Prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet

Haro Valenzuela, Edward Vladimir

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Postgrados

Maestría en Gerencia de Sistemas

Trabajo de titulación, previo a la obtención del título de Magíster en Gerencia de Sistemas

PhD. Marcillo Parra, Diego Miguel

15 de octubre de 2020

URKUND

URKUND

Document Information

Analyzed document	Edward_Haro_Tesis.docx (D84467450)
Submitted	11/9/2020 1:48:00 PM
Submitted by	Diego Marcillo Parra
Submitter email	dmmarcillo@espe.edu.ec
Similarity	0%
Analysis address	dmmarcillo.espe@analysis.urkund.com

Sources included in the report

SA **Universidad de las Fuerzas Armadas ESPE / TrabajoTitulacionFinal_AguinagaGomez.docx**
Document TrabajoTitulacionFinal_AguinagaGomez.docx (D54526579)
Submitted by: dmmarcillo@espe.edu.ec
Receiver: dmmarcillo.espe@analysis.urkund.com
2

Firma:

Firmado electrónicamente por:
DIEGO MIGUEL MARCILLO PARRA

PhD. Marcillo Parra, Diego Miguel

DIRECTOR

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet**" fue realizado por el señor **Haro Valenzuela, Edward Vladimir** el mismo que ha sido revisado y analizado en su totalidad, por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 15 de septiembre de 2020

Firma:

Firmado electrónicamente por:
DIEGO MIGUEL MARCILLO PARRA

.....

PhD. Marcillo Parra, Diego Miguel

Director

C.C.: 1710802925

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

RESPONSABILIDAD DE AUTORÍA

Yo **Haro Valenzuela, Edward Vladimir**, con cédula de ciudadanía n°1003197132, declaro que el contenido, ideas y criterios del trabajo de titulación: **Prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 15 de octubre de 2020

Firma (s)

.....

Haro Valenzuela, Edward Vladimir

C.C.:1003197132

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

AUTORIZACIÓN DE PUBLICACIÓN

Yo **Haro Valenzuela, Edward Vladimir** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Título: Prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 15 de octubre de 2020

Firma

.....

Haro Valenzuela, Edward Vladimir

C.C.:1003197132

Dedicatoria

El trabajo de la gerencia es transmitir el mensaje de liderazgo de una manera convincente e inspiradora. No solo en las reuniones sino también dando el ejemplo. **Jefrrey Gitomer**.

Dedico el presente trabajo de titulación de maestría a toda mi familia, en especial a mi hija que ha sido fuente de inspiración para desear ser cada día mejor, a mi querida y añorada abuelita Bachi que me enseñó a no rendirme y alcanzar mis metas, dedico a todas las personas que durante este proceso me han acompañado y me han dado palabras sabias y de aliento que me han incentivado a superarme cada día.

Agradecimientos

Agradezco primero a Dios por darme salud en estos tiempos críticos y por darme sabiduría y paciencia para continuar y terminar lo que tanto he anhelado como lo es el título de magister en gerencias de sistemas, agradezco a todos los docentes por el conocimiento brindado y a mi familia por acompañarme en las noches de estudio y de desvelo.

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

Los signos vitales son importantes indicadores de salud del ser humano que permiten obtener el estado fisiológico de órganos vitales como el corazón, cerebro, pulmones, clasificados en cuatro principales que son Temperatura, Frecuencia respiratoria, Frecuencia cardiaca y Presión arterial.

El problema se centra en las personas con enfermedades crónicas o con un estado de salud delicado que necesitan un constante monitoreo, pero, debido a diversas circunstancias entre la que destaca; los costos de la salud, los precios elevados de dispositivos tecnológicos que permitan controlar los diferentes signos vitales, falta de espacio en las instituciones para ser atendidos, no logran acceder a este servicio.

Por lo que se ha planteado el objetivo de Desarrollar un prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet.

En este proyecto se ha desarrollado un prototipo low cost aplicando una metodología propia dividida en definición del problema, definición y obtención de signos vitales, diseño e implementación del prototipo, análisis de resultados. Permitiendo recabar, ordenar y analizar los datos obtenidos otorgando validez y rigor científico en el proceso de estudio y análisis.

En cuanto a los resultados obtenidos se tiene un prototipo armado con dispositivos low cost como es un sensor de temperatura DS18B20, un sensor de pulso todo esto conectado a un Raspberry Pi, capaz de ejecutar un proyecto desarrollado en Node.js que funciona como Backend y levantar un servidor MQTT, capaz de transmitir información entre los publicadores a los subscritores de un determinado tópico, así como también el consumo de una API de Twilio capaz de enviar mensajes WhatsApp a los números registrados para alertas en la aplicación. Y para consumir todos estos servicios, una aplicación Web adaptativa para cualquier dispositivo desarrollado con el framework Angular 9, capaz de mostrar toda la información de los pacientes y permitir el monitoreo de los signos vitales.

Palabras clave:

- **ANGULAR**
- **IOT**
- **NODE.JS**
- **RASPBERRY PI**
- **TELEMEDICINA**

Abstract

Vital signs are important indicators of human health that allow obtaining the physiological status of vital organs such as the heart, brain, lungs, classified into four main ones which are Temperature, Respiratory rate, Heart rate and Blood pressure.

The problem focuses on people with chronic diseases or with a delicate state of health who need constant monitoring, but, due to various circumstances, among which it stands out; Health costs, high prices for technological devices that allow monitoring of different vital signs, lack of space in institutions to be cared for, do not manage to access this service.

Therefore, the objective of developing an e-health prototype based on low-cost embedded systems for monitoring vital signs through the internet has been set.

In this project, a low cost prototype has been developed applying its own methodology divided into definition of the problem, definition and obtaining of vital signs, design and implementation of the prototype, analysis of results. Allowing to collect, order and analyze the data obtained, granting validity and scientific rigor in the study and analysis process.

Regarding the results obtained, there is a prototype armed with low cost devices such as a DS18B20 temperature sensor, a pulse sensor all connected to a Raspberry Pi, capable of executing a project developed in Node.js that works as a Backend and build an MQTT server, capable of transmitting information between publishers to subscribers of a certain topic, as well as the consumption of a Twilio API capable of sending WhatsApp messages to registered numbers for alerts in the application. And to consume all these services, an adaptive web application for any device developed with the Angular 9 framework, capable of displaying all the information of the patients and allowing monitoring of vital signs.

Keywords:

- **ANGULAR**
- **IOT**
- **NODE.JS**
- **RASPBERRY PI**
- **TELEMEDICINE**

Capítulo 1: Introducción

El presente trabajo de tesis tiene como objetivo principal desarrollar, aplicando el modelo low cost, un prototipo e-health que pueda ser usado en el ámbito de la telemedicina, basado en sistemas empotrados de bajo costo que permita el acceso en el ámbito económico a pacientes que necesitan un constante monitoreo de sus signos vitales por parte de los profesionales de salud, y así ayudar a resguardar la salud de los pacientes a través del internet para dar una atención personalizada y además descongestionar los centros de salud.

La importancia de realizar este prototipo, se considera el caso de necesitar un monitoreo constante de la salud por cualquier situación médica y poder tener acceso a un dispositivo que me permita conocer los signos vitales desde la comodidad del hogar y que un profesional de la salud pueda estar al tanto de esta información sin necesidad de acudir personalmente a un control, otro tema por lo que es importante este estudio radica en servir a la sociedad es decir plasmar los conocimientos académicos en un producto que ayude a solventar los requerimientos de las personas que necesitan un cuidado continuo y por ultimo pero no menos importante permitir el acceso a la gente de cualquier situación económica a un recurso que les puede cuidar su salud.

La metodología que se aplica en este proyecto es una metodología propia (Ad-Hoc) la cual se encuentra divida en: definición del problema, definición y obtención de signos vitales, diseño e implementación del prototipo, análisis de resultados. Permitiendo recabar, ordenar y analizar los datos obtenidos otorgando validez y rigor científico en el proceso de estudio y análisis.

Este trabajo también se centra en realizar una evaluación de costos basado en el modelo low cost que permita reducir costos para bajar los precios y también ofrecer un servicio complementario que en este caso es una aplicación web accesible desde el navegador de cualquier dispositivo y poder ver no solo datos en tiempo real, sino poder revisar un historial de la información de los pacientes y permitir alertar mediante un mensaje de WhatsApp a los profesionales de la salud en el caso de haber signos vitales que no se encuentren en el rango normal.

Para cumplir con una organización en el desarrollo de la tesis, se ha dividido el contenido en cuatro capítulos.

En el Capítulo 1 se realiza una revisión literaria acerca de antecedentes acerca del tema, permitiendo plantear un problema para definir el objetivo general y los específicos, realizar la justificación y plantear la importancia y definir el alcance del proyecto, plantear una hipótesis y categorizar las variables para la investigación.

En el Capítulo 2 se habla del estado de la cuestión, es decir conocer acerca de todos los temas usados en el desarrollo del presente trabajo; temas como signos vitales, e-health, IoT, sistemas embebidos, sensores para signos vitales, herramientas de desarrollo de software, la metodología de la investigación utilizada; también se considera el estado del arte, donde se referencia información de trabajos relacionados o similares con el presente trabajo.

En el Capítulo 3 se realiza el desarrollo del prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet. Para lo cual se propone el uso de dispositivos low cost que se van a encargar de recolectar la información al ser conectados a una Raspberry PI 4. La información recolectada se enviará hacia un bróker MQTT que comunicará la

información a un servicio REST que permitirá almacenar los datos obtenidos en una base de datos MongoDB, así como realizar una validación de acuerdo con los rangos normales para enviar una alerta o mensaje en el caso de que haya valores fuera del rango normal. La información se podrá ver mediante una aplicación web a los múltiples usuarios médicos, pacientes, así como también se podrá obtener los signos vitales del paciente mediante la aplicación web. Se concluye el capítulo con un análisis de costos con base al modelo low cost.

En el Capítulo 4 se detallan las conclusiones que el desarrollo de esta tesis ha permitido obtener.

Se incluyen anexos con información importante que van a permitir dar seguimiento a temas como sensores y sus características, encuestas, código fuente de la aplicación entre otros.

Antecedentes

Los signos vitales, son indicadores importantes que permiten obtener el estado fisiológico de órganos vitales como el corazón, cerebro, pulmones. Los cuatro principales signos vitales son: Temperatura, Frecuencia respiratoria, Frecuencia cardiaca, Presión arterial.

En la tesis de (Araujo Mena, 2015), se desarrolla la implementación de un sistema de video vigilancia para la UPS (Universidad Politécnica Salesiana), el cual tiene como componentes principales, tres estaciones con Raspberry PI trabajando bajo la distribución Linux, Raspbian, para el desarrollo como un servidor, en donde se aloja una página web para el monitoreo permanente en línea desde todo el mundo de tres cámaras fijas de video vigilancia. El proyecto permite concluir en el uso de Raspberry PI y Raspbian para levantar un servidor para una aplicación. El uso de

componentes de bajo costo y su disponibilidad en línea es otro de los temas con los que se relaciona la investigación.

En la tesis (Pardo Agudelo & Guacaneme Valbuena, Gerardo, 2016), se aplican las características principales del Internet de las Cosas, en un sistema orientado a la medición de variables propias de un ambiente demótico. Permitiendo el fácil acceso a redes conectadas a Internet. En conclusión, en este trabajo se puede obtener la información necesaria para el desarrollo del Frontend y Backend y la comunicación con los dispositivos electrónicos.

En el artículo de (Núñez, Peña, Jairo Cardona, & Antolinez, Iván Saavedra, 2014), donde realiza un desarrollo de un sistema de información que permite obtener y administrar la información relacionada con signos vitales como presión arterial, frecuencia cardiaca y respiratoria y la saturación de oxígeno en la sangre de un paciente, la implementación del sistema se basa en una solución Web, permitiendo que médicos y especialistas puedan monitorear a sus pacientes desde cualquier punto conectado a la red en tiempo real y dar indicaciones críticas al personal médico que se encuentra en el lugar del paciente. Se relaciona con el presente trabajo, en la obtención de los signos vitales y poder mostrar la información en un sistema para que pueda ser evaluado por los médicos especialistas.

En el artículo de (Barillaro, y otros, 2016), donde busca crear una solución que monitoree la salud del usuario y la reporte a familiares, médicos o personas a cargo a través de Internet durante las 24hs del día, 7 días de la semana, además emitir alertas en el caso de que el paciente requiere atención médica inmediata. Todo esto mediante el uso tecnologías actuales como sistemas embebidos con sensores de movimiento, sensores biométricos, conexión inalámbrica, geoposicionamiento; haciendo uso de estos dispositivos y la computación en la nube para brindar a los

pacientes mayor comodidad, autonomía, reducción de costos, mejores controles y mayor respuesta en las emergencias.

En el artículo de (Zárate-Ocaña, Ramos-Cuevas, MD, Contreras-Cariño, LB, Bélen-Luna, JC, & González-Morán, CO, 2018), se desarrolló un prototipo móvil para la obtención de datos de los signos vitales de pacientes utilizando tecnologías del internet de las cosas (IoT). Al paciente se le colocan sensores de: temperatura corporal, frecuencia respiratoria y frecuencia cardiaca, la información se adquiere a una interfaz digital, estos datos se envían a través de un microcontrolador, vía Bluetooth y WiFi, el médico puede ver los datos mediante: una aplicación hecha en Android usando un módulo Bluetooth, y también en una página web utilizando una dirección IP configurada en el prototipo. El proyecto se limita solo de manera local (red local). Este artículo se relaciona con el presente trabajo ya que Se desarrolló el prototipo de un dispositivo de obtención de datos de signos vitales de un paciente utilizando tecnologías del Internet de las cosas (IoT) de bajo costo, el cual transmite sus mediciones a una base de datos, un display LCD para consulta del paciente o personal de salud y una conexión inalámbrica (WiFi y Bluetooth).

En el trabajo de grado de (Gonzalez Mejia & Rodríguez Sarmiento), consiste en el diseño e implementación una red de sensores para el monitoreo de frecuencia cardiaca y respiratoria utilizando requerimientos del IoT. Esta aplicación web permite la visualización de la medición actual y del histórico de datos, así como la configuración de alarmas según la frecuencia cardiaca del paciente, proporcionando así una herramienta de apoyo para el diagnóstico y seguimiento de enfermedades por parte de los profesionales de la salud.

Planteamiento del problema

En el Ecuador existen personas con enfermedades crónicas o con estado de salud delicado que ameritan un monitoreo constante por parte de sus médicos o cuidadores lo que puede derivar en una molestia para los pacientes ya que los canales telefónicos y virtuales no prestan el mismo servicio que se presta físicamente; y además inconvenientes logísticos en la obtención del estado general de los pacientes debido a los elevados niveles de usuarios que requieren el monitoreo de su salud.

Las instituciones de salud, no cuenta con suficiente espacio para atender a cada uno de los pacientes al mismo tiempo, así como debido a los costos elevados tampoco cuentan con la cantidad suficiente de dispositivos y el tiempo de uso de cada uno no permite asignar los materiales necesarios a cada paciente por lo que es posible que muchas de estas personas, no reciban un tratamiento adecuado a tiempo para prevenir complicaciones en el estado de su salud.

Las personas con enfermedades crónicas o con un estado de salud delicado, necesitan un constante monitoreo de signos vitales; pero por problemas en el personal como falta de experiencia, personal insuficiente, sobrecarga laboral, desconocimiento de proceso; en los materiales como insuficientes dispositivos, costos elevados, tiempos de uso; problemas en el sistema falta de mantenimiento de los equipos y elevados números de usuarios; ocasiona que no se pueda realizar un constante monitoreo de los signos vitales a toda la lista de personas que lo requieren, por lo que es necesario desarrollar un dispositivo de bajo costo que permita enviar alertas y la información en tiempo real del paciente hacia la nube para que el profesional de la salud, pueda dar un seguimiento y un control de acuerdo a la información generada por el prototipo.

Objetivos

Objetivo General

Desarrollar un prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet.

Objetivos Específicos

- Investigar los signos vitales que presenta el ser humano y evaluar las herramientas tecnológicas hardware y software que permiten monitorear los signos vitales del ser humano.
- Diseñar el prototipo e-health mediante sistemas empotrados, dispositivos IoT y servicios en la nube.
- Implementar el prototipo e-health para monitoreo y alerta de signos vitales.
- Validar el prototipo e-health mediante la aplicación de pruebas en tiempo real.
- Evaluar las bondades y los costos del prototipo e-health mediante el modelo de low cost.

Justificación, importancia y Alcance del Proyecto

En la actualidad, los sistemas y aplicaciones orientadas a saludo mediante el uso de dispositivos electrónicos han ganado popularidad, debido a mejoras logradas en la calidad del servicio. La electrónica y la informática han alcanzado altos niveles de transmisión y análisis de datos gracias a la conexión con redes inalámbricas y la conexión con el Internet, así como la aparición de dispositivos de bajo costo capaces de efectuar procesos computacionales costosos (Pardo Agudelo & Guacaneme Valbuena, Gerardo, 2016).

El desarrollo del dispositivo e-health se relaciona con la creación de un prototipo que permite la recolección de información de los principales signos vitales del ser humano a través de sensores diseñados para obtener esta información; realizando un diseño mediante sistemas empotrados de bajo costo para reducir los precios de adquisición buscando minimizar los problemas de las instituciones de salud o de los pacientes para la compra de estos y así aumentar la cantidad de dispositivos para solventar el elevado nivel de usuarios y poder realizar un monitoreo constante de los pacientes ya sea en el hogar de cada uno o en las instituciones de salud.

Por esta razón se ha seleccionado este tema de investigación como una mejora a corto plazo para obtener un prototipo que cumpla con las bondades de: portabilidad, modelo low cost y acceso vía internet desde cualquier lugar, que permita ejecutar las acciones de monitoreo y alerta de la temperatura y del pulso del paciente hacia el médico tratante.

Hipótesis de Investigación

Entre las preguntas de investigación generadas a partir de la información obtenida se tiene:

¿Desarrollar un prototipo e-health basado en sistemas empotrados, dispositivos IoT y servicios en la nube cumple con el modelo de low cost?

Hipótesis: Es posible desarrollar prototipo e-health basado en sistemas empotrados, dispositivos IoT y servicios en la nube que cumplan con el modelo de low cost.

Categorización de las Variables de Investigación

Variable dependiente: Dispositivos IoT.

Variable independiente: Signos vitales temperatura y pulso.

Capítulo 2: Estado del Arte y de la Cuestión

En el capítulo dos se habla del estado de la cuestión, es decir conocer acerca de todos los temas usados en el desarrollo del presente trabajo; temas como signos vitales, e-health, IoT, sistemas embebidos, sensores para signos vitales, herramientas de desarrollo de software, la metodología de la investigación utilizada; también se considera el estado del arte, donde se referencia información de trabajos relacionados o similares con el presente trabajo.

Estado de la Cuestión

Signos Vitales

Las funciones vitales básicas en el ser humano como la respiración, la circulación, el pulso se manifiestan de manera externa a través de los signos vitales, los cuales se evalúan mediante un examen físico y se los miden a través de instrumentos simples. Considerando rangos de valores como normales, en donde las variaciones son resultado de cambios que ocurren en el organismo ya sea fisiológicos o patológicos. Los cuatro principales signos vitales del ser humano son: frecuencia respiratoria, frecuencia cardiaca o pulso, tensión arterial y la temperatura (Cobo & Daza, 2011).

Los signos vitales son una herramienta, la cual nos permite conocer el estado funcional de un paciente, los resultados de las mediciones de estos deben representar a una evaluación clínica confiable del paciente por parte de una enfermera permitiendo realizar una interpretación de estos con la finalidad de que el medico pueda decidir cómo manejar la situación funcional del paciente.

