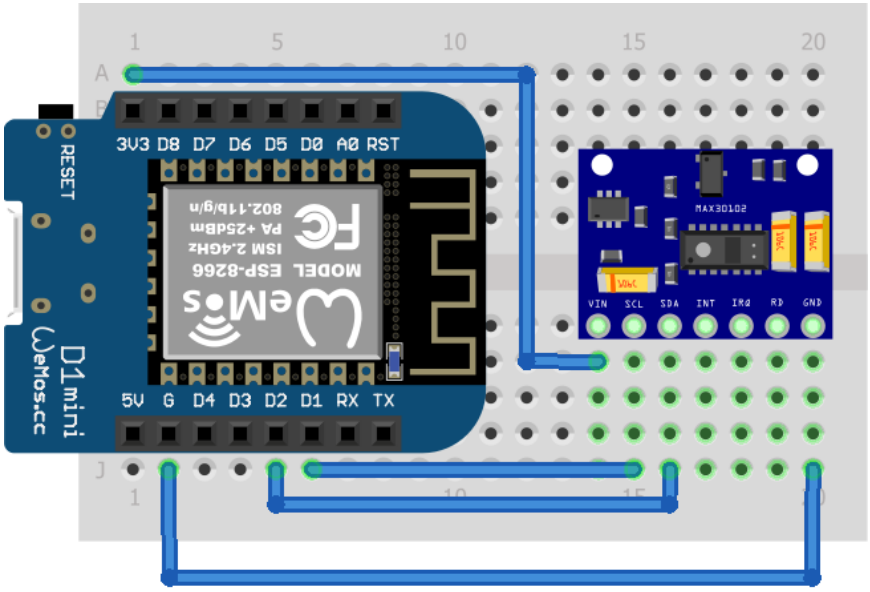


Figura 9.- La instalación del microcontrolador (en la parte derecha) y el sensor de pulso (en la parte izquierda) y los 4 cables necesarios para su correcto funcionamiento.

Por otro lado, el estándar USB proporciona por dos de sus líneas alimentación de 5v y tierra, lo que facilita que el propio smartphone mediante su conexión Micro-USB o USB-C proporcione alimentación al circuito. Como he dicho antes el WemosD1 implementa el microcontrolador ESP8266 pero le añade varias funcionalidades. Una de estas funcionalidades es un regulador de voltaje que transforma los 5 voltios continuos en 3.3 voltios continuos.

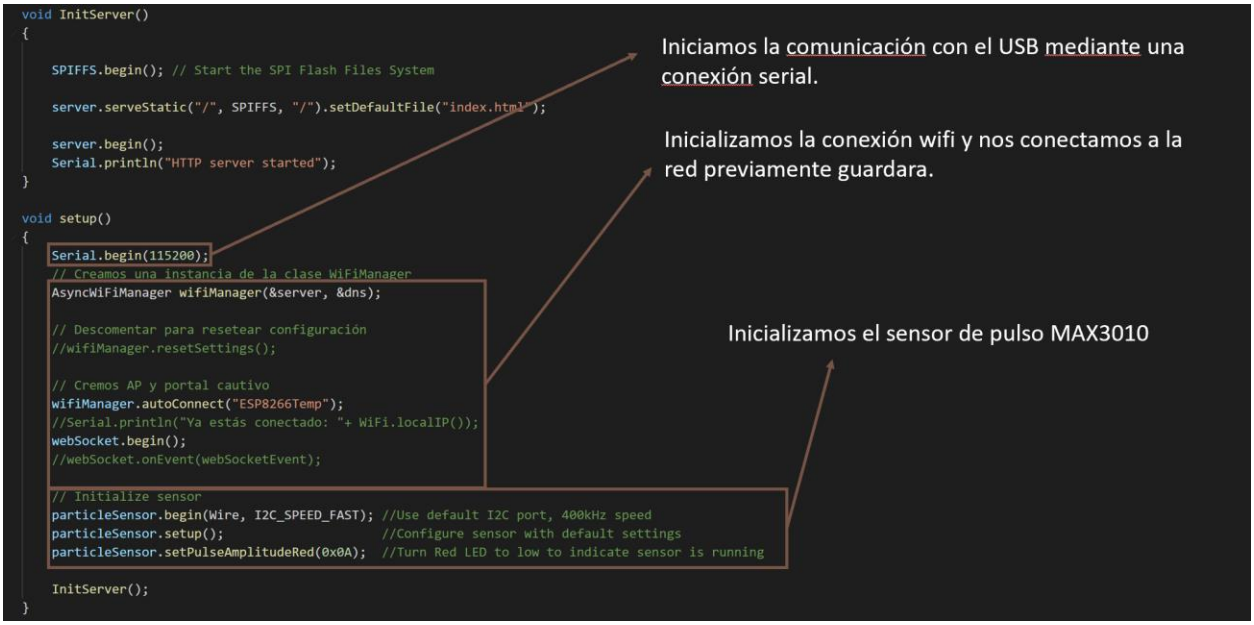
De la misma manera que con la medición mediante la cámara, se proporciona el código completo en los anexos del proyecto. Sin embargo, aquí se destaca lo más relevante.

Al ser un programa de Arduino, el "Sketch" consta de dos funciones principales, una de configuración y otra de bucle infinito donde a cada tick de reloj se repetirá una función. Estas funciones son las de setup() y la de loop().

En la función setup() inicializamos el WifiManager (se encarga de crear una red wifi para que configuremos la conexión al router, en caso de no poder conectarse a ninguna red al arrancar) e inicializamos el sensor de pulso ("***particleSensor***").

