Escuela Politécnica Superior de Linares

Grado en ingeniería telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

SISTEMA PARA LA MONITORIZACIÓN DE LA SATURACIÓN DE OXÍGENO EN SANGRE

Alumno: David Padilla Alvarado

Tutor: D. Manuel Fuentes Conde.
Depto.: Ingeniería Electrónica y Automática.

Tutor: D. José Enrique Muñoz Expósito.
Depto.: Ingeniería de Telecomunicación.

Resumen: Este trabajo fin de grado se sustenta en la implantación y diseño de un prototipo electrónico basado en plataformas de hardware libre para el registro remoto de datos biomédicos de pacientes y su posterior envío a un servidor web para que se pueda analizar y representar los datos de los mismos de una manera gráfica.

El envío y recepción de los datos siempre se lleva a cabo mediante un microcontrolador, y se enviara de forma asíncrona al servidor mediante Wifi. Es en este servidor donde se realizará un monitoreo de cada usuario adscrito a la aplicación y su posterior almacenamiento en la base de datos.

Desde la aplicación web basada en Java se podrá acceder a cada información única para cada uno de los usuarios, se podrá visualizar los datos obtenidos por el sensor de manera alfanumérica o gráfica, así como vincular de manera única cada usuario con el hardware correspondiente

Palabras clave: Monitorización, Sensor, Microcontrolador, Servidor, Aplicación.

MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E

Firmado digitalmente por MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Nombre de reconocimiento (DN): c=ES, serialNumber=27386513E, sn=MUÑOZ EXPOSITO, givenName=JOSE ENRIQUE, cn=MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Fecha: 2019.02.08 16:52:10 +01'00'

Agradecimientos: Quiero dar las gracias a toda la gente que me ha apoyado en llegar hasta aquí.

A mi familia, mi madre y mi hermana Pilar, a mis amigos que han aguantado mis subidas y bajadas en el transcurso de la carrera, a los profesores que he conocido, así como a grandes personas que he podido conocer al estudiar esta carrera como son Luis, Víctor, Emilio, Antonio y más.

Pero especialmente a una persona que junto con mi madre y mi hermana forma parte de un pilar fundamental en mi vida. Siempre me apoyó a que pudiera seguir adelante con la carrera cuando lo veía todo muy difícil, y que se sacrificó para que pudiera ser ingeniero. Soy consciente de que en parte me está viendo, y lo llevaré siempre conmigo por ser mi referencia, gracias papá, lo conseguimos.

ÍNDICE

1. Introducción. 12
2. Antecedentes. 13
2.1. Pulsioximetría 14
3. Objetivos. 16
4. Descripción de las soluciones adoptadas. 17
4.1. Estructura general del proyecto. 17
4.2. Hardware. 18
4.2.1. Plataformas de hardware libre. 18
4.2.2. Arduino. 20
4.2.3. Pantalla LCD. 28
4.2.4. Fuente de alimentación. 31
4.2.5. SensorPulse. 34
4.2.6. MAX30100. 36
4.2.7. Modo punto de acceso. 38
4.2.8. Modo estación. 44
4.3. Aplicación web. 48
4.3.1. Inicio sesión. 49
4.3.2. Administrador 53
4.3.3. Registro temporal. 54
4.3.4. Aplicación. 56
4.3.5. Evaluación. 57
4.3.6. phpMyAdmin. 59
4.3.7. XAMPP. 61
4.3.8. Eclipse. 62
4.3.9. Java. 63
4.3.10. MVC. 67
4.3.11. Spring. 70
4.3.12. JSON vs XML. 75
5. Discusión de los resultados. 76
5.1. Resultados 77
6. Conclusiones. 85
7. Líneas futuras. 87
8. Bibliografía. 91
9. Presupuestos. 94
9.1. Análisis de mercado 94

9.2. Matriz DAFO. 94
9.3. Precio. 95
10. Anexos. 96
10.1. Manuales. 96
10.1.1 Manual de usuario. 96
10.1.2 Manual de instalación y mantenimiento. 101

Índice de figuras.

Figura 1. Internet de las cosas……14

Figura 2. Logo Arduino……20

Figura 3. Arduino UNO……21

Figura 4. Arduino Leonardo……22

Figura 5. Arduino Mega……23

Figura 6. Esp8266 01……24

Figura 7. NodeMCU v3……25

Figura 8. Arduino nano……27

Figura 9. Pantalla LCD 16x2……28

Figura 10. Librería LiquidCrystal……28

Figura 11. Bus I2C……29

Figura 12. Diagrama de Flujo SCA y SCL……30

Figura 13. Adaptador LCD a I2C……31

Figura 14. Fuente de Alimentación YwRobot……32

Figura 15. Diagrama de flujo……33

Figura 16. SensorPulse……34

Figura 17. Prototipo……35

Figura 18. MAX30100……36

Figura 19. Librería MAX30100……37

Figura 20. Métodos oxígeno y saturación……37

Figura 21. Protocolo http……38

Figura 22. Librería WI-FI……38

Figura 23. Modo punto de acceso……39

Figura 24. Red NodeMCU……39

Figura 25. Script punto de acceso……40

Figura 26. Página html……41

Figura 27. Escaneo de redes……42

Figura 28. Guardar configuración……..42
Figura 29. Grabar.....……43
Figura 30. Conectando a Internet……..44
Figura 31. Conectado a internet.……45
Figura 32. Conexión via WI-FI...……46
Figura 33. Estado conectado..……46
Figura 34. Envío Identificador……47
Figura 35. Diagrama de flujo de la aplicación…….48
Figura 36. Inicio de sesión……..49
Figura 37. HomeController……50
Figura 38. LeerNickname……51
Figura 39. Formulario.....…….....52
Figura 40. Panel de Administrador…….53
Figura 41. Formulario dar de baja.....…….....53
Figura 42. Borrar usuario.....……53
Figura 43. Registro temporal……..54
Figura 44. Registro.....…….....55
Figura 45. Aplicación.……...56
Figura 46. Hash...…….57
Figura 47. Vista Evaluacion Spring……..…….58
Figura 48. Evaluacion HomeController.……58
Figura 49. Peticion AJAX.……59
Figura 50. DatosEvaluacion.....……59
Figura 51. XAMPP.....……61
Figura 52. Conexión a la base de datos……66
Figura 53. Conexión a base de datos con JdbcTemplate……..66
Figura 54. Patrón MVC.....…….....66
Figura 55. UsuarioDAOJdbc...……67

Figura 56. InterfazDao..…….67
Figura 57. Etiqueta EL..…….....68
Figura 58. HomeController..……..70
Figura 59. Ejemplo DI..…….....71
Figura 60. Arquitectura de Spring……..72
Figura 61. Pom.xml..……...…73
Figura 62. Servlet-context.xml……73
Figura 63. Recogida datos en Spring.…….…..74
Figura 64. JSON vs XML..…….……..75
Figura 65. Subconsulta SQL..……75
Figura 66. Flujo final..……...76
Figura 67. Pulsioxímetro real…….…..77
Figura 68. Max30102.…….…87
Figura 69. Sensor latidos del corazón.……88
Figura 70. Amazon Web Services.…….……90
Figura 71. Servicios REST..……..……90
Figura 72. Administrador…… ……...97
Figura 73. Principal……..97
Figura 74. Registrarse..……..97
Figura 75. Vista central..……98
Figura 76. Vista Evaluacion..……...……..98
Figura 77. Contacto.……....99
Figura 78. Quienes somos…….99
Figura 79. Devoluciones..……..……100
Figura 80. Modifica tus datos……100
Figura 81. Opciones avanzadas..……...……101
Figura 82. Propiedades del sistema……..……..102
Figura 83. Path……102
Figura 84. Instalando XAMPP……102

Figura 85. IDE Eclipse……103

Figura 86. Jerarquización del proyecto……103

Figura 87. Servidor funcionando……..104

Índice de tablas.

Tabla 1. Resultados……...78-83

Tabla 2. Porcentaje de error Usuario 1……..…..84

Tabla 3. Porcentaje de error Usuario 2……84

Tabla 4. Porcentaje de error Usuario 3……..……84

Tabla 5. Comparacion pulsioximetros……...……85

Tabla 6. Presupuesto del Proyecto……...…….95

Tabla 7. Precio total……..…..95

1. Introducción.

A lo largo de la historia de la Humanidad, se han ido produciendo grandes avances en el ámbito de la ciencia para satisfacer las necesidades de las personas. Más concretamente desde el punto de vista médico, la ingeniería ha sustentado un papel primordial en el avance de muchos aspectos, como pueden ser el desarrollo de técnicas preventivas, aplicaciones o prototipos que puedan hacer frente a distintos problemas que han acontecido a las personas desde siempre.

Cabe destacar también el papel que tiene a día de hoy la electrónica, ya que permite el desarrollo de distintos dispositivos que permiten que los especialistas puedan obtener datos de distintos pacientes mediante técnicas no invasivas.

En la última década, la fabricación en masa de distintos componentes electrónicos ha dado lugar a que, en el ámbito de la telemedicina, se pueda hacer uso de distintos elementos para poder desarrollar pequeños dispositivos que permitan obtener datos biomédicos como es el caso de los pulsioximetros. Incluso, se estima que en los años venideros siga habiendo un aumento exponencial del desarrollo de este tipo de dispositivos lo que influirá positivamente en la recolección de nuevas técnicas de desarrollo para poder paliar cualquier tipo de anomalía en el ámbito de la salud.

Concretamente, en este trabajo final de grado se pretender dar énfasis en la necesidad de poder desarrollar ciertos tipos de dispositivos electrónicos que puedan hacer más útil la vida de las personas, concretamente aquellas que posean cualquier tipo de problema relacionada con la saturación de oxígeno en sangre y la frecuencia cardiaca.

Focalizaremos el contenido de este trabajo final de grado en conseguir un prototipo basado en microcontroladores y obteniendo parámetros biomédicos por medio de un sensor, poder hacerle ver a un determinado paciente cuáles son sus diferentes parámetros por medio de una aplicación web.

Debido a la vinculación que se obtendrá con cada dispositivo hardware y cada paciente, es conveniente recalcar el hecho de que se puedan introducir distintos sensores que no limiten al paciente, sino que puedan visualizar distintos valores biomédicos como si de un desarrollo de telemedicina global se tratara.

El uso de una aplicación web permite consultar los valores medidos en cualquiera ubicación, siempre que se disponga de un ordenador o móvil con conexión a Internet.

Por otra parte, cualquier modificación o mejora de la aplicación será transparente al usuario, sin tener que instalar nuevas versiones en un dispositivo local.

Es por esto que con este trabajo se busca el desarrollo de un servicio que pueda ser utilizado por la mayoría de la población y que funcione como un ¨sistema de alarma precoz¨ para que toda aquella persona en situación de riesgo pueda detectar, un problema a tiempo.

2. Antecedentes.

En los últimos veinte años en España han fallecido aproximadamente unas cuatrocientas mil personas debido al cáncer de pulmón. Un dato, cuando menos alarmante ya que perecieron más personas por esta enfermedad, que la suma de las fallecidas por otras enfermedades como el cáncer de mama, colon y próstata juntos [1].

Uno de los factores que más inciden en el desarrollo de este tipo de patologías es el tabaquismo, ya que el 84,1% de las personas que fallecieron eran fumadores o lo fueron [1].

Por esto, a día de hoy, es un problema al que la sociedad debe poner solución. Cada año se dedican más recursos para paliar esta enfermad y, paradójicamente se producen más diagnósticos.

A nivel mundial, anualmente, se diagnostican 1.6 millones de nuevos casos de cáncer de pulmón de los que 1.3 millones derivan en fallecimiento [2].

En países donde la accesibilidad a pruebas para alcanzar este tipo de diagnósticos se hace más complicada por métodos propios, debido a la escasa información, a los ínfimos recursos económicos que se dedican a la salud y la sanidad en países en vías de desarrollo, o territorios sumidos en conflictos bélicos o sociales, este problema se agrava de forma considerable. Es aquí donde entra el papel del IOT (Internet de las cosas) ya que está siendo una gran evolución y se espera que en los próximos años haya más de veinte millones de dispositivos conectados a la red.

La interconexión de estos dispositivos producirá un mejor uso de los recursos y una optimización de los servicios que se deseen ofrecer de cara a los usuarios. Con respecto al campo de la salud, esta tecnología tendrá un impacto significativo ya que se estimará en costes ciertos dispositivos que a día de hoy son indispensables y que ciertos prototipos electrónicos podrán suplir perfeccionando las mismas características e incluso aumentándolas, pero a un coste inferior.

Esta nueva tecnología de dispositivos electrónicos generara cantidades enormes de información que serán almacenadas en servidores y donde se podrá hacer uso de ellos de forma remota para su posterior visualización, análisis o almacenamiento.

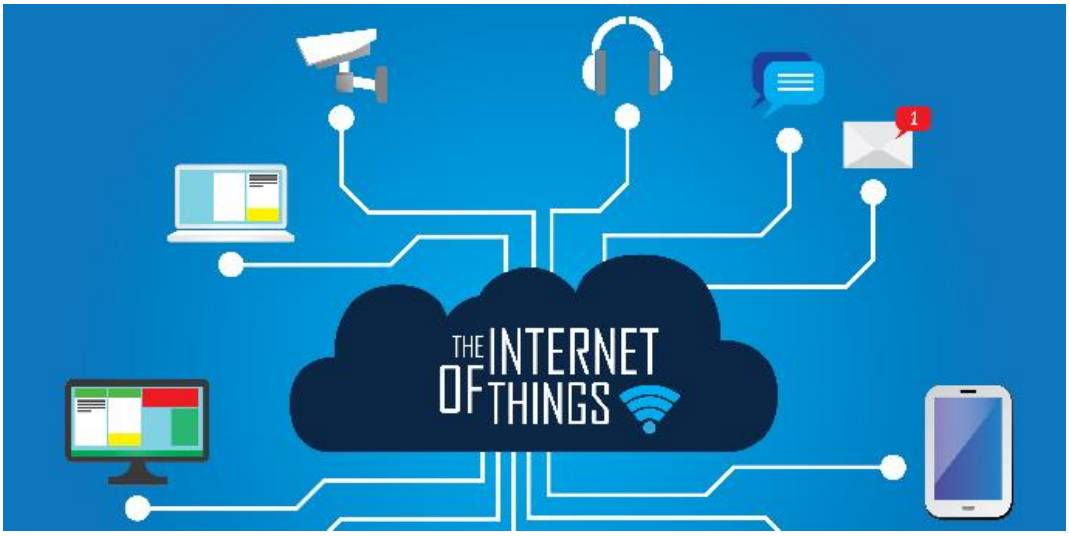

Figura 1. Internet de las cosas.

El proyecto estará destinado a determinados grupos de riesgo como pueden ser personas adultas fumadoras, personas de una edad avanzada, con algún tipo de patología relacionada con niveles de oxígeno en sangre o aquellas propensas a sufrir distintos problemas como el infarto de miocardio, asma, ansiedad, síndrome de fatiga crónica, etc…

A continuación, se dispondrá a introducir los conceptos y definiciones necesarios para entender los parámetros necesarios para entender cómo se puede llegar a detectar las patologías anteriormente citadas.

2.1. Pulsioximetría.

Para comprender la importancia de la realización de este proyecto, hay que tener en cuenta el concepto de pulsioximetria. Es la medición no invasiva del oxígeno transportado por la hemoglobina en el interior de los vasos sanguíneos [3]. Cada una de las pruebas correspondientes se realiza con un dispositivo llamado pulsioximetro.

Éste, emite luz con dos longitudes de onda de 660 nanómetros (roja) y 940 nanómetros (infrarroja). La mayor parte de la luz es absorbida por el tejido conectivo, piel, hueso y sangre venosa en una cantidad constante, produciéndose un pequeño incremento de esta absorción en la sangre arterial con cada latido [3].

Uno de los datos a tener en cuenta es que las afecciones que se observan con más frecuencia en los consultorios médicos de atención primaria son aquellas relacionadas con enfermedades respiratorias, como puede ser el caso de la enfermedad pulmonar obstructiva crónica (EPOC), el asma, alergias…

Estos profesionales médicos necesitan de herramientas que puedan determinar, monitorear y gestionar cualquier tipo de anomalía que se presente en algún paciente.

Dado que la pulsioximetria trabaja con técnicas no invasivas, ha sido implementado correctamente por parte de los profesionales médicos. Si bien es cierto que la tecnología ha ido aumentando cada año de forma exponencial no es menos cierto que los facultativos han ido obteniendo distintos dispositivos tecnológicos que mejorasen la vida del paciente.

Es por esto que entra el juego el papel de los pulsioximetros, ya que en países donde es más difícil el acceso a la sanidad, este tipo de aparatos se limitan a un cierto número de personas bajo supervisión médica y lo que esto acarrea, el poder trasladarse a un sitio médico para observar, monitorear y gestionar los distintos parámetros biomédicos y consecuentemente, proporcionar al paciente un diagnóstico correcto.

Las alteraciones del haz de luz se recogen por el receptor del pulsioxímetro, ahí se interpretan, y dan un valor numérico que representa el porcentaje de oxígeno que hay en la sangre. La prueba dura unos instantes, no representa ninguna molestia para el enfermo, y puede ayudar a iniciar un tratamiento o a realizar otras pruebas rápidamente.

Al llegar a la consulta o al servicio de urgencias el facultativo realizará unas preguntas generales sobre tu estado de salud (enfermedades importantes, factores de riesgo, estilo de vida, lugar de trabajo…), y sobre todo insistirá en los síntomas que te han llevado a consultar. El síntoma que se relaciona con la insuficiencia respiratoria es la disnea, fatiga o falta de aire [4].

Después suele llevar a cabo una exploración física general, y valorará la realización de una pulsioximetría si cree que puede ser útil para el diagnóstico y tratamiento. Al ser un test sencillo se suele realizar en todas las personas de forma directa [4].

En ese mismo momento, un médico o una enfermera procederá a colocar el pulsioxímetro. Se suele colocar en los dedos de la mano; cualquiera es válido. Pero hay situaciones en las que la sangre no llega bien a la punta de los dedos. Algunas de ellas son fisiológicas, como una exposición al frío, y otras son situaciones patológicas, como sepsis, enfermedad de Raynaud, etcétera. En esos casos se puede intentar colocar el pulsioxímetro en otras localizaciones, como por ejemplo el lóbulo de la oreja.

Los resultados de la pulsioximetría facilitan un porcentaje que representa la cantidad de oxígeno que hay en la sangre unido a la hemoglobina; es lo que se conoce como saturación de oxígeno (SatO2). Este dato también se puede obtener con una gasometría, pero la pulsioximetría permite obtenerlo directamente, sin estimaciones.

La saturación de oxígeno no es el método instaurado para diagnosticar la insuficiencia respiratoria, solo la presión parcial de oxígeno (pO2) es el dato válido para ello, y sólo se puede obtener mediante una gasometría arterial. Sin embargo, la saturación de oxígeno tiene una correlación muy estrecha.

Se consideran normales unos niveles de saturación de oxígeno entre el 98 y 100%. Con la edad es habitual que el paso de oxígeno no sea óptimo, por lo que también se puede aceptar como normal hasta el 95%. En los enfermos pulmonares crónicos hasta el 90% puede considerarse normal [5]. El motivo de considerar el 95% como el límite de la normalidad es que se corresponde con una presión parcial de oxígeno de 60 mmHg, límite para diagnosticar una insuficiencia respiratoria.

Si una pulsioximetría es normal el médico atenderá otros problemas, pero si es menor de 95% será su prioridad. Primero pedirá una gasometría arterial, para poder confirmar el dato y observar exactamente el nivel de oxígeno en sangre y otras alteraciones metabólicas. Luego pondrá oxígeno a través de unas gafas nasales o una mascarilla, según la gravedad de la situación.

3. Objetivos.

Los objetivos de este Trabajo de Fin de Grado son:

- Desarrollo de un sistema hardware y un conjunto de aplicaciones de servidor (programadas en Java) destinadas a la monitorización de la saturación de oxígeno en sangre.
- Captación de los parámetros médicos mediante la recepción de un sensor
- Los parámetros médicos se almacenarán en un servidor, para extraer conocimiento para una posterior visualización de los mismos por parte del paciente.

Como conclusión, comentar que el prototipo a desarrollar será portátil/portable y con autonomía energética.

4. Descripción de las soluciones adoptadas.

Para la elaboración de este prototipo electrónico el cual busca la obtención de parámetros biomédicos se ha realizado una serie de pasos para poder llevarlo a cabo.

A continuación, se presenta distintos procesos por los cuales se han podido anexionar y culminar con el proyecto electrónico que nos presenta.

4.1. Estructura general del proyecto.

En una primera fase se realiza un análisis del problema, focalizando aquellos puntos en los que se requiera un mayor esfuerzo y siguiendo unas premisas dadas.

La segunda fase es la adecuada para la de construcción del hardware necesario, así como la captación y recepción por parte del usuario de los parámetros provenientes del sensor.

La tercera fase es la del desarrollo de una aplicación donde se podrán visualizar los datos provenientes del sensor por medio de una conexión wifi y crear una interacción con estos.

En la fase de líneas futuras se pretende conseguir un paso para el avance de desarrollos tecnológicos a posteriori, sustentados en este proyecto y que conlleven un enriquecimiento y mejora del mismo.

El apartado de bibliografía recoge todas las fuentes consultadas para el desarrollo histórico-científico de este proyecto.

Para finalizar, en el apartado de anexo se recoge el material extra que se ha utilizado para formalizar este proyecto, códigos de programación, hoja de características…

4.2. Hardware.

4.2.1. Plataformas de hardware libre.

Inicialmente, antes de poner en tesitura cada uno de los componentes utilizados para la realización de este proyecto, y concretamente en el uso del hardware, se identificará cada una de las plataformas de hardware libre más utilizadas en el mercado y la justificación de la elegida.