A demás los signos vitales reflejan el estado fisiológico de órganos vitales como son corazón, pulmones, cerebro, expresando los cambios funcionales que

ocurren en el organismo de manera inmediata, permitiendo cualificarlos y cuantificarlos.

Los cuatro principales signos vitales son: Frecuencia cardiaca, medida por el pulso en latidos por minuto; Frecuencia respiratoria; Tensión (presión) arterial; Temperatura (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Pulso Arterial.

Es la onda pulsátil de la sangre, se origina al momento de la contracción del ventrículo izquierdo del corazón dando como resultado la expansión y contracción regular del calibre de las arterias, esta onda pulsátil representa el rendimiento del latido cardiaco, es decir la cantidad de sangre que entra en las arterias con cada contracción ventricular y la adaptación de las arterias (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Los latidos por minuto, frecuencia cardiaca, varía de acuerdo con:

Sexo: ya que después de la pubertad el pulso es más lento en el hombre que en la mujer.

Edad: el pulso varía desde el nacimiento hasta la madurez.

Actividad física: el pulso sufre un aumento de velocidad cuando se realizar ejercicios físicos.

Estado emocional: la actividad cardiaca aumenta cuando emociones, ansiedad, dolor, miedo estimulan el sistema simpático.

Medicamentos: existen medicamentos que pueden aumentar o disminuir el pulso.

Fiebre: Aumenta el pulso cardiaco.

Hemorragias: La pérdida de sangre puede aumentar el pulso convirtiéndose en una taquicardia.

Existen nuevos puntos anatómicos para la palpación del pulso (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Pulso temporal: se mide sobre el hueso temporal, en la región externa que se encuentra entre la ceja y el cuero cabelludo.

Pulso carotideo: se encuentra en el cuello entre la tráquea y el musculo esternocleidomastoideo.

Pulso branquial: se localiza en la cara interna del bíceps**.**

Pulso radial: se puede palpar en la cara interna de la muñeca al realizar una suave presión.

Pulso femoral: se encuentra debajo del ligamento inguinal, en la arteria femoral.

Pulso poplíteo: se localiza detrás de la rodilla en la fosa poplítea.

Pulso tibial posterior: se palpa la arteria tibial localizada por detrás del maléolo interno.

Pulso pedio: se obtiene al palpar la arteria dorsal del pie.

Existen recomendaciones importantes para la valoración del pulso cardiaco, entre las que podemos mencionar las siguientes:

El paciente debe adoptar una posición cómoda y relajada.

Verificar si el paciente ha recibido medicación que pueda alterar la frecuencia cardiaca.

En el caso de actividad física el paciente debe esperar entre 10 y 15 minutos para realizar un control.

Los valores normales de la frecuencia cardiaca se los especifica en la Tabla 1 (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Tabla 1

Valores normales frecuencia cardiaca

EDAD	PULSACIONES POR MINUTO
Recién nacido	120 – 170
Lactante menor	120 – 160
Lactante mayor	110 – 130
Niños de 2 a 4 años	100 – 120
Niños de 6 a 8 años	100 – 115
Adulto	60 – 80

Respiración

La respiración es un proceso mediante el cual ingresa oxigeno del aire del medio ambiente y se expulsa anhídrido carbónico del organismo, este ciclo respiratorio tiene una fase de inspiración y otra de espiración (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Existen factores que influyen a la respiración como son:

- El estrés.
- La edad.
- Aumento de temperatura en el ambiente.

- Medicamentos que disminuyan la frecuencia respiratoria.
- Disminución de oxígeno en el aire, debido al ascenso a grandes alturas.

Para realizar una valoración de la respiración, se puede tomar como referencia los valores normales que se encuentran detallados en la Tabla 2 (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Tabla 2

Valores normales de frecuencia respiratoria

EDAD	RESPIRACIONES POR MINUTO
Recién nacido	30 – 80
Lactante menor	20 – 40
Lactante mayor	20 – 30
Niños de 2 a 4 años	20 – 30
Niños de 6 a 8 años	20 – 25
Adulto	15 – 20

Presión Arterial (PA) o Tensión Arterial (TA)

Es la presión que ejerce la sangre sobre las paredes arteriales en su impulso a través de las arterias. Ya que la sangre se mueve en forma de ondas, existen dos tipos de medidas de presión: presión sistólica que es la presión de la sangre cuando los ventrículos se contraen llamada también la presión máxima; el otro tipo de presión llamada diastólica es la presión cuando los ventrículos se relajan, llamada la presión mínima.

En la Tabla 3 (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005) se detallan los valores normales de acuerdo con las edades. Mientras que en la Tabla 4 (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005) se detalla el tamaño del manguito que debe escogerse de acuerdo con el diámetro del brazo. La desigualdad relativa entre el tamaño del brazo y el manguito puede ser causa de error.

Otra de las recomendaciones que se debe tomar en cuenta es que el brazo y el ante brazo deben estar descubiertos para que las prendas no ejerzan presión (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

Tabla 3

Presiones sanguíneas normales

EDAD	Presión sistólica (mmHg)	Presión diastólica (mmHg)
Lactante	60 – 90	30 – 62
2 años	78 – 112	48 – 78
8 años	85 – 114	52 – 85
12 años	95 – 135	58 – 88
Adulto	100 – 140	60 – 90

Tabla 4

Tamaño de los manguitos de presión sanguínea

EDAD	Ancho (cm)	Longitud (cm)
Recién nacido	2,5 – 4,0	5,0 – 10,0
Lactante	6,0 – 8,0	12,0 – 13,5
Niño	9,0 – 10,0	17,0 – 22,5
Adulto, estándar	12,0 – 13,0	22,0 – 23,5
Adulto, brazo grande	15,5	30,0
Adulto, muslo	18,0	36,0

Temperatura

La temperatura es el equilibrio entre la producción de calor por el cuerpo y su perdida, cuando la temperatura sobrepasa el nivel normal se activan mecanismos como sudoración, hiperventilación los cuales promueven la perdida de calor. En el caso de que la temperatura, baje del nivel normal se activan mecanismos como contracciones espasmódicas que producen escalofríos para generar calor. Existen sitios para obtener la temperatura, como el Oral, Rectal donde el resultado tiende a ser 0.5 a 0.7 grados centígrados mayor que la temperatura oral y axilar donde el resultado es 0.5 grados centígrados menor que la temperatura oral (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005).

En la Tabla 5 (Penagos, Salazar, Luz Dary, & Vera, Fanny E, 2005), se muestran los valores normales de temperatura de acuerdo con la edad.

Tabla 5

Valores normales temperatura

EDAD	Grados centígrados (ºC)
Recién nacido	36,1 – 37,7
Lactante	37,2
Niños de 2 a 8 años	37,0
Adulto	36,0 – 37,0

E-Health

E-health describe a aplicaciones de Internet relacionadas con la salud que ofrecen una variedad de contenido, conectividad y atención clínica, promueve un mecanismo que reduce costos, genera crecimiento y mejora los procesos de la atención pública (Wilson & Lankton, Nancy K, 2004).

También se define como el uso de las tecnologías de la información para apoyar a campos relacionados con la salud como: atención médica, vigilancia, educación conocimiento e investigación sobre salud.

Los sistemas de información pueden proveer un enorme apoyo para la atención médica en diferentes entornos, apoyan a un trabajador de salud realizando un seguimiento a pacientes en programas de VIH en donde la tasa de abandono de los tratamientos por los pacientes puede llegar a un 76% (Blaya, Fraser, & Holt, 2010).

E-health tiene diez principios los cuales podemos observar en la Tabla 6 (Eysenbach, 2001).

Tabla 6

Principios e-health

PRINCIPIO	DESCRIPCION
Eficiencia	Aumentar la eficiencia en la atención médica, disminuyendo los costos, evitando intervenciones duplicadas o innecesarias, mejorando la comunicación entre el paciente y los establecimientos de atención de salud.
Mejorar la Calidad	E-health puede mejorar la calidad de atención médica permitiendo comparar entre proveedores de atención para dirigir a los pacientes a los proveedores de mejora calidad.
Basado en la evidencia	La efectividad y la eficiencia no deben ser asumidas, sino probadas por una evaluación científica rigurosa.
Empoderamiento	Permitir la elección del paciente basado en la evidencia mediante el acceso a bases de conocimientos de medicina y registros electrónicos personales a través de Internet.
Estimulación	Las decisiones se toman de manera compartida mediante una asociación entre el paciente de la salud y el profesional.
Educación	Información preventiva adaptada para los consumidores a través de fuentes en línea.
Permitir	Intercambio de información y comunicación entre los establecimientos de atención médica.

PRINCIPIO	DESCRIPCION
Extensible	Extender la atención de salud más allá de los limites convencionales, e-health permite a los consumidores obtener servicios de salud en línea incluso con proveedores mundiales.
Ética	Implica nuevas formas de interacción entre el médico y el paciente, planteando nuevos desafíos y amenazas a problemas éticos.
Equidad	La brecha digital se extiende entre poblaciones rurales y urbanas, personas ricas y pobres, jóvenes y viejas. Por lo que e-health trata de hacer que la atención médica sea más equitativa.

IoT (Internet of Things)

El Internet de las cosas, se refiere a la conexión entre el mundo digital y físico, donde las cosas físicas y virtuales tienen identidades, atributos físicos y personalidades virtuales, estas se integran en la red de información a través de interfaces inteligentes basados en protocolos de comunicación estándar (Ray, 2018).

Para el 2020, alrededor de 25 mil millones de dispositivos estarán conectados a Internet facilitando información que permita analizar, planificar, gestionar y tomar decisiones de manera autónoma. En este contexto podemos ver qué servicios como transporte, ciudades inteligentes, domótica, salud, educación, comercio, agricultura, negocios entre otros, han sido beneficiados por diferentes formas de arquitectura de IoT (Ray, 2018).

En el Internet de las Cosas objetos, máquinas y personas se conectan para comunicarse entre sí en un entorno ubicuo, en una red global dinámica capaz de auto configurarse de acuerdo con estándares y protocolos de comunicación entre objetos virtuales y físicos integrados en la red de información, por esto se le considera como una idea de futuro.

En los últimos años el “Internet de las Cosas”, se ha dado a conocer de manera acelerada; en el 2005 ya se lo podía encontrar en títulos de libros, mientras que la Unión Internacional de Telecomunicaciones (UIT) los describía como “una promesa de un mundo de dispositivos interconectados que proveen contenido relevante a los usuarios”.

El Internet de las Cosas, da lugar a un enorme flujo de información de manera continua, por lo que es necesario disponer una red que soporte la conexión de los objetos y una gran base de datos para almacenar toda la información que circula ya que los servicios del Internet de las Cosas se basan en disponer de información en el lugar correcto y en el momento y contexto adecuados. En la Figura 1 (García Salvatierra, 2012) se observa un diagrama que representa la disponibilidad del Internet de las Cosas.

Figura 1

Disponibilidad del Internet de las Cosas

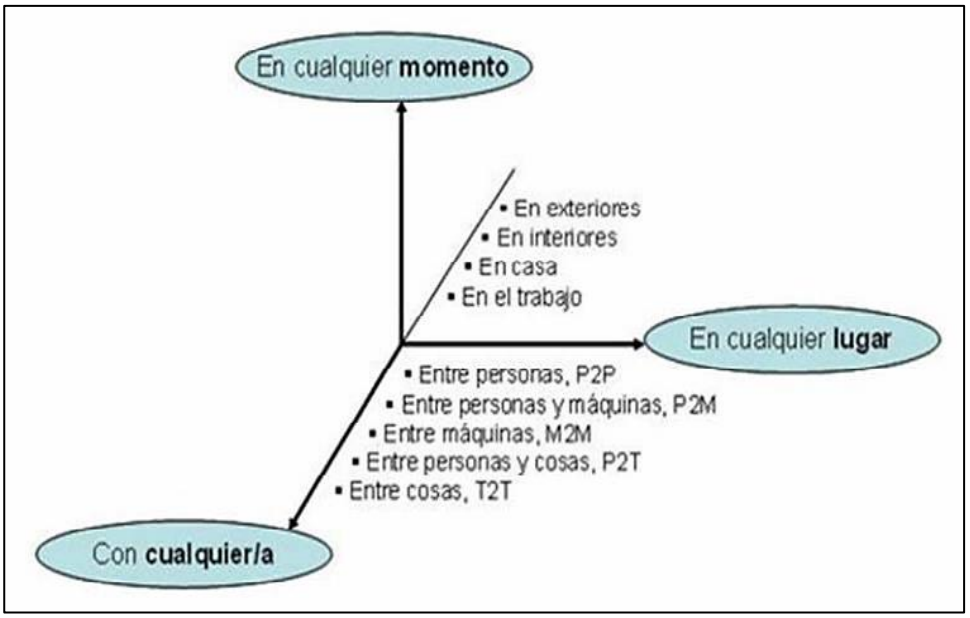


Tabla 7

Comparación de plataformas de hardware soportadas por IoT

Parámetros	Arduino Uno	Intel Galileo Gen 2	Intel Edison	Beagle Bone Black	Raspberry Pi B+	ARMmbed NXP LPC1768
Procesador	ATMega 328P	Intel QuarkTM SoC X1000	Intel QuarkTM SoC X1000	Sitara AM3358BZCZ100	Broadcom BCM2835 SoC based ARM1176JZF	ARM Cortex M3
GPU	-	-	-	PowerVR SGX530 @520 MHz	Video Core IV Multimedia@ 250 MHz	-
Voltaje de Operación	5V	5V	3,3V	3,3V	5V	5V
Velocidad del Reloj (MHz)	16	400	100	1GHz	700	96
Ancho de Bus (bits)	8	32	32	32	32	32
Memoria del Sistema	2K	256 MB	1 GB	512 MB	512 MB	32 KB
Memoria Flash	32 KB	8 MB	4 GB	4 GB	-	512 KB
EEPROM	1 KB	8 KB	-	-	-	-
Comunicación soportada	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/g/ n, 433RF, IEEE 802.15.4, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/ g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial

Parámetros	Arduino Uno	Intel Galileo Gen 2	Intel Edison	Beagle Bone Black	Raspberry Pi B+	ARMmbed NXP LPC1768
Ambiente de desarrollo	Arduino IDE	Arduino IDE	Arduino IDE, Eclipse, Intel XDK	Debian, Android, Ubuntu, Cloud9 IDE	NOOBS	C/C++ SDK, Online Compiler
Lenguaje de programación	Wiring	Wiring, Wyliodrin	Wiring, C, C++, Node.JS, HTML5	C, C++, Python, Perl, Ruby, Java, Node.js	Python, C, C ++, Java, Scratch, Ruby	C, C++
Conectividad I/O	SPI, I2C, UART, GPIO	SPI, I2C, UART, GPIO	SPI, I2C, UART, I2S, GPIO	SPI, UART, I2C, McASP, GPIO	SPI, DSI, UART, SDIO, CSI, GPIO	SPI, I2C, CAN, GPIO

En la Tabla 7 (Ray, 2018), se presentan plataformas de hardware clasificadas por parámetros como: Procesador, GPU, Voltaje de Operación, Velocidad de reloj, Ancho de bus, Memoria del sistema, Memoria Flash, EEPROM, Soporte para comunicación, Ambiente de desarrollo, Lenguaje de programación, Conectividad de entrada y salida I/O. En la tabla podemos ver cómo estas plataformas fomentan el crecimiento de IoT.

Sistemas Embebidos

Existen varias definiciones de sistemas embebidos en las que tenemos:

- “Las personas usan el término sistema embebido para referirse a cualquier sistema de cómputo escondido en algún producto o dispositivo” (Simon, 1999).

- “Un sistema embebido es cualquier dispositivo que incluye un computador programable, pero en sí mismo no es un computador de propósito general” (Wolf, 2012).

En un sistema embebido, el dispositivo encapsula un computador que sirve para un propósito específico a diferencia de los computadores de propósito general, diseñados para cumplir funciones previamente definidas, que no se puede cambiar fácilmente su funcionalidad, están compuestos por componentes hardware, circuitos integrados y software con grandes requerimientos en términos de confiabilidad (Camargo Bareño, 2011).

Los sistemas embebidos se distinguen de otros sistemas computacionales debido a ciertas características propias (Pérez, 2009):

1. Funcionamiento específico, ejecuta de forma repetitiva un programa.
2. Limitaciones, deben ser de bajo costo, tamaño reducido, buen desempeño, procesar datos en tiempo real, bajo consumo de energía por lo que posee limitaciones en su diseño y en sus métricas.
3. Reactivos y tiempo real, deben reaccionar ante cambios en el ambiente, procesar datos en tiempo real sin retrasos, realizar cálculos precisos a diferencia de un computador normal que la frecuencia y la demora de los cálculos no produce fallas en él sistema.

Sensores

Los sensores, son dispositivos fundamentales para el monitoreo de signos vitales, ya que son los encargados de obtener y enviar la información recogida en señales eléctricas a través de medios alámbricos o inalámbricos, esta señal puede ser digital o analógica, abarcan distintos campos de diseño que sirven para obtener la

información de la temperatura, presión, contacto, corriente entre otros (Ruiz Sánchez, 2016).

Los sensores responden a un estímulo que se transforma en una señal eléctrica de salida, la salida puede ser en forma de carga, corriente o voltaje, una forma de comparar varios sensores entre sí es a través de las especificaciones de cada uno, siendo principalmente las siguientes: Sensibilidad, Resolución, Estabilidad, Selectividad, Exactitud, Condiciones Ambientales, Rapidez de respuesta, Características de sobrecarga, Formato de salida, Costos, Tamaño, Peso (Valencia Zambrano, 2018).

Herramientas de Desarrollo de Software

Las herramientas para el Desarrollo de software para dar una explicación más clara de cada una de sus funciones se las va a dividir en: Base de datos, Backend, Protocolos de comunicación y Frontend.

Bases de Datos

Una base de datos es un conjunto de datos almacenados de forma estructurada, independientes de las aplicaciones que los utilizan, sin redundancias innecesarias, capaz de brindar acceso a múltiples usuarios o aplicaciones al mismo tiempo. La base de datos contiene información importante para una empresa, con el propósito de permitir a los usuarios de una manera práctica y eficiente almacenar y recuperar información (Silberschatz, y otros, 2002).

Las bases de datos se pueden clasificar en SQL y NoSQL como podemos ver en la Tabla 8.

Tabla 8

Bases de datos SQL y NoSQL

SQL	NoSQL
Oracle: Desarrollada por Oracle, es un sistema de gestión de base de datos de tipo objeto relación.	**MongoDB:** Sistema de base de datos NoSQL orientado a documentos.
PostgreSQL: Sistema de gestión de base de datos de código abierto, relacional orientado a objetos.	**Casandra:** Base de datos NoSQL distribuida por lo que permite grandes cantidades de datos de manera distribuida con una estructura clave valor.
MySQL: Sistema de gestión de base de datos relacional, con un modelo cliente servidor.	**Redis:** Motor de base de datos en memoria muy rápida con una estructura clave valor.

Las bases de datos SQL, hacen referencia al modelo relacional, es decir las relaciones entre las diferentes tablas y columnas, mientras que las bases de datos NoSQL guardan información no estructurada que puede recuperarse con el uso de lenguajes diferentes al SQL.

Backend

"Un desarrollador backend es quien trabaja del lado del servidor, utilizando lenguajes tales como Java, C#, Python etc. interactuando con bases de datos, verificando sesiones de usuario y montando una página en el servidor" (Luna, Peña, CM, & Iacono, M, 2014).

El backend es el extremo del servidor de aplicaciones, permite el acceso a la base de datos, donde se encuentra la lógica de la aplicación, la parte administrativa donde se gestionan los contenidos que van a ser mostrados al usuario final (Ortega Checa, 2019).