La conexión con el sensor de pulso se hace mediante el protocolo I2C. Al utilizar los puertos D1 y D2 del GPIO del Arduino, que son los que él espera para una conexión I2C, podemos directamente cargar el objeto ***Wire*** y pasárselo al ***particleSensor*** para inicializar el sensor.

También abrimos la conexión "***Serial***" con el que se conecte por USB. En la fase de desarrollo es el ordenador y una terminal serial, y una vez terminado será el smartphone mediante el USB-OTG.




```
void InitServer()
{

    SPIFFS.begin(); // Start the SPI Flash Files System

    server.serveStatic("/", SPIFFS, "/").setDefaultFile("index.html");

    server.begin();
    Serial.println("HTTP server started");
}

void setup()
{
    Serial.begin(115200);
    // Creamos una instancia de la clase WiFiManager
    AsyncWiFiManager wifiManager(&server, &dns);

    // Descomentar para resetear configuración
    //wifiManager.resetSettings();

    // Cremos AP y portal cautivo
    wifiManager.autoConnect("ESP8266Temp");
    //Serial.println("Ya estás conectado: "+ WiFi.localIP());
    webSocket.begin();
    //webSocket.onEvent(webSocketEvent);

    // Initialize sensor
    particleSensor.begin(Wire, I2C_SPEED_FAST); //Use default I2C port, 400kHz speed
    particleSensor.setup();                     //Configure sensor with default settings
    particleSensor.setPulseAmplitudeRed(0x0A);  //Turn Red LED to low to indicate sensor is running

    InitServer();
}
```

Figura 10.- Código de inicialización del microcontrolador, inicialización de la red WIFI y del sensor de pulso.

Por otro lado, en la función loop() empezamos por comprobar si la intensidad de infrarrojos es mayor que un umbral (dedo encima del sensor) y esperamos que haya un pulso ("***checkForBeat()***").

Cuando lo encontramos (valor booleano de si o no) guardamos ese milisegundo y hacemos cálculos con los milisegundos anteriores para calcular la media de pulsaciones. Volvemos a estar en una situación en la que tenemos un si o no a "hay un pulso", y es el recuento en un delta de tiempo el que nos da las pulsaciones por minuto.

Al final mandamos el pulso por el USB ("***Serial.println()***").

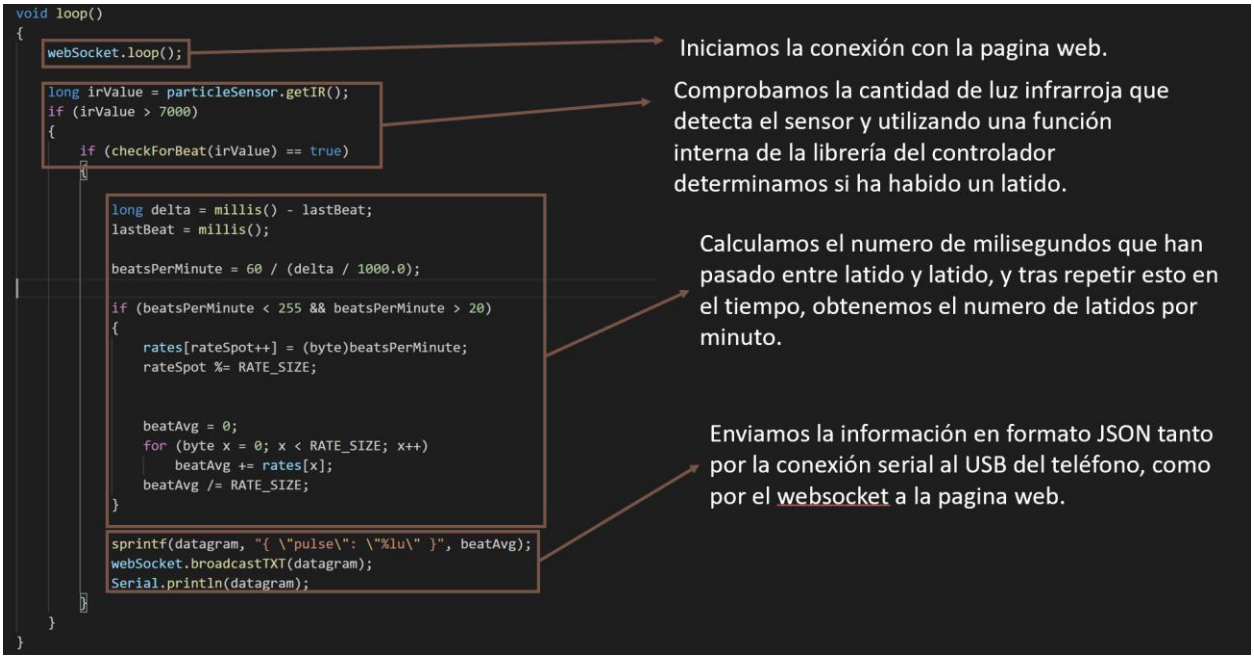


Figura 11.- Código que se repite indefinidamente. Se realiza aquí la medición en sí, la detección de latido y la agregación de varios latidos para calcular los bpm.

Adicionalmente, aunque solo sirve de manera orientativa ya que no interactúa con la app, se manda la información del pulso obtenida mediante websockets a una página web simple que el propio microcontrolador sirve.