- Arduino: Es una plataforma de hardware libre, basado en un microcontrolador ATmega168, ATmega328 y el ATmega1280, con varios puertos digitales y analógicos de entrada y salida, a través de los cuales podemos leer información del medio ambiente utilizando diversos sensores (de temperatura, humedad, luminosidad, magnestismo, corriente, coordenadas de un gps, etc.), mostrar información o notificaciones o hasta interactuar con el usuario [6]
- Netduino: Esta es la plataforma de Microsoft para programación de microcontroladores, la cual fue desarrollada por Secret Lab LLC, basándose en la idea de Arduino. Tomaron un procesador de ATmel (el mismo fabricante que los procesadores de Arduino) de 32 bits y 48 MHz y desarrollaron una placa compatible con Arduino. Esta idea permite acercar a los desarrolladores al mundo de la programación de microcontroladores utilizando una placa compatible con todos los sensores y dispositivos que ya existen para Arduino [6].
- Gadgeteer: Esta plataforma, fue lanzada en 2011, está basada en la idea de poder conectar de forma muy sencilla distintos dispositivos a nuestra placa base, sin necesidad de pensar en resistencias, consumo de los componentes ni nada por estilo. Al momento esta plataforma posee muchos dispositivos en el mercado: display, cámaras, interfaces de red, lectores/grabadoras de tarjetas, etc. Se programa con Visual Studio 2010 en cualquier versión inclusive la Express [6]
- Raspberry: Esta palca fue creada por la fundación “Fundación Raspberry Pi” del Reino Unido para estimular la enseñanza de las ciencias de la computación en las escuelas. Su salida a la venta fue en principios de 2012 con un total de 10.000 placas fabricadas en China y Taiwan.

Esta placa se diferencia en mucho a las anteriores, para empezar, ya viene con puertos HDMI, RCA (útil para conexiones a televisiones.) y USB (al que se le puede conectar un ratón y un teclado), GPU compatible con OpenGL es 2.0 y OpenVG que puede decodificar video con calidad blue-ray. Pero lo que hace la mayor diferencia viene con Linux, más específicamente con Raspbian, una versión derivada de Debian, aunque le podemos instalar la distribución que queramos, inclusive Android [6].

Como conclusión destacar que existen otras plataformas como Freaduino (basado y compatible con el Seeduino (también compatible con Arduino), AMICUS 18 (que es muy parecido a Arduino pero con un procesador PIC 18F25K20 de 64MHz y el compilador Proton BASIC), Stellaris LaunchPad (de Texas Instruments), ODROID-U2 (que salió hace unas semanas y posee un procesador ARM Cortex-A9 con cuatro núcleos de 1.7GHz con acelerador gráfico y mide la mitad de una tarjeta de crédito) y muchos otros, pero mi intensión en este artículo era que conocieran las placas más usadas y accesibles del momento.

La plataforma que se ha elegido finalmente has sido Arduino,
Alguna de las justificaciones que se han llevado a elegir esta plataforma son las siguientes:

- Es la plataforma más madura de las presentadas aquí.
- Es la que posee más versiones a la hora de elegir una placa, desde la LilyPad, que posee un procesador de 8 MHz y una placa flexible para poder coserla a cualquier prenda de vestir, hasta la Due con 84Mhz.
- El código generado no es “manejado”, por lo cual, se ejecuta en tiempo real y se tiene acceso directo al hardware, lo cual puede ser bueno por la performance o malo si no se programa adecuadamente.
- El entorno de programación es multiplataforma.
- Posee muchísimos módulos y sensores, y hasta se pueden conectar dispositivos que no sean específicamente para Arduino.
- No se puede debugear fácilmente el código (básicamente el IDE que trae no puede debugear, pero se puede usar otro IDE).
- Al ser una plataforma con tantos años, posee muchísimas librerías disponibles.
- Los shields a veces traen conflictos entre sí.

4.2.2. Arduino.

La primera solución que se contempló fue la utilización de Arduino, ya que es un hardware libre, ¿Qué quiere decir esto? Que sus esquemas, especificaciones y planos están disponibles para el público de forma gratuita. Esto muestra una clara ventaja con otro hardware ya que tiene una gran accesibilidad y bajo coste.

Figura 2. Logo Arduino.

Se presenta como una placa con diferentes entradas y salidas analógicas y digitales, su corazón es el microcontrolador definido, así como un circuito integrado programable, capaz de ejecutar las ordenes grabadas en su memoria, éste, está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Concretamente el microcontrolador que tiene Arduino se llama Atmega8, que es un chip sencillo y de bajo coste que permite el desarrollo de múltiples diseños. Al ser open source tanto su diseño como su distribución es libre, puede utilizarse para desarrollar cualquier tipo de proyectos sin tener que adquirir ningún tipo de licencia.

Esto es una ventaja bastante considerable a la hora de realizar este proyecto ya que existe esa libertad para poder desarrollar un prototipo con un acceso de forma pública, incluso se puede retomar el desarrollo del mismo para una posible innovación en alguna línea futura.

En específico, el Arduino UNO posee unas ciertas características que son:

- Microcontrolador: ATmega328
- Voltaje de funcionamiento: 5 V
- Pines I/O digitales: 14 (de los cuales 6 proveen salida PWM)
- Pines de entradas análogas: 6

- Corriente DC por cada pin I/O: 40 mA
- Corriente DC en el pin de 3.3 V: 50 mA
- Memoria Flash: 32 KB (ATmega328) de los cuales 0.5 KB son utilizados por el bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocidad de reloj: 16 MHz

Figura 3. Arduino UNO.

Debido a que Arduino es una de las primeras plataformas microcontroladoras open source en el mundo cabe esperar que se desarrollan varias versiones de ésta. A continuación, se observa algunas de las características de los diferentes modelos oficiales de Arduino,

Arduino Leonardo es un modelo oficial de Arduino que presenta las siguientes características:

Microcontrolador**:** ATmega32u4.

- Voltaje de funcionamiento: 5 V.
- Pines I/O digitales: 20.
- Canales PWM: 7.

- Pines de entradas análogas**:** 12.
- Corriente DC por cada pin I/O: 40 mA.
- Corriente DC en el pin de 3.3 V: 50 mA.
- Memoria Flash: 32 KB (ATmega32u4) de los cuales 4 KB son utilizados por el bootloader.
- SRAM: 2 KB (ATmega32u4).
- EEPROM: 1 KB (ATmega32u4).
- Velocidad de reloj: 16 MHz.

Figura 4. Arduino Leonardo.

Arduino Due presenta las siguientes características:

- Microcontrolador: AT91SAM3X8E.
- Voltaje de funcionamiento: 3.3 V.
- Pines I/O digitales: 54 (de los cuales 12 proveen salida PWM).
- Pines de entradas análogas**:** 12.
- Corriente DC total en todos los pines I/O: 130 mA.
- Corriente DC en el pin de 5 V**:** 800 mA.
- Corriente DC en el pin de 3.3 V: 800 mA.
- Memoria Flash: 512 KB disponibles para las aplicaciones de usuario.
- SRAM**:** 96 KB (dos bancos: 64KB Y 32 KB).

- Velocidad de reloj: 84 MHz.

Otro de los modelos más famosos que existen es Arduino Mega que presenta las características siguientes:

- Microcontrolador: ATmega2560.
- Voltaje de funcionamiento: 5 V.
- Pines I/O digitales: 54 (de los cuales 15 proveen salida PWM).
- Pines de entradas análogas: 16.
- Corriente DC por cada pin I/O: 40 mA.
- Corriente DC en el pin de 3.3 V: 50 mA.
- Memoria Flash: 256 KB de los cuales 8 KB son utilizados por el bootloader.
- SRAM: 8 KB (ATmega328).
- EEPROM: 4 KB (ATmega328).
- Velocidad del reloj: 16 MHz.

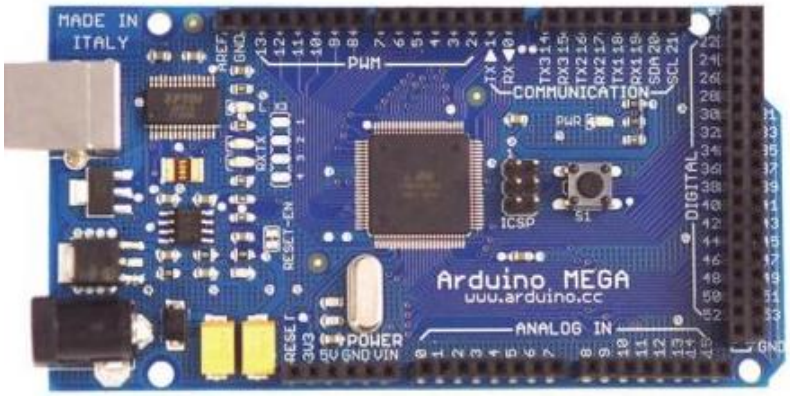

Figura 5. Arduino Mega.

Se han presentado algunos de los diferentes modelos oficiales de Arduino existentes actualmente en el mercado, pero como se quiere realizar una comunicación con un servidor web y puesto que Arduino no dispone de un módulo Wifi implementado en la placa habría que comprar un módulo Wifi externo para poder realizar la comunicación.

Se pensó en utilizar el componente ESP-8266 01.

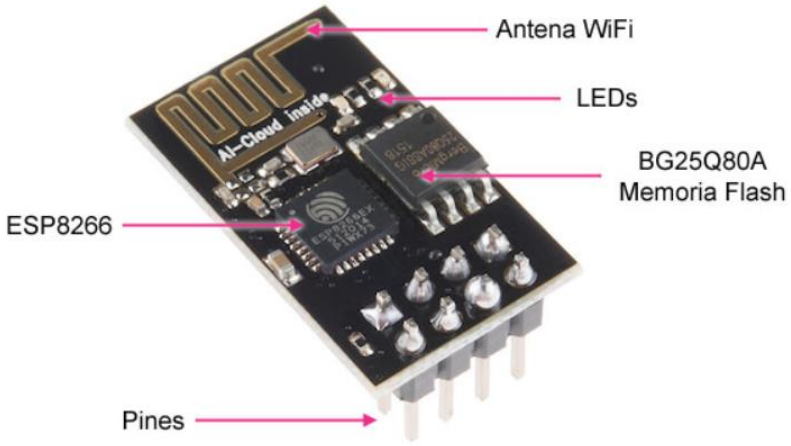

Figura 6. Esp8266 01.

Dispone de una antena wifi, leds para la recepción y envío de parámetros, una memoria flash interna donde se puede subir los scripts provenientes del IDE de Arduino y ocho pines de conexión en los que se encuentra alimentación, tierra (GND), transmisión (TX) y recepción (RX) entre otros.

El inconveniente principal que tiene este módulo wifi es que, a pesar de su bajo precio, presenta unas características que son obsoletas comparadas con otros módulos como el esp8266 12, o el propio NodeMCU v3. También cabe considerar la idea de que produce muchos fallos a la hora de conectar a una red wifi lo que conlleva retardos en el programa, redirección de subida de script y un consumo de energía excesivo.

NodeMCU es el módulo más característico que emplea el SoC ESP8266. Es un nombre que recoge tanto un firmware Open Source como a una placa de desarrollo.
Las características en las que se sustenta son: Código abierto, interactivo, programable, bajo coste, simple, inteligente, habilitado para Wifi [7].

Es una de las mejores opciones para el desarrollo de aplicaciones de IoT.

Viene integrado con un módulo WI-Fi MCU ESP8266 que es muy útil para el desarrollo de prototipos.

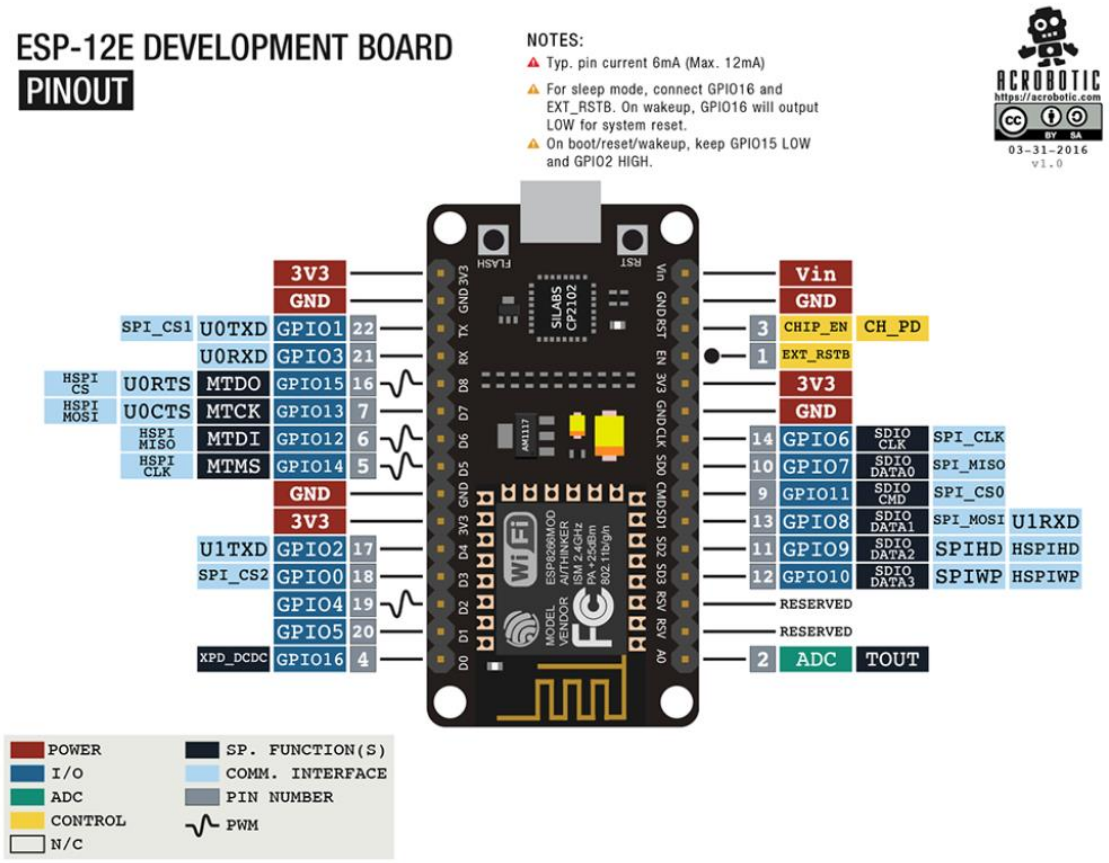

Figura 7. NodeMCU v3.

Especificaciones Técnicas:

- Voltaje de Alimentación (USB): 5V DC.
- Voltaje de Entradas/Salidas: 3.3V DC.
- SoC: ESP8266 (Módulo ESP-12).
- CPU: Tensilica Xtensa LX3 (32 bit).
- Frecuencia de Reloj: 80MHz/160MHz.
- Instrucción RAM: 32KB.
- Data RAM: 96KB.
- Memoria Flash Externa: 4MB.
- Pines Digitales GPIO: 17 (pueden configurarse como PWM a 3.3V).
- Pin Analógico ADC: 1 (0-1V).
- UART: 2.
- Chip USB-Serial: CP2102.
- Certificación FCC.

- Antena en PCB.
- 802.11 b/g/n.
- Wi-Fi Direct (P2P), soft-AP.
- Stack de Protocolo TCP/IP integrado.
- PLLs, reguladores, DCXO y manejo de poder integrados.
- Potencia de salida de +19.5dBm en modo 802.11b.
- Corriente de fuga menor a 10uA.
- STBC, 1×1 MIMO, 2×1 MIMO.
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval.
- Wake up and transmit packets in < 2ms.
- Consumo de potencia Standby < 1.0mW (DTIM3).

Finalmente se optó por trabajar con este módulo ya que ofrecía una conexión más robusta para el estándar 802.11 que el anterior chip, el ESP8266-01.

Es necesario destacar el hecho de que a redes Intranet (802.11.x) presenta problemas de conexión ya que soporta estándares (802.11 b/g/n).

Es el encargado de interactuar con la pantalla LCD que verá el usuario y el sensor, para que finalmente envíe esos parámetros vía Wi-Fi al servidor.

En el siguiente diagrama de flujo se explica el trabajo principal en el que se encuentra el NodeMCU.

Para la obtención de valores provenientes del sensor (MAX30100) y el posterior envío de estos al microcontrolador que gestionará esa información y hará el posterior envío mediante http, se ha utilizado el Arduino Nano.

El Arduino Nano es un pequeño, pero poderoso chip basado en el microcontrolador ATmega328. Posee las mismas funcionalidades que un Arduino Uno, solo que en tamaño reducido. Esto es una ventaja a tener en cuenta para la implantación en este proyecto ya que corresponderá con un consumo menor, y con respecto a un encapsulado es más óptimo que el Arduino Uno que tiene un tamaño mayor.

Figura 8. Arduino nano.

Para que un prototipo electrónico se considere optimo es necesario que ofrezca una simplicidad a la vez que una robustez considerable.

Por este hecho, y debido al desconocimiento aparente que debe de presentar un usuario es necesario el uso de una pantalla lcd para que el usuario observe cada uno de los pasos que debe realizar sin preocuparse en nada más que poner el dedo en el sensor de manera correcta y esperar que se mande esa información al servidor.

Configuración de los pines:

- Pin 1 – Vss: GND o tierra.
- Pin 2 – Vdd: Alimentacion Vcc o +5V.
  (Algunos pueden alimentarse a 3 Vcc).
- Pin 3 – V0: Control del contraste del display, conectamos este pin al terminal variable de un potenciómetro conectado a Vcc y Masa en sus terminales extremos.
- Pin 4 – RS: Selección de Registro.
  0 lógico: Registro de comandos (escritura).
  1 lógico: Registro de datos (escritura,lectura).
- Pin 5 – R/W:
  0 lógico: Escritura del LCD.
  1 lógico: Lectura del LCD.
- Pin 6 – Enable: El famoso Enable de casi todos los componentes de la electrónica digital. Un 1 lógico señala el inicio de escritura o lectura del LCD, un 0 lógico, desactiva todas las funciones.

- Pin 7-10 – D0/D3: Pines correspondientes al bus de datos.
  D0 corresponde al bit menos significativo.
  Estos pines no se utilizan si realizamos operaciones sobre el LCD de4 bits.
- Pin 11-14 – D4/D7: Pines correspondientes al bus de datos.
  D7 corresponde al bit más significativo y puede utilizarse como “Busy Flag”, si leemos sobre este pin, un 1 lógico nos indicará que el LCD se encuentra ocupado, no permitiéndonos realizar ninguna operación hasta que se deshabilite.
- Pin 15 – Ánodo de la retroiluminación: R + 5V.
- Pin 16 – Cátodo de la retroiluminación: GND.

4.2.3. Pantalla LCD.

Figura 9. Pantalla LCD 16x2.

Para utilizar la pantalla lcd, es necesario incluir en Arduino la librería: LiquidCrystal_I2C

```
#include <LiquidCrystal_I2C.h>
```

Figura 10. Librería LiquidCrystal.

En esta librería se utilizan métodos, como, por ejemplo:

- LiquidCrystal_I2C () Constructor de la clase, configura el hardware.
- init () – Prepara el LCD para su uso.
- clear () – Borra todos los caracteres de la pantalla LCD.
- setCursor (col, row) – Permite mover el cursor a la posición indicada en sus parámetros.
- print () – Imprime una variable o literal en la pantalla
- scrollDisplayLeft () y scrollDisplayRight() – Recorre el contenido de la pantalla a la izquierda o a la derecha
- backlight () y noBacklight () – Métodos para encender / apagar la iluminación de fondo
- createChar (num, data) – Crear un carácter definido por el usuario en la memoria del controlador de pantalla.

Para que pueda darse de forma óptima la intercomunicación entre la pantalla lcd, el microcontrolador y el sensor de forma síncrona es necesario nombrar el uso del bus I2C.

Los ingenieros de “Philips” vieron la necesidad de la simplificación y normalización de las líneas de datos que viajan entre los diversos circuitos integrados en sus productos. Esto redujo un gran número de cables a sólo dos (SDA = datos, y SCL = reloj).

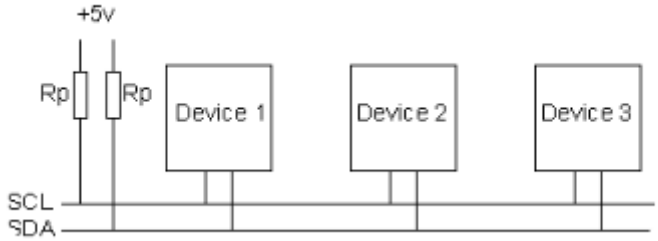

Figura 11. Bus I2C.

Se observa que tiene cuatro pines, se realiza la conexión de la siguiente manera:

- Pin GND del adaptador I2C se conecta a GND de la protoboard.
- Pin VCC se conecta a los cinco voltios que darán alimentación al prototipo.
- Pin SDA se conecta al pin D2 del NodeMCU v3.
- Pin SCL se conecta al pin D1 del NodeMCU v3.

Las características principales del bus I2C son:

- Utilización únicamente de dos líneas, la de datos (SDA) y la de reloj (SCL).
- Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software.
- Permite la conexión de varios Maestros porque incluye un detector de colisiones.
- El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.
- Los datos y direcciones se transmiten con palabas de ocho bits [8].

El Maestro es el dispositivo que inicia la transferencia en el bus y generar la señal de Reloj, el NodeMCU actúa como Maestro y la pantalla LCD como esclavo.