Para el desarrollo del backend existen algunos lenguajes de programación entre los cuales vamos a mostrar en la Tabla 9:

Tabla 9

Lenguajes de programación backend

LENGUAJE	CARACTERÍSTICAS
JAVA	Es un lenguaje orientado a objetos, que una vez compilado se ejecuta en la Máquina Virtual de Java.
C#	Es un lenguaje desarrollado por Microsoft, elegante, tipado y orientado a objetos.
NODE.JS	Es un lenguaje de programación basado en JavaScript asíncrono que se ejecuta en el lado del servidor.
PYTHON	Es un lenguaje de programación interpretado que soporta orientación a objetos, programación imperativa incluso programación funcional.

Protocolos de Comunicación IoT

Los protocolos IoT permiten conectar dispositivos con plataformas IoT, existen protocolos Cliente/Servidor es decir que él cliente se conecte con él servidor para realizar solicitudes que él servidor va a responder con información obtenida previamente y Publish/Subscribe es decir que los clientes se subscriban a un tópico para publicar y recibir información que un bróker va a administrar para redirigir los datos (Semle & eFalcom, K, 2016).

En la Tabla 10 se puede observar los protocolos de comunicación más comunes en el uso de IoT.

Tabla 10

Protocolos de comunicación para IoT

PROTOCOLO	CARACTERÍSTICA
MQTT (MQ Telemetry Transport)	Protocolo Publish/Subscribe de Servicio de Mensajes que actúa sobre TCP, ligero y sencillo de usar. Soporta gran número de clientes de forma simultánea.
AMQP (Advanced Message Queuing Protocol)	Protocolo Publish/Subscribe de cola de mensajes, no resulta para dispositivos IoT de bajos recursos.
WAMP (Web Application Messaging Protocol)	Protocolo abierto Publish/Subscribe y Remote Procedure Calls que se ejecuta sobre WebSockets.
CoAP (Constrained Application Protocol)	Para dispositivos IoT de baja capacidad bajo el modelo REST sobre HTTP.
WMQ (WebSphere MQ)	Protocolo de cola de mensajes desarrollado por IBM.

Frontend

El Frontend se encuentra disponible al público con funcionalidades y contenidos de acceso restringido o libre. El usuario interactúa directamente con tecnologías de aplicaciones web desarrolladas con lenguajes como HTML, JavaScript, CSS, con el objetivo de desarrollar la interfaz gráfica para el uso del usuario final (Valdivia-Caballero, 2016).

En la Tabla 11 podemos ver los frameworks para el desarrollo de Frontend más usados por los desarrolladores.

Tabla 11

Frameworks para Frontend

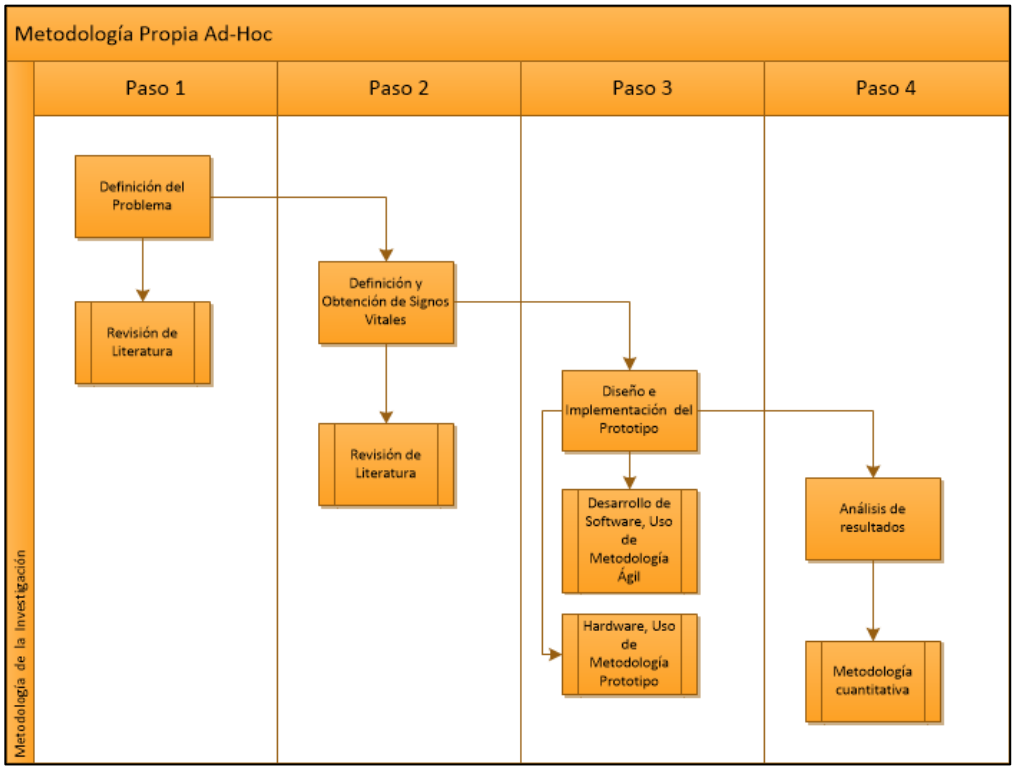
FRAMEWORK	CARACTERÍSTICA
ANGULAR IO	Utiliza el patrón Modelo Vista Controlador, el lenguaje que usa es TypeScript, permite desarrollar aplicaciones de una sola página. Es mantenido por Google.
VUE.JS	Es un framework JavaScript que permite construir interfaces de usuario de manera sencilla. El patrón que utiliza es modelo-vista-vista-modelo en aplicaciones de una sola página.
REACT	Es una biblioteca de JavaScript, permite crear interfaces de usuario en aplicaciones de una sola página.

Metodología de la Investigación

La metodología que se aplica en este proyecto es una metodología propia (Ad-Hoc) la cual se encuentra divida en: definición del problema, definición y obtención de signos vitales, diseño e implementación del prototipo, análisis de resultados. Permitiendo recabar, ordenar y analizar los datos obtenidos otorgando validez y rigor científico en el proceso de estudio y análisis. En la Figura 2 podemos observar un diagrama de flujo con la metodología propuesta para el desarrollo del trabajo.

Figura 2

Metodología propia (Ad-Hoc)



Definición del Problema

Para definir el problema se va a realizar un análisis de documentos como tesis, artículos científicos, libros, revistas es decir se va a aplicar una metodología de revisión preliminar de literatura, para buscar si existe o no una solución similar para el objetivo propuesto y en el caso de que si exista conocer cuáles son las falencias o vacíos que se deben cubrir y/o realizar una mejora a estos para cumplir con el objetivo propuesto.

Definición y Obtención de Signos Vitales

Al igual que en la definición del problema se va a realizar un análisis de documentos como tesis, artículos científicos, libros, revistas es decir se va a aplicar una metodología de revisión preliminar de literatura, que permita identificar los

principales signos vitales del ser humano y establecer la información necesaria para continuar con el siguiente paso en base a los datos principales de cada signo vital, es decir obtener la información que permita cumplir con el objetivo de obtener la información de cada signo a través de dispositivos disponibles para este propósito.

Diseño e Implantación del Prototipo

Para realizar el prototipo se va a usar una metodología ágil ya que es necesario agilizar el proceso del desarrollo del prototipo el cual está conformado por dos partes una parte hardware y otra parte software.

Para el desarrollo del hardware se usa una metodología de prototipo, es decir hacer una muestra de la solución que permita validar su esencia funcional y permita hacer los cambios que sean necesario o fundamentales para entregar una solución final.

Para el desarrollo del software, la metodología ágil eXtreme Programming al ser la más sencilla y fácil de implementar va a permitir tener una retroalimentación continua entre el cliente y el desarrollador, planificación flexible entregas rápidas mediante un desarrollo iterativo o en cascada.

Para conocer más acerca de las metodologías ágiles la siguiente referencia tiene información básica permite entender más acerca de la misma (IEBS, 2020).

Análisis de Resultados

Para el análisis de resultados se va a realizar una metodología cuantitativa que permita obtener y procesar las muestras de los sensores y compararlos con muestras de dispositivos comerciales disponibles para validar la que exista un rango de error tolerable y finalmente comparar con el modelo low cost los valores invertidos en el prototipo.

Herramientas Hardware y Software del Dispositivo e-health

Para el presente prototipo, se ha determinado medir dos signos vitales, pulso y temperatura, para la cual se justifica las herramientas hardware y software utilizadas y los sensores escogidos para realizar el prototipo de acuerdo con las características que cada uno ofrece.

A continuación, se describe cada uno de los componentes escogidos para el desarrollo del prototipo.

Sensor de Pulso

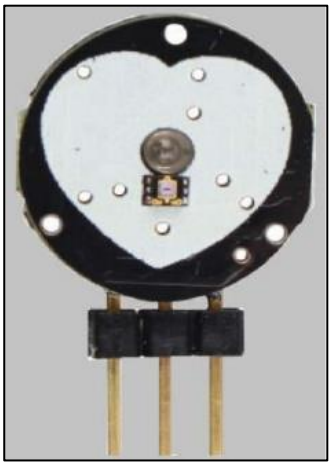
El sensor de pulso o foto pletismógrafo debido a que funciona con la respuesta de los cambios en la intensidad relativa de la luz, el valor de la señal será de 512, es decir el valor del rango medio de ADC de 10 bits, mientras mayor sea la intensidad de la luz, el valor ADC será mayor. El sensor produce una onda llamada foto pletismograma (PPG) que en la medicina ayuda a medir la frecuencia cardiaca, así como también la frecuencia respiratoria (Rahadian & Arifin, Zaenal, 2016).

El sensor de pulso cuenta con tres pines para la conexión, uno para el suministro de 5V, otro par GND y finalmente un pin para transmitir la señal analógica (Goel[1], Srivastava, Sharad, Pandit, Dharmendra, Tripathi[4], & Goel[5], 2018).

En el mercado ecuatoriano, es un sensor fácil de conseguir y se promociona en varios precios siendo el más común seis dólares. En la Figura 3 (Goel[1], Srivastava, Sharad, Pandit, Dharmendra, Tripathi[4], & Goel[5], 2018) podemos observar el sensor de pulso con sus tres pines.

Figura 3

Sensor de pulso



Toda la información del sensor de pulso se encuentra en el Anexo 1 a esta tesis, en este anexo se encuentra el Datasheet del sensor.

Sensor de Temperatura

El sensor de temperatura DS18B20, es un dispositivo que trabaja bajo el protocolo 1-Wire, lo que permite el uso de más de un sensor de temperatura con el mismo cable de datos para la interacción entre dispositivos sin necesidad de ampliar la infraestructura por cada sensor que se use. Para poder usar varios sensores a la vez, cada sensor tiene una dirección única de 64 bits que lo distingue de los otros. Este sensor provee medidas de temperatura con una resolución configurable de 9 a 12 bits (Gomez Blazquez, 2016).

Posee un rango de mediciones entre -55 y 125 grados centígrados, con un error entre ±0.5 grados para mediciones entre -10ºC hasta 85ºC y un rango de error de ±2 grados para las mediciones restantes hasta -55ºC y 125ºC. Está compuesto por tres cables para la conexión, un cable amarillo para la transmisión de datos DQ, en formato digital, cable negro para la conexión a tierra GND y un cable rojo para la

alimentación de 5V VDD, aunque también puede trabajar en modo parásito tomando energía a través del cable de transmisión de datos DQ (Martínez Jimeno, 2018).

En la Figura 4 (Martínez Jimeno, 2018) se puede ver el sensor DS18B20 con los cables disponibles para la conexión, en la Figura 5 (Martínez Jimeno, 2018) se muestra la conexión 1-Wire tipo VDD y en la Figura 6 (Martínez Jimeno, 2018) se representa la conexión 1-Wire tipo parasito.

Figura 4

Sensor de Temperatura DS18B20

Figura 5

Conexión 1-Wire tipo VDD

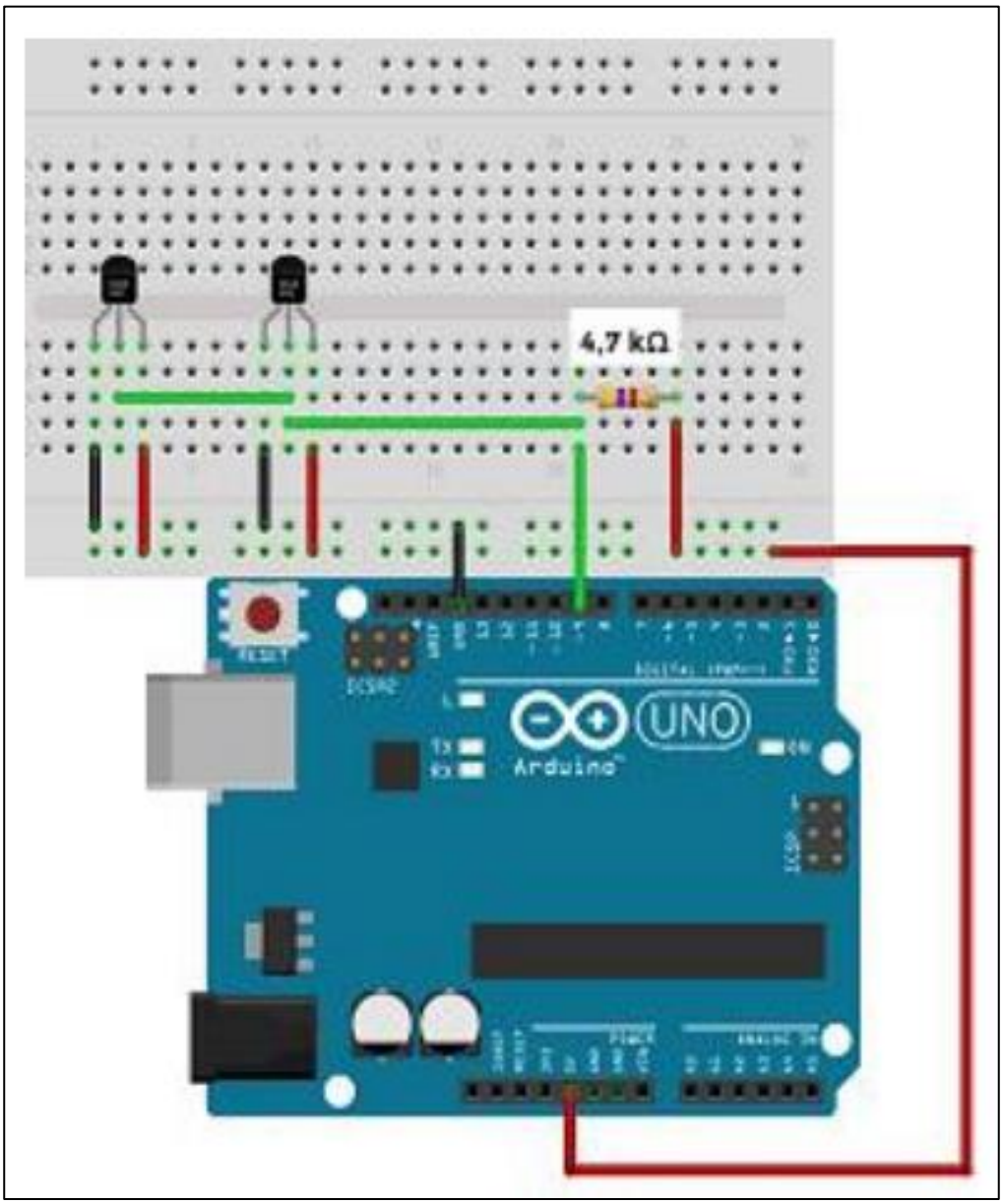
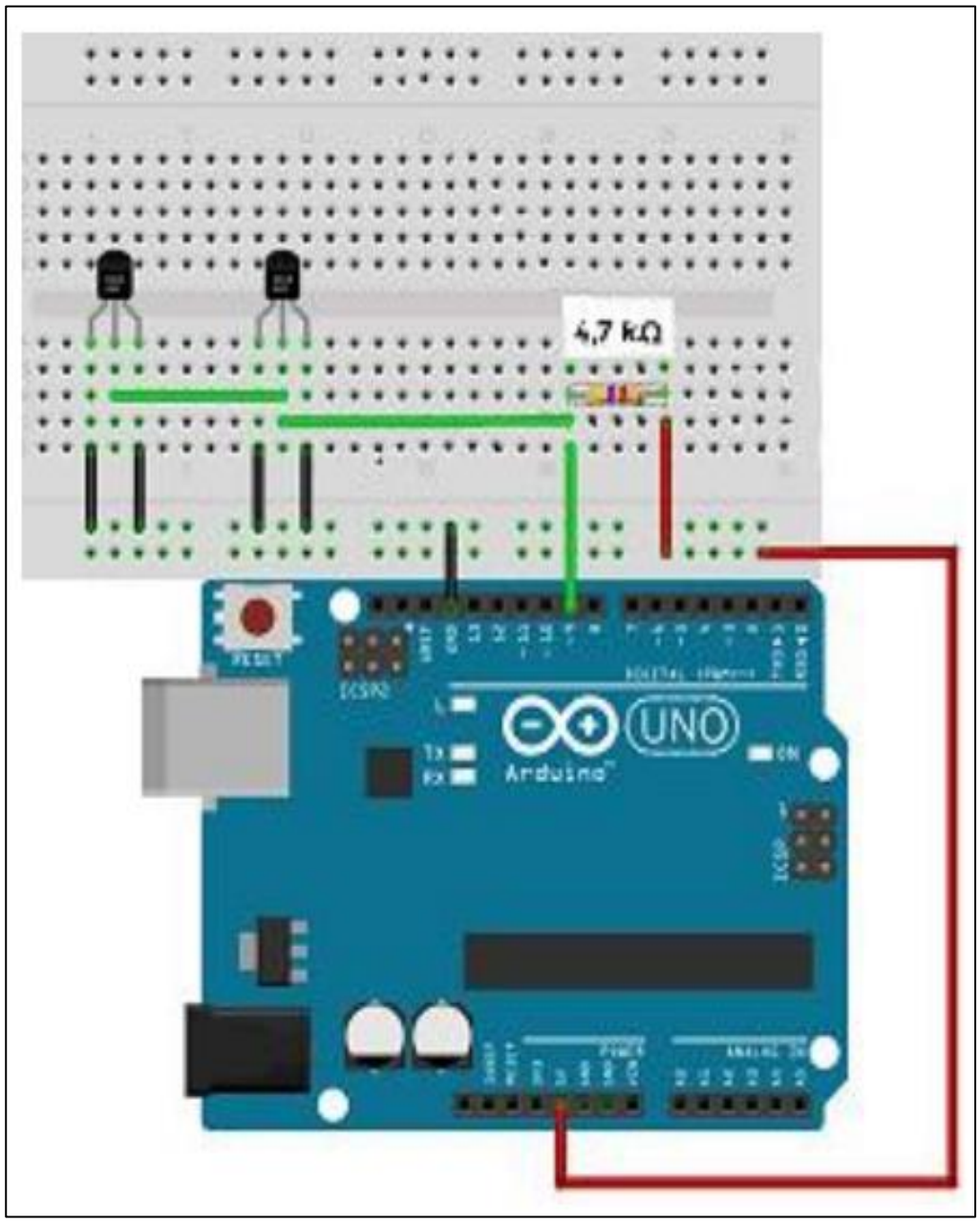


Figura 6

Conexión 1-Wire tipo Parásito



Toda la información del sensor de pulso se encuentra en el Anexo 2 a esta tesis, en este anexo se encuentra el Datasheet del sensor.

Raspberry Pi

Del tamaño de una tarjeta de crédito y con la capacidad de hacer cualquier cosa, casi como una computadora de escritorio, reproducir música, juegos, navegación web, procesamiento de texto, transmisión de videos, además de 27 pines GPIO, pines SPI, UART I2C, conexión a tierra y fuentes de energía entre 3.3V y 5V (Kumbhar, Mulla, Kanagi, & Shah, 2018).