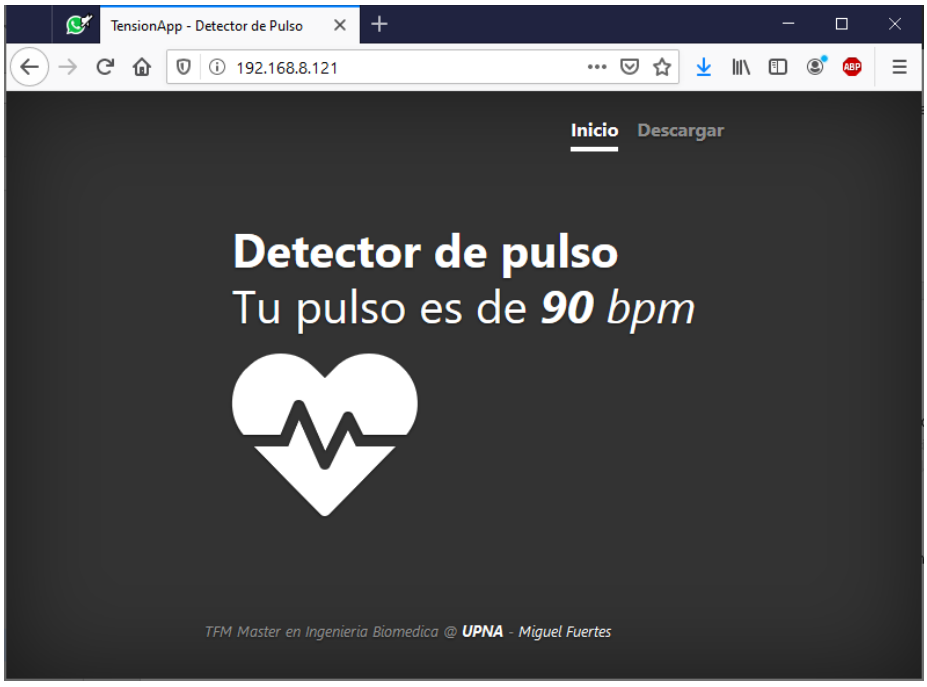


Figura 12.- Página web servida por el microcontrolador para ver lo que está midiendo. (Complementario al visor en la aplicación)

La cadena de texto con la información en formato JSON es luego capturada por la clase Java nativa de Android y procesada.

Cuando todo está correcto y el usuario pulsa el botón “Guardar”, se le devuelve el valor “_beat” al widget de ***Flutter*** para su posterior envío al servidor.

Como dije al principio, no todo se puede hacer con ***Flutter***, y aunque esto sí, era más claro desarrollarlo nativamente en Android y conectarlo a ***Flutter***. En cualquier caso, iOS es un sistema muy cerrado y no permite el acceso al USB a cualquier aplicación.

```
@Override
public void onNewData(byte[] data) {
    try {
        JSONObject pulse = new JSONObject(new String(data));

        _beat = pulse.getInt("pulse");
        runOnUiThread(() -> {
            //_heart.setColorFilter(_red ? Color.RED : Color.BLACK);
            _text.setText(_beat + " bpm");
        });
    } catch (JSONException e) {
        e.printStackTrace();
    }

}
```

Figura 13.- Pieza de código encargada de recoger el dato de pulso desde el microcontrolador en la aplicación Flutter (Android)

Siendo ***data = “{pulse:’120’}”***

5 Resultados de la implementación

5.1 ¿Se ha conseguido desarrollar/implementar completamente cada una de las técnicas?

Se han conseguido implementar las dos técnicas propuestas, esto es, la medición mediante el smartphone y la medición mediante el sensor del microcontrolador Arduino.

Para realizar la medición del smartphone se necesitó usar la API nativa de Android, ya que como se comenta anteriormente el framework de Flutter no es capaz a día de hoy de manejar la cámara y el flash al mismo tiempo.

5.2 ¿Funcionan cada una de las técnicas correctamente, se han comparado los valores de frecuencia cardiaca?

Si bien ambas técnicas funcionan correctamente, sin embargo, la precisión en cada una de ellas difiere. Es más precisa la medición mediante el microcontrolador, ya que el receptor del sensor (MAS 3010, o cualquier sensor de pulso) solo capta la luz infrarroja reflejada del dedo, obviando todas las demás frecuencias, con lo que la medida es más clara, y es más fácil determinar el latido, importante para el cálculo de pulsaciones por minuto.

En esencia, el sensor del microcontrolador hace lo mismo que el método del smartphone, pero donde éste deshecha los canales verde y azul en posproducción, el sensor del microcontrolador directamente emite solo luz roja/infrarroja, de manera los otros dos canales no interfieren en la medición.

Aquí se muestra una tabla con mediciones hechas con un tensiómetro estándar, el smartphone (la Cámara y el flash), y el microcontrolador. (Las unidades son pulsaciones por minuto, bpm).

TENSIÓMETRO (BPM)	**SMARTPHONE** (BPM)	**MICROCONTROLADOR** (BPM)
63	75	70
60	67	64
62	62	60
74	83	80
69	75	67

Tabla 1.- Comparación de medidas entre ambos sistemas.

5.3 Comparación de ambas técnicas desde el punto de vista computacional y desde el punto de vista del programador.

A nivel computacional, la complejidad es similar, sin embargo, por como son ambas arquitecturas, el smartphone y el Arduino, es mucho más rápido y potente el microcontrolador, ya que se trata de un dispositivo que solo va a hacer una cosa, en vez del smartphone que tiene múltiples procesos funcionando a la vez.