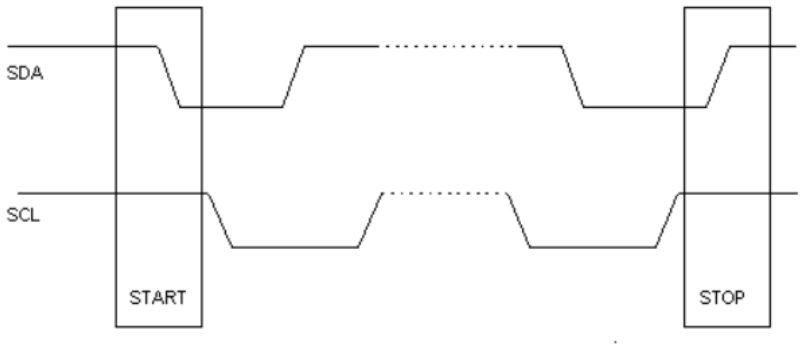

Figura 12. Diagrama de Flujo SCA y SCL.

Métodos para Comienzo y Final de informacion:

En primer lugar, antes de que se produzca una interaccion entre el maestro y el esclavo, el maestro debe transmitir el comienzo de la comunicación (condición de comienzo): SDA cae a cero mientras SCL permanece en nivel alto. A partir de este momento comienza la transferencia de datos. Una vez finalizada la comunicación se debe informar de esta situación (condición de Final). La línea SDA pasa a nivel alto mientras SCL permanece en estado alto [8].

Transferencia de datos:

El Maestro genera la condición de Comienzo. Cada palabra puesta en el bus SDA debe tener 8 bits, la primera palabra transferida contiene la dirección del Esclavo seleccionado. Luego el Master lee el estado de la línea SDA, si vale 0 (impuesto por el esclavo), el proceso de transferencia continúa. Si vale 1, indica que el circuito direccionado no valida la comunicación, entonces, el Maestro genera un bit de stop para liberar el bus I2C.

Figura 13. Adaptador LCD a I2C.

Posee un potenciómetro para calibrar la luminosidad de la pantalla LCD.

Si hablamos de la alimentación que presentara nuestro dispositivo, es conveniente recalcar el hecho de que deberíamos utilizar una fuente de alimentación externa.

Como este prototipo que se ha desarrollado va a ser portátil, es necesario transformar los voltios de alimentación de entrada en unos voltios de salida óptimos para el circuito electrónico.

4.2.4. Fuente de alimentación.

Debido a que tanto el Arduino nano se alimenta a cinco voltios y el sensor a tres voltios, se ha optado por utilizar esta función de alimentación externa ya que presenta esas dos salidas.

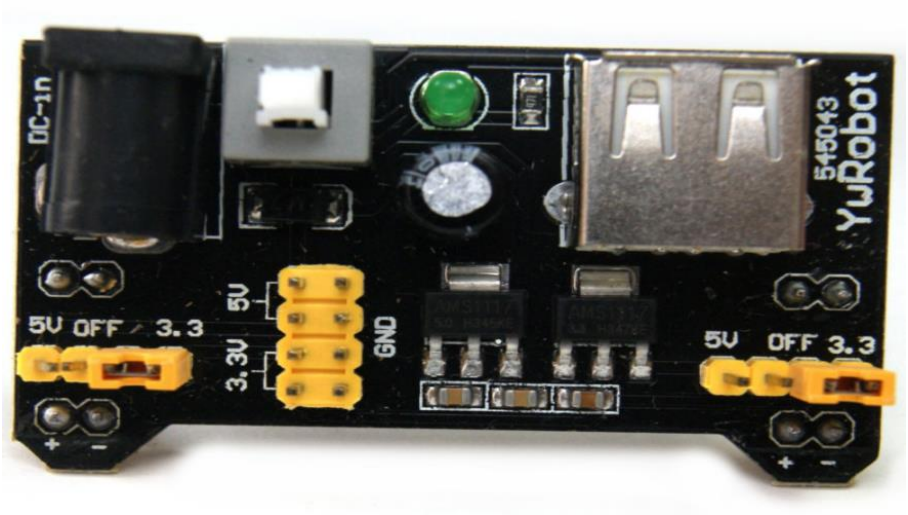

Figura 14. Fuente de Alimentación YwRobot.

Especificaciones:

- Compatibilidad: Protoboard de 400 y 830 puntos
- Voltaje de entrada: 6.5 V a 12 V.
- Voltaje de salida 1: 3.3 V o 5 V (seleccionable).
- Voltaje de salida 2: 3.3 V o 5 V (seleccionable).
- Salida de voltaje USB: 5 V.
- Entrada de voltaje USB: 5 V.
- Corriente máxima de salida: 700 mA.
- Conectores entrada:
- Plug invertido 5.5 mm x 2.1 mm.
- Jack USB-A.
- LED indicador de encendido.
- Botón de encendido / apagado.
- Dimensiones: 53 mm X 23 mm X 33 mm.
- Marca: YwRobot.

Este módulo suministra directamente sobre la protoboard alimentación entre 5 y 3.3 voltios.

La corriente máxima que ofrece se encuentra en torno a los setecientos miliamperios, debido al consumo total que proporcionara nuestro circuito, este valor de amperaje es muy optimo, ya que, en ningún momento, sobrepasaremos esta línea.

Puede contener alimentación de entrada vía USB o Jack con valores comprendidos entre los seis voltios y medio y doce voltios.

Una vez que se ha tenido en cuenta que microcontroladores van a formar parte del proyecto, se realiza un diagrama de flujo para que se observe de manera general, la síntesis del proyecto con respecto al hardware:

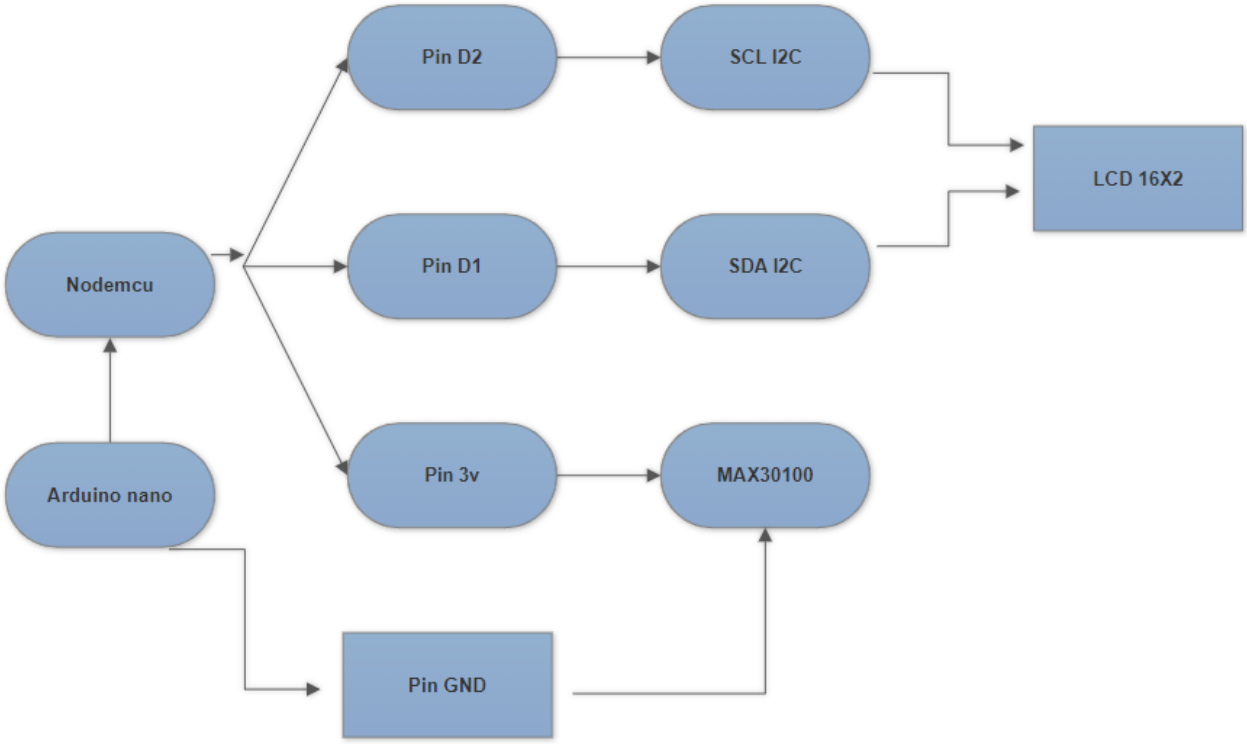

Figura 15. Diagrama de flujo.

A continuacion se presenta disintos sensores que se han utilizado, notando las caracterisitcas de cada uno y decidiendose finalmente por el sensor: MAX30100.

Uno de los primeros sensores que se quiso utilizar para desarrollar este proyecto fué el sensor: sensorpulse.

4.2.5. SensorPulse.

Figura 16. SensorPulse.

El sensorpulse es un tipo de sensor que mide la frecuencia cardiaca que presenta la gran ventaja de que esta compuesto internamente para que solo tengamos que conectar tres cables que son la de alimentacion, la de datos. Y la de tierra.

Se podría utilizar un puerto analogico de los que posee el microcontrolador (NodeMCU) y podriamos obtener los datos biomedicos.Otra de las ventajas que nos ofrece este sensor es que la calibracion de este sensor se produce via software , ya que ofrece unas librerías oficiales del sensor donde se puede hacer una serie de llamadas a los metodos de estas librerias y obtendriamos la frecuencia cardiaca de un determinado paciente. Tambien cabe destacar el hecho de que combina un sensor óptico de frecuencia cardíaca simple con circuitos de amplificación y cancelación de ruido que lo hacen rápido y fácil de obtener lecturas de pulso confiables. Además, consume energía con solo un consumo de corriente de 4 mA a 5 V, por lo que es ideal para aplicaciones móviles [9].

Pero precisamente el hecho de que solo mida la frecuencia cardiaca de un paciente y no mida inclusive la saturación de oxígeno en sangre hizo que se quedara corto el hecho de desarrollarlo como motor principal de este proyecto y se optó por descartarlo.

Llegado a este punto, se dispone a realizar una serie de postulados por el cual se ha decidido trabajar con el sensor de Texas Instruments: MAX30100.

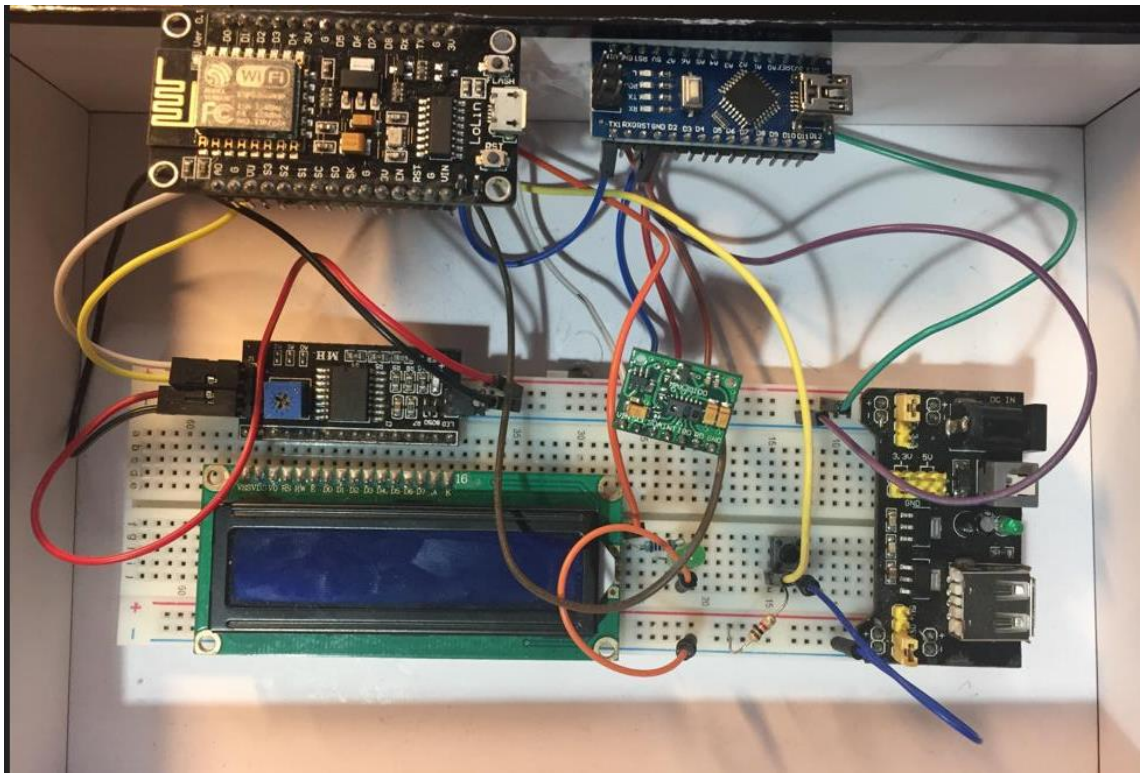

Figura 17. Prototipo.

En primer lugar, se observa que mediante una fuente de energía externa se alimenta el circuito electrónico. A continuación, se recogerán los datos biomédicos del sensor que, posteriormente haciendo uso de un módulo wifi, se enviarán a un servidor web para su posterior visualización.

El proyecto está dividido en una parte electrónica donde se recogen los datos y se envían y otra telemática donde se recogen esos parámetros y los visualiza el usuario.

En los siguientes capítulos se verán las medidas adoptadas y como se ha sustentado estos dos campos para la realización del mismo.

4.2.6. MAX30100.

Figura 18. MAX30100.

El chip MAX30100 es un sensor capaz de detectar niveles de saturacion de oxigeno en sangre y frecuencia cardiaca en un paciente. Disponde de un led rojo y otro infrarrojo (el MAX30101 cuenta ademas con un tercer led verde) que iluminan alternativamente durante cierto tiempo (ancho de pulso) la zona expuesta (por ejemplo, un dedo).

La luz reflejada se detecta con un fotodiodo donde la intensidad de luz que es captada se le corrige la desviacion que produce la iluminacion ambiental, el ruido electrico de baja frecuencia (los 50 Hz o 60 Hz) y la influencia de la temperatura, para lo que cuenta con un termometro interno.

La señal obtenida se convierte en valores digitales y se almacena en un buffer al que se puede acceder desde un microcontrolador mediante el bus i2c con que cuenta el MAX30100.

La configuracion que se ha realizado en este proyecto para poder configurar de manera exitosa la implantacion del sensor es muy sencilla.

La alimentación que necesita el sensor dado la hoja de caracteristica se encuentra en torno a los 3-5 voltios. Por esta razon, se ha dispuesto a alimentarlo mediante el pin de 3 voltios que presenta el NodeMCU [10].

El pin de toma a tierra se ha conectado al pin equivalente al Arduino nano, y, finalmente tanto la señal de datos (SDA) como la señal de reloj (SCL) se han conectado a los pines A4 y A5 respectivamente tambien de este microcontrolador.

El hecho de decantarnos por este sensor es la comprobacion de la eficiencia asi como una linea de innovacion en versiones posteriores que dara lugar a una exatitud mas precisa de los datos biomedicos asi como la correccion de errores por parte del MAX30100.

La gestion de la obtencion de los datos biomedicos llevados a cabo por el sensor se ha realizado por medio de unas librerias oficiales del mismo, obtenido por medio de un repositorio oficial. Teniendo esto en cuenta y utilizando el IDE de Arduino, simplemente importaremos las librerias y llamaremos a cada de los metodos que queramos utilizar, en este caso se hace la llamada a la obtencion de la frecuencia cardiaca y de la saturacion del oxigeno en sangre.

```
#include "MAX30100_PulseOximeter.h"
```

Figura 19. Librería MAX30100.

Se puede apreciar en la siguiente como se obtiene, vía software, cada uno de estos valores.

```
Serial.print("@");
Serial.print(pox.getHeartRate());
Serial.print(";");
Serial.print(pox.getSpO2());
Serial.print(";");
Serial.println("");
```

Figura 20. Métodos oxígeno y saturación.

En primer lugar se establecerá una conexión serie mediante el nano y el nodemcu. La funcionalidad que tendrá el arduino nano será la de obtener los dos parámetros biomédicos que son, la saturación de oxígeno en sangre (SPO2) así como el pulso del paciente y enviarlo al Nodemcu.

El Nodemcu recogerá esos parámetros que posteriormente enviará al servidor por medio del protocolo http.

Es por este motivo, y una vez comentado la sensorizacion, nombrar el hecho del envio de esos datos mediante http, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos html [11]. Es la base de cualquier intercambio de datos en

la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada, por el elemento que recibirá los datos (el cliente), normalmente un navegador Web.

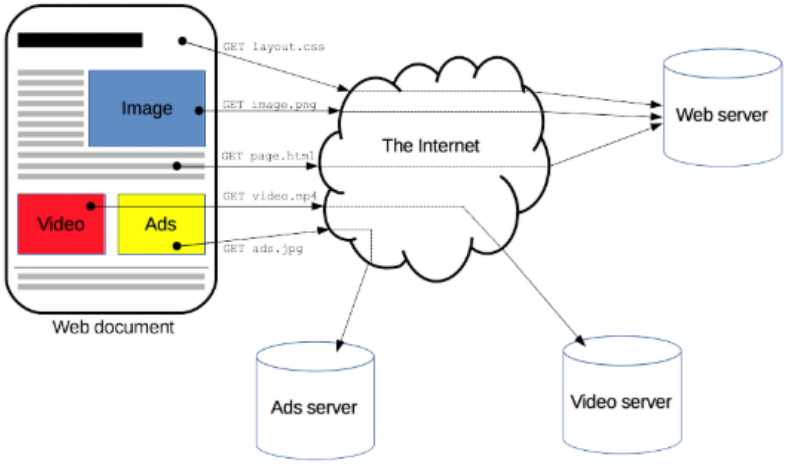

Figura 21. Protocolo http.

Para hacer uso del envio de datos por parte del microcontrolador al servidor web es necesario recalcar el uso de la siguiente librería:

```
#include <ESP8266WiFi.h>
```

Figura 22. Librería WI-FI.

Una vez que se ha importado con éxito esta librería, y de cara oculta al usuario, el microcontrolador debe distinguir dos modos de funcionamiento distintos.

El modo de punto de acceso y el modo estación.

Para que se pueda trabajar en un modo de funcionamiento u otro, el usuario simplemente hará uso de un botón para que, al pulsarlo entre por defecto en el modo punto de acceso y si no se pulsa acceda al modo estación.

4.2.7. Modo punto de acceso.

Inicialmente, se debe vincular el microcontrolador con el servidor por medio de la interacción de un script. Es por esto que se debe de iniciar este primer modo de funcionamiento para guardar en la memoria EPROM cada una de las credenciales.

La forma de vincular el microcontrolador y el servidor, de forma directa para el usuario se limita en pulsar un botón, ya que si se pulsa, el microcontrolador trabajará en modo punto de acceso, pero si no se pulsa, por defecto el microcontrolador trabajara en modo estación, esto conllevará una gran ventaja ya que al almacenarse la información mediante un tipo de memoria no volátil, estarán almacenadas las credenciales y no se le obliga al usuario a que cada vez que encienda el prototipo, establezca los para metro de vinculación con la red, ya que en ese caso sería inviable.

De forma interna se está produciendo una seria de interacciones que se describen a continuación.

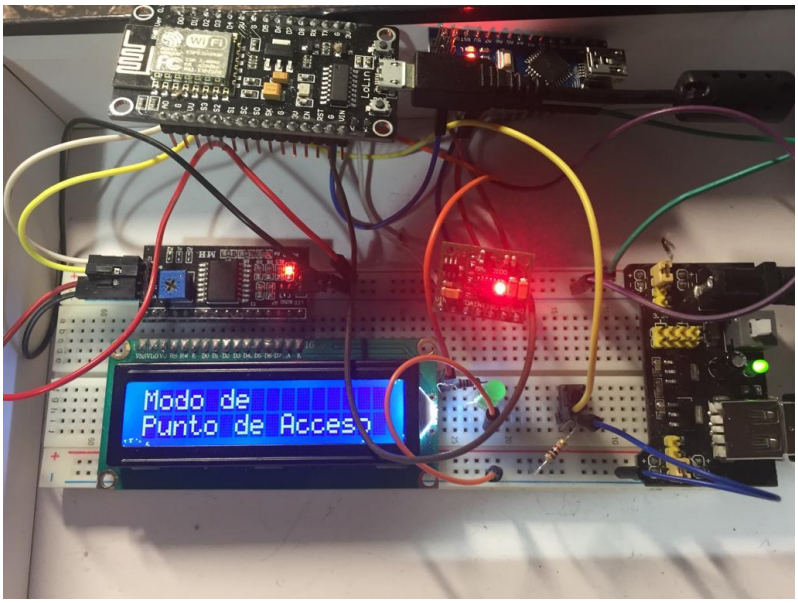

Figura 23. Modo punto de acceso.

Lo primero que observará el usuario cuando acceda a este modo de funcionamiento del NodeMCU, es la creación de una red abierta, la cual podrá conectarse sin mayor dificultad. Para que sea lo más cómodo posible de cara a posibles usuarios que, normalmente, serán personas de avanzada edad o con algún tipo de anomalía relacionadas con la frecuencia cardiaca o la saturación de oxígeno en sangre, con un conocimiento poco técnico con respecto a redes, se ha decidido prescindir de la seguridad de la red que se ha creado para que el usuario pueda conectarse fácilmente.

```
//nombre de la red del accespoint
const char *ssidConf = "OXIPADI";
```

Figura 24. Red NodeMCU

Para inicializar el modo punto de acceso, y de forma oculta al usuario, se hace uso del método “softAPIP”.

A continuación, este método por defecto siempre devolverá una dirección IP por defecto que será 192.168.4.1. Debido a que este proyecto tiene la gran ventaja de que no necesariamente tiene que utilizarse un determinado NodeMCU asociado a cada aplicación, sino que se puede hacer uso de otro NodeMCU si se precisa, se ha considerado dejar esta ip por defecto para mayor comodidad de cara al usuario.