Es un ordenador de bajo costo diseñado por la Raspberry PI Foundation, una organización sin fines de lucro con el objetivo de educar a niños y adultos en el campo de la informática. Requiere un teclado y un mouse estándar para el ingreso de comandos, una fuente de alimentación para el dispositivo low cost con un precio inicial a partir de los 35 euros según la página oficial del creador del dispositivo.

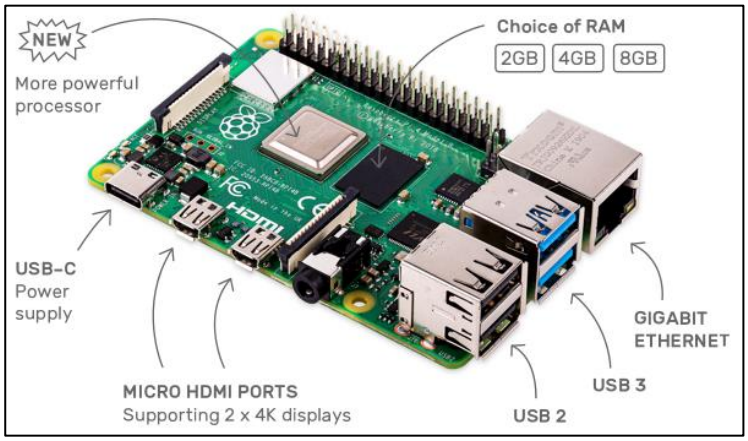
Raspbian es el sistema operativo por defecto, siendo este una distribución de Linux gratuita y de código abierto que favorece al bajo precio de la plataforma y permite una amplia gama de usos (Maksimović, Vujović, Davidović, Milošević, & Perišić, 2014).

La Raspberry PI para ejecutar la distribución de Linux derivada de Debian, Raspbian, necesita un hardware potente que facilite la programación de aplicaciones multitarea con threads, comunicación TCP/IP, esto se lo hace mediante sockets (Catalán Cantero & Blesa Gascón, 2016).

Entre otras características de la Raspberry que le diferencian de una computadora normal es que no tiene un disco duro, sino una tarjeta MicroSD, así como tampoco un case de fábrica, sino que se lo debe adquirir como accesorio. En la Figura 7 (Raspberrypi, 2020) se muestra la plataforma Rasberry.

Figura 7

Plataforma Raspberry PI



Arduino

El concepto de hardware y software libre para el uso de la sociedad se aplica en esta plataforma construida con prototipos electrónicos constituido básicamente en una placa microcontrolador y un lenguaje de programación para el desarrollo que soporta la entrada y salida de datos y señales, Arduino es open source es decir que cualquiera de sus programas es libre para copias, modificaciones y mejoras de cualquier usuario (Pedrera, 2017).

Basado en C/C++ el lenguaje de Arduino soporta las funciones de C y algunas de C++. Aunque debido a la transmisión de datos mediante el puerto de serie, Arduino, también es compatible con lenguajes como Java, Node.js, Python entre otros que soportan la comunicación serial. Incluso para otros lenguajes que no soportan la comunicación serial, mediante software de terceros es posible traducir los mensajes entre el emisor y el receptor para permitir la comunicación (Sánchez, 2012).

En el mercado actual, realizando varias consultas para la adquisición el valor de la placa Arduino, que es la más común encontrar en el mercado nacional, el precio medio es de 10 dólares.

Arduino tiene la finalidad de ser fácil de programación, aplicación, así como también puede ser configurado en cualquier plataforma Windows, Linux y Mac, en la Figura 8 (Sánchez, 2012) podemos observar la placa Arduino.

Figura 8

Placa Arduino

Node.js

Para construir aplicaciones de red escalables, JavaScript desarrolló un entorno de ejecución orientado a eventos asíncronos, es decir, si no hay trabajo que hacer JavaScript del lado del servidor permanece dormido hasta que haya una llamada a los servicios (Haro Valenzuela, Guarda, Peñaherrera Zambrano, & Ninahualpa Quiña, 2019).

Node.js tiene una arquitectura de I/O asíncrona basada en eventos sin bloqueos siendo una buena opción para aplicaciones en tiempo real y uso masivo de datos. Entre la comunidad de usuarios se encuentran LinkedIn, eBay, Microsoft

haciendo de Node.js una plataforma probada para desarrollar y ejecutar servidores para aplicaciones (Bermúdez-Ortega, Besada-Portas, López-Orozco, Bonache-Seco, & De la Cruz, 2015).

Node.js se ejecuta con el motor de JavaScript V8 de Google, con enlaces de C++ para llamadas al sistema (Davis, Williamson, & Lee, 2018).

MQTT

Message Queue Telemetry Transport (MQTT) basado en el método publish/subscribe, es un protocolo de comunicación de mensajes, ligero y fácil de implementar donde los clientes no se conocen, pero cada cliente conoce la dirección y puerto del bróker él cual se encarga de redirigir según el tópico a cada cliente subscrito al mismo. Cada tópico debe contener por lo menos un carácter y se puede generar una estructura jerárquica mediante el carácter "/" como podemos ver en estos ejemplos:

1)"departamento/dormitorio/luz",

2)"departamento/cocina/luz",

3)"departamento/sala/tv"

Además, existen dos caracteres usados como comodín "+" que actúa sobre un nivel es decir si se publica al tópico "departamento/+/luz" se van a enviar al primer y segundo tópico y el carácter "#" que actúa de forma multinivel, si se publica de la siguiente manera "departamento/#" los tópicos uno, dos y tres van a recibir el mensaje. MQTT maneja calidad en el servicio (QoS) que permite obtener una confirmación en la entrega de paquetes: QoS 0 el mensaje es mandado una sola vez; QoS 1 el mensaje puede enviarse por lo menos una vez es decir pueden existir

duplicados; QoS 2 mediante una comprobación se asegura que el mensaje se entregue una sola vez (Moreno Cerdà, 2018).

MongoDB

La base de datos NoSQL MongoDB desarrollada por la empresa 10gen bajo el concepto de código abierto y bajo la licencia AGPL(Affero General Public License) permite guardar una estructura de datos en formato JSON llamados documentos que se guardan en colecciones pueden contener un número indeterminado de estos documentos, haciendo referencia con una base de datos relacional las colecciones se comparan con tablas mientras que los documentos se los compara con filas de una tabla con la diferencia que en una base de datos relacional cada fila tiene el mismo número de campos, mientras que en MongoDB, cada documento puede tener diferentes campos, los cuales se pueden agregar, modificar, eliminar, renombrar sin afectar el modelo de los datos (Martín, Chávez, Susana Beatriz, Murazzo, María Antonia, Rodríguez, Nelson R, & Valenzuela, Adriana, 2015).

MongoDB es una base de datos multiplataforma desarrollada en C++ que funciona en sistemas operativos Linux, Windows, Mac OS y Solaris (Narváez, Grefa, Pablo Ronny Calapucha, Caisa, Marco Vinicio Tarco, & Guisñan, Pamela Alexandra Buñay, 2020).

Al reemplazar el concepto de fila por documento, permite representar relaciones jerárquicas complejas en un solo registro, es libre de esquema por lo que no obligan a que los datos tengan la misma forma, ofreciendo flexibilidad a los desarrolladores para trabajar con el modelo de datos, lo cual la hace flexible, escalable y rápida (Vele Zhingri, 2016).

El esquema dinámico que usa para guardar los documentos es similar a JSON llamado BSON, los cuales mantienen una lista ordenada de los documentos, cada documento tiene tres componentes: 1) nombre del campo, 2) tipo de dato y 3) valor (Castillo, Garcés, Navas, Segovia Jácome, & Naranjo Armas, 2017).

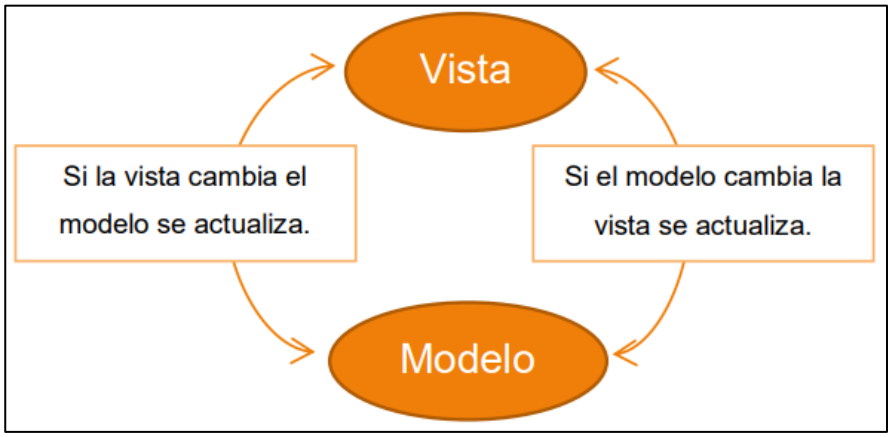
Angular IO

Angular IO, es un framework desarrollado por Google y contribuyentes de código abierto, con el objetivo de desarrollar aplicaciones web, de escritorio y móviles siendo compatible con navegadores modernos que soporten JavaScript. Su antecesor es AngularJS, el cual es un marco de referencia para el desarrollo de páginas SPA (Single Page Application), siendo completamente reescrito utilizando una versión extendida de JavaScript llamada TypeScript (Schmiedehausen, 2018).

Angular IO conocida comúnmente solamente como Angular, fue lanzada en el 2016 con la modificación de sintaxis de su predecesor, mejorando el rendimiento e incorporando TypeScript como lenguaje, en el 2017 se lanzó un cambio semántico del framework. Entre las características principales se tiene Two-way data binding que consiste en que los cambios se actualicen en tiempo real, es decir si en la vista existen cambios, también se actualice el modelo y viceversa; vistas y rutas, ya que se está trabajando en una SPA; expresiones que permite evaluar valores que se los colocan dentro de llaves dobles (Mafla Flores, 2019). En la Figura 9 (Mafla Flores, 2019) podemos ver cómo funciona el two-way data binding.

Figura 9

Two-way data binding



Estado del Arte

En (Terán Flores, 2019) construye un dispositivo e-Health de tamaño y peso reducido, conseguido gracias al uso de una tarjeta ESP 32 con pantalla OLED que incluye un módulo WiFi. El dispositivo construido permite obtener datos de temperatura ambiente, temperatura corporal y ritmo cardiaco. Primero desarrolló la caracterización de cada sensor de forma independiente obteniendo un error que posteriormente a través de software se corrigió el error mediante una compensación.

Para capturar la temperatura ambiente se utiliza el sensor LM35 que presentaba un error de 0.68% y que posterior a la compensación el error se redujo al 0.41%.

El sensor de temperatura corporal que se usa es el sensor Termistor NXFT15H103FA2B100 con un error inicial de 1.39% y que con la compensación se redujo al 0.08%.

El ritmo cardiaco se mide a través del sensor HRM-2511B con un error inicial de 2.77% y luego de la compensación reducido al 1.38%.

La información se transmite por medio de una conexión WiFi, cuando esta no está disponible se guarda la información en una tarjeta de memoria en un archivo con la extensión txt.

Para transmitir información se usa Mosquitto como bróker del protocolo MQTT debido a la rapidez en la transmisión de la información en tiempo real por ser un protocolo asíncrono y no es necesario esperar la respuesta del servidor para continuar con las mediciones.

La aplicación web está desplegada en una Infraestructura como Servicio (IaaS) con el proveedor Digital Ocean en donde se instaló Mosquitto, Node-RED y MySQL como base de datos.

La aplicación web está desarrollada con la herramienta Node-RED la cual permite el desarrollo de flujos para obtener la interfaz multiusuario. Asimismo, Node-RED se usa para guardar y consultar datos de la base de datos mediante nodos programados con sentencias SQL.

En (Castro Navarro, 2019) implementa un prototipo automático capaz de monitorear en tiempo real los signos vitales de los trabajadores del área de producción de Copeinca. El prototipo se compone de los siguientes componentes: un sensor de gas, un pulsómetro-oxímetro, un microcontrolador, una pantalla OLED I2C unificados en un prototipo capaz de obtener la información de la temperatura corporal, pulso cardiaco, oxigenación en la sangre, gases del medio ambiente como amoniaco NH3, dióxido de carbono CO2, monóxido de carbono CO los cuales se van a registrar en PPM (partes por millón).

El microcontrolador que usa es un µC PIC18F4550 capaz de convertir los parámetros captados con la finalidad de emitir una alerta de emergencia en caso de que alguna de las señales tenga valores de riesgo, también se usa es microcontrolador debido a que tiene pines disponibles para acoplamiento de más variables en el futuro, posee tecnología SMD y no es limitado.

Tomando como principio tener una autonomía y bajo consumo de energía, opta por elegir el sensor pulsómetro Max 30100 con la finalidad de obtener la frecuencia cardiaca mediante un LED rojo y un LED infrarrojo que iluminan alternamente durante cierto tiempo la zona de medición como un dedo o la muñeca detectando con fotodiodos la luz reflejada para convertirla en señal digital con un ADC (Conversor de señal analógica a digital) para acceder desde el microcontrolador mediante el bus I2C a un buffer donde se almacena la información.

Para capturar la temperatura corporal, se utiliza el mismo sensor MAX 30100 debido a que este posee un termómetro interno convirtiendo la señal recibida con un ADC (Conversor de señal analógica a digital) de manera similar a la obtención de la frecuencia cardiaca.

Como sensor de gases el autor usa el dispositivo MQ135 que permite capturar gases peligrosos y controlar la calidad de aire en un determinado ambiente.

Todos estos sensores se los une en una PCB (Placa de Circuito Impreso) logrando miniaturizar lo máximo posible para que sea usado en la muñeca del trabajador con la finalidad de medir los signos vitales sin perturbar las labores del usuario.

En base a las conclusiones del autor se tiene que el prototipo tiene una margen de error de más menos 1.14% respecto a pruebas comparando con

dispositivos comerciales teniendo en cuenta que no son portables y necesitan de cableado para funcionar a diferencia del prototipo planteado.

En (Balamba Camacho & Sacristán Vargas, 2019) se desarrolla un prototipo funcional de un servicio e-Health el cual permite de manera remota comunicar, almacenar y monitorear el estado de la presión arterial de pacientes crónicos hipertensos.

Mediante un proceso de ingeniería inversa se utiliza un tensiómetro comercial para obtener la información, adaptan un dispositivo bluetooth para enviar los datos a una base de datos creada en ThinkSpeak para ser mostrados en una página web desarrollada en Azure.

Los autores realizan una comparación entre tres tensiómetros escogiendo para el uso del prototipo al tensiómetro CK 1000 debido a que es un tensiómetro de muñeca con dimensiones y peso más bajo, asimismo se realiza la comparación entre tarjetas de desarrollo escogiendo Arduino UNO ya que los pines y la memoria disponible se ajustan a los requerimientos del proyecto aprovechando al máximo los recursos del dispositivo. También se usa un microcontrolador compatible con la tarjeta desarrollo con el nombre ATmega328P capaz de leer mientras escribe de bajo consumo de energía, así como de bajo costo, además un módulo Bluetooth HC-05 dado que puede trabajar en el rol de maestro esclavo ya que el tensiómetro solo envía la información hacia el dispositivo, la disponibilidad en el mercado y la documentación también fueron tomados en cuenta para el uso de este dispositivo. Otro dispositivo usado es un analizador lógico para mostrar las señales digitales recibidas, finalmente el uso de una tarjeta Shield GPRS GSM M95, la cual va a permitir enviar SMS en caso de tener medidas fuera de los rangos considerados normales.

Para la validación de datos, los autores realizan una comparación con un tensiómetro comercial de marca Omron concluyendo que existe una variación de más menos 5 puntos entre las medidas. El costo del proyecto es de $ 658.000 pesos colombianos.

En (Tejada Pardo, 2019) se propone un dispositivo que permite la captura de información con la concentración de monóxido de carbono y metano en un ambiente mediante una interfaz de bajo costo que mide constantemente y envía la información mediante WiFi a una aplicación en un celular en tiempo real para que el usuario pueda ver las mediciones en cualquier momento y saber si se encuentra en situación de riesgo en el caso de que se superen los rangos máximos permitidos aunque el prototipo tiene la capacidad de enviar una notificación que pueda brindar ayuda en caso de dicha situación.

Los dispositivos usados son una tarjeta de desarrollo ESP32 debido a los puertos ADC que posee, sensores MQs que son diseñados para detectar la presencia de partículas químicas en el aire, el sensor MQ4 capaz de detectar el gas metano y el MQ9 que puede detectar el Monóxido de carbono.

Los sensores envían la información mediante la plataforma Blynk disponible para dispositivos IOS y Android que permite el control del Arduino mediante una interfaz gráfica para controlar botones y el envío de mensajes y notificaciones.

Para mostrar al usuario que tan altos están los niveles de los gases se utiliza 10 leds que se van encendiendo de acuerdo con niveles de medida de concentración de los gases siendo el de color rojo cuando supera el límite máximo determinado por los autores.

Para validar los datos se compara el prototipo con el dispositivo comercial detector de gas combustible APROBE GSD600 con el sensor MQ4 y el dispositivo comercial detector de monóxido de carbono AS8700A con el sensor MQ9, obteniendo un error en algunos datos que supero el 10% teniendo que hace ajustes a los sensores para minimizar el error y obtener mejores resultados.

Como se puede observar, los trabajos nombrados anteriormente se concentran en hacer un dispositivo low cost que se comunique con aplicaciones para celulares, incluso se propone también el envío de SMS con una notificación de alerta; otro trabajo adapta un tensiómetro comercial para obtener los signos vitales; MQTT aparece como la tecnología usada para la transmisión de datos, mientras que ninguno de los trabajos se concentra en la obtención de los cuatro principales signos vitales sino que se concentran en base a los negocios específicos para los que se desarrollaron.

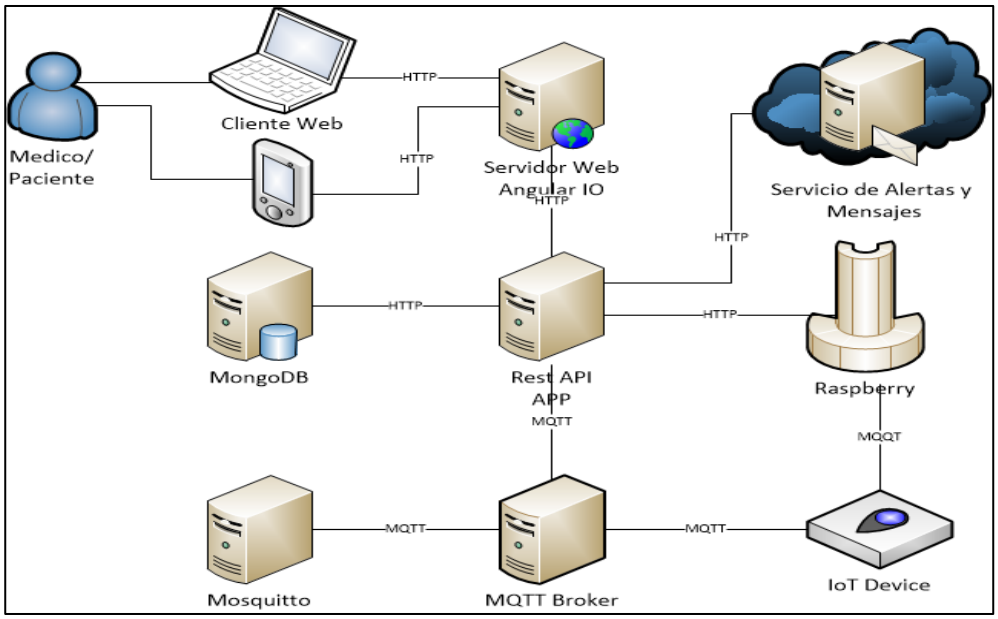
Capítulo 3: Desarrollo del Prototipo

En este capítulo se realiza el desarrollo del prototipo e-health basado en sistemas empotrados de bajo costo para monitoreo de signos vitales a través de internet. Para lo cual se propone el uso de dispositivos low cost que se van a encargar de recolectar la información al ser conectados a una Raspberry PI 4. La información recolectada se enviará hacia un bróker MQTT que comunicará la información a un servicio REST que permitirá almacenar los datos obtenidos en una base de datos MongoDB, así como realizar una validación de acuerdo con los rangos normales para enviar una alerta o mensaje en el caso de que haya valores fuera del rango normal. La información se podrá ver mediante una aplicación web a los múltiples usuarios médicos, pacientes, así como también se podrá obtener los signos vitales del paciente mediante la aplicación web.