Mientras que el smartphone captura fotogramas al máximo de su resolución para hacer una agregación y sacar un único valor, el microcontrolador toma una imagen mucho más reducida y de un mapa de colores distinto y más optimizado, con lo que el cálculo matricial es menor. Sin embargo, la potencia del procesador de un smartphone actual hace que el cálculo de toda esa imagen por minuto sea instantáneo, con lo que al final el rendimiento es similar.

En caso de teléfonos antiguos con menos RAM o con sistemas operativos más potentes, esta diferencia sí que podría ser mayor.

A la hora de programar, al ser más inteligente el smartphone, el nivel de programación puede ser más alto, o de más alto nivel, abstrayendo mucha funcionalidad, haciendo mucho más sencilla su programación. Por su lado el microcontrolador está programado en C (un lenguaje de muy bajo nivel) con una programación más compleja.

5.4 Comparar las dos técnicas desde un punto de vista de operatividad para el médico o del usuario

Desde el punto de vista del médico, la complejidad es la misma, puesto que desde la propia aplicación se indican los pasos a seguir si queremos utilizar un método de medición u otro. Para el caso del microcontrolador solo tenemos que conectarlo mediante el cable al puerto USB del teléfono (al puerto por donde se carga).

En el caso del usuario final que se baje la aplicación en un futuro, y como se sobreentiende que no tendrá acceso al Arduino, la operatividad es tan sencilla como poner el dedo encima de la cámara cuando entras en la pantalla de medición. En cualquier caso, la aplicación a la hora de hacer este documento solo está enfocada a médicos. En las líneas futuras se propone una aplicación para pacientes en un futuro.

5.5 Comparar las dos técnicas desde un punto de vista económico

Desde el punto de vista económico la respuesta es obvia, si partimos de la base que el usuario potencial ya posee un smartphone. Sin embargo, el desembolso económico para realizar el montaje del microcontrolador no supera en la mayoría de las plataformas de venta online los 10 euros.

5.6 ¿En qué escenarios/entornos recomiendas una técnica y la otra?

Siempre que el centro tenga la posibilidad de adquirir/fabricar el microcontrolador, es preferible el microcontrolador.

5.7 Tabla de fortalezas y debilidades

A continuación, se describen las principales fortalezas y debilidades de ambos procesos de lectura de frecuencia cardiaca.

5.7.1 Arduino (WeMosD1) + Sensor MAX3010

Fortalezas	**Debilidades**
Más sensible y preciso	Aparato externo
No le afecta la luz ambiental.	Compatibilidad con smartphone (OTG)
Económico	MAX3010 difícil de encontrar
Actualizable (por ejemplo, con sensores nuevos)	

Tabla 2.- Fortalezas y debilidades del microcontrolador

5.7.2 Smartphone (Cámara + Flash)

Fortalezas	**Debilidades**
Accesibilidad (siempre contigo)	Menos preciso
Facilidad de uso (dedo sobre la cámara)	Requiere que la cámara y el flash estén cerca entre sí

Tabla 3.- Fortalezas y debilidades del smartphone (Cámara + Flash)

6 Conclusiones

Se ha logrado implementar con éxito dos técnicas de detección de la frecuencia cardiaca utilizando la tecnología disponible en los teléfonos móviles actuales: una primera técnica basada en la operatividad del flash y la cámara del teléfono, y otra segunda técnica utilizando conjuntamente un microcontrolador y un sensor de pulso externos junto con el teléfono móvil.

Basándonos en resultados obtenidos, se tiene confianza en que esta aplicación es de gran utilidad para los médicos: tanto para gestionar a sus pacientes y ver rápidamente una posible patología, como para almacenar una gran cantidad de datos que poder utilizar luego anónimamente para realizar estudios estadísticos. La confianza en las técnicas implementadas en este proyecto se ha reforzado tras recibir la valoración positiva de un médico cardiólogo interesado en el proyecto, el cual valoró el trabajo del siguiente modo: “le veo futuro a la aplicación en el campo de seguimiento de pacientes, y en el campo de la investigación”.

7 Líneas futuras

Una de las carencias que la cardióloga detectó es que la aplicación a día de hoy solo le es funcional al médico, y solo tiene sentido en el tiempo en que el paciente está en consulta. A futuro tiene sentido dividir la aplicación en dos, una parte para el paciente y otra para el médico. Así la toma de mediciones quedara del lado del paciente (y escondida en un submenú para el medico) y la estadística de datos o posible mensaje de diagnóstico de parte del médico.

Visto que cada médico puede atender a muchos pacientes y para no sobrecargar el sistema, la arquitectura que se propone a futuro es la siguiente (Figura 14).