A continuación, se muestra un mensaje al usuario indicándole que acceda a esta dirección ip cuando esté conectado a la red abierta citada anteriormente.

```
void modoconf() {

  lcd.setCursor(0,0);
  lcd.print("Modo de");
  lcd.setCursor(0,1);
  lcd.print("Punto de Acceso");
  delay(2000);
  lcd.clear();
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);

  WiFi.softAP(ssidConf);
  IPAddress myIP = WiFi.softAPIP(); //inicia el modo acces point
  Serial.print("IP del acces point: ");
  Serial.println(myIP); // por defecto la ip que da el node para el ap es 192.168.4.1
  lcd.setCursor(0,0);
  lcd.print("Introduzca la ip:");
  lcd.setCursor(0,1);
  lcd.print(myIP);
  delay(3000);
  Serial.println("WebServer iniciado...");
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Webserver");
  lcd.setCursor(0,1);
  lcd.print("iniciado...");
  delay(10000);
  lcd.clear();

  server.on("/", paginaconf); //esta es la pagina de configuracion

  server.on("/guardar_conf", guardar_conf); //Graba en la eeprom la configuracion

  server.on("/escanear", escanear); //Escanean las redes wifi disponibles

  server.begin();

  while (true) {
      server.handleClient();
  `
```

Figura 25. Script punto de acceso.

Llegado a este punto, el usuario observara una página, creada en el propio script, donde se encuentran tres campos en el que tendrá que introducir cada una de las credenciales y dos botones para enviar esa información o para escanear las redes a las que se quiere conectar.

```
//-----------CODIGO HTML PAGINA DE CONFIGURACION---------------
String pagina = "<!DOCTYPE html>"
"<html>"
"<meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>"
"<head>"
"<title>OXIPADI</title>"
"<style type='text/css'> body,td,th { color: #036; } body { background-color: #999; } </style>"
"</head>"
"<body>"
"<center><h1>WIFI CONF</h1></center><br>"
"<center><h2>OXIPADI</h2></center><br>"
"<form action='guardar_conf' method='get'>"
"<center><fieldset align='center' style='border-style:solid; border-color:#336666;>"
"<legend><strong>Configurar WI-FI</strong></legend>"
"SSID:<br> <input class='inputl' name='ssid' type='text' size='24'/><br><br>"
"PASSWORD: <br> <input class='inputl' name='pass' type='password' size='24'/><br><br>"
"IDENTIFICADOR: <br> <input class='inputl' name='Identificador' type='text' size='24'/><br><br>"
"<br><input class='boton' type='submit' value='GUARDAR'/><br><br>"
"</form>"
"<form action='escanear' method='get'>"
"<input class='boton' type='submit' value='ESCANEAR REDES'/><br><br>"
"</form>";

String paginafin = "</body>"
"</html>";

void onBeatDetected()
{
    lcd.print("Bien");
    delay(1000);
```

Figura 26. Página html.

Cuando se accede a este panel de administrador se pueden observar tres elementos:

- SSID: Aquí se establece el nombre de la red Wifi a la que se quiere acceder.
- Password: La contraseña de la red a la que se quiere acceder.
- Identificador: Aquí se debería de introducir el identificador único que autogenera la aplicación para cada usuario y vincular la aplicación con el prototipo electrónico.

Si el usuario pulsa el botón de “Escanear redes”, automáticamente la página irá, mediante el método GET al método “escanear”.

```
//---------------------------ESCANEAR-----------------------------
void escanear() {
  int n = WiFi.scanNetworks(); //devuelve el número de redes encontradas
  Serial.println("escaneo terminado");
  if (n == 0) { //si no encuentra ninguna red
    Serial.println("no se encontraron redes");
    mensaje = "no se encontraron redes";
  }
  else
  {
    Serial.print(n);
    Serial.println(" redes encontradas");
    mensaje = "";
    for (int i = 0; i < n; ++i)
    {
      // agrega al STRING "mensaje" la información de las redes encontradas
      mensaje = (mensaje) + "<p>" + String(i + 1) + ": " + WiFi.SSID(i) + " (" + WiFi.RSSI(i)
      //WiFi.encryptionType 5:WEP 2:WPA/PSK 4:WPA2/PSK 7:open network 8:WPA/WPA2/PSK
      delay(10);
    }
    Serial.println(mensaje);
    paginaconf();
  }
}
```

Figura 27. Escaneo de redes.

Mediante una variable entera, se almacena el número de redes escaneadas mediante el método “scanNetworks ()”. En este punto, se recorrerá mediante un bucle cada una de las redes escaneadas y se mostrarán al usuario.

Cuando se haya establecido tanto el SSID de la red a la que quiere conectarse, el password y el identificador único que se proporcionará por parte de la aplicación, se pulsará el botón enviar.

```
void guardar_conf() {

  Serial.println(server.arg("ssid"));//Recibimos l
  grabar(0,server.arg("ssid")); // 0 es la posicio
  Serial.println(server.arg("pass"));
  grabar(50,server.arg("pass"));
  Serial.println(server.arg("Identificador"));
  grabar(100,server.arg("Identificador"));

  mensaje = "Configuracion Guardada...";
  paginaconf();
}
```

Figura 28. Guardar configuración.

En este punto, se almacenarán las credenciales en la memoria EPROM. Los cincuentra primeros caracteres estarán reservados para el ssid, los siguientes cincuenta para el password, y los últimos cincuentas estarán destinados para almacenar el identificador.

El método para grabar cada una de las credenciales dentro de la memoria EPROM del NodeMCU es el siguiente.

```
//----------------Función para grabar en la EEPROM-------------------
void grabar(int addr, String a) {
  int tamano = a.length();
  char inchar[50];
  a.toCharArray(inchar, tamano+1); //en la eprom trabaja con caracteres no string
  for (int i = 0; i < tamano; i++) {
    EEPROM.write(addr+i, inchar[i]);
  }
  for (int i = tamano; i < 50; i++) {
    EEPROM.write(addr+i, 255);
  }
  EEPROM.commit();
}
```

Figura 29. Grabar.

Como se observa, se introducen como parámetros de entrada al método, la dirección de memoria donde se quiere dejar alojado una variable, y el segundo método es un String con el nombre de esa variable.

Como en la memoria EPROM trabaja con caracteres, no con String, es necesario pasar cada una de las variables como tipo carácter. En este punto, se ira recorriendo un bucle guardando en ese espacio de memoria cada carácter.

Esto tiene la gran ventaja de que cuando se encienda de nuevo el prototipo electrónico accederá a la red que ya está configurada y se vinculara con el proyecto.

Es por esto que se hace una comprobación en la base de datos por medio del envío del identificador por parte del cliente al servidor, y es en éste donde se compara con la base de datos de la aplicación y si coincide devolverá al cliente un código http que será de 200 (aceptación).

La medida de comprobación que obtendrá el usuario dado el envío de ese identificador único es que, se ha transformado ese código http para que, en el caso de que devuelva el servidor un código 200, el cliente observara un mensaje: éxito. En caso contrario, la pantalla lcd mostrará un mensaje de error.

Cabe mencionar que a la hora de crear la red Wifi que tendrá acceso el usuario se pueden introducir ciertos valores como pueden ser el nombre de la red, el tipo de encriptación, el nivel de seguridad de la misma, etc…

De igual manera que la creación de la red nombrado anteriormente, también se puede implementar cierto tipo de variaciones a la hora de presentar al usuario las distintas

redes a la que tiene acceso, como por ejemplo la potencia que se encuentra cada red, el tipo de canal entre otros aspectos. Viene representado de la siguiente manera:

4.2.8. Modo estación.

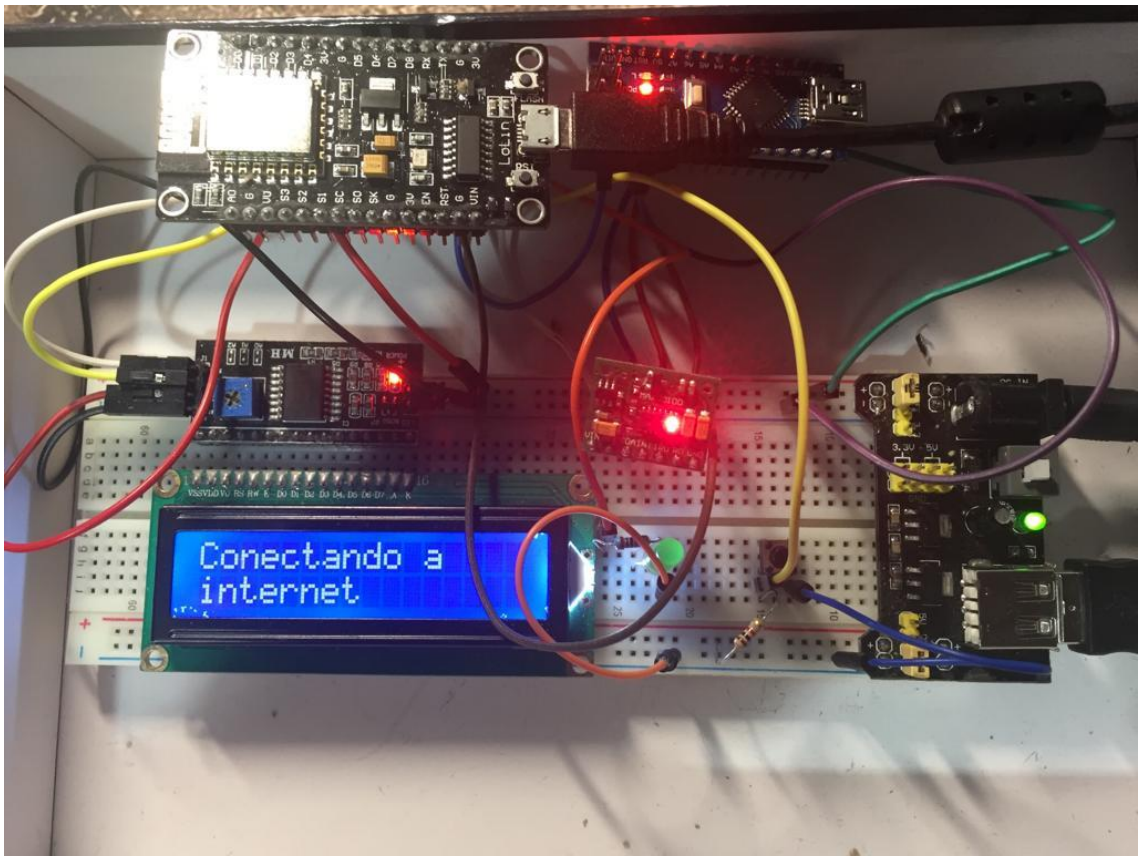

Figura 30. Conectando a internet.

El modo de estación que se inicializa por defecto es debido gracias a que tanto el ssid como el password, en este caso de la red que se ha configurado previamente se almacenan en la memoria EPROM. Es un tipo de memoria no volátil, esto quiere decir que se almacena la información que se carga en esta memoria y aunque se produzca un reseteo del microcontrolador, la información seguirá permaneciendo en el mismo [12].

Es por esto que, si se enciende el dispositivo y no se tiene pulsado el botón, el microcontrolador accederá a la memoria EPROM y accederá al modo estación por defecto y consecuentemente a esto, se conectará con éxito a la red que hemos configurado, esto tiene la gran ventaja de no tener que configurar la red a la que

queremos acceder cada vez que apaguemos el dispositivo o lo reseteemos, algo que sería inviable.

El usuario cuando encienda el dispositivo, accederá con éxito a la red que se configuró previamente.

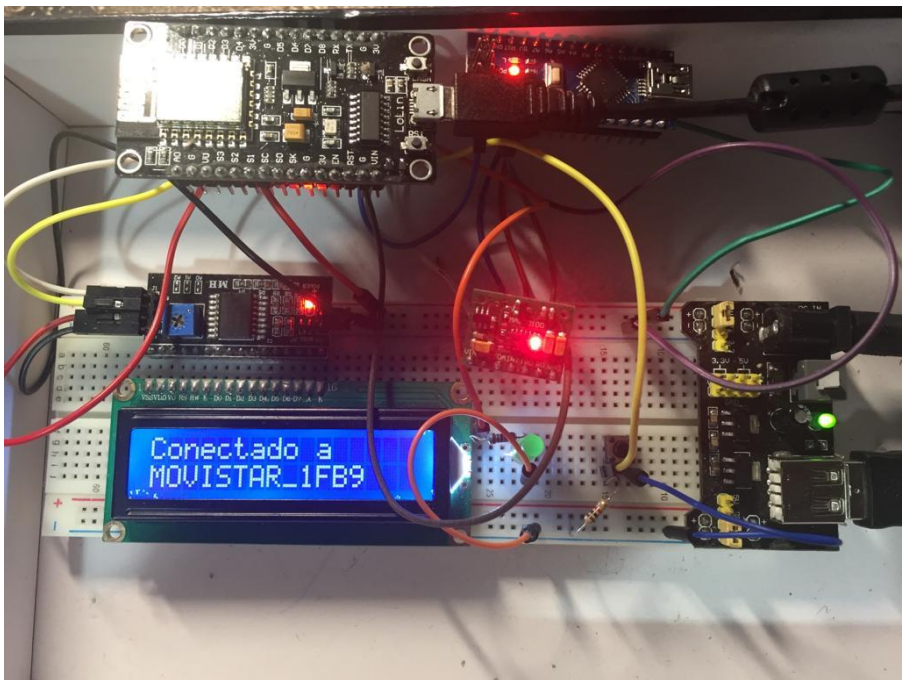

Figura 31. Conectado a internet.

En cambio, si no se puede acceder a la red, existirá un retardo de 500 milisegundos y al final de un determinado tiempo mostrará un error de conexión.

Para que esta conexión se lleve a cabo, se hace uso del método “mode”, indicando al microcontrolador que actuará en modo estación.

```
void setup_wifi() {

   lcd.setCursor(0,0);
   lcd.print("Conectando a ");
   lcd.setCursor(0,1);
   lcd.print("internet");
   delay(2000);
   lcd.clear();
// Conexión WIFI
  WiFi.mode(WIFI_STA); //para que no inicie el SoftAP en el r
  WiFi.begin(ssid, pass); //ssid y pass que lo coge de la epr
  while (WiFi.status() != WL_CONNECTED and contconexion <50)
    ++contconexion;
    delay(250);
    Serial.print(".");
    digitalWrite(13, HIGH);
    delay(250);
    digitalWrite(13, LOW);
  }
```

Figura 32. Conexión via WI-FI.

Si el método “status” indica que el estado es conectado, se dispondrá a leer de la memoria EPROM cada una de las variables que se han introducido con anterioridad.

```
 if (contconexion <50) {
     Serial.println("");
     Serial.println("WiFi conectado");
     lcd.setCursor(0,0);
     lcd.print("Conectado a");
     lcd.setCursor(0,1);
     lcd.print(ssid);
     delay(2000);
     lcd.clear();
     Serial.println(WiFi.localIP());
     digitalWrite(13, HIGH);

     //Envio del identificador
      String encrip=leer(100);
      String Identificador="Identificador="+encrip;
      Serial.println("Identificador es: ");
      Serial.println(encrip);
      lcd.setCursor(0,0);
      lcd.print("Identificador es");
      lcd.setCursor(0,1);
      lcd.print(encrip);
      delay(1000);
      lcd.clear();
```

Figura 33. Estado conectado.

El usuario observará, por medio de la pantalla lcd que se ha conectado de manera exitosa a la red. Así mismo, se mostrará el identificador que ha introducido para que, posteriormente sea enviado al servidor.

El método para leer de la memoria la variable del identificador único, se realiza, pasándole a este método, el espacio de memoria que contendrá al Identificador explicado con anterioridad, es decir, a partir del carácter cien.

Cuando se muestre este identificador, y una vez ya esté vinculado el NodeMCU con el servidor, se decide a mostrar el envío de este Identificador al servidor.

```
USE_SERIAL.printf("Enviando Identificador\r\n");

lcd.setCursor(0,0);
lcd.print("Enviando");
lcd.setCursor(0,1);
lcd.print("Identificador...");
delay(500);
lcd.clear();

HTTPClient http;
http.begin("http://192.168.1.36:8080/tfg/Aplicacion");
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
int httpCode= http.POST(Identificador);
http.writeToStream(&Serial);
http.end();
USE_SERIAL.printf("Se ha enviado !!! \r\n\r\n");

lcd.setCursor(0,0);
lcd.print("Se ha");
lcd.setCursor(0,1);
lcd.print("enviado !!!");
delay(500);
lcd.clear();

Serial.println("[Response:]");
Serial.print("HTTP response code "); //imprime el codigo de estado
Serial.println(httpCode);
```

Figura 34. Envío Identificador.

Mediante el método POST, se produce el envío al método “Aplicación” dentro de la app que se describe en el capítulo siguiente.

Cuando llega esta información al servidor, se comprueba con la base de datos y si el identificador que se envía por parte del usuario coincide con el identificador que se ha autogenerado por parte de la aplicación, mandará un mensaje de “Created” http (Código

201) al cliente. Debido a la transparencia por parte del usuario, se ha parseado este código de respuesta del servidor, y si se recibe un código 201, el usuario observara en la pantalla lcd un mensaje tal como: “Correcto” en caso contrario mostrara un error.

4.3. Aplicación web.

Para la aplicación que se ha realizado, se ha tenido en la cuenta la jerarquización de distintos bloques, en el que, conjuntamente, interactúan entre sí para proporcionar distintas funciones al usuario por medio de unas vistas.

Esta aplicación está sustentada en el lenguaje de programación Java, concretamente, en uno de los frameworks más utilizados que es Spring, se verá con más detalle más adelante.

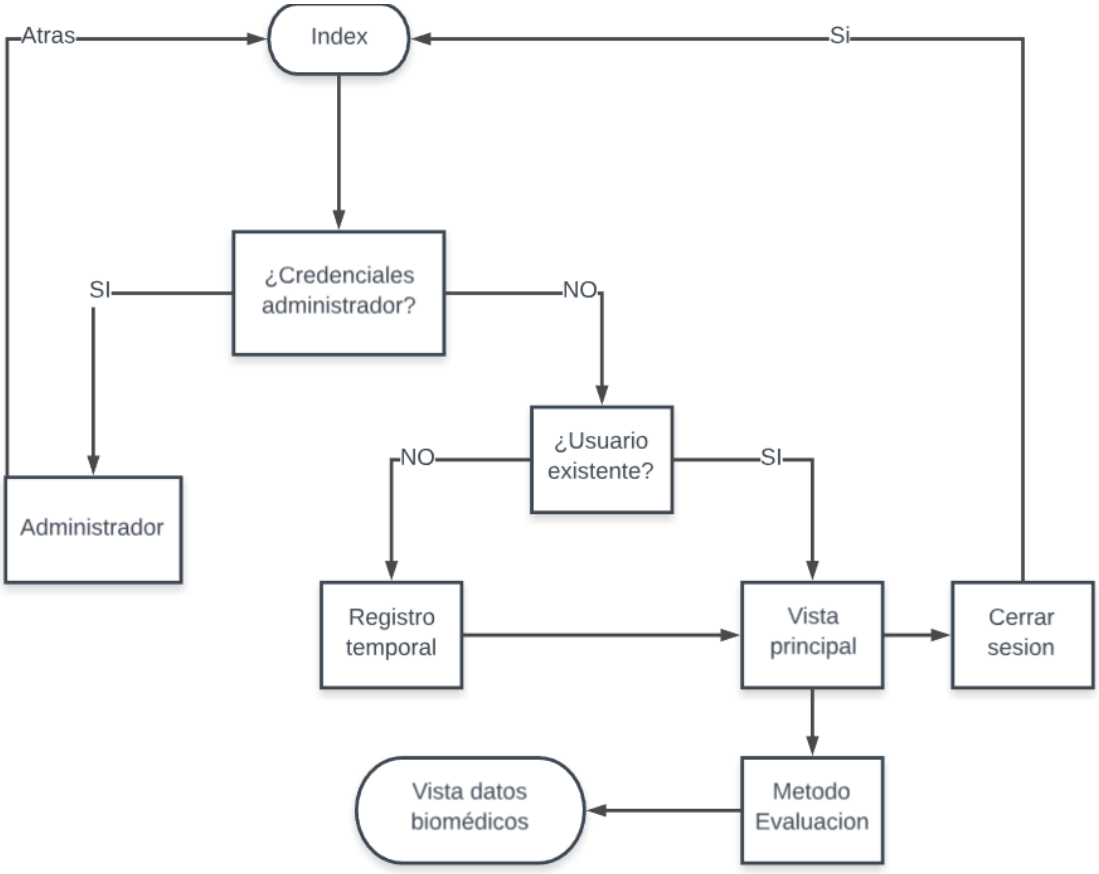

Figura 35. Diagrama de flujo de la aplicación.

4.3.1. Inicio sesión.

Este primer bloque, se centra en la interacción principal que tiene el usuario con la aplicación. Si introduce unas credenciales que ya existen en la base de datos, la aplicación ejecutara unas funciones que si esas credenciales no se encuentran en la base de datos que ejecutara otras funciones distintas.

De igual modo, si las credenciales que introduce un cierto usuario coinciden con las de administrador se proporcionara un flujo de datos distinto que si no coinciden.

En este postulado, y debido a la anterior explicación de cómo se ha estructurado el proyecto vía hardware, es conveniente describir como se ha realizado la recepción de esos datos por parte del cliente en un servidor web y la manera que tiene de gestionarlos y visualizarlos de cara al usuario final.

En primer lugar, y como ocurre en cualquier aplicación, lo primero que debe realizar a la hora de ejecutar una aplicación es el proceso de inicio de sesión.

La manera que tiene Spring de recoger las llamadas a cada uno de los métodos mediante URL es a través de la anotación @RequestMapping.

Es por esto que cuando se ejecuta el proyecto, el directorio raíz deberá redireccionar a la vista inicial que es la que se muestra a continuación.