Análisis

Figura 10

Arquitectura del prototipo



La arquitectura del prototipo consta de dos partes principales, que son el hardware y el software, cuya descripción se encuentra a continuación:

- Hardware: Sensores y equipos físicos que permiten obtener la información de los signos vitales.
- Software: Una aplicación backend que permite la transmisión y el almacenamiento de los valores obtenidos por los sensores y otra Frontend que permite visualizar los datos obtenidos e interactuar con los sensores para ponerlos en funcionamiento.

En la Figura 10 se puede observar el diagrama de la arquitectura propuesta del prototipo con las conexiones entre el hardware y software necesarios para el funcionamiento para obtener y mostrar los valores obtenidos de los signos vitales por los sensores. Un médico o un paciente que se conecta a un dispositivo sea un pc o un dispositivo móvil el cual hace una petición a un servidor web donde se encuentra alojada la aplicación web; esta se comunica con el servidor de aplicaciones para obtener la información de la base de datos y también se comunica con la Raspberry PI para hacer las peticiones a los sensores para obtener información; además se comunica con el servicio web para el envío de alertas.

Requerimientos de Hardware

En esta sección se describe el hardware que se necesita para la implementación del prototipo propuesto, en los que se incluye los siguientes dispositivos:

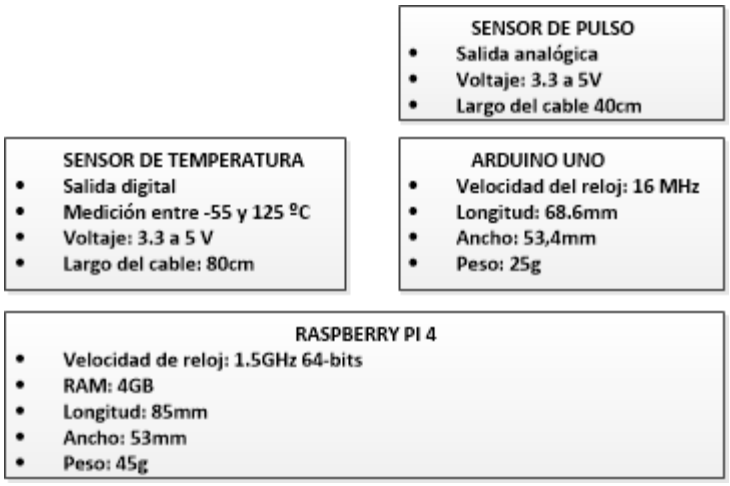
- Sensor de temperatura DS18b20 TO-92 encapsulado.
- Sensor de pulso Pulse Sensor.

- Raspberry Pi 4.
- Placa Arduino Uno.

En la Figura 11 podemos ver una representación de los componentes de hardware que se van a utilizar.

Figura 11

Requisitos del hardware



Requerimientos de Software

En esta sección se describe el software que se necesita para la implementación del prototipo propuesto para lo cual se detallan los requisitos funcionales y no funcionales del sistema.

Requisitos funcionales

Los requisitos funcionales, son las especificaciones acerca de que debe hace cada uno de los servicios que ofrece el sistema, en el caso del prototipo planteado se considera: aplicación Frontend, aplicación backend, base de datos, MQTT bróker, servicio de alertas y mensajes.

a) Aplicación Frontend

En la aplicación Frontend se debe poder visualizar los datos de monitoreo, realizar el monitoreo en tiempo real, administrar los parámetros de la aplicación, creación de usuarios, página de información relevante, dashboard.

- **Inicio de sesión:** El usuario ingresa su usuario y contraseña y de acuerdo con su rol, van a aparecer las opciones habilitadas.
- **Monitoreo en tiempo real:** El usuario puede realizar un monitoreo en tiempo real de los signos vitales disponibles.
- **Visualizar datos de monitoreo:** El usuario puede visualizar sus datos de los monitoreos realizados, se puede filtrar por fecha o por signo vital. En el caso de ser un usuario con el rol de Doctor, este puede ver los datos de todos los pacientes, se puede filtrar por fecha o por signo vital.
- **Creación de usuarios:** Un usuario con el rol de administrador puede crear usuarios para el uso de la aplicación. Se ingresa los necesarios para que un usuario puede funcionar correctamente en la aplicación.
- **Página de información:** La página de información contiene datos importantes acerca de la aplicación.
- **Dashboard:** En esta página se muestra un diagrama de barras y de líneas con la información de los signos vitales, un usuario con el rol de paciente solo puede ver sus datos, mientras que un doctor o administrador, puede ver la información de todos los pacientes.
- **Crear un cliente MQTT:** Crear un cliente MQTT para recibir la información de los sensores en la aplicación Frontend.

b) Aplicación Backend

En la aplicación Frontend se debe poder ofrecer los servicios web necesarios para creación, edición, borrado, consulta de los datos provenientes de la parte Frontend, integración con MQTT y la integración con el servicio de mensajes y alertas.

- **Servidor de aplicaciones:** Habilitar un servidor que permita la comunicación con la aplicación Frontend.
- **Token sesión:** Proveer de un token para la seguridad en la comunicación con la aplicación Frontend.
- **Conexión con la base de datos:** Permitir la conexión con la base de datos para consultar, editar, borrar, insertar datos en la base desde la aplicación Frontend.
- **Servicios REST:** Servicios REST que permitan crear, modificar, borrar, consultar datos para interactuar con las diferentes tablas disponibles en la base de datos.
- **Integración con MQTT:** Servicios de integración con MQTT para la transmisión de información y procesamiento de esta.
- **Integración con Servicio de alertas y mensajes:** Servicios que permitan enviar la información necesaria al Servicio de alertas y mensajes

c) Base de datos

Guardar la información procesada en el backend proveniente de la aplicación Frontend.

- Definir las tablas para el almacenamiento de la información.
- Crear las tablas para el almacenamiento de la información.
- Almacenar la información de las variables para cada tabla creada.

d) MQTT bróker

Bróker MQTT para la transmisión de la información desde los sensores hacia la aplicación.

- Levantar un servidor MQTT para la aplicación.
- Crear un cliente MQTT para el backend.
- Crear los tópicos para las subscripciones.
- Enviar la información de cada sensor por medio de los tópicos creados.

e) Servicio de alertas y mensajes

Envío de alertas y mensajes al médico con información relevante de las mediciones de los signos vitales.

- Enviar mensajes WhatsApp con la información personalizada de la aplicación hacia el Doctor.

Requisitos no Funcionales

Los requisitos no funcionales, son especificaciones para los aspectos visuales de la aplicación Frontend para la interacción del usuario sean estos pacientes doctores o el administrador. A continuación, se detallan los requisitos que van a permitir cumplir con las expectativas del Frontend del prototipo.

- Aplicación web amigable e intuitiva con los usuarios.
- Colores obscuros para no causar molestias en la visibilidad.
- Mensajes informativos y entendibles.
- Adaptabilidad a los distintos dispositivos de uso sean estos computadores, celulares o tablets.

Diseño del Prototipo

Para el diseño del prototipo, se ha divido en diseño de hardware y diseño de software.

Diseño de Hardware

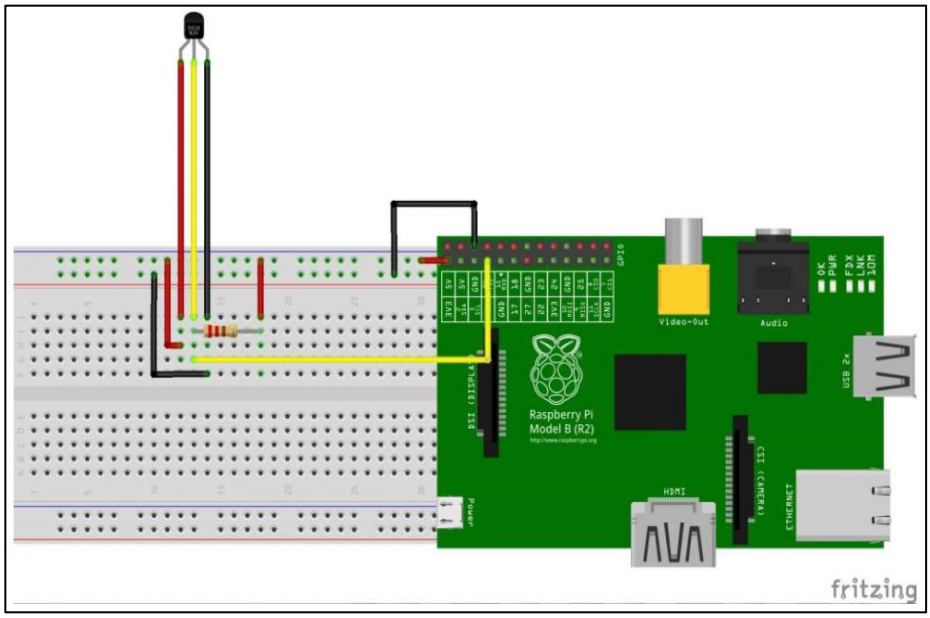
En la sección 3.2.1 se especifican los sensores que se van a usar en la construcción del prototipo, cada sensor tiene su configuración particular para su funcionamiento.

Sensor de temperatura DS18b20 TO-92 encapsulado

El sensor de temperatura utiliza el protocolo 1-Wire, la alimentación se lo hará mediante el pin de 3.3V o 5V que en la Raspberry corresponden a los pines 1 y 2 respectivamente, el cable de tierra se lo va a conectar al pin 6 de la Rasberry, mientras que el cable de transmisión de datos se debe conectar en el pin 4 que es el predefinido para este tipo de conexiones, para terminar se debe colocar una resistencia de 4K7 Ohm entre el pin positivo del sensor y el pin de datos; en la Figura 12 (Fritzing, 2020) se puede observar el diagrama para la conexión.

Figura 12

Diseño DS18B20

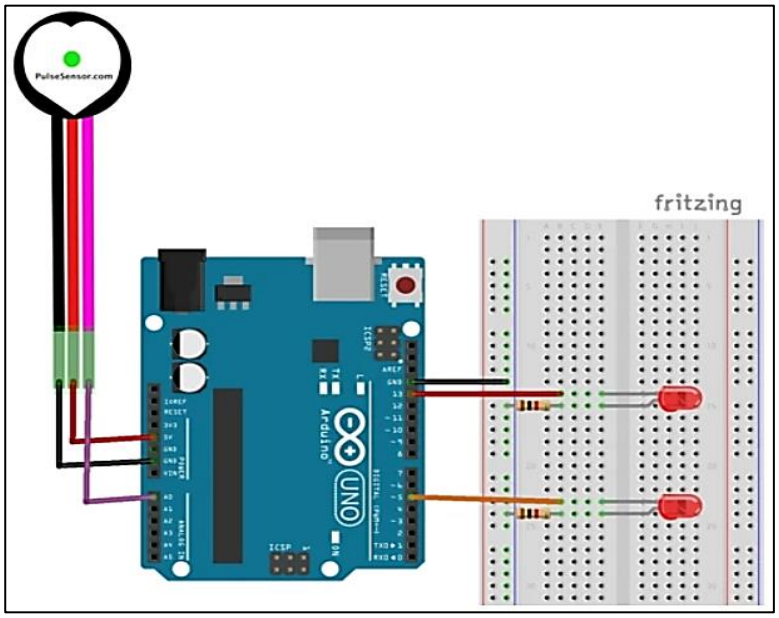


Sensor de Pulso, Pulse Sensor

El sensor de pulso, para su uso es necesario alimentarlo con un voltaje entre 3.3V y 5V que la placa Arduino va a ser la encargada de proveer la alimentación, el cable de tierra de la misma manera se lo debe conectar en la placa Arduino a uno de los pines GND y el pin de datos se lo debe conectar a cualquier entrada analógica de la placa Arduino, no es necesario conectar a ninguna resistencia debido a que el circuito integrado del sensor ya tiene todos los componentes necesarios para su funcionamiento. En la Figura 13 (Fritzing, 2020) se puede observar el diagrama para la conexión.

Figura 13

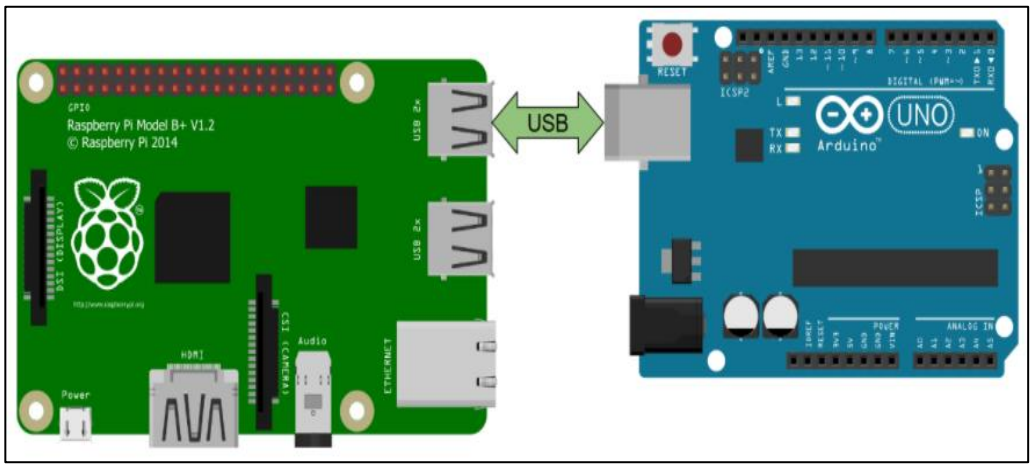
Diseño Sensor de Pulso



Placa Arduino UNO

Figura 14

Diseño Arduino + Raspberry PI



La placa Arduino Uno se debe conectar mediante cable serial con la Raspberry Pi 4 para transmitir la información obtenida del sensor de pulso conectado

al pin analógico A0, al pin de 5V y al pin GND, en el microchip va a contener un sketch (programa Arduino) capaz de obtener la información del sensor de pulso conectado a la placa y transmitir mediante el puerto serial la información obtenida para que sea enviada hacia la Raspberry Pi 4. Como podemos ver en la Figura 14 (Back-End, 2020) a la izquierda se encuentra la Raspberry Pi 4 mientras que la placa de color azul de la derecha es la Arduino Uno y en el centro la flecha bidireccional de color verde con la leyenda USB podemos ver cómo es la conexión entre ambas placas.

Raspberry Pi 4

La Raspberry Pi 4 provee pines GPIO, el pin GPIO 4 sirve para conectar el sensor de temperatura DS18B20, el pin 2 va a ser utilizado para suministrar corriente de 5V a los sensores mientras que el pin 6 se lo utiliza para la conexión GND.

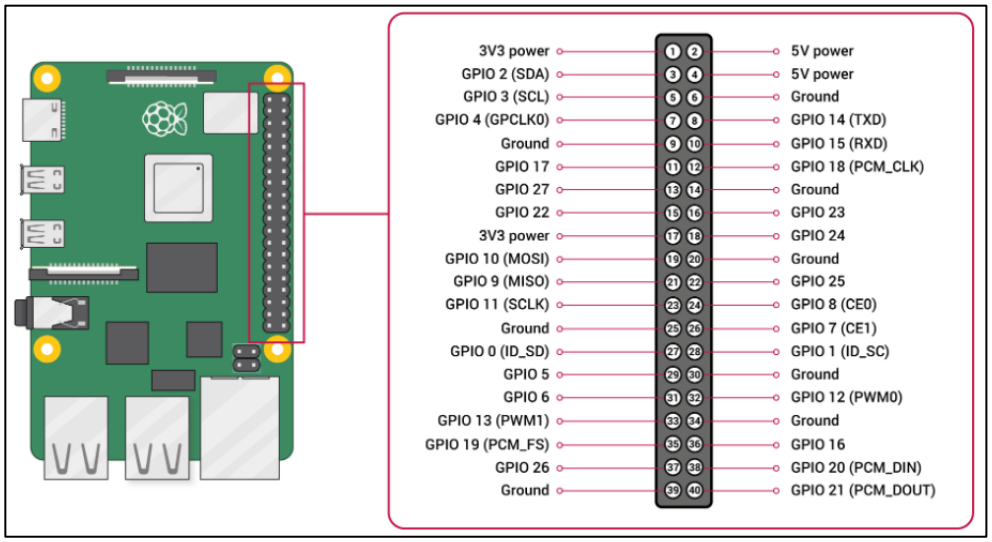
Mediante conexión USB, se debe recibir los datos enviados por la placa Arduino Uno mediante la conexión serial.

Además, se debe implementar un bróker MQTT que permita recibir las comunicaciones de los sensores.

Finalmente va a levantar un servidor REST API que permita alojar el backend de la aplicación.

Figura 15

Diseño Raspberry Pi 4



Como podemos ver en la Figura 15 (Raspberrypi, 2020), se encuentra la Raspberry Pi 4 en donde se puede observar el detalle de los pines disponibles en la última versión que van a permitir la interacción con los dispositivos sensores y la placa Arduino Uno.

Diseño de Software

En la sección 3.2.2 se especifican los requerimientos de software que son necesarios para la construcción de la parte del Backend y Frontend del prototipo.

Diseño de la Base de Datos

La base de datos debe contener una colección que permita guardar parámetros que van a funcionar en toda la aplicación para hacer que funcionalidades como el envío de mensajes sean dinámicos los datos y no sea necesario hacer cambios en toda la aplicación para actualizar la información.

Una tabla de usuarios donde se debe crear toda la información de cada usuario incluso la información para inicio de sesión y contacto. En esta tabla se va a encontrar incluso una relación con el rol que va a disponer el usuario.

Una tabla de roles, en donde se debe crear la información básica de un rol que va a permitir distinguir a los usuarios.

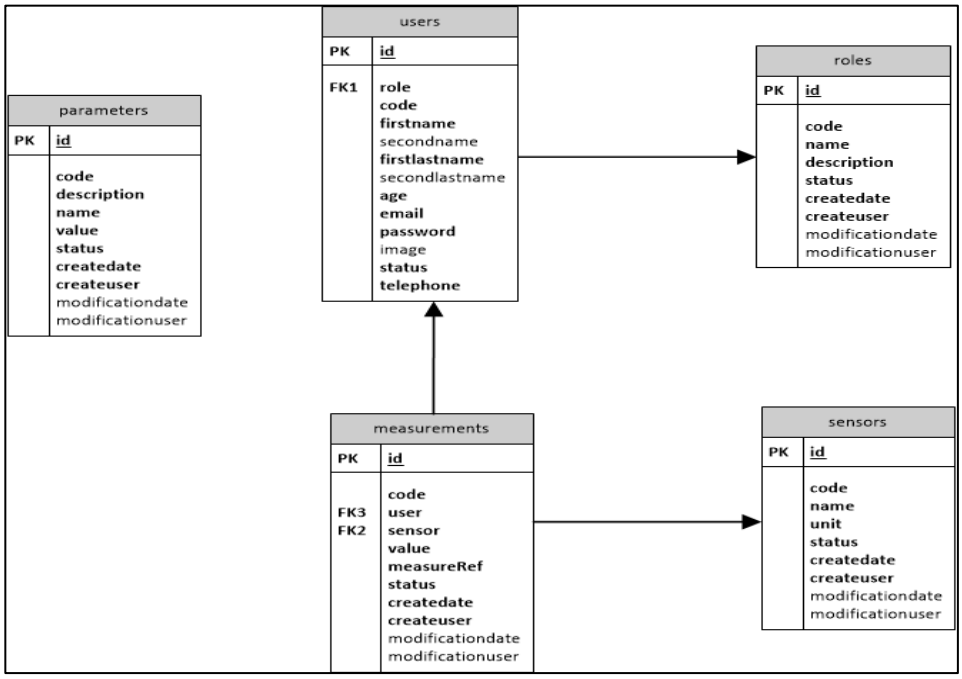
Una tabla de sensores, que va a permitir guardar la información de cada sensor que va a estar disponible en el prototipo con la configuración que va a permitir mostrar los valores recibidos.