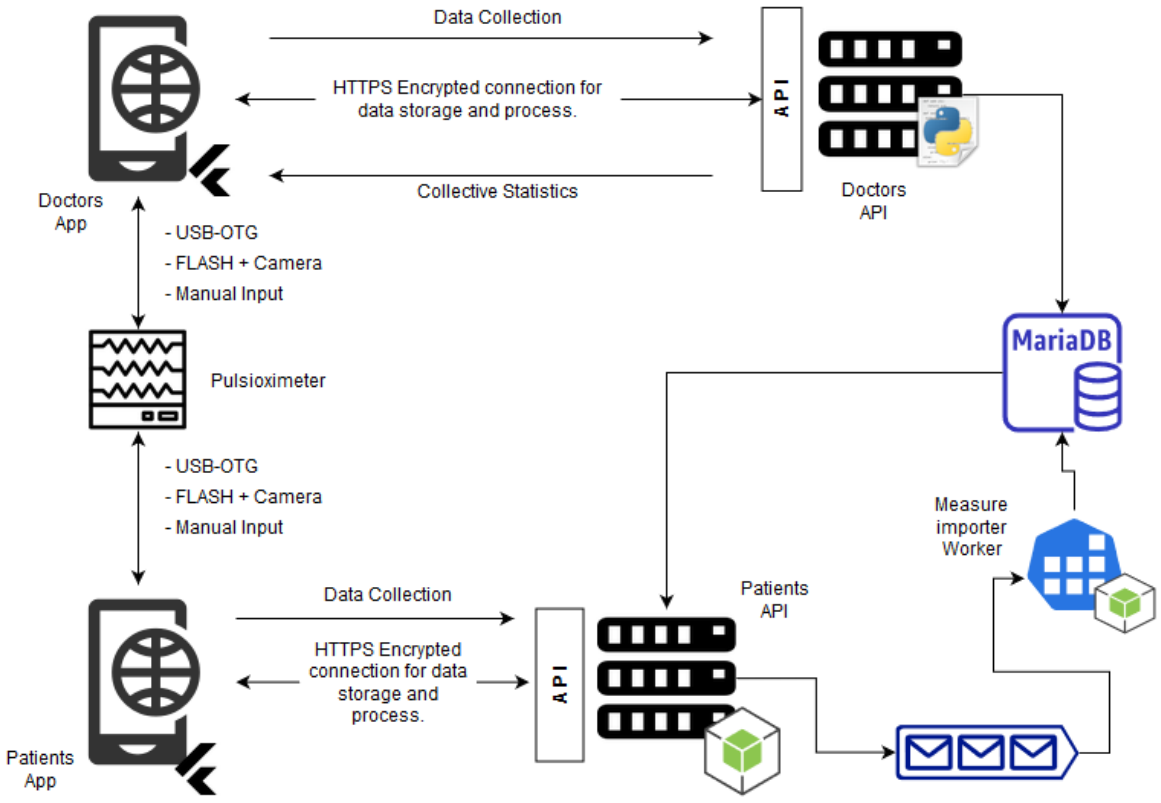


Figura 14.- Futuro del sistema (Aplicaciones y Servidores)

La idea es separar la aplicación en dos, como ya se ha comentado, e incorporar un servidor más que dependa de la misma base de datos. Las mediciones realizadas por los pacientes se encolarán y se procesarán a medida que el encargado de procesar los mensajes tenga tiempo de ejecución disponible. Así aceleramos el proceso y hacemos todo más confiable. A futuro podemos añadirle más colas y replicación de mensajes para asegurar el servicio.

Aunque a día de hoy ya se cuenta con un campo “tratamiento” donde el medico puede guardar información relevante al paciente y a su tratamiento, a día de hoy solo se puede ver y editar desde la aplicación del médico, la única que hay. Con el nuevo sistema, se espera que el medico vea y edite el campo, y que el paciente, instantáneamente vea lo que el doctor le ha pautado.

Adicionalmente se le puede incorporar un servicio de mensajería entre el paciente y el doctor.

En el campo de la adquisición de datos, se podría perfeccionar el sistema de captura de datos con la cámara, haciéndolo más fiable con más pruebas contra pulsioxímetros reales, así como integrar una comunicación persistente entre la aplicación del paciente y el posible wearable que este pueda poseer.

8 Referencias

- Siddiqui, S. A., Zhang, Y., Feng, Z., & Kos, A. (2016). A Pulse Rate Estimation Algorithm Using PPG and Smartphone Camera. *Journal of medical systems*, *40*(5), 126. https://doi.org/10.1007/s10916-016-0485-6

- Jubran, A. (2015). Pulse oximetry. Critical Care, 19(1). doi:10.1186/s13054-015-0984-8 (https://doi.org/10.1186/s13054-015-0984-8)

- (https://www.healthline.com/health/pulse-oximetry) Medically reviewed by Carissa Stephens, RN, CCRN, CPN — Written by Ana Gotter — Updated on August 2, 2017

- Mobile selling trends since 2007: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

9 Anexos

9.1 Anexo 1: Documentación de la API (Servidor)

En este anexo se describen tanto los endpoints a los que llama la app (o cliente) como ejemplos de respuestas para todas las llamadas.

9.1.1 Endpoints

Aquí se detallan los enpoints en los que la API está escuchando peticiones.

9.1.1.1 Endpoints para el acceso y el refresco de autenticación

La API usa tokens JWT para autenticar y autorizar todas las llamadas siguientes. Si el token queda invalido, también se provee un token de refresco para renovar el token de acceso. Las llamadas son las siguientes.

Para obtener un token valido o refrescarlo, la llamada es la siguiente.

Acción	URL	Método	Header	Body
Login	`/auth`	**POST**		`{"username": <string>,"password":<string>}`
Refresh	`/refresh`	**POST**	Refresh JWT "Bearer"	

La respuesta de login será similar a esta, en caso de no error (Código HTTP `200`):


```
{
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1ODAxN
TI2ODUsIm5iZiI6MTU4MDE1MjY4NSwianRpIjoiZTExMGY5MjYtZjlhNi00ODRlLWJiNDItNWQ3
NTc4NmNhY2I5IiwiZXhwIjoxNTgwMTUzNTg1LCJpZGVudGl0eSI6ImhrZnVlcnRlc0BnbWFpbC5
jb20iLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MifQ.tnG5P7oCI4TvNgugnx9Cl2acCw0
snE0MtwGm9vhYHKE",
    "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1ODAx
NTI2ODUsIm5iZiI6MTU4MDE1MjY4NSwianRpIjoiNzg2N2QwMDQtNDI0YS00M2ZmLTg3YWUtMzM
yMzhmOTVhNjIxIiwiZXhwIjoxNTgyNzQ0Njg1LCJpZGVudGl0eSI6ImhrZnVlcnRlc0BnbWFpbC
5jb20iLCJ0eXBlIjoicmVmcmVzaCJ9.9hJ8v5j2KRyiA9vdus5UFQzCnGUJoh9FeJ_ayOpffcw"
}
```