Figura 36. Inicio de sesión.

Como se ha comentado, si el usuario introduce unas credenciales u otras, el flujo de información repercutirá en un cierto tipo de ejecuciones u otras. Por lo tanto, es conveniente saber cómo realiza Spring de forma interna, este tipo de actuaciones de forma transparente al usuario.

```
@Controller
public class HomeController {
    private String nickname_global=null;
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);


    @Autowired
    private UsuariosDaoInterface dao;

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(HttpServletRequest request,Locale locale,HttpServletResponse response, Model model) {
        Cookie[ ] cookies = request.getCookies();
        String cookieName = "emailCookie";
        String email="";
        HttpSession session = request.getSession(true);
        if(cookies!=null) {
            for(Cookie cookie:cookies) {
                if(cookieName.equals(cookie.getName())) {
                    email = cookie.getValue();
                }
            }
        }

        if(email.equals("")) {
            return "index";
        }
        else {
            UsuarioDTO usuario = dao.LeerNickname(email);
            if(usuario!=null) {
                session.setAttribute("usuario", usuario);
                if(usuario.getNickname()=="admin" && usuario.getContraseña()=="admin") {
                    return "Administrador";
                }
                else return "Aplicacion";
            }
            else return "index";
        }
}
```

Figura 37. HomeController.

Inicialmente, declaramos la clase como @Controller, para indicar a esa clase que va a ser un controlador, dentro del patrón MVC que se ha explicado en este trabajo.

A continuación, mediante la etiqueta @Autowired se hará una inyección de la interfaz DAO que contendrá todos los métodos requeridos para la conexión con la base de datos.

Como se ha citado con anterioridad, se mapea la URL con la dirección raíz mediante el método GET, y por medio de las cookies de sesión, que simplemente por definición son un conjunto de clave-valor, se hace una comprobación si el email que se ha introducido se encuentra en la sesión.

En caso de que se encuentre en la sesión, habrá que diferenciar tipo de credenciales que se han introducido. Si coinciden en la base de datos con algún usuario activo en la

aplicación, el flujo de información ira directamente a la vista principal de la aplicación, en caso contrario irá a la vista de administrador.

El método de comprobación que se lleva a cabo para ver si existe un usuario en la base de datos o no es el siguiente:

```
public UsuarioDTO LeerNickname(String Nickname){
    String sql = "select * from usuarios where Nickname = ?";
    Object[ ] parametros = {Nickname};
    UsuarioMapper mapper = new UsuarioMapper();
    List<UsuarioDTO> usuarios = this.jdbcTemplate.query(sql, parametros, mapper);
    if (usuarios.isEmpty()) return null;
    else return usuarios.get(0);
}
```

Figura 38. LeerNickname.

Como parámetro de entrada, se introduce el nickname que un determinado usuario establece y se hace una búsqueda dentro de la tabla usuarios donde el nickname coincida con el parámetro de entrada a la función. Si coincide, querrá decir que ese usuario con ese nickname que ha introducido un determinado paciente existe y por tanto devolverá esa variable, en caso contrario la función devolverá null.

Llegado a este punto, y como el administrador del sistema también forma parte de la base de datos, se diferencia si el usuario es administrador del sistema o no, mediante un condicional.

Si las cookies de sesión no se encuentran en el navegador, el flujo de información se ejecutará por medio del formulario de inscripción. Éste, se enviar a un método llamada “Formulario”, en el que se recoge en Spring de la siguiente manera.

```java
@RequestMapping(value = "/Formulario", method = RequestMethod.POST)
public String sesion(@RequestParam("txtNickname") String nickname,@RequestParam("txtPass") String contraseña,HttpServl

    UsuarioDTO db= new UsuarioDTO();

    //Comprueba si los datos introducidos coiniciden con el administrador
    if(db.CompruebaAdmin(nickname,contraseña)) {
        HttpSession session = req.getSession(true);
        List<UsuarioDTO> usuario=dao.leeUsuarios();
        session.setAttribute("usuario", usuario);

        return"Administrador";



        }else {


            //Comprueba si los datos introducidos coinciden con un usuario de la base de datos, en ese caso, se dirije
            //si devuelve null es que ese usuario con esas credenciales no existe en la base de datos se pide que se r
            if(dao.LeerCredenciales(nickname,contraseña)==null) {
                return "BadCredenciales";
            }else {

                //si no devuelve null es que ese nickname coincide con un usuario que ya esta en la base de datos
                HttpSession session = req.getSession(true);
                UsuarioDTO paciente=dao.LeerNickname(nickname);
                session.setAttribute("paciente", paciente);
                Cookie c = new Cookie("emailCookie", paciente.getNickname());
                c.setMaxAge(60*60*24*31);//caducan cada mes
                c.setPath("/");
                response.addCookie(c);
                return "Aplicacion";
            }
```

Figura 39. Formulario.

En este método, se recogerán mediante la anotación @RequestParam, cada una de las variables procedentes del formulario de inscripción, en este caso el nickname y la contraseña.

A continuación, se crea un objeto de la clase DTO y se comprueba mediante el método “CompruebaAdmin”, si las credenciales coinciden con el administrador.

En ese caso, se llamará a un método que leerá todos los usuarios de la base de datos y los introducirá en la sesión para que puedan ser visualizados en la vista de administrador.

Si las credenciales que se han introducidos son incorrectas, el flujo de información redirigirá la información a una vista indicándole al usuario que las credenciales que ha introducido son incorrectas.

En caso contrario, se hará una comprobación en la base de datos, llamando al método “LeerNickname”, explicado con anterioridad.

Se establecen unas cookies de sesión que formara parte del nickname del usuario, y finalmente el flujo de datos se ejecutara y se mostrara la vista principal de la aplicación.

4.3.2. Administrador.

En este punto, el administrador de la aplicación tendrá un control total para llevar a cabo la monitorización de cada uno de los usuarios dados de alta en la aplicación llevados a cabo por medio de un panel de control.

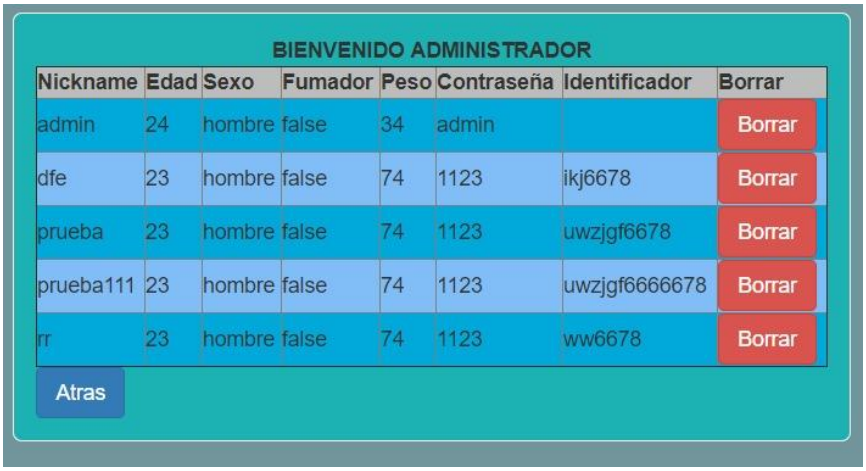

Figura 40. Panel de administrador.

Cabe destacar el hecho de que el administrador de la aplicación puede dar de baja un usuario.

```
<form action="Borrar" method="POST">
    <input type="hidden" name="email" value="${user.nickname}">
    <button type="submit" class="btn btn-danger">Borrar</button>

</form>
```

Figura 41. Formulario dar de baja.

Mediante un input de tipo oculto, se envía al método “Borrar” el nickname del usuario que se quiera dar de baja de la aplicación.

```
@RequestMapping(value = "/Borrar", method = RequestMethod.POST)
public String Borrar(HttpServletRequest request,Locale locale, Model model) {

    String nickname=request.getParameter("email");
    dao.BorrarUsuario(nickname);

    return "index";

}
```

Figura 42. Borrar usuario

Por medio de la anotación de @RequestMapping, se mapea la url proveniente del formulario, a continuacion, se recoge mediante un String el email asociado del usuario que se quiera dar de baja de la aplicación, y mediante un método dao, se hace un delete de la base de datos dado ese nickname.

En este espacio, se llevará a cabo la gestión por parte de los usuarios. En el caso de que algún usuario inicie sesión y esté autenticado con anterioridad, accederá directamente a la vista de la aplicación general, en caso contrario y como ocurre normalmente en cualquier aplicación, se indiciará un registro temporal para que el paciente pueda introducir sus datos.

La última columna muestra el identificador único que se ha creado para anexionar a cada paciente con cada microcontrolador.

4.3.3. Registro temporal.

La vista que se le ofrece al usuario, teniendo en cuenta el flujo principal de la aplicación consiste en que si un usuario, establece unas credenciales donde se comprueba que no existen en la base de datos, o introduce unas credenciales que son erróneas, o hace click en el enlace para registrarse, estas tres acciones conllevaran a la vista siguiente del registro temporal.

Este registro temporal se recoge como muestra la siguiente imagen.

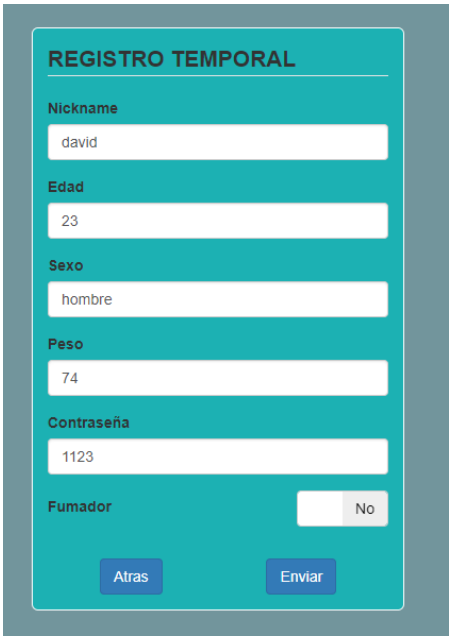

Figura 43. Registro temporal.

Por medio de la introducción de distintas variables, Spring deber recoger esas variables y almacenarlas.

```
@RequestMapping(value = "/Registro", method = RequestMethod.POST)
public String registropost(HttpServletRequest request,Locale locale, Model model,HttpServletResponse response) {

    if(request.getSession(false)==null) {


        return "Caduca";


    }else {


        String nickname= (String)request.getParameter("txtUsername");
        nickname_global=nickname;
        String edad= (String)request.getParameter("txtEdad");
        String sexo=request.getParameter("txtSexo");
        System.out.println(sexo);
        String peso=request.getParameter("txtPeso");
        System.out.println(peso);
        String contraseña=request.getParameter("txtContras");
        System.out.println(contraseña);
        String fumar=request.getParameter("txtFumador");

        Boolean fumador=Boolean.valueOf(false);
        if(fumar==null) {
            fumador=false;
        }else {
            fumador=true;
        }



        //No insertar dos usuarios con el mismo email
        UsuarioDTO usuario= dao.LeerNickname(nickname);

        //Si el usuario existe por su email dara error
        if(usuario!=null) {

            return "UsuarioExistente";

        //Si el usuario no existe dara acceso a la aplicacion
        }else {
            HttpSession sesion = request.getSession();
            UsuarioDTO paciente=new UsuarioDTO(nickname, edad,sexo,fumador,peso,contraseña);
            sesion.setAttribute("paciente", paciente);
            Cookie c = new Cookie("emailCookie", paciente.getNickname());
            c.setMaxAge(60*60*24*31);//caducan cada mes
            c.setPath("/");
            response.addCookie(c);
            //Insertamos un nuevo usuario
            dao.NuevoUsuario(paciente);

            return "Aplicacion";
```

Figura 44. Registro.

Se recogen las distintas variables de entrada del formulario, también se inicializa una variable global y se iguala al nickname. Esto se explicará con posterioridad.

Para dar de alta un usuario en la aplicación, hay que tener en cuenta que no se pueden introducir dos usuarios con el mismo email, es por esto que se comprueba antes en la base de datos.

En el caso de que el nickname que haya introducido un determinado usuario no exista en la base de datos, se dará de alta en la aplicación como un nuevo usuario y el flujo de información re direccionará a la vista principal de la aplicación.

Cuando se acceda de forma exitosa, la visualizacion de la vista que se obtendrá es la misma que si se inicia sesión con un usuario registrado. Esto se lleva a cabo por medio de las cookies de sesión.

4.3.4. Aplicación.

La vista principal de la aplicación será la siguiente.

Figura 45. Aplicación.

Se observa como la propia aplicación, y de forma oculta al usuario, se ha creado un identificador único para ese paciente en concreto, mostrando la ventaja de no tener que adquirir de forma obligatoria un microcontrolador en específico, si no que se puede utilizar cualquier NodeMCU que después se vinculará a la aplicación. Produciendo así

un aspecto positivo en el desarrollo de posibles avances futuros y manteniendo la cohesión entre los distintos bloques tanto vía software como hardware.

```
//Metodo de encriptacion
String mensaje=nickname+contraseña;
System.out.println(mensaje);
char array[] = mensaje.toCharArray();
for(int i = 0; i<array.length; i++) {
    array[i] = (char)(array[i]+ (char)5);
}
```

Figura 46. Hash.

La forma de creación de este identificador único se ha realizado mediante la implantación de un hash.

Declaramos una variable como String concatenando el nombre y la contraseña de un determinado paciente, a continuación, lo pasamos a caracteres y lo movemos cinco bytes produciendo así un hash único para cada usuario. Sería más robusto si a parte del nombre y la contraseña, asignásemos más variables.

Es este identificador el que tenemos que poner en el panel de control del microcontrolador cuando se trabaje en modo punto de acceso

A continuación, se interactúa con el servidor web para obtener una consulta de la base de datos indicándonos los parámetros biomédicos.

La principal función de un servidor web es almacenar los archivos de un sitio y emitirlos por internet para que las páginas que aloja puedan ser visualizadas por los usuarios.

4.3.5. Evaluación.

En este proceso, se observa varias pestañas, la mas importante es la que dará a conocer al usuario final sus parámetros biomédicos, para ello debería clickar en la pestaña ¨Evalucion¨ donde finalmente veria la siguiente vista donde se irán recogiendo y actualizando la gráfica cada ciertos segundos.

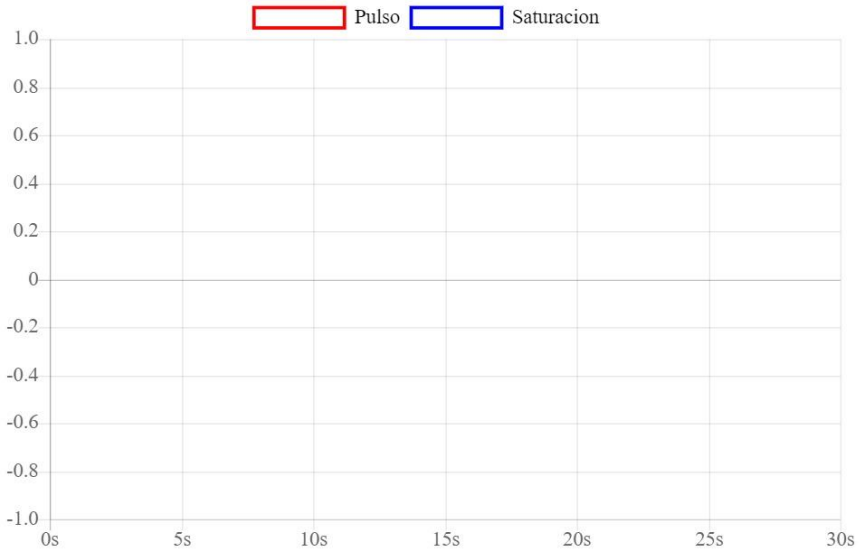

Figura 47. Vista Evaluacion Spring.

Cabe mencionar que la gráfica que se muestra de forma dinámica se ha realizado con Chart.js que es una herramienta, potente y simple para la creación de gráficos, es una de las librerías de Javascript mas usadas para la creación de gráficos, existen librerías más potentes pero a la vez son más complejas como D3.js

El proceso que sigue el flujo de información en Spring es el siguiente, cuando el usuario accede a la vista ¨Evaluacion¨ , el HomeController recibe esta peticion de la siguiente forma:

```
//vista de la aplicacion .jsp
@RequestMapping(value = "/Evaluacion", method = RequestMethod.GET)
public String Evaluacion(HttpServletRequest request,Locale locale, Model model) {

    String usuario=(String) request.getAttribute("usuario");
    model.addAttribute("usuario",usuario);


    return "Evaluacion";
}
```

Figura 48. Evaluacion HomeController.

Obtiene el usuario de la sesión y los encapsula en un objeto de tipo model que se lo pasa a la vista, y es dentro de ésta donde se hace una petición utilizando la tecnología en AJAX para que los datos esten serializados en JSON.

```
function request(){
    $.ajax({
          url: "/tfg/DatosEvaluacion",///http://808
          data: {
               txtNickname: "${paciente.nickname}"
           },
           type: "GET",
           dataType: "json",
           success: function( result ) {
               //result tiene un json id y valor qu
             console.log(result);
             $('#datosjson').text(result.pulso)
             //la funcionalidad push guardar el obj
             valores.push(result.pulso);
             valores1.push(result.saturacion);

           }
         });
};
setInterval(() => {
    request();
    chartJs(valores,valores1);
},5000);
</script>
```

Figura 49. Peticion AJAX.

Lo que hace esta función es que llama internamente dentro de la vista a este método ¨DatosEvaluacion¨ que será el que le devuelva los datos serializados en JSON.

```
@RequestMapping(value = "/DatosEvaluacion", method = RequestMethod.GET)
public @ResponseBody PulsoDTO DatosEvaluacion(HttpServletRequest request,Locale locale, Model model,@RequestParam("txtNickname") String nickname) {

    PulsoDTO valor=dao.BuscarDatos(nickname);
    return valor;
}
```

Figura 50. DatosEvaluacion.

Se observa que realiza una peticion a un método dao para obtener el id y el valor.Se inserta como parametros de entrada el nombre del usuario de la sesion, se introduce en el metodo de la base de datos para obtener la ultima medicion de los parametros biomedicos y debido a que la interfaz Serializable del DTO es capaz de parsear los datos a JSON que es lo que devuelve a la vista, la cual se encargara de situar tanto el pulso como la saturación de oxígeno en la sangre en la gráfica.

4.3.6. phpMyAdmin.

Para este proyecto se utilizará phpMyAdmin como herramienta principal para alojar una base de datos de gestión de usuarios en el que pueda ser monitorizada por el administrador del sistema.

Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 72 idiomas. Se encuentra disponible bajo la licencia GPL Versión 2 [13].

Este proyecto se encuentra vigente desde el año 1998, siendo el mejor evaluado en la comunidad de descargas de SourceForge.net como la descarga del mes de diciembre del 2002. Como esta herramienta corre en máquinas con Servidores Webs y Soporte de PHP y MySQL, la tecnología utilizada ha ido variando durante su desarrollo.

Las principales características de phpMyAdmin son:

- Interfaz web intuitiva.
- Soporte de la mayoría de características de MySQL.
- Explorar, eliminar bases de datos, tablas, vistas, campos e índices.
- Crear, copiar, eliminar, renombrar y modificar bases de datos, campos e índices.
- Mantenimiento de servidor, bases de datos y tablas, de cara a la configuración del servidor.
- Ejecutar, editar y marcar cualquier instrucción SQL, incluso peticiones por lotes.
- Administrar procesos almacenados.
- Importar datos desde archivos CSV y SQL.

Una vez que se ha estructurado cada una de las características de phpMyAdmin, nos disponemos a definir la importancia que tiene MySQL en este proyecto, ya que MySQL es un sistema de gestión de base de datos relacional. Posee una licencia publica general/licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web [14].

Esto ofrece una gran versatilidad a nuestro proyecto, ya que, si se pretende hacer un hincapié general en el hecho de desarrollar mejoras futuras, migrar el proyecto, ofrecer una accesibilidad óptima, este sistema de gestión de bases de datos estará más encauzada a realizar este tipo de procesos que otros sistemas que se basen en un software privado.

Las ventajas que presenta este software son las siguiente:

- Es OpenSource.
- Velocidad al realizar operaciones, lo que hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una maquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de Sistemas Operativos.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet.

Algunas de sus desventajas son:

- Un gran porcentaje de las utilidades de MySQL no están documentadas.
- No es intuitivo, como otros programas (ACCESS).
- Un gran porcentaje de las utilidades de MySQL no están documentadas.

Una vez llegado a este punto donde tenemos claro que interfaz vamos a utilizar (phpMyAdmin) y que sistema de gestión de base de datos (MySQL), debemos nombrar la importancia de interactuar con un servidor independiente que estará alojado en nuestro ordenador en local. Este es el caso de XAMPP:

4.3.7. XAMPP.

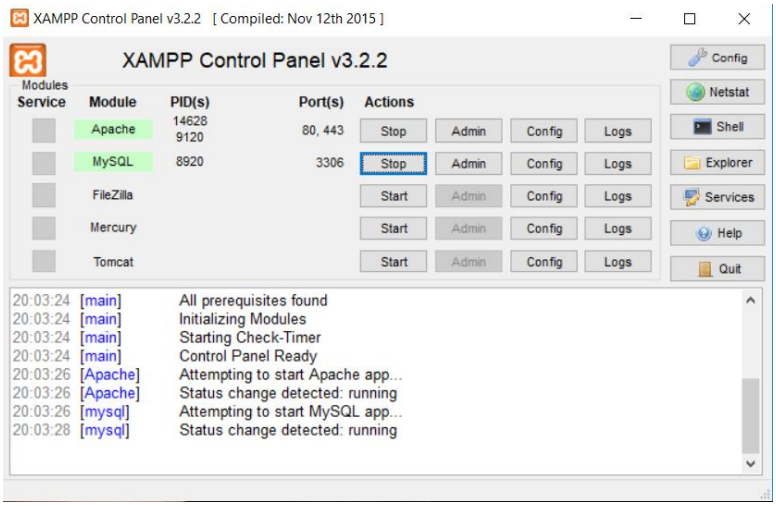

Figura 51. XAMPP.