Una tabla para las mediciones, en donde se guarde la información de las mediciones que se han recibido de los sensores, que sea capaz de distinguir la medición de cada sensor realizada.

En la Figura 16 se muestra un diseño del diagrama de la base de datos propuesto para el desarrollo del prototipo con las tablas especificadas y los principales campos de cada una de las mismas.

Figura 16

Diseño de la base de datos

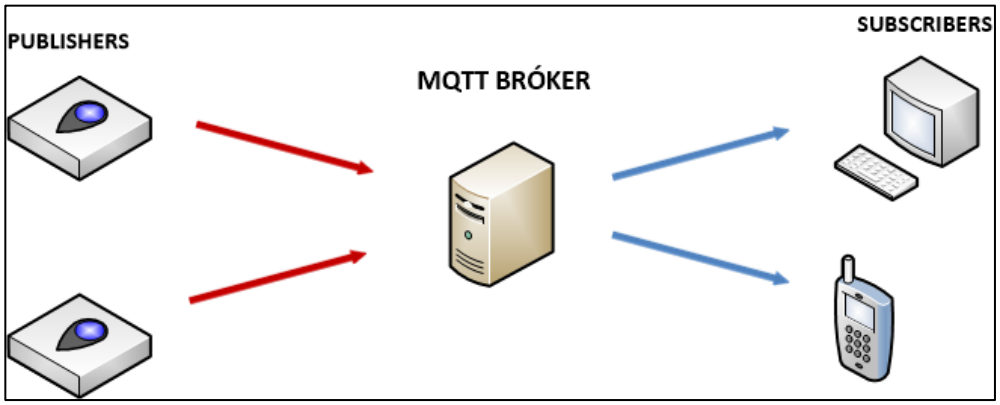


Diseño del Broker MQTT

El diseño del bróker MQTT, cada sensor va a publicar la información obtenida a través de un tópico al que deben estar subscritos los clientes que requieran mostrar la información de su interés. Como podemos ver en la Figura 17, a la parte izquierda se encuentran los sensores que van a ser los publicadores, en el centro se encuentra un servidor MQTT encargado de distribuir la información que recibe de los sensores a cada cliente subscrito al tópico que son los subscriptores que se encuentran en al lado derecho de la imagen.

Figura 17

Diseño del Bróker MQTT



Diseño de los Servicios REST

El diseño de los servicios REST, se debe considerar un servidor que permita conectarse con la base de datos obtener información y enviarla hacia el cliente a través de los métodos HTTP disponibles, enviar información en formato JSON con información útil para el usuario. Los servicios web entre los principales se encuentran:

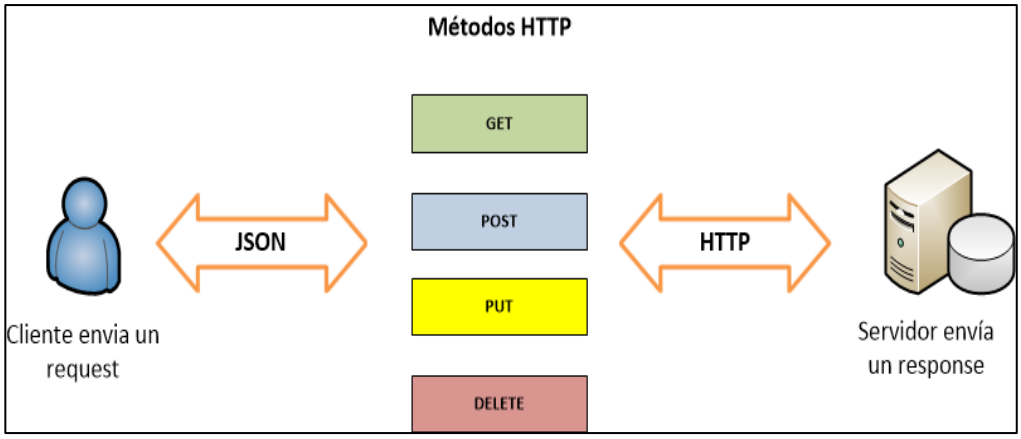
- Servicio para validar el ingreso al sistema.
- Servicio que permita generar un token con los datos de sesión.
- Guardar, Editar, Eliminar usuarios.
- Guardar, Editar, Eliminar sensores.
- Guardar, Editar, Eliminar roles.
- Guardar, Editar, Eliminar mediciones.
- Medir cada signo vital de manera independiente.
- Monitorear cada signo vital de manera independiente.

- Obtener la información necesaria para armar el dashboard de cada paciente.
- Obtener una lista de pacientes para realizar la medición de signos vitales.
- Obtener una lista de doctores para realizar la medición de signos vitales.
- Servicio que permita enviar un mensaje a los doctores.
- Validaciones de los datos recibidos.

En la Figura 18 podemos ver el diseño de los servicios REST, en donde a la izquierda el cliente hace un request a través de métodos HTTP enviando información necesaria en formato JSON y el servidor debe responder información obtenida desde la base de datos o a desde un servicio de validaciones.

Figura 18

Diseño de los servicios web

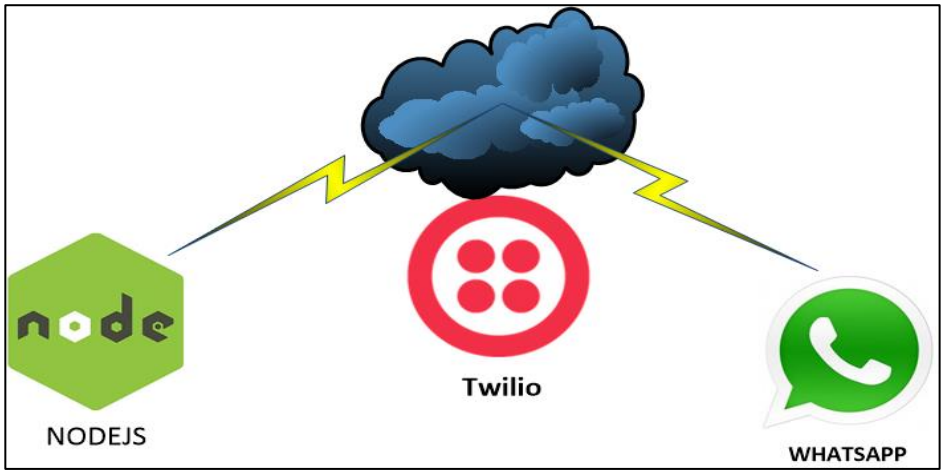


Diseño del Servicio de Alertas y Mensajería

El diseño del servicio de mensajería se envían una petición desde el Backend con información como mensaje, destinario hacia el API de Twilio (Twilio, 2020), el cuál envía un WhatsApp al número de destino que resuelve de la información enviada por él Backend. En la Figura 19 a la izquierda se encuentra la aplicación Backend, en el centro la nube, el servicio de Twilio y a la derecha, el WhatsApp del destinatario.

Figura 19

Diseño del servicio de alertas



Diseño de Ingreso al Sistema

El diseño de ingreso al sistema consta de una pantalla con un Login, el cual debe contener un campo en el que se ingrese el usuario, en este caso el correo electrónico con el que se registró el usuario y otro campo con la contraseña, al presionar Login se realiza una validación de los datos ingresados y si ingresa, se debe generar un token para validar la sesión y el uso de la aplicación con datos controlados con seguridad. Al ingresar debe mostrar a la izquierda un menú con los

accesos a las diferentes opciones permitidas de acuerdo con el rol del usuario que ingresó al sistema.

En la Figura 20 se muestra el diseño de la pantalla principal, donde en la parte de arriba se encuentra un input de tipo texto que permita ingresar los datos del usuario, mientras que, en el segundo input, se ingresa la contraseña, al final se encuentra el botón de Login.

Figura 20

Ingreso al sistema

Diseño del Módulo de Mediciones

El diseño para el módulo de mediciones se debe poder seleccionar al doctor que solicita la medición de signos vitales, automáticamente se selecciona el paciente que ingreso al sistema en el caso de que tenga con el rol de paciente, en el caso de que tenga el rol de médico, puede escoger cualquiera de los pacientes registrados en la aplicación.

Cuando es un administrador puede tener una checkbox que le permita ingresar manualmente los datos de cada signo vital disponible.

Cada signo vital debe tener un botón Medir, encargado de hacer una medición del signo vital seleccionado.

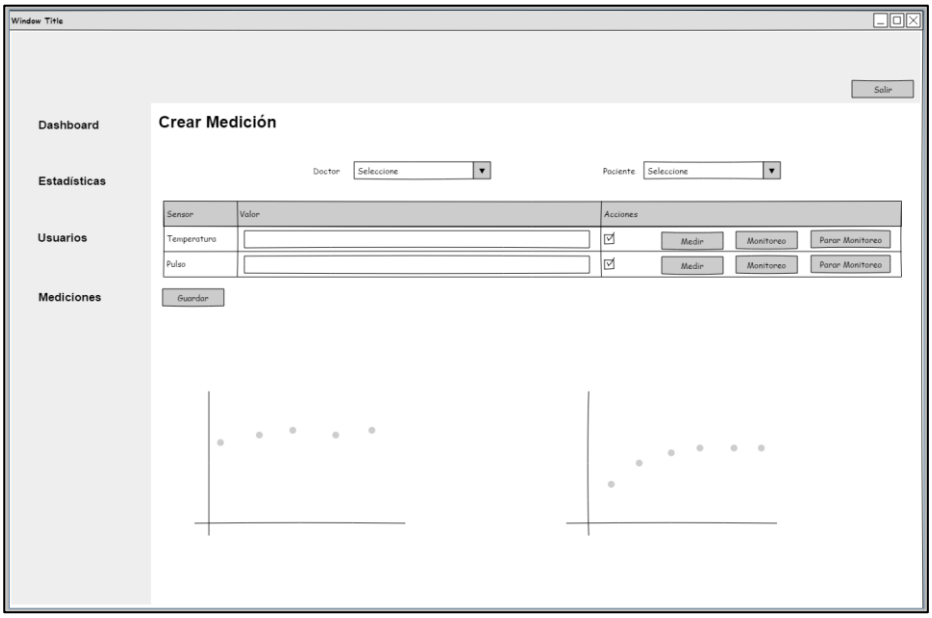
Cada signo vital debe tener un botón para iniciar el monitoreo y otro para finalizar el monitoreo en tiempo real. La información del monitoreo se debe mostrar en un gráfico para dar un seguimiento visual.

Una vez obtenida la información requerida se debe habilitar el botón guardar que va a enviar los datos registrados hacia el servicio del Backend, que se va a encargar de hacer las validaciones respectivas para guardar todos los datos en la base y en el caso de que sea necesario, enviar un mensaje de WhatsApp.

En la Figura 21 se puede ver a la izquierda el menú de navegación, en el centro los campos necesarios para el ingreso de información y los botones disponibles para medir y monitorear los signos vitales.

Figura 21

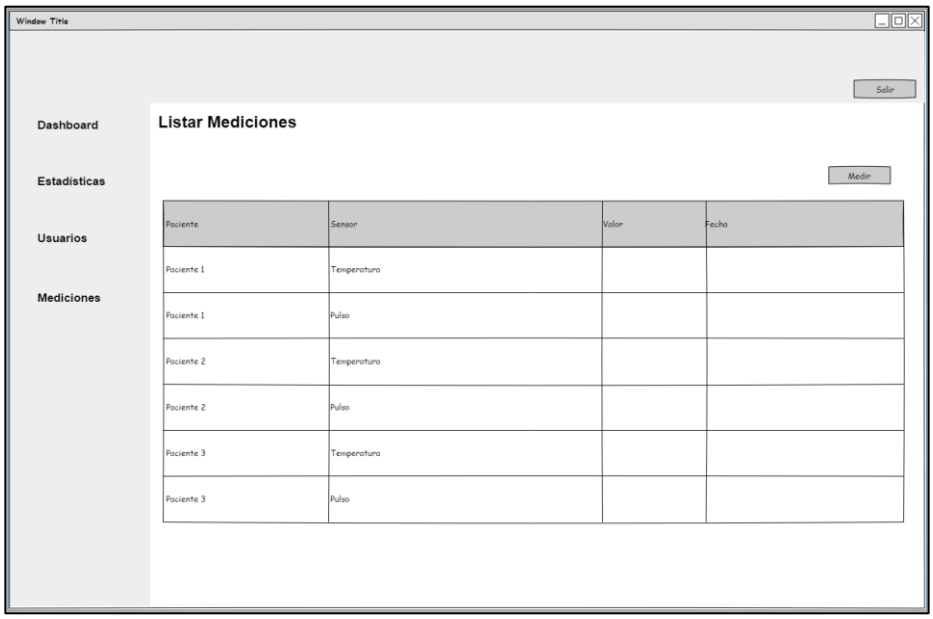
Crear Medición



Para mostrar los datos guardados, se debe mostrar una en una pantalla, una tabla con las columnas Paciente, Sensor, Valor del sensor, fecha de la medición. En la parte superior de la tabla se encuentra un botón que va a redirigir a la pantalla de mediciones. En la Figura 22 se puede ver el diseño de la tabla con los campos necesarios que se deben mostrar.

Figura 22

Listar Mediciones



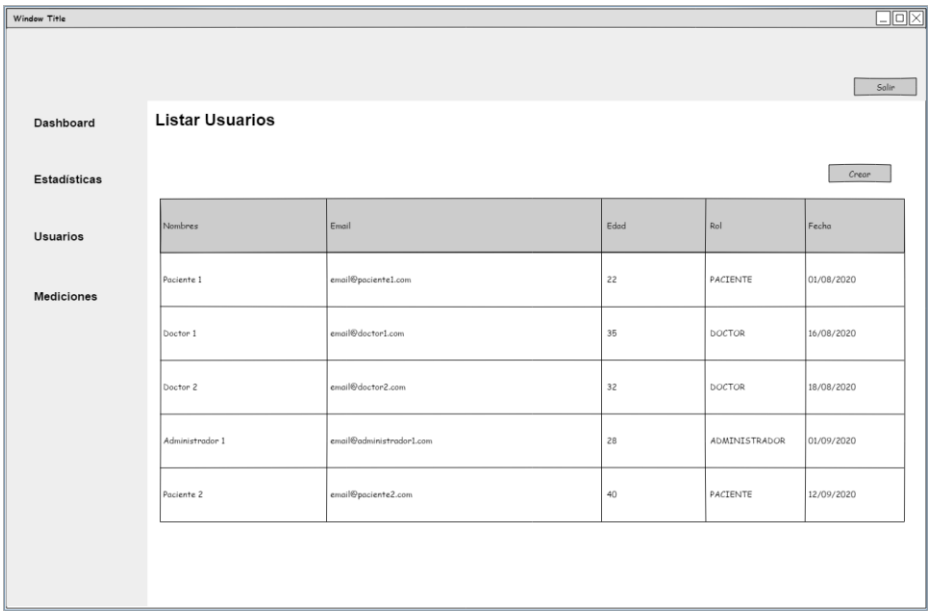
Diseño del Módulo de usuarios

El diseño del módulo de usuarios debe mostrar una tabla con la lista de todos los usuarios registrado en la base de datos, debe mostrar una tabla con la información principal como nombre, correo electrónico, edad, rol y fecha de registro, en la parte de arriba de la tabla debe haber un botón que permita acceder a la pantalla de creación de un usuario.

En la Figura 23 se puede ver a la izquierda el menú para acceder a las funcionalidades del sistema, en el centro de la pantalla se puede ver la tabla con las columnas propuestas para visualizar la información de los usuarios.

Figura 23

Listar Usuarios

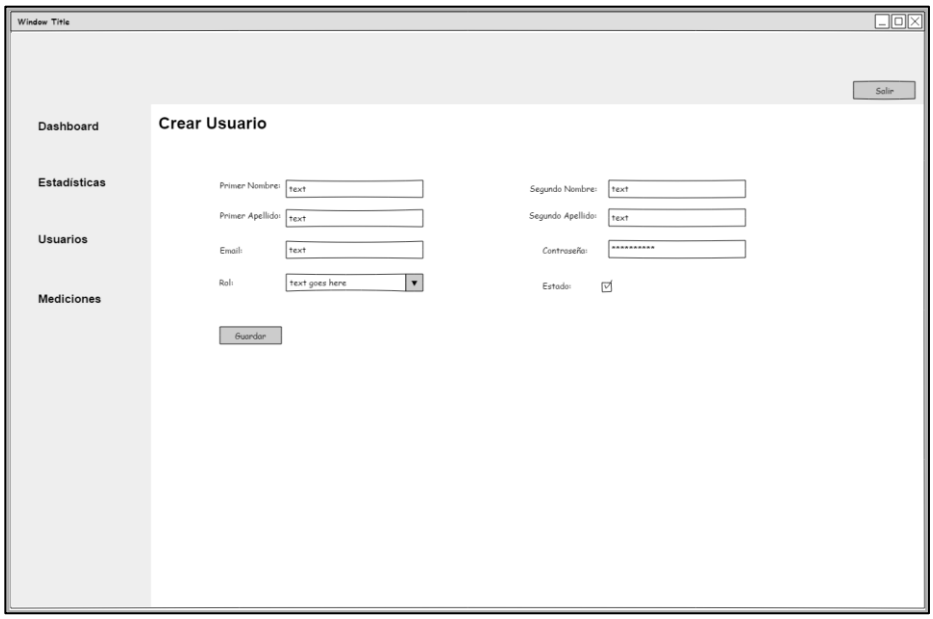


En la pantalla de creación de usuarios, se debe mostrar inputs que permitan el acceso de información principal como nombres, apellidos, correo electrónico, edad, contraseña, rol y estado de registro.

En la Figura 24 se muestra a la izquierda el menú para acceder a las funcionalidades del sistema, mientras que en el centro de la pantalla se muestra el formulario de acceso con los campos principales como primer nombre, segundo nombre, primer apellido, segundo apellido, email, contraseña, rol y un checkbox que es para el estado del registro, en la parte final del formulario un botón que se va a activar cuando todos los campos requeridos se encuentren con datos y va a permitir enviar la información hacia los servicios del Backend destinados para guardar la información de los usuarios.