La respuesta para refresh será similar a esta, en caso de no error (Código HTTP `200`):


```
{
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1ODAxN
TI3OTEsIm5iZiI6MTU4MDE1Mjc5MSwianRpIjoiYzFjZTFkMDYtNzE2YS00NWY3LThmMDMtMWQw
OWExZDM3ZDc0IiwiZXhwIjoxNTgwMTUzNjkxLCJpZGVudGl0eSI6ImhrZnVlcnRlc0BnbWFpbC5
jb20iLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MifQ.M89-uHvKNGOqsTjkfpwT5MU-IBO
fIFdMfuJvUpWpDOY"
}
```

9.1.1.2 *Endpoints relacionados con los Médicos*

Las llamadas a la API para manejar información sobre el doctor. Actualmente solo se usa el GET y el PUT en la app.

Acción	URL	Método	Autenticación
Crear Médico	`/api/v1/doctor`	**POST**	Private Key "PSK" in Header
Actualizar Médico	`/api/v1/doctor`	**PUT**	JWT in Header "Bearer"
Borrar Médico	`/api/v1/doctor`	**DELETE**	JWT in Header "Bearer"
Ver Médico	`/api/v1/doctor`	**GET**	JWT in Header "Bearer"

9.1.1.3 *Endpoints relacionados con los Pacientes*

Las llamadas a la API para manejar información sobre el paciente.

Acción	URL	Método	Autenticación
Pacientes para el doctor del `JWT`	`/api/v1/patients`	**GET**	JWT in Header "Bearer"
Ver perfil del paciente `<id>`	`/api/v1/patient/<id>`	**GET**	JWT in Header "Bearer"
Crear Paciente	`/api/v1/patient`	**POST**	JWT in Header "Bearer"
Actualizar Paciente `<id>`	`/api/v1/patient/<id>`	**PUT**	JWT in Header "Bearer"

9.1.1.4 *Endpoints relacionados con las medidas*

Las llamadas a la API para manejar información sobre las medidas.

Acción	URL	Método	Autenticación
Enviar pulso para `<id>`	`/api/v1/patient/<id>/pulse`	**POST**	JWT in Header "Bearer"
Enviar peso para `<id>`	`/api/v1/patient/<id>/weight`	**POST**	JWT in Header "Bearer"
Enviar tensión para `<id>`	`/api/v1/patient/<id>/pressure`	**POST**	JWT in Header "Bearer"
Ver pulso para `<id>`	`/api/v1/patient/<id>/pulse`	**GET**	JWT in Header "Bearer"
Ver peso para `<id>`	`/api/v1/patient/<id>/weight`	**GET**	JWT in Header "Bearer"
Ver tensión para `<id>`	`/api/v1/patient/<id>/pressure`	**GET**	JWT in Header "Bearer"

9.1.1.5 Endpoints relacionados con las estadísticas

Las llamadas a la API para obtener las estadísticas, los datos de todos los pacientes, sin nombres ni información. Puede ser interesante dejar estos endpoints abiertos en el futuro.

Acción	URL	Método	Autenticación
Ver los pulsos	`/api/v1/stats/pulse`	**GET**	JWT in Header "Bearer"
Ver los pesos	`/api/v1/stats/weight`	**GET**	JWT in Header "Bearer"
Ver las tensiones	`/api/v1/stats/pressure`	**GET**	JWT in Header "Bearer"
Ver todas las Estadisticas	`/api/v1/stats`	**GET**	JWT in Header "Bearer"

9.1.2 Respuestas

Las respuestas tienen todas la misma forma:

- En caso de no error (Código HTTP `200`):

```
{
    "data": { ... },
    "result": "success"
}
```

- En caso de error:

```
{
    "data": { ... },
    "result": "failed"
}
```

Donde en **"{ ... }"** se mostrar el modelo de dato específico para cada endpoint/petición/respuesta.

9.1.3 Modelo de Datos

Aquí se muestran los distintos modelos de datos devueltos por la plataforma. Son las respuestas de todos los endpoints del apartado anterior.

Modelo de datos:

- Doctor

```
{
  "data": {
      "email": <email>,
      "lastName": <string>,
      "name": <string>
  },
  "result": "success"
}
```

- Paciente

```
{
  "data": {
      "birthday": "1994-06-04",
      "gender": "male",
      "height": 175,
      "id": 1,
      "lastName": "Fuertes",
      "name": "Javier"
  },
  "result": "success"
}
```

- Pesos


```
{
  "data": [
      {
          "timestamp": "2020-01-20 18:49:50",
          "weight": 120.6
      },
      {
          "timestamp": "2020-01-23 13:16:03",
          "weight": 120.0
      },
      ...
  ],
  "result": "success"
}
```


- Frecuencia cardiaca


```
{
  "data": [
      {
          "high": null,
          "low": null,
          "pulse": 100,
          "timestamp": "2020-01-21 18:35:12"
      },
      ...
  ],
  "result": "success"
}
```

- Presión Sanguínea


```
{
  "data": [
      {
          "high": 140,
          "low": 80,
          "pulse": 101,
          "timestamp": "2020-01-24 22:57:10"
      },
      ...
  ],
  "result": "success"
}
```