Es un paquete de software libre que consiste principalmente en albergar el sistema de gestión de base de datos, que en nuestro caso es MySQL, el servidor web Apache y los interprete para lenguajes de script PHP y Perl.

Como se observa en la imagen, se ha inicializado tanto Apache como MySQL por medio de los llamados “puertos bien conocidos”, que son el puerto 80 y el puerto 3306 respectivamente.

Una vez inicializados estos procesos, lo último que cabe mencionar es el IDE de desarrollo de la aplicación web, que será Eclipse.

4.3.8. Eclipse.

Es una plataforma software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados como el IDE de Java llamado Java Develpment Toolkit (JDK) y el compilador (ECJ) que se entrega como parte de Eclipse [15].

La base para Eclipse es la Plataforma de cliente enriquecido (del inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins.
- OSGi - una plataforma para bundling estándar.
- JFace - manejo de archivos, manejo de texto, editores de texto.
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes.

En cuanto a las aplicaciones cliente, Eclipse provee al programador con frameworks de distintos lenguajes de programación como Java para el desarrollo de aplicaciones gráficas.

Para entender este concepto de framework, hay que tener claro el lenguaje de programación que se va a emplear, para este proyecto, la aplicación se ha desarrollado en Java, ya que posee diferentes características como ser un lenguaje concurrente, de propósito general, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

4.3.9. Java.

Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en el año 1991. El equipo (Green Team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road, en Menlo Park, en su desarrollo [16].

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarse Green tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas, y finalmente se renombró como Java [16].

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++ [16].

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture.

La primera característica, orientado a objetos (“OO”), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos) [16].

El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa.

Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean

fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos.

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo.

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode" (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino [16].

El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time) [16].

Es necesario nombrar que la sintaxis de java se compone de lo siguiente:

- Todo en Java está dentro de una clase, incluyendo programas autónomos.
- El código fuente se guarda en archivos con el mismo nombre que la clase que contienen y con extensión ".java". Una clase (class) declarada pública (public) debe seguir este convenio. Si tenemos una clase llamada Hola, su código fuente deberá guardarse en el fichero "Hola.java".
- El compilador genera un archivo de clase (con extensión ".class") por cada una de las clases definidas en el archivo fuente. Una clase anónima se trata como si su nombre fuera la concatenación del nombre de la clase que la encierra, el símbolo "$", y un número entero.
- Los programas que se ejecutan de forma independiente y autónoma, deben contener el método" main.
- La palabra reservada "void" indica que el método main no devuelve nada.
- El método main debe aceptar un array de objetos tipo String. Por acuerdo se referencia como "arg", aunque puede emplearse cualquier otro identificador.

- La palabra reservada “static” indica que el método es un método de clase, asociado a la clase en vez de a una instancia de la misma. El método main debe ser estático o’’de clase’’.
- La palabra reservada public significa que un método puede ser llamado desde otras clases, o que la clase puede ser usada por clases fuera de la jerarquía de la propia clase. Otros tipos de acceso son” private ” o” protected ”.
- La utilidad de impresión (en pantalla, por ejemplo) forma parte de la biblioteca estándar de Java: la clase ‘’’System’’’ define un campo público estático llamado ‘’’out’’’. El objeto out es una instancia de ‘’’PrintStream’’’, que ofrece el método ‘’’println (String)’’’ para volcar datos en la pantalla (la salida estándar).
- Las aplicaciones autónomas se ejecutan dando al entorno de ejecución de Java el nombre de la clase cuyo método main debe invocarse. Por ejemplo, una línea de comando de la forma java –cp. Hola ejecutará el programa del ejemplo (previamente compilado y generado “Hola.class”). El nombre de la clase cuyo método main se llama puede especificarse también en el fichero “MANIFEST” del archivo de empaquetamiento de Java (.jar) [16].

Para ahorrarnos tiempos de compilación en el código, líneas de programación innecesarias, cargas de buffer, etc.… se ha considerado el hecho de implementar la aplicación web por medio de un framework. Como se ha mencionado con anterioridad y una vez comprendido cual es la definición de lenguaje de programación, como por ejemplo Java, hay que destacar que un framework no es más que un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

En el caso de Java, uno de los frameworks más utilizados es Spring utilizando JDBCTemplate.

Las razones por las que se utiliza un framework son las siguientes:

- Evitar escribir código repetitivo. En concreto para el acceso a la base de datos se resumen en tres simples líneas de codigo que si no utilizamos JdbcTemplate.

Por ejemplo, si no se utilizase JdbcTemplate, a parte de que pesaria mas la aplicación, tendria un retardo mayor que utilizando JdbcTemplate, asi , una simple conexión con la base de datos que se ve a continuación:

```
{
getConnect();
Statement stm = null;
try{
            …
            …
            …
}catch (Exception e) { System.out.println(e); }
    finally{
              if (stmt!=null)  {
                        try{ stmt.close(); } catch(SQLException e){e.printStackTrace();}          }
             if (conn!=null)  {
                        try{ conn.close(); } catch(SQLException e){e.printStackTrace(); }          }
    }
}
```

Figura 52. Conexión a la base de datos.

Se puede resumir así utilizando JdbcTemplate:

```
public List<Usuario> leeUsuarios(){
    String sql = "select * from usuarios";
    UsuarioMapper mapper = new UsuarioMapper();
    List<Usuario> usuarios =  this.jdbcTemplate.query(sql, mapper);
    return usuarios;
}
```

Figura 53. Conexión a base de datos con JdbcTemplate.

- Utilizar buenas prácticas. Los frameworks estan basados en patrones de desarrollo, normalmente MVC (modelo-vista-controlador) que ayudan a separar los datos y la lógica de negocio de la interfaz con el usuario.

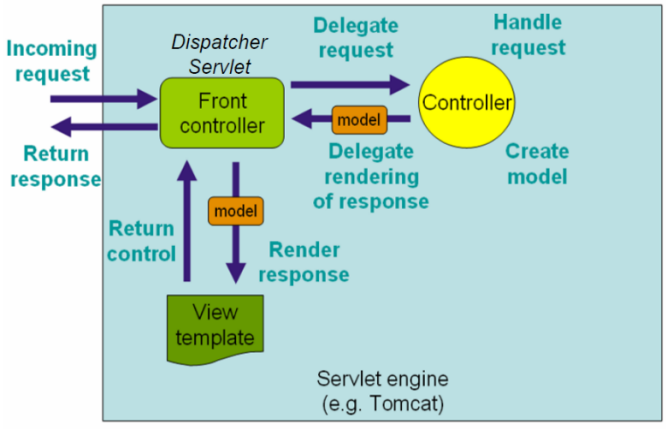

Figura 54. Patrón MVC.

4.3.10. MVC.

Este patrón MVC está desarrollado en tres capas principalmente que tienen independencia por sí mismas pero que interaccionan entre ellas.

Estas tres capas son:

- Modelo. Esta es la capa donde se trabaja con los datos , por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán las correspondientes sentencias SQL.

```
@Repository
public class UsuarioDAOJdbc implements UsuariosDaoInterface {
    private JdbcTemplate jdbcTemplate;
    private DataSource dataSource;
    @Autowired
    public void setDataSource(DataSource dataSource) {
        this.dataSource = dataSource;
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }
```

Figura 55. UsuarioDAOJdbc.

Deben implementar una interfaz donde se encuentran todos los métodos que se van a utilizar para la conexión con la base de datos [17].

```
public interface UsuariosDaoInterface {

    public List<UsuarioDTO>leeUsuarios();
    public void NuevoUsuario(UsuarioDTO usuario);
    public UsuarioDTO LeerNickname(String Nickname);
    public void BorrarUsuario(String nickname);
    public UsuarioDTO LeerCredenciales(String nickname, String contraseña);
    public void ActualizarDatos(String edad, String sexo, String peso, String contraseña, Boolean fumador, String nickname);
    public void InsertarDatos(PulsoDTO datos, String nikcname);
    public PulsoDTO BuscarPulso(String Nickname);
}
```

Figura 56. InterfazDao.

- Vista. Contienen el código de nuestra aplicación que va a producir la visualizacion de las interfaces de usuario, es decir, el código que nos permitirá renderizar los estados de nuestra aplicación HTML.

Generalmente, las vistas requerirán elementos especiales para acceder a los atributos del objeto request desde un JSP tranto de evitar asimismo el uso de código Java: Este es el caso de las etiquetas ´EL´.

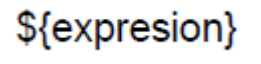

Figura 57. Etiqueta EL.

Una expresion EL puede devolver cualquier tipo de objeto. Si el objeto tiene una determinada propiedad se pueden usar los operadores [ ] ó . para acceder a ella.

Además del acceso a los atributos del objeto request, EL también le ofrece al desarrollador su propio conjunto de objetos implícitos. Algunos de ellos:

1. param: un mapa conteniendo todos los parámetros de la petición HTTP El nombre de los parámetros forma la clave del mapa. Los valores del mapa están formado por el primer valor de cada parámetro.
2. paramValues: a diferencia del anterior, el valor para cada clave del mapa está formado por un array de strings con todos los posibles valores.
3. header: un mapa con las cabeceras de la petición HTTP. Para cada cabecera solo almacena el primero de sus valores.
4. headerValues: de forma análoga, usado en caso de querer acceder a múltiples valores de la cabecera de la petición HTTP.
5. cookie: un mapa con todas las cookies de la petición. Cada valor del mapa es un objeto de tipo Cookie [17].

Las declinación por utilizar la tecnologia de JSP y no otro tipo de archivos que puedan ser indexados en la vistas es que, JSP muestra las siguientes ventajas frente a otras tecnologias:

1. Contra Active Server Pages (ASP).
   ASP es una tecnología similar de Microsoft. Las ventajas de JSP estan duplicadas. Primero, la parte dinámica está escrita en Java, no en Visual Basic, otro lenguaje específico de MS, por eso es mucho

más poderosa y fácil de usar. Segundo, es portable a otros sistemas operativos y servidores Web.

2. Contralos Servlets.

   JSP no nos da nada que no pudierámos en principio hacer con un servlet. Pero es mucho más conveniente escribir (y modificar!) HTML normal que tener que hacer un billón de sentencias println que generen HTML. Además, separando el formato del contenido podemos poner diferentes personas en diferentes tareas: nuestros expertos en diseño de páginas Web pueden construir el HTML, dejando espacio para que nuestros programadores de servlets inserten el contenido dinámico.

3. Contra JavaScript.

   JavaScript puede general HTML dinámicamente en el cliente. Este una capacidad útil, pero sólo maneja situaciones donde la información dinámica está basada en el entorno del cliente. Con la excepción de las cookies, el HTTP y el envió de formularios no están disponibles con JavaScript. Y, como se ejecuta en el cliente, JavaScript no puede acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etc [17].

- Controlador. Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar elemento, visualizar los datos del paciente para ver su evolucion temporal, entre demás funciones.
  Es una capa que sirve de enlace entre las vistas y los modelos, repondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación.

  No obstante, su responsabilidad no es manipular directamente datos, ni mostrar ningun tipo de salid, sino servir de enlace entre los modelos y las vistas.

  En este trabajo se ha establecido un controlador principal que sera el encargado de direccionar todas las ordenes provenientes del usuario y enlazarlo con distintos metodos.

  Para ellos se ha uso de una anotcion: @Controller , que sera la encargada de decirle al framework que esa clase cumplira la accion de Controlador.

  Se puede ver en este fragmento de código:

```
1
2 @Controller
3 public class HomeController {
```

Figura 58. HomeController.

- Desarrollar mas rapido y eficiente. Si tenemos en cuenta los puntos anteriores, sabremos que desarrollar una aplicación con un framework nos permite hacer más rápido, más limpio y más seguro.
  Para estre proyecto se ha utilizado el framework de Spring, ya que esta basada en el patrón MVC explicada con anterioridad y utiliza Java como lenguaje de programación.

4.3.11. Spring.

El Framework Spring es una plataforma Java que da soporte al desarrollo de aplicaciones. Hoy en día es de los Frameworks de java más usados y demandados.

• Ofrece un modelo de programación basado en el uso de POJOs - plain

old Java objects -, siendo más ligero en comparación con otras

alternativas.

• Cualquier aplicación en Java se puede beneficiar de Spring en términos

de sencillez, testeabilidad y un acoplamiento ligero entre componentes.

• El núcleo de Spring está basado en técnicas conocidas como inyección

de dependencias e inversión de control [18].

La inyeccion de dependencias de Spring es un patrón de diseño que sirve para “inyectar” componentes a las clases que tenemos implementadas. Esos componentes son contratos que necesitan nuestras clases para poder funcionar, de ahí el concepto de “dependencia”. La diferencia sustancial en este patrón de diseño es que nuestras clases no crearán esos objetos que necesitan, sino que se les suministrará otra clase “contenedora” perteneciente al Framework DI que estemos utilizando y que inyectarán la implementación deseada a nuestro contrato, y todo ello sin tener que hacer un solo “new”.

Supongamos dos componentes, A y B, donde A depende de B:

```
public class A{
    public void metododeA( ) {
            B b = …. // instancia de B
            b.metododeB( );
            ….
    }
    ….
}
```

Figura 59. Ejemplo DI.

También se puede decir que B es una dependencia de A, y A debe obtener una instancia de B para usarlo.
Si B es una interfaz con distintas implementaciones, A se ve obligado a escoger una implementación, reduciendo así su propia reusabilidad.

• Spring es modular, permitiendo usar sólo aquellas partes que se necesiten para el proyecto en concreto

Algunos de los módulos disponibles en Spring:

- Spring Core

- Spring AOP (programación orientada a aspecto)

- Spring MVC Web

- Spring Data (relacionales y no-relacionales)

- Spring Cloud

- Spring Social

- Spring Security

La arquitectura principal en la que se sustenta Spring es la siguiente:

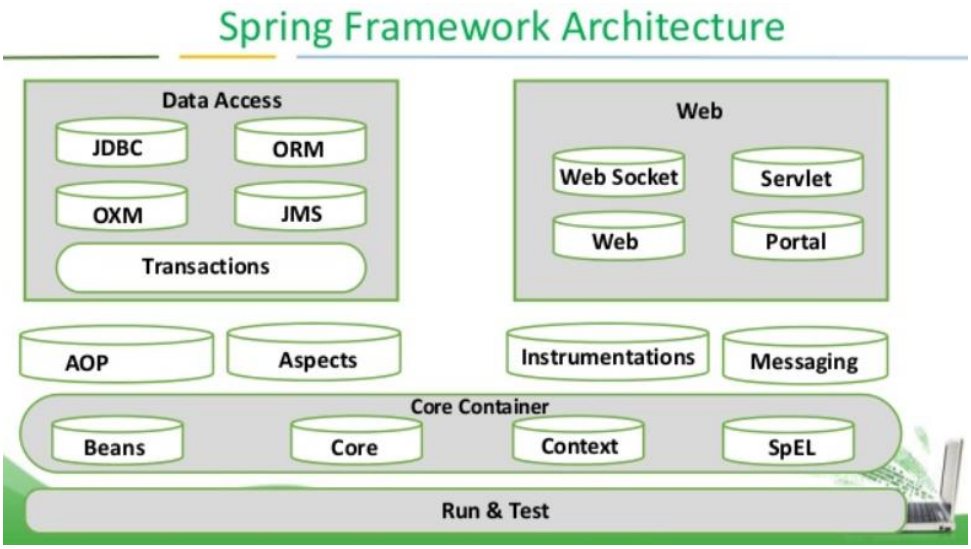

Figura 60. Arquitectura de Spring.

Cada uno de los módulos que tiene Spring son:

- Core: La parte fundamental de este framework es el módulo Core, y los adyacentes Bean y Context. Proveen toda la funcionalidad para la inyección de dependencias, permitiéndole administrar la funcionalidad del contenedor de Beans. Además, también proveen de los servicios Enteprise como JDNI, EJB, …
- AOP: Se trata de un módulo que nos permitirá utilizar el paradigma de Programación Orientada a Aspectos (Aspect Oriented Programming). Este paradigma permite mejorar la modularización y separar las responsabilidades. De esta forma, podemos separar las funcionalidades comunes, que se utilizan transversalmente a lo largo de toda la aplicación, de aquellas que son propias de cada módulo.
- Data: Se trata de un gran módulo, formado por múltiples submódulos, y que nos permite simplificar el acceso y persistencia de datos. Spring Data nos proporciona soporte para usar base de datos relacionales (JDBC), ORMs (como por ejemplo JPA, Hibernate, …) e incluso modelos de persistencia NoSQL (como por ejemplo, MongoDB).
- Web: Este módulo nos permitirá implementar el patrón Modelo-Vista-Controlador (MVC) de una manera sencilla y limpia, haciendo uso de forma transparente también de otros patrones de diseño, como FrontController. De esta forma, podemos separar limpiamente la lógica de negocio de la presentación de los datos y el acceso a los mismos. Además de aplicaciones que implican el uso de

vistas y formularios, también podremos crear servicios web (por ejemplo, al estilo REST) de una forma sencilla y rápida.

Nuestro proyecto en Spring cotendran distintos archivos que se describen a continuacion:

Pom.xml. Es un archivo de configuracion del proyecto donde se haran uso de todas las librerias, vease el ejemplo.

```
        </dependency>
        <!-- Libreria para importar mysql -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
        </dependency>
        <!-- Libreria para importar jdbc -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>3.1.0.RELEASE</version>
        </dependency>
        <!-- Permite el uso de JSON -->
        <dependency>
            <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-mapper-asl</artifactId>
            <version>1.9.13</version>
        </dependency>
    </dependencias>
```

Figura 61. Pom.xml.

Servlet-context.xml. Es el necesario para establecer una configuracion a nivel local de nuestra base de datos. Su estructura es: ¨jdbc:mysql://localhost/nombre_basededatoss¨

```
<!-- CREACION DEL BEAN DATASOURCE PARA CONEXION CON LA BASE DE DATOS -->
<beans:bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <beans:property name="driverClassName"
        value="com.mysql.jdbc.Driver" />
    <beans:property name="url"
        value="jdbc:mysql://localhost/tfg" />
    <beans:property name="username" value="root" />
    <beans:property name="password" value="" />
</beans:bean>
```

Figura 62. Servlet-context.xml.

Una vez que se ha definicion el framework que se va a utilizar, pasamos a explicar la interaccion en el flujo de informacion que se produce entre el NodeMCU y la Aplicación Web.

Cuando NodeMCU envía la informacion , se ve puede visualizar en la figura 8, se muestra que redirecciona a un método llamada ¨Evaluacion¨.

En este punto, el HomeController lo recoge de la siguiente manera:

```
//evaluacion proveniente del arduino
@RequestMapping(value = "/Evaluacion", method = RequestMethod.POST)
public ResponseEntity<Void> creardatosprueba(@RequestParam("pulso") float pulso, @RequestParam("saturacion") int saturacion){
    PulsoDTO datos= new PulsoDTO(pulso,saturacion);
    System.out.println("El pulso es:"+datos.getPulso()+"La saturacion es:"+datos.getSaturacion());
    dao.InsertarDatos(datos,nickname_global);
    ResponseEntity<Void> respuestaHTTP = new ResponseEntity<Void>(HttpStatus.CREATED);

    return respuestaHTTP;

}
```

Figura 63. Recogida datos en Spring.

La función que realiza este método es, obtener los datos provenientes del sensor y junto con una variable externa llamada ¨nickname_global¨ que es la encargada de diferenciar cada usuario en la aplicación, almacenarlo en la base de datos a continuacion manda un mensaje al servidor con un estado doscientos uno indicando que se ha creado con éxito.

Como parametros de entrada a este metodo, se ofrecen las variables de pulso y saturacion que son las que se envia por medio del microcontrolador.

Debido a que son procesos que se envían de manera asíncrona, el HomeController seguira su flujo de datos en cada una de las vistas, por esto, y cuando el usuario acceda con éxito al proceso de registro, contemplará la siguiente vista que será la página principal de la aplicación:

4.3.12. JSON vs XML.

Cabe recalcar que JSON presenta una serie de ventajs con respecto a XML ya que su formato es mucho mas sencillo ya que solo utiliza pares de nombre-valor delineados de manera concisa, en cambio, XML obliga a detallar el uso de llaves.

JSON presenta un menor tamaño lo que proporciona una mayor viabilidad en proyectos grandes donde se lleven a cabo una cantidad elevada de lineas de código, pero la gran ventaja que nos ofrece es que es un subconjunto de Javascript, por lo que el codigo para analizar y elpaquete se ajusta de forma muy natural al código Javascript [19].

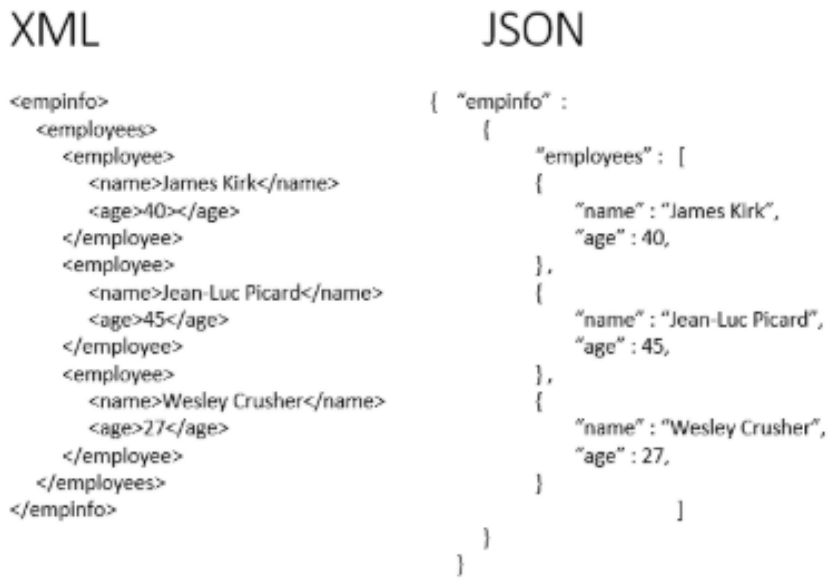

Figura 64. JSON vs XML.