Figura 24

Crear Usuario



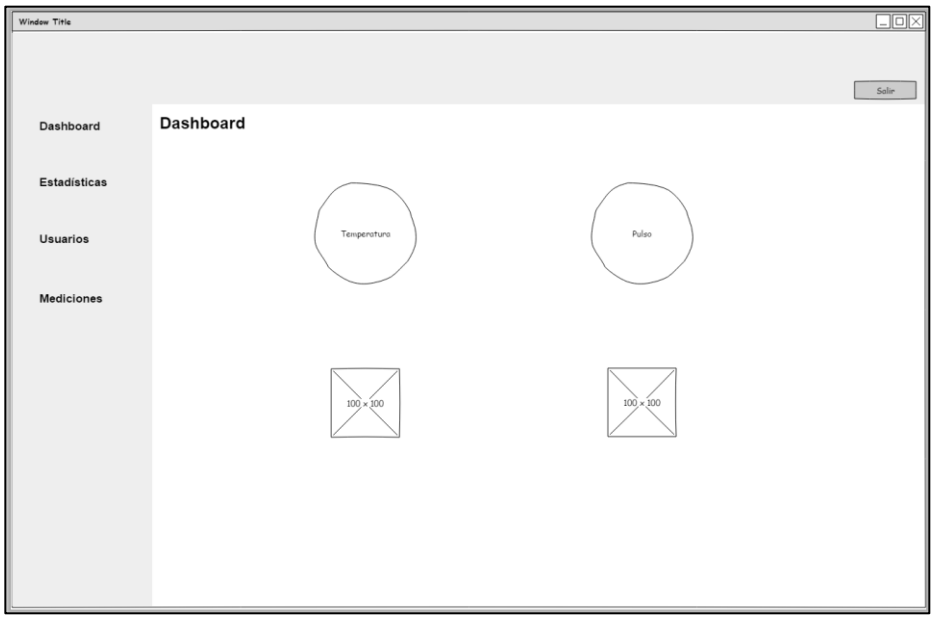
Diseño del Módulo de Dashboard y Estadísticas

El diseño de la pantalla del Dashboard debe mostrar información de los principales indicadores de los signos vitales transmitiendo información que se transforme en conocimiento para la persona interesada.

En la Figura 25 podemos ver en la parte izquierda se encuentra el menú de navegación disponible para la aplicación, en la parte del centro, se va a mostrar los indicadores principales de cada sensor disponible y en la parte de abajo imágenes, indicadores, notificaciones o alertas que indiquen información relevante para los profesionales. Como por ejemplo una alerta de color verde cuando existan datos exitosos, azul para cuando se requiere mostrar una información, amarillo cuando haya datos de advertencia y rojo para cuando existan datos alerta, error o peligro.

Figura 25

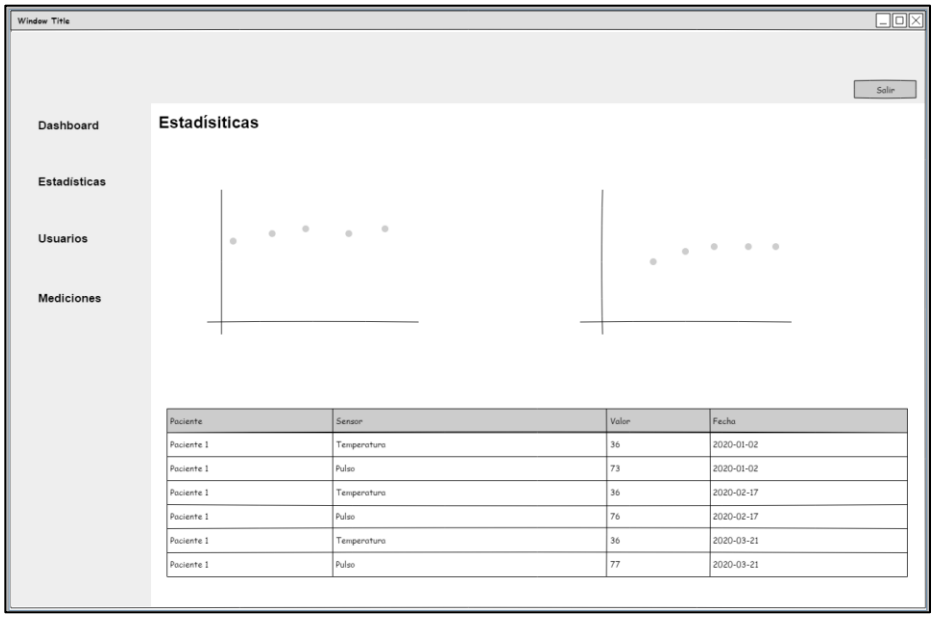
Dashboard



Para el diseño de la pantalla de estadísticas, se debe mostrar un gráfico de líneas y de barras que muestre el historial de las mediciones de los signos vitales, además una tabla con todos los datos de las mediciones. En la Figura 26 podemos ver en la parte izquierda se encuentra el menú de navegación disponible para la aplicación, y en el centro de la pantalla se muestra en la parte superior los gráficos, mientras que en la parte inferior la tabla con la información de las mediciones.

Figura 26

Estadísticas



Implementación del Prototipo

En la fase de implementación, se toma como referencia la fase del diseño del para armar el prototipo y dar funcionalidad a la obtención de los signos vitales, generar el código de la aplicación Backend y Frontend, implementar el servidor MQTT y consumir los servicios del API de Twilio para enviar mensajes WhatsApp.

Durante el proceso de implementación de cada uno de los diseños propuestos, se generó varios incrementos en el desarrollo que permitieron añadir funcionalidades y requisitos a la aplicación.

Para la implementación del prototipo se definieron las siguientes actividades:

- Configuración de la Raspberry Pi 4.
- Conectar el sensor de pulso a la placa Arduino UNO.
- Escribir el sketch de Arduino para obtener el valor del sensor de pulso.

- Conectar el sensor de temperatura DS18B20 a la Raspberry Pi 4.
- Instalar Node.js en la Raspberry Pi 4.
- Escribir el JavaScript para obtener la información del sensor DS18B20.
- Escribir el JavaScript para obtener la información del sensor de pulso a través del puerto serial de la Raspberry Pi 4 y la placa Arduino UNO.
- Configuración de la base de datos MongoDB.
- Configuración del servidor MQTT.
- Desarrollar el Backend.
- Desarrollar el Frontend.

Configuración de la Raspberry Pi 4

Para configurar la Raspberry Pi 4, es importante conocer el sistema operativo instalado, en este caso se encuentra la distribución Raspbian. En la Figura 27 se encuentra la información del sistema operativo instalado.

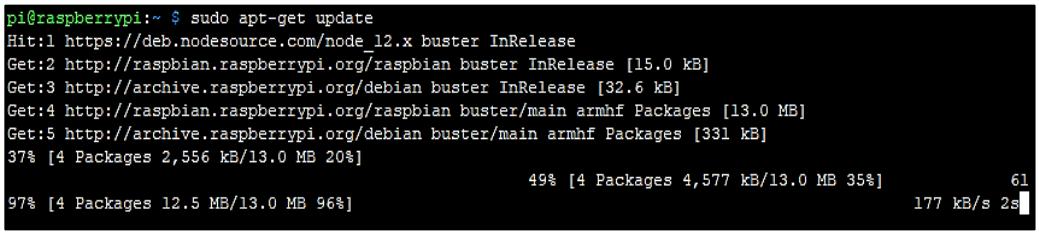
Figura 27

Sistema operativo instalado en la Raspberry Pi 4

La Primero se debe actualizar la información de los paquetes de todas las fuentes configuradas, como se puede observar en la Figura 28.

Figura 28

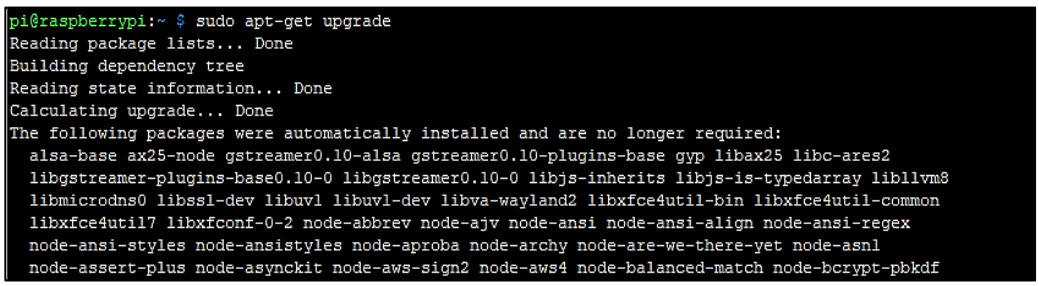
Actualizar la información de los paquetes



A continuación, se instala las últimas versiones disponibles para actualizar todos los paquetes previamente instalados en el sistema operativo.

Figura 29

Instalar la última versión de los paquetes

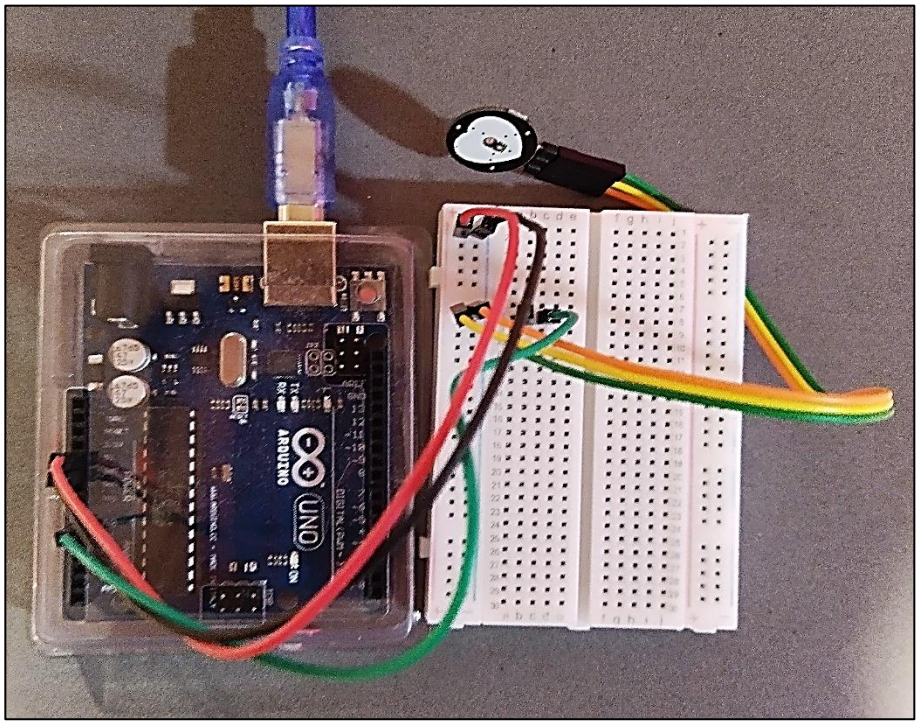


Conectar el sensor de pulso a la placa Arduino UNO

El sensor de pulso, se lo conecta a la placa Arduino UNO de la siguiente manera: el cable rojo se encarga de transmitir la corriente de 5V, el cable negro es GND, mientras que el cable verde es el encargado de la transmisión de datos desde el sensor hacia el pin A0 de la placa Arduino UNO, también se puede observar el cable para la conexión serial que será el encargado de enviar la información hacia la Raspberry Pi, en donde se encuentra el script que va a permitir obtener la información del sensor y enviar la misma hacia el Frontend como podemos ver en la Figura 30.

Figura 30

Conexión del Sensor de Pulso



Escribir el sketch de Arduino para obtener el valor del sensor de pulso

El sketch de Arduino comienza con la inclusión de la librería del sensor de pulso, la definición de las variables que se utilizan como por ejemplo PulseWire, que es el pin Analógico al que va a estar conectado el sensor de pulso. Se crea una instancia del sensor de pulso llamada pulseSensor; en la sección de configuración del script es decir en el setup (), antes de todo se inicializa la frecuencia de transmisión que en este caso de es de 9600 y se configura la variable pulseSensor con las opciones requeridas.

Una vez que se ha realizado la configuración del pulseSensor, en la sección del loop (), se invoca a la función lógica del sensor que va a permitir obtener los latidos por minuto, una vez obtenido esta información se la imprime en el puerto

serial para que sea transmitido, cada iteración tendrá un retorno configurado de 200 milisegundos.

En la Figura 31 se encuentra el sketch escrito en IDE por defecto para la placa de Arduino.

Figura 31

Sketch para obtener el BPM

getting_BPM §

```
/*  Getting_BPM_to_Monitor prints the BPM to the Serial Monitor, using the least lines of code and PulseSensor Library.
    Tutorial Webpage: https://pulsesensor.com/pages/getting-advanced

  --------Use This Sketch To----------------------------------------
  1) Displays user's live and changing BPM, Beats Per Minute, in Arduino's native Serial Monitor.
  2) Print: "♥  A HeartBeat Happened !" when a beat is detected, live.
  2) Learn about using a PulseSensor Library "Object".
  4) Blinks LED on PIN 13 with user's Heartbeat.
  ----------------------------------------------------------------------*/

#define USE_ARDUINO_INTERRUPTS true    // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h>     // Includes the PulseSensorPlayground Library.

//  Variables
const int PulseWire = 0;       // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
const int LED13 = 13;          // The on-board Arduino LED, close to PIN 13.
int Threshold = 550;           // Determine which Signal to "count as a beat" and which to ignore.
PulseSensorPlayground pulseSensor;  // Creates an instance of the PulseSensorPlayground object called "pulseSensor"
void setup() {
  Serial.begin(9600);          // For Serial Monitor
  // Configure the PulseSensor object, by assigning our variables to it.
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13);       //auto-magically blink Arduino's LED with heartbeat.
  pulseSensor.setThreshold(Threshold);
  if (pulseSensor.begin()) {
    Serial.println("We created a pulseSensor Object !");  //This prints one time at Arduino power-up,  or on Arduino reset.
  }
}

void loop() {
  int myBPM = pulseSensor.getBeatsPerMinute();  // Calls function on our pulseSensor object that returns BPM as an "int".
  // "myBPM" hold this BPM value now.
  if (pulseSensor.sawStartOfBeat()) {            // Constantly test to see if "a beat happened".
    Serial.println("♥  A HeartBeat Happened ! "); // If test is "true", print a message "a heartbeat happened".
    Serial.print("BPM: ");                        // Print phrase "BPM: "
    Serial.println(myBPM);                       // Print the value inside of myBPM.
  }
  delay(200);                    // considered best practice in a simple sketch.
}
```

Conectar el sensor de temperatura DS18B20 a la Raspberry Pi 4

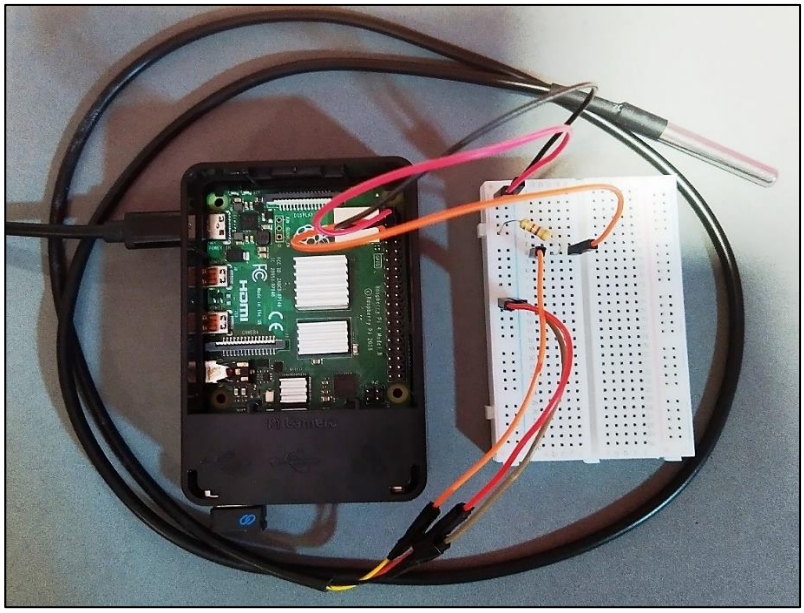
El sensor de temperatura, se lo conecta a la Raspberry Pi 4 con el cable de color amarillo en el pin GPIO 4, el cual es exclusivo para el uso de dispositivos 1-Wire, mientras que el cable de color rojo se lo conecta en el pin de alimentación de 5V, y por último el cable de color negro, se lo conecta con el pin GND.

El cable de color amarillo, encargado de transmitir la información, se debe poner una resistencia de 4.7 kΩ entre el paso de datos y la alimentación de 5V.

En la Figura 32 podemos ver la Rasberry Pi 4 que se conecta con un proto en donde se encuentra la resistencia y los cables del sensor de temperatura.

Figura 32

Conexión del sensor DS18B20



Instalar Node.js en la Raspberry Pi 4

Para instalar Node.js se utiliza el comando que se muestra en la Figura 33, es necesario tener permisos de administrador.

Figura 33

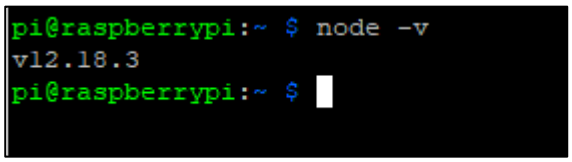
Instalar Node.js

```
pi@raspberrypi:~ $ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
nodejs is already the newest version (12.18.3-1nodesource1).
The following packages were automatically installed and are no longer required:
```

Una vez terminada la instalación se puede comprobar la información ejecutando el comando que se puede ver en la Figura 33.

Figura 34

Comprobar la versión instalada de Node.js



Escribir El JavaScript Para Obtener La Información Del Sensor DS18B20

Para obtener la información del sensor DS18B20, se debe importar el paquete ds18b20-raspi, el cual es una librería para Node.js con las funcionalidades que permiten obtener la información desde el sensor. Se crea una clase en la que se inicializa en el constructor el objecto con el que se va a trabajar. Se implementa una función getTemperature, que obtiene los datos del sensor y los publica por MQTT.

La función monitoringTemperature es la encargada de obtener los datos del sensor en modo de monitoreo y los publica por MQTT. Como podemos ver en la Figura 35 se encuentra la clase escrita en JavaScript para Node.js.