- Estadísticas de frecuencia cardiaca


```
{
  "data": [
      {
          "age": 25,
          "gender": "male",
          "height": 175,
          "pulse": 70,
          "timestamp": "2020-01-20 18:49:50"
      },
      ...
  ],
  "result": "success"
}
```

- Estadísticas de peso

```
{
  "data": [
      {
          "age": 25,
          "gender": "male",
          "height": 175,
          "timestamp": "2020-01-20 18:49:50",
          "weight": 120.6
      },
      ...
  ],
  "result": "success"
}
```

- Estadísticas de presión sanguínea

```
{
  "data": [
      {
          "age": 25,
          "gender": "male",
          "height": 175,
          "high": 120,
          "low": 90,
          "pulse": 70,
          "timestamp": "2020-01-20 18:49:50"
      },
      ...
  ],
  "result": "success"
}
```

9.2 Anexo 2: Aplicación móvil

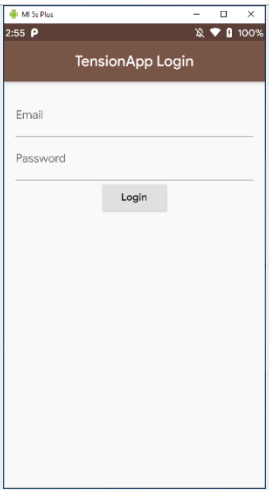
La aplicación tiene varias 3 pantallas principales, como son pacientes, estadísticas y perfil, a parte de la página de login que solo veremos la primera vez que nos logueemos.

9.2.1 Pantallas Generales

Pantalla de Acceso

Para poder acceder al sistema, el médico deberá hacer login en el sistema.

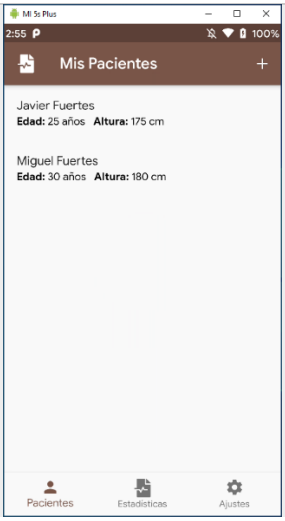
A dia de hoy la única forma de generar un usuario (un doctor) nuevo en la plataforma es mediante una llamada a la API hecha externa a la aplicación.



Pantalla de pacientes

Nada más entrar en la aplicación se nos presenta esta pantalla.

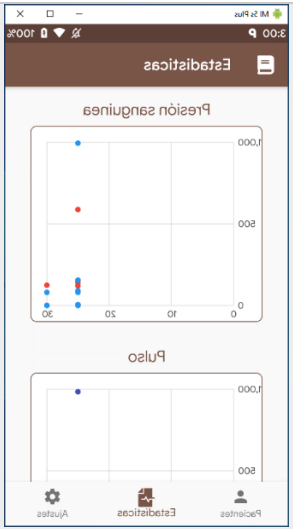
Aquí veremos nuestros pacientes y podremos, clicando sobre ellos, consultar el histórico de las medidas, así como realizar nuevas mediciones.



Pantalla de Estadísticas

La segunda pestaña que nos encontramos es la de estadística.

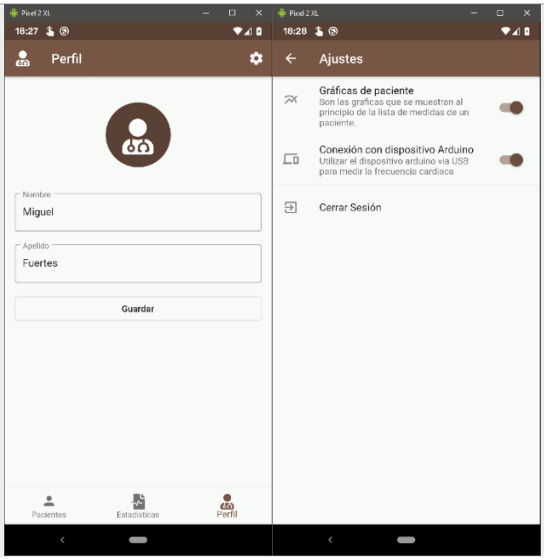
Aquí se cargarán todos los datos numéricos de todos los pacientes de todos los Médicos en forma de grafica.



Pantalla de Perfil

La tercera pestaña es la del perfil de usuario.

Aquí se encuentran todas las configuraciones de la plataforma, así como los datos privados de médico actual; nombre, apellidos, contraseña, etc.



9.2.2 Pantallas del paciente

Si nos centramos en la pantalla de pacientes y entramos en su histórico podremos introducir nuevas mediciones de peso, presión y pulso.

Pantalla de histórico de Paciente

Al seleccionar un paciente de la lista de pacientes llegaremos a esta pantalla.

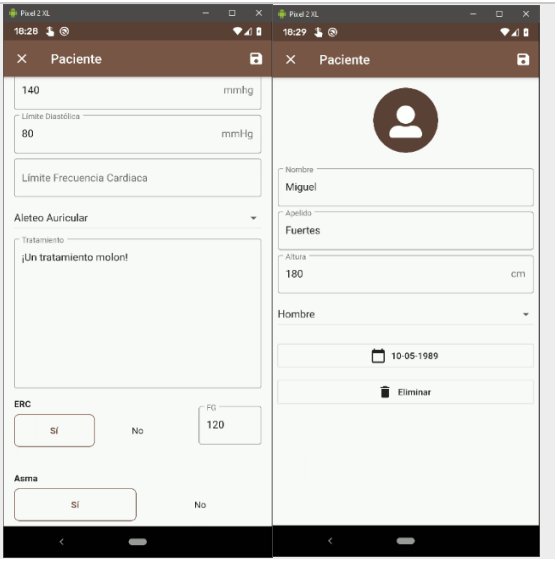
Aquí podemos ver las últimas mediciones, así como realizar nuevas.