Como se ha mencionado anteriormente, la funcionalidad que tendrá la página

mediante la petición AJAX será la de ir recargando la página de forma asíncrona con respecto a la información que le llega del NodeMCU, debido a esto se hace una subconsulta dentro de ese método que es la siguiente:

```
public PulsoDTO BuscarDatos(String Nickname) {
    String sql="SELECT * FROM datos WHERE Id = (SELECT MAX(id) FROM `datos` WHERE Nickname_usuario = ?)";
    Object[] parametros = {Nickname};
    PulsoMapper mapper = new PulsoMapper();
    List<PulsoDTO> pulso = this.jdbcTemplate.query(sql, parametros,mapper);
    if(pulso.isEmpty()) return null;
    else return pulso.get(0);
}
```

Figura 65. Subconsulta SQL.

Se sustenta en la funcionalidad de que, debido a un ¨Nickname¨ que será la variable de entrada que se le proporcionará al método, se produce una búsqueda del útlimo ¨Id¨ que se ha introducido en la base de datos y el ¨pulso, saturacion¨ asociado a ese id donde el nombre de usuario coincida.

Con esto se consigue la independencia de poder enviar parámetros al servidor y que éste cada ciertos segundo coja el último valor que se ha introducido en la base de datos dependiendo del usuario.

El flujo final de este proceso en la aplicación quedaria de la siguiente manera:

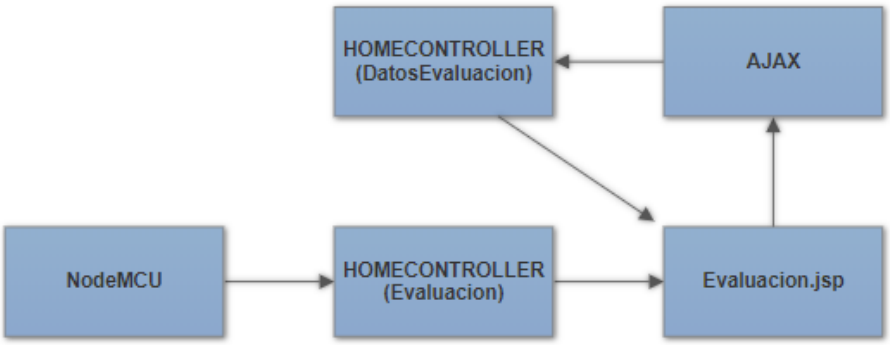

Figura 66. Flujo final.

5. Discusión de los resultados.

Una vez estructurado el funcionamiento del proyecto por medio de su composición vía hardware y software, es necesario testear la viabilidad de este proyecto en un entorno real.

Para ello, se ha dispuesto a realizar distintas pruebas mediante un pulsioxímetro real utilizado en varios centros hospitalarios.

Figura 67. Pulsioxímetro real.

Se ha querido comparar las medidas que ofrece este proyecto desarrollado con uno real y estableciendo un margen de mejora.

Para la realización de las pruebas se ha introducido en un dedo del paciente el pulsioximetro real y en el otro dedo se ha puesto sobre el sensor MAX30100.

El porcentaje de error de las medias se ha obtenido de la siguiente manera: El dato obtenido menos el dato del patrón entre el dato del patrón por cien,

Constatará un tema importante para llevar a cabo la robustez del desarrollo de este prototipo.

El oximetro de pulso se real, se encuentra de forma accesible para los pacientes en un centro hospitalario, ya que se ha querido comprobar la posible mejorar que acarearía el uso de este prototipo desarrollado.

Finalmente comentar que, el paciente debe estar en un estado de reposa para conseguir las mediciones optimas del prototipo.

No debe hacerse movimientos bruscos que puedan desviar las mediciones biomédicas del prototipo, y que esto conlleve, a una mala obtención de los datos.

Los dos parámetros que se van a considerar son:

- BPM. Frecuencia cardíaca (pulsos por minuto).
- SPO2(%). Niveles de saturación de oxígeno en la sangre.

5.1. Resultados.

A continuación, se muestran cada uno de los resultados obtenidos con el prototipo desarrollado y uno real

	Muestras cada 5 segundos Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 1 Persona joven.	61	95	72	96
	68	94	73	96
	77	95	74	97
	73	94	74	97
	81	96	75	97
	76	95	77	97
	82	96	78	97
	79	94	81	97
	76	95	77	97
	77	94	79	96
	75	96	78	94
	Media: 75	Media: 95	Media: 76	Media: 97

	Muestras cada 10 segundos. Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 1	64	95	71	96
	69	97	73	98
	72	97	72	96
	64	96	70	98
	64	97	70	97
	67	96	72	96
	Media: 69	Media: 96	Media: 71	Media: 97

	Muestras cada 20 segundos. Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 1	64	95	71	97
	71	96	70	97
	62	96	67	95
	Media: 68	Media: 96	Media: 69	Media: 96

	Muestras cada 5 segundos Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 2 Adulta.	61	95	72	96
	68	94	73	96
	77	95	74	97
	73	94	74	97
	81	96	75	97
	76	95	77	97
	82	96	78	97
	79	94	81	97
	76	95	77	97
	77	94	79	96
	75	96	78	94
	Media: 76	Media: 95	Media: 76	Media: 96

	Muestras cada 10 segundos. Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 2 Adulta.	69	95	71	96
	69	97	73	96
	72	97	72	96
	68	96	70	97
	68	97	70	97
	67	96	72	96
	Media: 69	Media: 97	Media: 70	Media: 96

	Muestras cada 20 segundos. Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 2 Adulta.	64	95	71	97
	71	96	70	97
	62	96	67	95
	Media: 68	Media: 96	Media: 69	Media: 96

	Muestras cada 5 segundos Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 3 Fumador.	61	95	72	96
	68	94	73	96
	77	95	74	97
	73	94	74	97
	81	96	75	97
	76	95	77	97
	82	96	78	97
	79	94	81	97
	76	95	77	97
	77	94	79	96
	75	96	78	94
	Media: 75	Media: 95	Media: 76	Media: 96

	Muestras cada 10 segundos. Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 3 Fumador.	64	95	71	96
	69	97	73	96
	72	97	72	96
	64	96	70	97
	64	97	70	97
	67	96	72	96
	Media: 66	Media: 96	Media: 68	Media: 96

	Muestras cada 20 segundos. Tiempo total de obtención de resultados: 1min.			
	PROTOTIPO		OXYM4000	
	BPM	SPO2 (%)	BPM	SPO2 (%)
Usuario 3 Fumador.	64	95	71	97
	71	96	70	97
	65	96	67	95
	Media: 67	Media: 96	Media: 69	Media: 96

Se han obtenido los parámetros en intervalos de tiempo cada cinco, diez, veinte y treinta segundos en un minuto, que se ha estimado que es el correcto para una estimación normal de los datos.

Porcentaje de error (%)		
Usuario 1. Joven.	BPM	Saturación de oxígeno en sangre
Datos cada 5 segundos	1	2
Datos cada 10 segundos	2.8	1
Datos cada 20 segundos	1.44	0

Tabla 2. Porcentaje de error Usuario 1.

Porcentaje de error (%)		
Usuario 2. Adulta.	BPM	Saturación de oxígeno en sangre
Datos cada 5 segundos	1	1
Datos cada 10 segundos	2.8	1
Datos cada 20 segundos	1.44	0

Tabla 3. Porcentaje de error Usuario 2.

Porcentaje de error (%)		
Usuario 3. Fumador.	BPM	Saturación de oxígeno en sangre
Datos cada 5 segundos	1.31	1
Datos cada 10 segundos	2.9	1
Datos cada 20 segundos	2.9	0

Tabla 4. Porcentaje de error Usuario 3.

6. Conclusiones.

Se puede llegar a la conclusión de que el margen de error producido entre el prototipo desarrollado para este trabajo y el pulsioxímetro real es óptimo ya que se encuentra en un rango inferior al 3%.

Dado el incremento de precios en el mercado y el costo para la realización de este desarrollo tecnológico, es precioso indicar el hecho de que la viabilidad del mismo es positiva y, por lo tanto, se considera un trabajo eficiente.

Una muestra de ello es la comparación con los diferentes modelos del mercado que se pueden observar a continuación.

MODELO		COMUNICACIONES	PRECIO
NONIN WristOx2 3150. [20]		Bluethooth	700 €
866-NONIN3230. [21]		Bluethooth	314€
OXYM7500. [22]		WIFI	250 €
Nonin Go2 LED. [23]		NO	156 €
OXYM6100. [24]		USB	122 €
OXYM6000USB. [25]		USB	63€

Tabla 5. Comparación pulsioxímetros.

Se ha seleccionado una plataforma de software y hardware libre como es Arduino, lo que ha permitido que el dispositivo sea completamente autónomo y modular. El diseño de este prototipo está caracterizado por dos componentes o estructuras como es la funcionalidad electrónica, donde interviene un primer proceso de alimentación externa como puede ser el caso de una pila, para proporcionar al dispositivo la autonomía suficiente como para abastecerse por sí mismo y no dependa de tener una alimentación fija para alimentarse, una captación, recepción e interacción de datos provenientes de un sensor jerarquizados por el NodeMCU que será el microcontrolador principal de la aplicación , responsable de la unión de estos dos campos de la electrónica y la telemática mediante el envío de parámetros por medio de un chip WIFI. Una pantalla LCD 16x2 que le hará saber en todo caso al usuario las pautas que debe seguir para parametrizar y enviar sus datos al servidor. Un adaptador LCD que será el encargado de recoger las órdenes previstas desde el NodeMCU y mostrarlas en la pantalla. Finalmente, el sensor MAX30100 obtendrá los parámetros biomédicos de un usuario.

El componente software encargado de realizar los procesos de la obtención de esos datos en el servidor y poder mostrárselos al usuario para que haga uso de ellos mediante el FrameWork de Spring.

Se han cumplidos los objetivos principales que se marcaron a la hora de diseñar e implementar el prototipo ya que dispone de capacidades de monitorización y procesado de datos.

Respecto a los demás dispositivos existentes en el mercado, se ha demostrado que tiene una serie de características propias y posee un avance de desarrollo tecnológico del mismo para hacerles frente por el elevado costo que tienen, o por no presentar las comunicaciones de este prototipo aun siendo este último más barato.

Teniendo en cuenta que el dispositivo se ha diseñado para una monitorización en tiempo real, se consigue un importante ahorro energético enviando los parámetros biomédicos cada cierto tiempo.

Otra de las conclusiones, es haber conseguido una independencia entre cada uno de los módulos electrónica y software, ya que los datos se envían de manera asíncrona, lo que supondría que no se limita al usuario a obtener la aplicación con su correspondiente elemento hardware, sino que se haría uso de la aplicación vía software y el usuario, si lo desea puede utilizar otro hardware y no se limita a la restricción de tener que comprar el producto que le ofrecemos.

Esto, aparte de una gran autonomía, ofrece otras opciones de suscripción que puede tener el usuario, pero siempre pudiendo obtener, en el caso en el que lo desee, nuestro producto por menos de unos treinta euros.

Por otro lado, la utilidad que presenta el hecho de que venga incorporado un módulo WIFI en el microcontrolador da lugar al envío de datos tanto a dispositivos como puede ser el caso de un ordenador, un móvil, o cualquier dispositivo similar.

Finalmente, lo más importante que se ha conseguido utilizar con este prototipo es llevar a cabo las técnicas necesarias para realizar las funcionalidades de otros dispositivos ya existentes en el mercado actualmente.

7. Líneas futuras.

Para que este proyecto contenga un proceso de desarrollo y posterior avance, se han propuesto las siguientes líneas futuras:

- Hardware:

En esta área, es conveniente reconocer el método de captación y recepción de parámetros provenientes de un sensor para ir acoplándolo al circuito y que envíe distintos parámetros a la aplicación de manera similar a como se ha implementado aquí.

Por ejemplo, haciendo uso de otros sensores como es el caso del Max30102 que mide la saturación de oxígeno, la frecuencia cardíaca, así como otros parámetros como la temperatura, y ofrece una mejora sustancial con respecto al sensor MAX30100 [26].

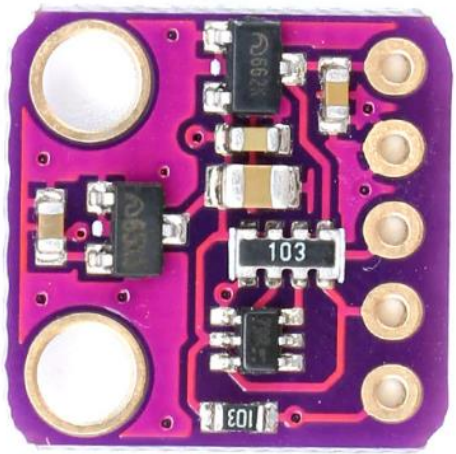

Figura 68. Max30102.

También cabe recalcar el uso que se le puede dar a otro tipo de sensores que se puedan ir adaptando a nuestra placa como es el caso del sensor de ritmo cardíaco de latidos del corazón, ya que proviene de tres tipos de señales de salida que se caracterizarían por: tierra, alimentación y la señal [27].

Figura 69. Sensor latidos del corazón.

Por último, y debido a la multiplicad de pines existentes en el microcontrolador (NodeMCU), se podría sustentar en el contínuo avance de este trabajo acoplando diferentes sensores como los aquí mencionados, u otros que cumplan con un carácter médico-tecnológico e implementar así, un sistema de salud a distancia consistente.

- Software:

En lo referente a los procesos de avance que se le puede dar al proyecto en relación al software empleado se podría mencionar el uso de Cloud Computing, más concretamente Amazon Web Services (AWS).

El cloud computing es un término general que se refiere a la prestación de servicios alojados a través de Internet. Se trata de una tecnología avanzada que hace que todos los archivos, programas e información estén almacenados en Internet, como en una “nube”, siendo completamente irrelevante las capacidades de almacenaje de los ordenadores instalados por el cliente y prescindiendo así de los discos duros.

Amazon Web Services (AWS) es una plataforma de servicios de nube que ofrece potencia de cómputo, almacenamiento de bases de datos, entrega de contenido y otra funcionalidad para ayudar a las empresas a escalar y crecer.

Lanzado oficialmente en 2006, Amazon Web Services ofrece servicios en línea para otros sitios web o aplicaciones del lado del cliente. La mayoría de estos servicios no están expuestos directamente a los usuarios finales, sino que ofrecen una funcionalidad que otros desarrolladores puedan utilizar en sus aplicaciones. Se accede a Amazon Web Services a través de HTTP, utilizando protocolos REST y SOAP. Todos los servicios son facturados en función del uso, pero la forma de uso por la que es medida la facturación varía de un servicio a otro.

Amazon Web Services ofrece herramientas en las siguientes categorías:

Cloud computing: todo lo necesario para la creación de instancias y el mantenimiento o el escalado de las mismas. Amazon EC2 es el rey indiscutible dentro de los servicios de computación en la nube de Amazon.

Bases de datos: distintos tipos de bases de datos pueden permanecer en la nube mediante el servicio Amazon RDS, que incluye distintos tipos a elegir como MySQL, PosgreSQL, Oracle, SQL Server y Amazon Aurora, o Amazon DynamoDB para NoSQL [28].

Creación de redes virtuales: permite la creación de redes privadas virtuales a través de la nube, gracias principalmente al servicio Amazon VPC.

Aplicaciones empresariales: Amazon WorkMail es el servicio de correo empresarial que ofrece Amazon, al que pueden unirse otros servicios como Amazon WorkDocs y Amazon WorkSpaces.

Almacenamiento y gestores de contenido: tipos de almacenamiento diferentes, tanto para archivos con acceso regular, poco frecuente o incluso como archivo. Amazon S3 es el servicio principal, aunque complementan la oferta otros como Amazon Glacier o Amazon EBS.

Es por esto que sería conveniente el hecho de poder subir la información al servidor y mediante un nombre de dominio, sustentar la aplicación en Internet.

Así, no sería necesario hacer uso de XAMPP o TOMCAT ya que estarán subidos en la nube.

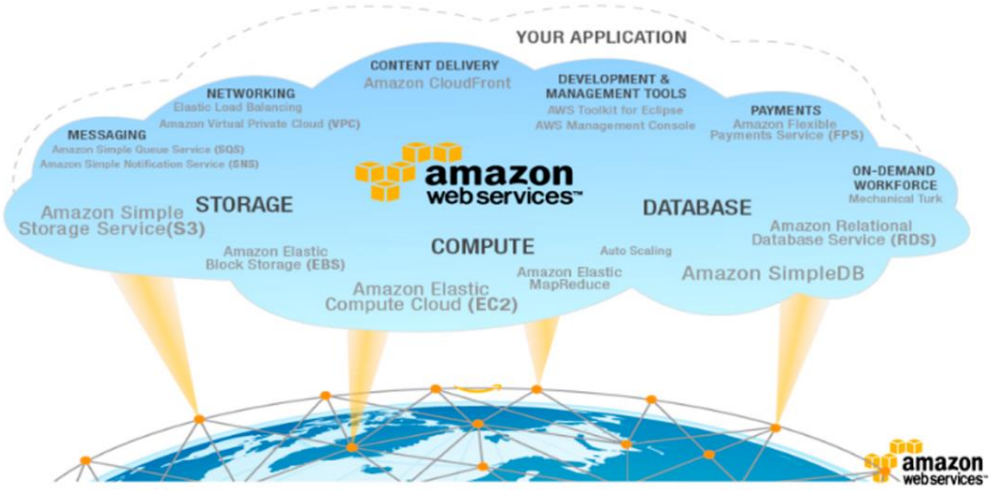

Figura 70. Amazon Web Services.

El término REST se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.

Hoy en día la mayoría de las empresas utilizan API REST para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web.

REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad, pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.

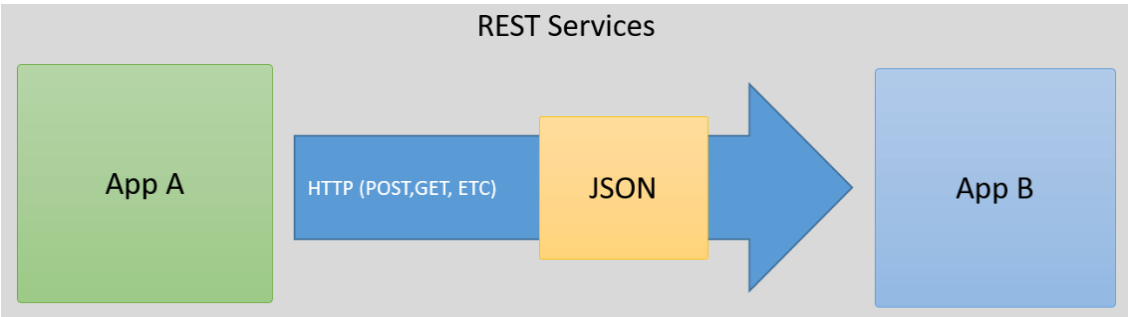

Figura 71. Servicios REST.

8. Bibliografía.

[1] LA VOZ DE GALICIA. *El cáncer de pulmón causa tantas muertes en España como la suma de los de mama, colon y próstata*. [Consulta: 17-11-2017]. Disponible en:

https://www.lavozdegalicia.es/noticia/sociedad/2017/11/17/cancer-pulmon-causa-tantas-muertes-espana-suma-mama-colon-prostata/0003_201711G17P29991.htm

[2] RODRIGO ROJAS. *Hasta el 40 por ciento de las mujeres mexicanas con cáncer de pulmón nunca fumaron.* [Consulta: 02-04-2019]. Disponible en:

https://www.saludiario.com/hasta-el-40-por-ciento-de-las-mujeres-mexicanas-con-cancer-de-pulmon-nunca-fumaron/

[3] NOGUEROL CASADO MJ, SECO GONZALEZ A. *Técnicas en AP: Pulsioximetria*. [Consulta: 4-01-2019]. Disponible en:

https://www.fisterra.com/material/tecnicas/pulsioximetria/pulsioximetria.pdf

[4] DAVID SACEDA CORRALO. *Pulsioximetria*. [Consulta: 15-12-2015]. Disponible en:

https://www.webconsultas.com/pruebas-medicas/como-se-hace-una-pulsioximetria-13530

[5] GUILLERMO PEREZ. *Saturación de oxígeno en sangre*. [Consulta: 20-03-2017]. Disponible en:

https://www.gasometria.com/saturacion_de_oxigeno_en_sangre

[6] GUSTAVO CANTERO. *Plataformas Hardware*. [Consulta: 10-12-2012]. Disponible en:

https://www.programandoamedianoche.com/2012/12/plataformas-de-hardware/

[7] LUIS LLAMAS. *NodeMCU, la popular placa de desarrollo con ESP8266.* [Consulta: 1-06-2018]. Disponible en:

https://www.luisllamas.es/esp8266-nodemcu/

[8] EDUARDO COQUET. *El bus I2C.* [Consulta: 14-10-2017]. Disponible en:

https://www.comunidadelectronicos.com/articulos/i2c.htm

[9] DZDUINO. *Store.* [Consulta 10-01-2019]. Disponible en:

https://www.dzduino.com/capteur-de-frequence-cardiaque-module-capteur-de-pouls

[10] MAXIM INTEGRATED. *MAX30100.* [Consulta: 20-01-2019]. Disponible en:

https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf

[11] SERGIO GONZALEZ COLLADO, LUIS GALICIA. *Generalidades del protocolo http*. [Consulta: 3-02-2019]

https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[12] SISTEMAS MASTER. *Definición de EPROM*. [Consulta: 20-07-2016]. Disponible en:

https://sistemas.com/eprom.php

[13] PHPMYADMIN. *Downloads*. [Consulta: 05-02-2019]. Disponible en:

https://www.phpmyadmin.net/

[14] MYSQL. *Downloads*. [Consulta: 05-02-2019]. Disponible en:

https://www.mysql.com/

[15] ECLIPSE. *Downloads*. [Consulta: 05-02-2019]. Disponible en:

https://www.eclipse.org/ide/

[16] ICTEA. *¿Qué es el lenguaje de programación Java?* [Consulta: 01-02-2019]. Disponible en:

http://cs.ictea.com/knowledgebase.php?action=displayarticle&id=8790

[17] UNIVERSIDAD DE ALICANTE. *Introducción a MVC en Spring*. [Consulta: 26-06-2014]. Disponible en:

http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion03-apuntes.html

[18] SPRING. *Downloads*. [Consulta: 05-02-2019]. Disponible en:

https://spring.io/

[19] REST API TUTORIAL. JSON vs XML. [Consulta: 03-06-2018]. Disponible en:

https://restfulapi.net/json-vs-xml/

[20] NONIN. *Model 3150 oem with bluethooth low energy*. [Consulta: 01-01-2019]. Disponible en:

https://www.nonin.com/products/3150-oem-ble/

[21] PRAXISDIENST. *Nonin Onyx II 950*. [Consulta: 02-02-2019]. Disponible en:

https://www.praxisdienst.es/es/Medica/Emergencia/Monitorizacion/Pulsioximetria/Pulsioximetros/Nonin+Onyx+II+9550+oximetro+de+pulso+de+dedo.html?speed=1&gclid=Cj0KCQiAheXiBRD-ARIsAODSpWOhB5-m9hW90GYVJmR05UQJqaJrh4pU5AX2q4WYbsTGijVyESiOA8MaAlLUEALw_wcB

[22] QUIRUMED. *Pulsioximeter.* [Consulta: 03-02-2019]. Disponible en:

https://www.google.com/search?q=OXYM7500&rlz=1C1CHBF_esES832ES832&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiVwsier6XgAhVGWBoKHTtPARUQ_AUIDygC&biw=1278&bih=1284#imgrc=NO7IO1NTOg6UiM:

[23] BECHDO. *Nonin Go2 led-9571 Pulse oximeter orange*. [Consulta 02-03-2014]. Disponible en:

https://bechdo.in/nonin-go2-led-9571-pulse-oximeter-orange/p/400193

[24] QUIRUMED. *Pulsioximetro de dedo con onda plestimografica , con alarma conecion a pc*. [Consulta: 05-02-2019]. Disponible en:

https://www.quirumed.com/es/pulsioximetro-de-dedo-con-onda-plestimografica-con-alarma.html

[25] QUIRUMED. *Pulsioximetro de dedo con onda plestimografica , conexión USB*. [Consulta: 05-02-2019]. Disponible en:

https://www.quirumed.com/es/pulsioximetro-de-dedo-con-onda-plestimografica-alarma-y-conexion-usb.html

[26] I+D ELECTRONICA. *Sensor de concentración de oxígeno y ritmo cardiaco MAX30102.* [Consulta: 03-02-2019]. Disponible en:

https://www.didacticaselectronicas.com/index.php/sensores/cardiacos/sensor-de-concentraci%C3%B3n-de-ox%C3%ADgeno-y-ritmo-cardiaco-max30102-pulso-cardiaco-pulsiox%C3%ADmetro-pulsioximetr%C3%ADa-gy-max30102-detail

[27] ELECTRON PERDIDO. *Sensor de latido de corazón.* [Consulta: 01-01-2019]. Disponible en:

https://electronperdido.com/shop/liquidacion/sensor-latido-corazon-en-dedo/

[28] AMAZON WEB SERVICES. *Downloads.* [Consulta: 01-01-2019]. Disponible en:

https://aws.amazon.com/es/

9. Presupuestos.

9.1. Análisis de mercado.

El prototipo desarrollado se combina con el servicio comercial, con el objetivo de satisfacer las necesidades primarias de un grupo de población, concretamente, haciendo un énfasis especial en aquellas personas de avanzada edad o que presenten algún tipo de anomalía en el sistema respiratorio.

A pesar de ello, se considera competencia cualquier empresa que desarrolle cierto tipo de prototipos que cumplen estas necesidades.

Las principales empresas competidoras provienen de financiación privada donde el coste de cada uno de los productos que se encuentran actualmente en el mercado, es elevado con respecto a este dispositivo.

9.2. Matriz DAFO.

Como resultado del estudio del mercado realizado se pueden identificar el siguiente conjunto de Debilidades, Amenazas, Fortalezas y Oportunidades.

Fortalezas:

- El equipo encargado del desarrollo del prototipo tiene un alto conocimiento en el sector y su funcionamiento interno.
- Servicio totalmente adaptado al usuario, haciéndolo sencillo y robusto.
- Personal especializado para desarrollar ciertos aspectos tecnológicos en el futuro, sugerencias o problemas de funcionamiento.

Debilidades:

- Implantación como nueva empresa en el mercado. Por lo tanto, acarrea un desconocimiento por parte del público potencial que se pretende dar servicio.
- Poca experiencia en el sector.
- Escasez de recursos en una etapa inicial.

Oportunidades:

- Sector en auge debido al desarrollo del Internet de las cosas y a la política de: “todo conectado”.

- Creciente preocupación debido a las distintas anomalías desde las más leves a las más graves que no se diagnostican a tiempo.
- Posibilidad de ir incorporando más sensores para crear un equipo e-health totalmente equipado.

Amenazas:

- Imitación por parte de la competencia de la actividad o aparición de nuevas empresas que satisfagan necesidades similares.
- Aparición de nuevas técnicas preventivas destinadas a la saturación de oxígeno y la frecuencia cardiaca.

9.3. Precio.

A continuación, se presenta un desglose del precio total del proyecto:

COMPONENTE	MODELO	UNIDADES	PRECIO UNITARIO	PRECIO
Microcontrolador	NodeMCU v3	1	2,07 €	2,07 €
Sensor de pulso	SensorPulse	1	9.87 €	9.87 €
Fuente de alimentación	Alimentación YwRobot	1	1,07 €	1,07 €
Pantalla lcd	Pantalla LCD 16x2	1	1,82 €	1,82 €
Adaptador lcd	Adaptador LCD	1		
Precio total				14,83 €

Tabla 6. Presupuesto del Proyecto.

Se procede a calcular el total, incluyendo el IVA:

Ítem	Precio (€)
Gastos en equipamiento	14.83
IVA (21%)	3.11
Total	17.94

Tabla 7. Precio total.

El precio total es de **DIECISIETE CON NOVENTA Y CUATRO**, un factor constituyente ya que se encuentra por debajo de otros dispositivos sin capacidad de comunicación inalámbrica, y este producto que tiene comunicación inalámbrica la diferencia es bastante notable con respecto a los dispositivos que se encuentran en el mercado con comunicación inalámbrica.

10. Anexos.

10.1. Manuales.

En esta sección se van a describir dos manuales, el primero, de usuario, que describe la utilización de la aplicación, y el segundo, de instalación y mantenimiento, para explicar el funcionamiento del prototipo.

10.1.1 Manual de usuario.

Este manual pretende dar una explicación para la utilización de la aplicación.

Primero hay que iniciar el servidor, para ello hay que poner en el navegador web la siguiente dirección: http://localhost:8080/tfg/ .

Como vista principal de la aplicación se encuentra index.jsp, que muestra el inicio de sesión que suelen tener todas las aplicaciones para que nos identifiquemos con nuestro usuario y contraseña.

Esta vista contiene un flujo de distintas funcionalidades:

- Si el usuario y la contraseña que se introducen ya se encuentran en la base de datos de la aplicación, direccionará directamente a la vista principal de la aplicación.
- Si el usuario y la contraseña que se introducen no coinciden con ningún usuario de la base de datos, se mostrará una vista diciéndonos que ese usuario no existe y que es necesario establecerlo mediante un registro temporal.
- Si el usuario y la contraseña coinciden con el administrador del sistema, se accederá a la vista de mantenimiento y gestión de usuarios, donde se hace referencia a la tabla: usuarios. Aquí, se puede visualizar los usuarios totales registrados en la aplicación, así como darlos de baja de la aplicación por nula utilización del sistema u otros aspectos.

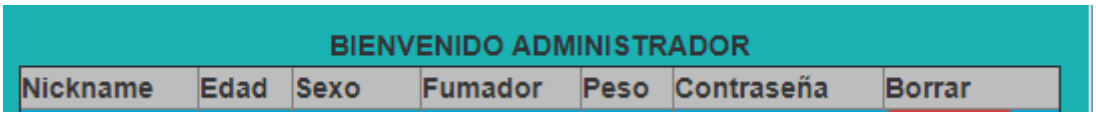

Figura 72. Administrador.

Si directamente no se tiene cuenta, hay que hacer click en el apartado que dice: “Si no tienes cuenta, Haz click aquí “. Se observa a continuación:

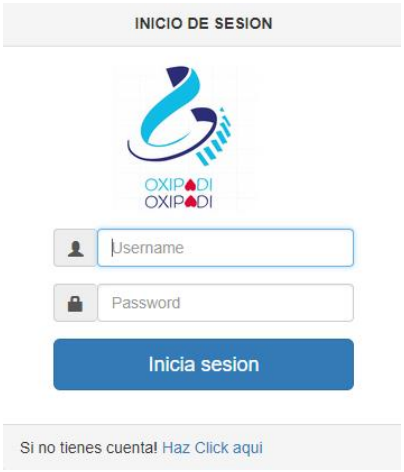

Figura 73. Principal.

En este apartado, tanto si se hace click como si se intenta acceder con un usuario que no existe en la aplicación, la aplicación redirigirá a la siguiente vista.

Figura 74. Registrarse.

Aquí, hay cinco valores que se deben completar para una identificación única de un usuario. El botón de Atrás, hará que se dirija a la vista principal de la aplicación citada anteriormente, y con el botón Enviar, se mandara la información al servidor, se dará de alta en la aplicación y se mostrara la siguiente vista, que es la principal de la aplicación.

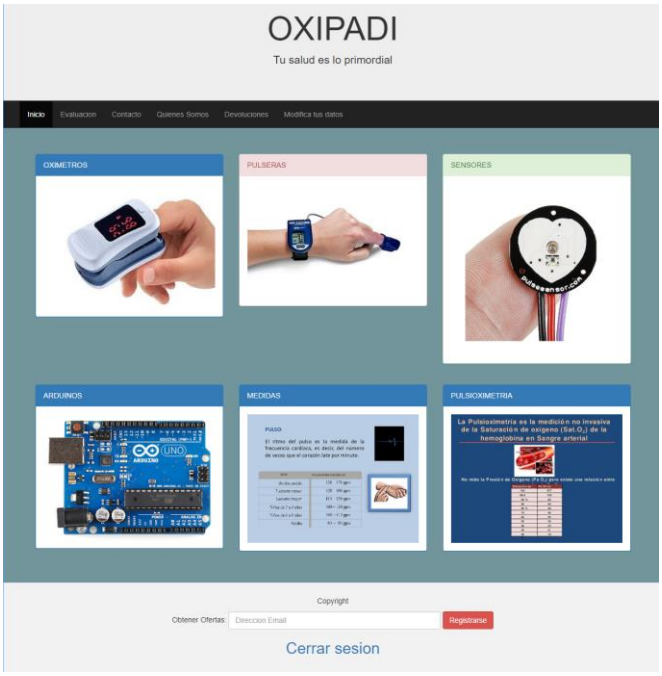

Figura 75. Vista central.

En la vista principal de la aplicación se mostrará varias pestañas.

- Inicio: Es la vista principal en la que se encuentra la aplicación.
- Evaluación: Es la más importante ya que si se hace click se puede observar los datos biomédicos que se están enviando por parte del microcontrolador.

Figura 76. Vista Evaluacion.

- Contacto: Es una vista que muestra al usuario la posibilidad de ponerse en contacto con el desarrollador de la aplicación en el caso de que hubiera algún tipo de problema o sugerencia.

Figura 77. Contacto.

- Quienes somos: Es una que muestra al usuario información con respecto a la empresa que se ha desarrollado la aplicación.

Figura 78. Quienes somos.

- Devoluciones: Si ocurre cualquier tipo de problema y se quiera devolver el producto, aquí se indica cómo hacerlo.

Figura 79. Devoluciones.

- Modifica tus datos: En el caso de que se haya introducido unos datos erróneos, o se quieran actualizar los mismo, se pone a disposición del usuario un apartado dentro de la propia aplicación donde se pueda llevar a cabo estos cambios.

Figura 80. Modifica tus datos.

Finalmente, si se quiere obtener ofertas de cada uno de los productos que se están desarrollando y se lancen próximamente al mercado, para los usuarios registrados

tienen prioridad con respecto al público en general con respecto a las ofertas, así como información anticipada.

El botón de cerrar sesión, finaliza la sesión y se produce el “logout” de la aplicación, por lo tanto, se muestra de nuevo la vista de inicio de sesión de la aplicación.

10.1.2 Manual de instalación y mantenimiento.

El equipo en el que se encuentra alojado el servicio presenta las siguientes características:

- Procesador: Inter i7-8700k
- Gráfica: GTX 1060
- RAM: 16GB
- Sistema operativo: Windows 10 de 64 bits.

Una vez tenemos el equipo, se ha de instalar una máquina virtual Java, JRE, y su entorno de desarrollo, JDK.

Seleccionamos nuestro sistema operativo y descargamos el propio de nuestro sistema, en este caso el de 64 bits. Para que sea compatible con el servidor Apache y el conteneder de Servlets Tomcat.

En este punto hay que declarar unas variables de entorno del sistema, para ello, accedemos al panel de control, dentro del apartado sistemas hacemos click en la opción de configuración avanzada.

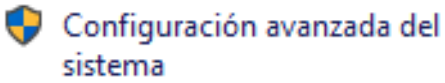

Figura 81. Opciones avanzadas.

En este punto nos saldrá una ventana emergente como esta:

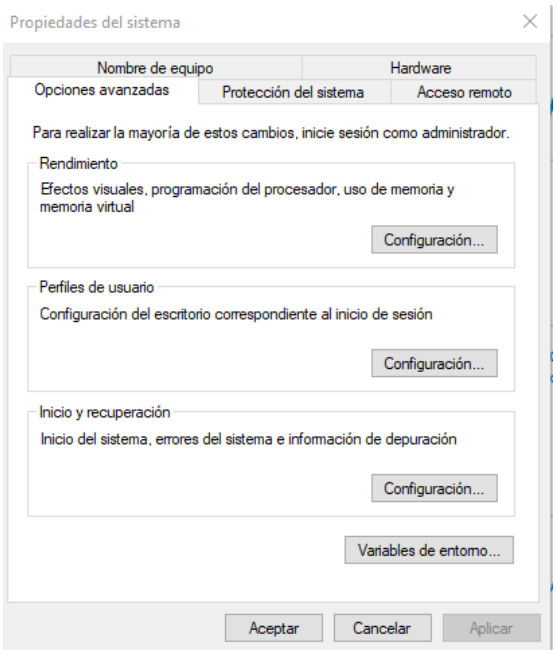

Figura 82. Propiedades del sistema.

Pulsamos ahora el botón de variables de entorno, y buscamos la variable path, le damos a editar y añadimos los directorios del JDK y JRE.

Figura 83. Path.

A continuación, hay que instalar XAMPP, elegimos de nuevo el sistema operativo correcto e instalamos Apache, MySQL, Tomcat y la interfaz gráfica que vamos a utilizar de la base de datos que es phpMyAdmin.

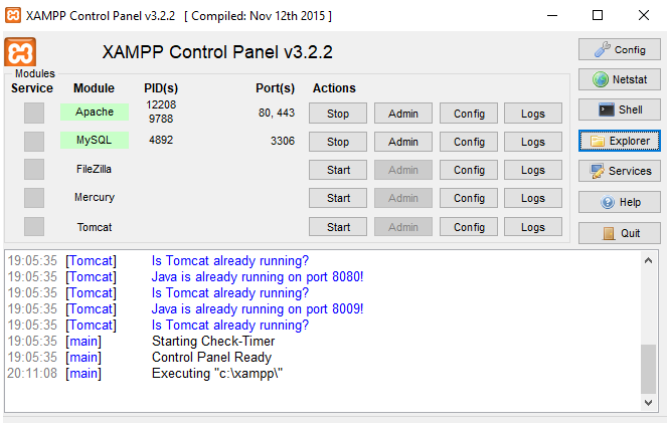

Figura 84. Instalando XAMPP.

Finalmente, instalamos el entorno de desarrollo que será Eclipse.

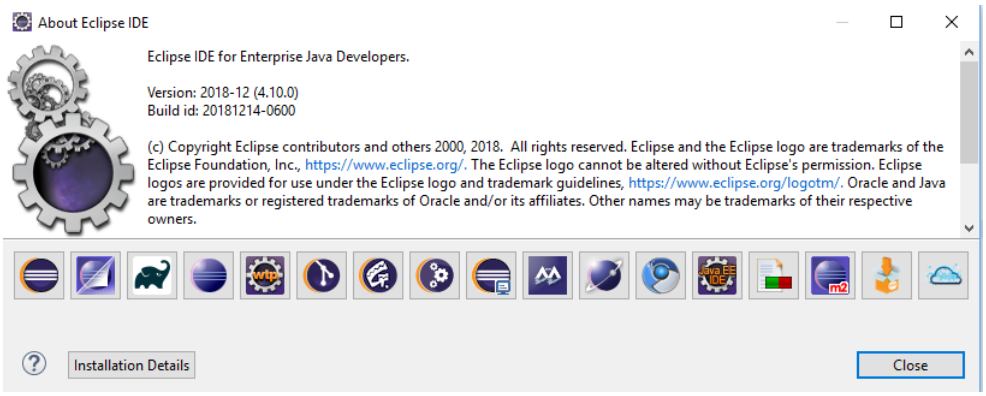

Figura 85. IDE Eclipse.

Se está utilizando el IDE for “Enterprise Java Developers” con la versión 4.10.0 del año 2018.

Dentro del proyecto, y debido a la jerarquización de carpetas, esta dividido en las clases en Java y las vistas.

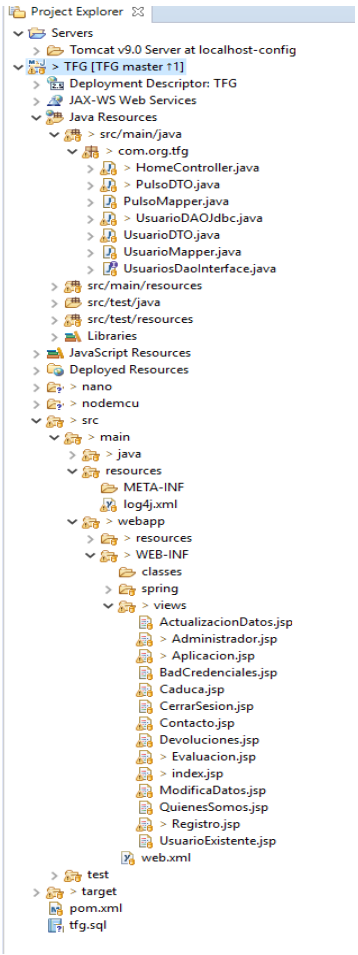

Figura 86. Jerarquización del proyecto.

Se observa que el servidor que se está utilizando es la versión 9.0 de Tomcat.

Cada una de las vistas las tenemos que localizar fuera de la carpeta WEB-INF, para que no se puedan acceder a ellas de manera pública.

Cada uno de los archivos.java estarán dentro del directorio: “src/main/java” y dentro de éste, se encuentra en el paquete: “com.org.tfg”.

Para poder lanzar el proyecto, tenemos que pulsar sobre la carpeta raíz del proyecto, llamada “TFG”, a continuación, hacer click derecho y pulsar en “Run As” y finalmente volver a pulsar en el botón “Run on server”

Si todo está correcto, se observará una información como esta, indicando que el servidor está funcionando:

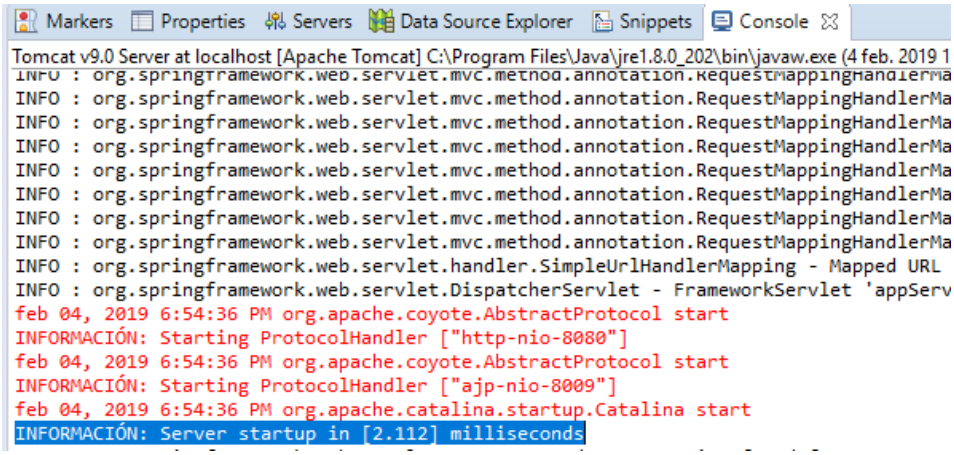

Figura 87. Servidor funcionando.