Edición de información estática del paciente

Desde estas pantallas se puede editar la información "estática" del paciente. En la primera es la información médica, la historia clínica, los antecedentes, ciertos indicadores, los limites para la presión sanguínea y frecuencia cardiaca, etc.

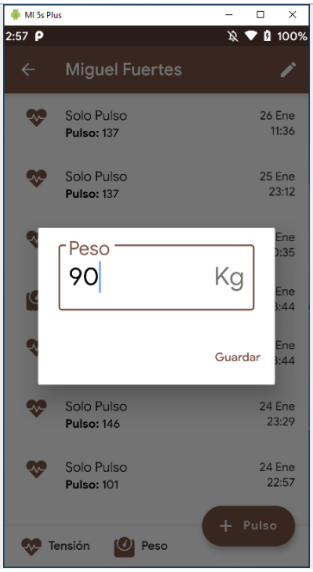
En la segunda se cambia el nombre, apellidos altura o fecha de nacimiento, así como se puede borrar el paciente.

(actualmente el botón borrar no borra los datos obtenidos, simplemente anonimiza el paciente y lo hace invisible en la lista general)



Pantalla de introducción de peso

Pulsando sobre el botón **Peso** se nos desplegara un dialogo para introducir la medida.

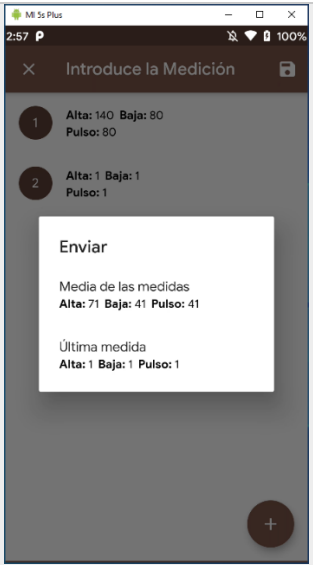


Pantalla de introducción de Presión Sanguínea

Pulsando sobre el botón *Tensión* se nos desplegara un dialogo para introducir la medida.
En este caso podemos introducir más de una toma, simplemente dándole al botón "+"

Pantalla de envío de Presión Sanguínea

Cuando tengamos todas las tomas deseadas, simplemente las enviamos dándole al botón de guardar.
Se nos abrirá un cuadro de dialogo preguntando que queremos enviar, si la media o la última.



Pantalla de Introducción de frecuencia cardiaca

Pulsando sobre el botón **Pulso** se nos desplegara una para seleccionar como queremos introducir la medida.

podrá ser manual, utilizando el flash y la Cámara del móvil, o mediante un aparato conectado al USB del móvil.

Una vez tengamos la medida, simplemente le damos a guardar.
(La opción de flash + Cámara, así como la del aparato USB solo está disponible para Android)

Introducción manual de la frecuencia cardiaca

Pulsando sobre el botón **introducir manualmente** se activará el campo de texto y podremos introducir la medida.

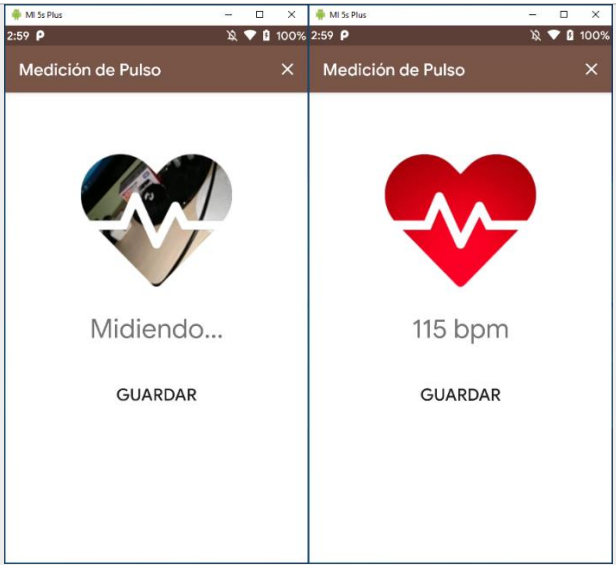
Medición mediante Cámara y Flash

Pulsando sobre el botón medir con el móvil se lanzará la siguiente pantalla y se encenderá el flash del teléfono.

(Es necesario que haya un visor de la Cámara, para que funcione, así que lo he camuflado en el corazón. Además, sirve para atinar donde poner el dedo)

Cuando veamos que tenemos "xxx bpm" tras un tiempo, ya podremos darle a guardar.

Volveremos a la pantalla anterior, y veremos que el campo se ha rellenado automáticamente.



Medición mediante el aparato USB

Conectamos el aparato por el puerto USB del teléfono mediante un cable OTG.

Una vez tengamos el aparato conectado, pulsando sobre el botón medir con el aparato se lanzará la siguiente pantalla.

Cuando veamos que tenemos "xxx bpm" tras un tiempo, ya podremos darle a guardar.

Volveremos a la pantalla anterior, y veremos que el campo se ha rellenado automáticamente.