Figura 35

JavaScript para obtener temperatura

```
const sensor = require( id: 'ds18b20-raspi');

class TemperatureGPIO {
    constructor() {
        this.ds18b20RaspiSen = sensor;
    }

    getTemperature(mqttHandler, topic, quantity) {
        let tempList = this.initDs18b20RasPiSensor(quantity);
        mqttHandler.getClient().publish(topic, tempList.toString());
    }


    monitoringTemperature(mqttHandler, topic) {
        let temperature = this.initMonitoringDs18b20RasPiSensor(Date.now());
        mqttHandler.getClient().publish(topic, temperature);
    }

    initDs18b20RasPiSensor(quantity :number = 20) {
        console.log('initDs18b20RasPiSensor');
        let tempArray = [];
        for (let i = 0; i < quantity; i++) {
            tempArray.push(this.initMonitoringDs18b20RasPiSensor(i));
        }
        return tempArray;
    }

    initMonitoringDs18b20RasPiSensor(index) {
        let deviceId = '28-0301a2790e70';
        let tempC = this.ds18b20RaspiSen.readC(deviceId, digits: 2);
        let tempF = this.ds18b20RaspiSen.readF(deviceId, digits: 2);
        console.log(`Temp: ${tempC}°C - ${tempF}°F`);
        return JSON.stringify( value: {tempC: tempC, tempF: tempF, index: index});
    }


}

module.exports = TemperatureGPIO;
```

Escribir El JavaScript Para Obtener La Información Del Sensor De Pulso A Través Del Puerto Serial De La Raspberry Pi 4 Y La Placa Arduino UNO

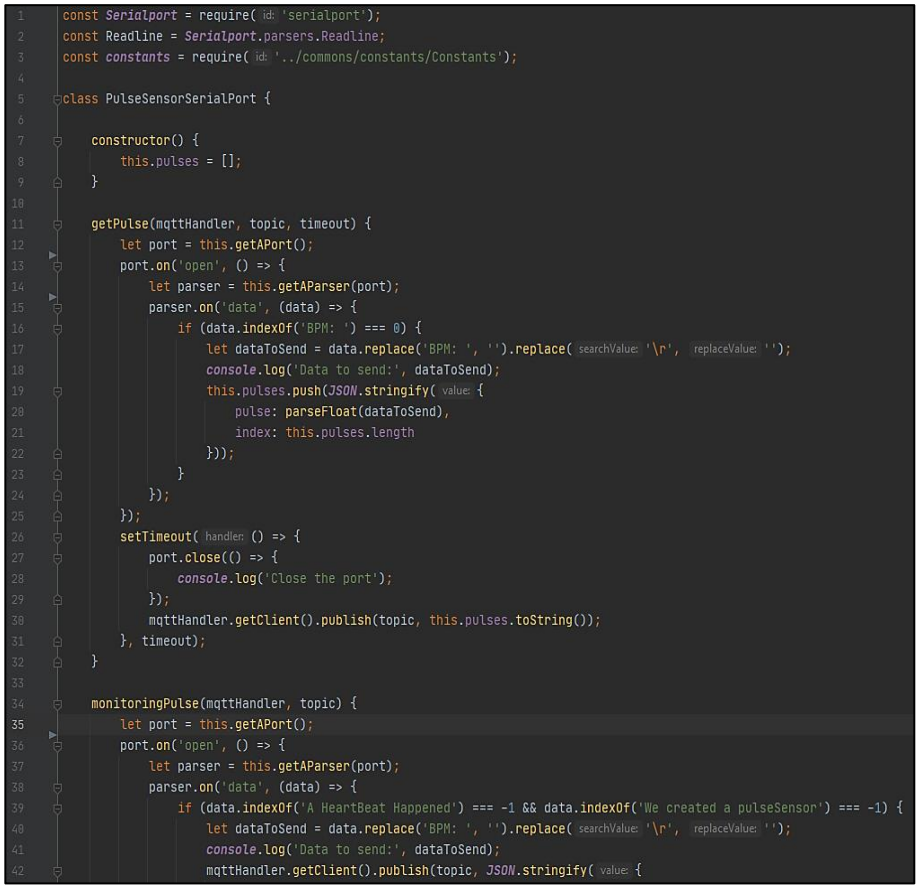
Para obtener la información del sensor de pulso, se debe importar el paquete serialport, el cual es una librería para Node.js con las funcionalidades que permiten obtener la información desde el puerto serial. Se crea una clase en la que se inicializa en el constructor el objecto con el que se va a trabajar. Se implementa una función getPulse, la que se va a encargar de obtener los datos provenientes del puerto serial

para publicarlos por MQTT y otra función monitoringPulse encargada de obtener los datos del puerto serial y publicarlos por MQTT para el monitoreo.

Como podemos ver en la Figura 36 se encuentra la clase escrita en JavaScript para Node.js.

Figura 36

JavaScript para obtener el pulso



```
const Serialport = require( id: 'serialport');
const Readline = Serialport.parsers.Readline;
const constants = require( id: '../commons/constants/Constants');

class PulseSensorSerialPort {

    constructor() {
        this.pulses = [];
    }

    getPulse(mqttHandler, topic, timeout) {
        let port = this.getAPort();
        port.on('open', () => {
            let parser = this.getAParser(port);
            parser.on('data', (data) => {
                if (data.indexOf('BPM: ') === 0) {
                    let dataToSend = data.replace('BPM: ', '').replace( searchValue: '\r',  replaceValue: '');
                    console.log('Data to send:', dataToSend);
                    this.pulses.push(JSON.stringify( value: {
                        pulse: parseFloat(dataToSend),
                        index: this.pulses.length
                    }));
                }
            });
        });
        setTimeout( handler: () => {
            port.close(() => {
                console.log('Close the port');
            });
            mqttHandler.getClient().publish(topic, this.pulses.toString());
        }, timeout);
    }

    monitoringPulse(mqttHandler, topic) {
        let port = this.getAPort();
        port.on('open', () => {
            let parser = this.getAParser(port);
            parser.on('data', (data) => {
                if (data.indexOf('A HeartBeat Happened') === -1 && data.indexOf('We created a pulseSensor') === -1) {
                    let dataToSend = data.replace('BPM: ', '').replace( searchValue: '\r',  replaceValue: '');
                    console.log('Data to send:', dataToSend);
                    mqttHandler.getClient().publish(topic, JSON.stringify( value: {
```

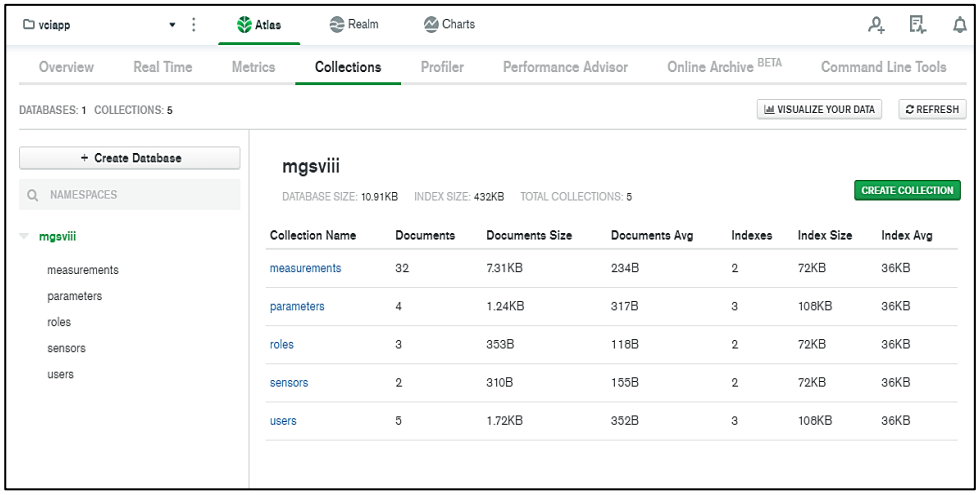
Configuración de la base de datos MongoDB

Para el almacenamiento de la información con los datos del monitoreo se utilizó la base de datos MongoDB, al ser una base de datos NoSQL, rápida, multiplataforma. La base de datos se ha desplegado en un servidor en la nube MongoDB Atlas (MongoDB, 2020) aprovechando la opción gratuita para aprender y

desarrollar aplicaciones pequeñas. La base de datos denominada “mgsviii” consta de cinco tablas: measurements, parameters, roles, sensors, users. En la Figura 37, podemos ver la estructura de la base de datos en la pantalla de MongoDB Atlas (MongoDB, 2020).

Figura 37

Base de datos MongoDB



En la Tabla 12 se muestra el contenido de cada tabla de la base de datos.

Tabla 12

Contenido de cada tabla de la base de datos

Tabla	**Contenido**
Measurements	Se almacena los datos de los monitoreos de cada signo vital, temperatura y pulso.
Parameteres	Se almacena información de parámetros que permita que la aplicación sea más dinámica.
Roles	Se almacena la información que se va a utilizar en la aplicación para permitir los accesos en las diferentes opciones.

Tabla	Contenido
Sensors	Se almacena la información de los sensores, para obtener el tipo de medida de cada uno, nombres.
Users	Se almacena la información para inicio de sesión, de acuerdo cada rol poder saber si es un paciente, doctor o un administrador.

Configuración del servidor MQTT

Para configurar el servidor de MQTT, se ha utilizado librería “mosca” (Collina, 2020), que es un bróker MQTT de Node.js. En la Figura 38 se puede ver la clase de configuración del servidor, en el constructor se va a recibir las configuraciones como parámetro o se las configura por defecto, habilitando el puerto 1883 para la comunicación MQTT y el puerto 8000 para la configuración por WebSockets. Una vez inicializado el servidor se registra los eventos con una funcionalidad por defecto para la transmisión de información.

Figura 38

Servidor MQTT

```
'use strict'
// MQTT broker
const mosca = require( id: 'mosca');
/** Events to use in the server MQTT ...*/
class MqttServer {
    constructor(settings) {
        if (settings) {
            this.settings = settings;
        } else {
            this.settings = {
                port: 1883,
                http: {
                    port: 8000,
                    bundle: true,
                    static: './public'
                }
            }
        }
        this.broker = null;
    }

    startMoscaServer() {
        this.broker = new mosca.Server(this.settings, (res) => {
            console.log('Mosca MQTT server upload!!', res);
        });
        this.broker.on('ready', this.onReady);
        this.broker.on('published', this.onPublished);
        this.broker.on('clientDisconnecting', this.onClientDisconnecting);
        this.broker.on('clientDisconnected', this.onClientDisconnected);
    }

    onReady() {...}

    onPublished(packet) {...}

    onClientDisconnecting(packet) {...}

    onClientDisconnected(packet) {...}
}

module.exports = MqttServer;
```

En la Figura 39 se puede ver la clase “MqttHandler”, encargada de conectarse y administrar las conexiones con el bróker Mosca. En primer lugar, se usa la librería mqtt.js la cual es un cliente para el protocolo MQTT, en la clase, se realiza la

conexión del cliente con el bróker MQTT y se provee los servicios que van a ser utilizados en toda la aplicación para publicar mensajes y subscribirse a tópicos para el envío y transmisión de datos.

Figura 39

MQTT Handler

```
'use strict'
const mqtt = require( id: 'mqtt');
const {SERVERS_URLS} = require( id: '../../commons/constants/Constants');
class MqttHandler {
    constructor(host, topic) {
        this.mqttClient = null;
        this.host = host || SERVERS_URLS.MOSCA;
        this.topic = topic;
        this.connect();
    }
    connect() {
        console.log('Connect Mqtthandler', this.host);
        // Connect mqtt with credentials (in case of needed, otherwise we can omit 2nd param)
        this.mqttClient = mqtt.connect(this.host);
        // Mqtt error calback
        this.onError();
        // Connection callback
        this.onConnect();
        // mqtt subscriptions
        this.subscribeInTopic(this.topic,  opts: {qos: 0});
        // When a message arrives, console.log it
        this.onMessage();
        this.onClose();
    }

    // Subscribers
    subscribeInTopic(topic, opts :{} = {}) {...}
    onError() {...}
    onConnect() {...}
    onMessage() {...}
    onClose() {...}
    disconnect() {...}
    // Publishers...
    sendMessage(topic, message) {...}
    getClient() {...}
    getHost() {...}
    getTopic() {...}
}

module.exports = MqttHandler;
```

Desarrollar el Backend

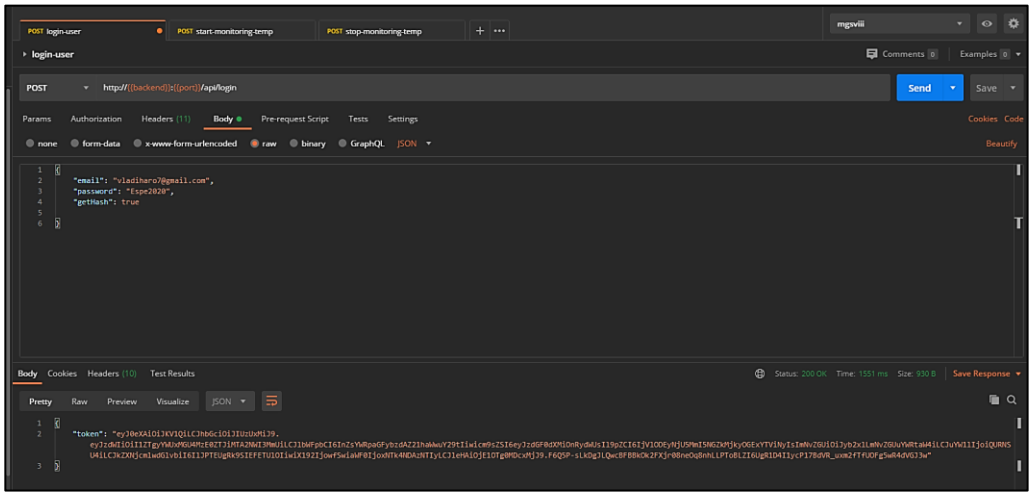
El backend de la aplicación, ha sido desarrollado con Node.js con la integración del framework flexible, robusto, rápido y muy simple de implementar Express.js (Express, 2020) para crear un servidor web básico capaz de responder a cualquier petición HTTP. En el Anexo 3 se puede encontrar el código fuente del proyecto. A continuación, se detallan imágenes del resultado del desarrollo del Backend.

En la Figura 40, se encuentra el servicio para obtener el Token que permite navegar en la aplicación con credenciales de seguridad, se envía el usuario, la contraseña y una opción más con el valor “true” que retorna el Token. El método HTTP utilizado es un POST.

El Token generado por la aplicación va a ser la llave en toda la aplicación, que permita realizar las transacciones solicitadas por el Frontend de manera segura.

Figura 40

Servicio para obtener token

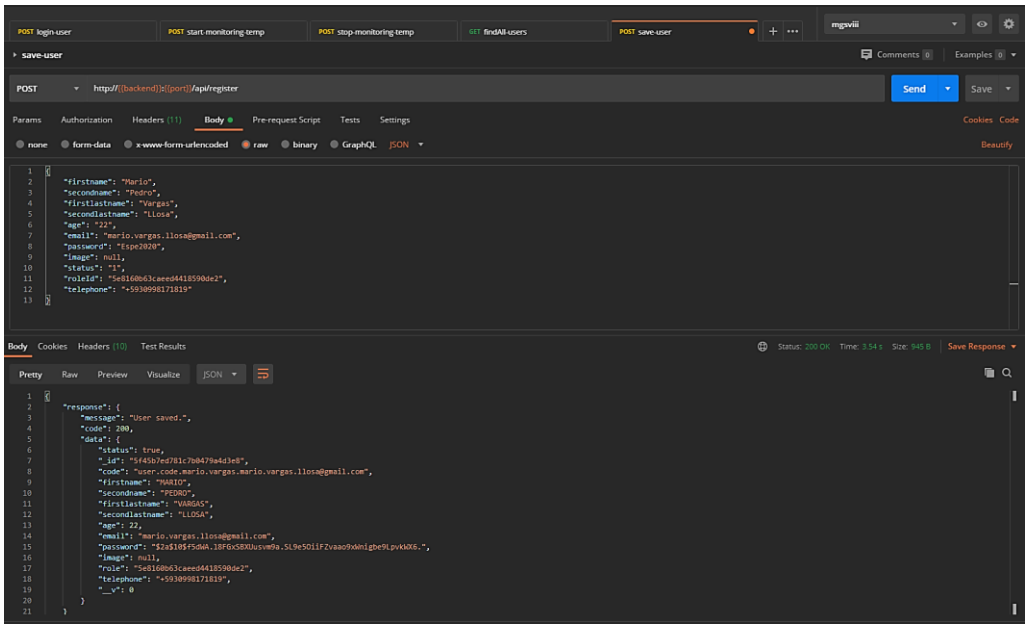


En la Figura 41 se muestra el servicio para guardar un usuario, una vez obtenido el token de acceso al sistema, en cada llamada a los servicios web, se debe mandar el valor de la cabecera de Autorización, el Token generado por el ingreso al sistema. El método HTTP utilizado es un POST.

En la sección del body mediante un objeto JSON, se envía la información necesaria para crear un usuario y se recibe un status 200 OK con un objeto JSON representando al usuario que se ha creado en la base de datos.

Figura 41

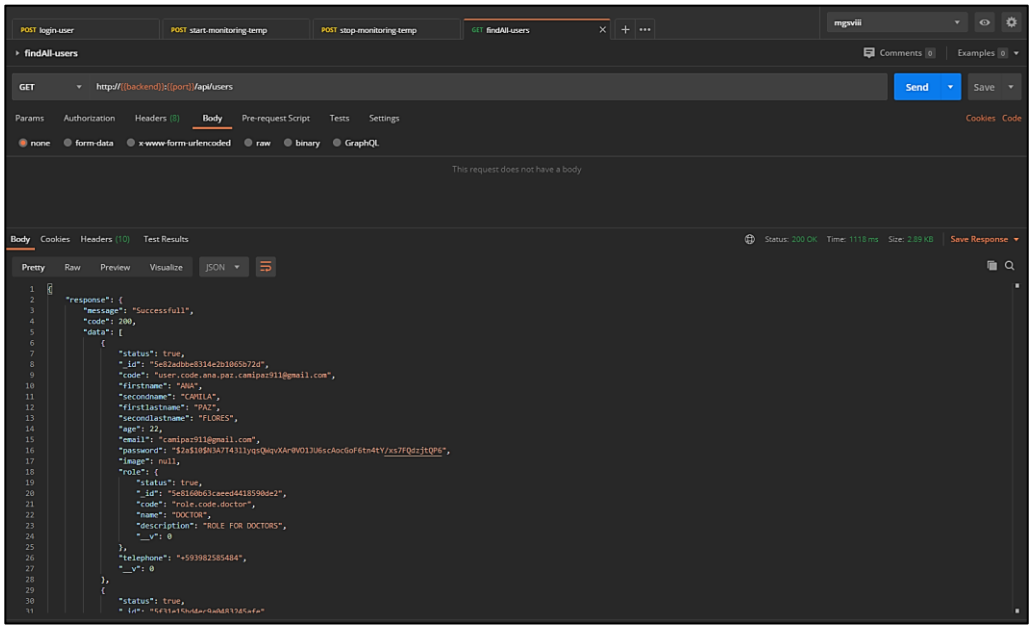
Servicio para guardar usuario



En la Figura 42 se puede observar el servicio para buscar usuarios, el método HTTP usado es GET, en el body de la petición no es necesario enviar ningún objeto JSON con datos para realizar la búsqueda, el método envía un status 200 con un JSON con los datos de todos los usuarios. Al igual que todos los métodos, es necesario enviar la cabecera de Autenticación con el Token generado.

Figura 42

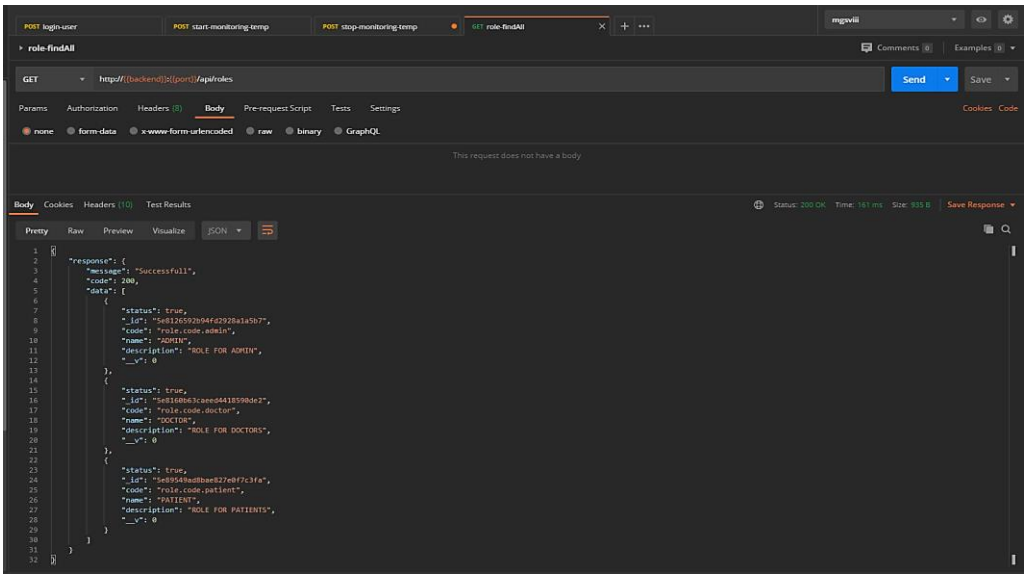
Servicio para buscar usuarios



En la Figura 43 se observa el servicio para obtener los roles guardados en la base de datos, el método HTTP usado es GET, en el body de la petición no es necesario enviar ningún objeto JSON con datos para realizar la búsqueda, el método envía un status 200 con un JSON con los datos de todos los roles. Al igual que todos los métodos, es necesario enviar la cabecera de Autenticación con el Token generado.

Figura 43

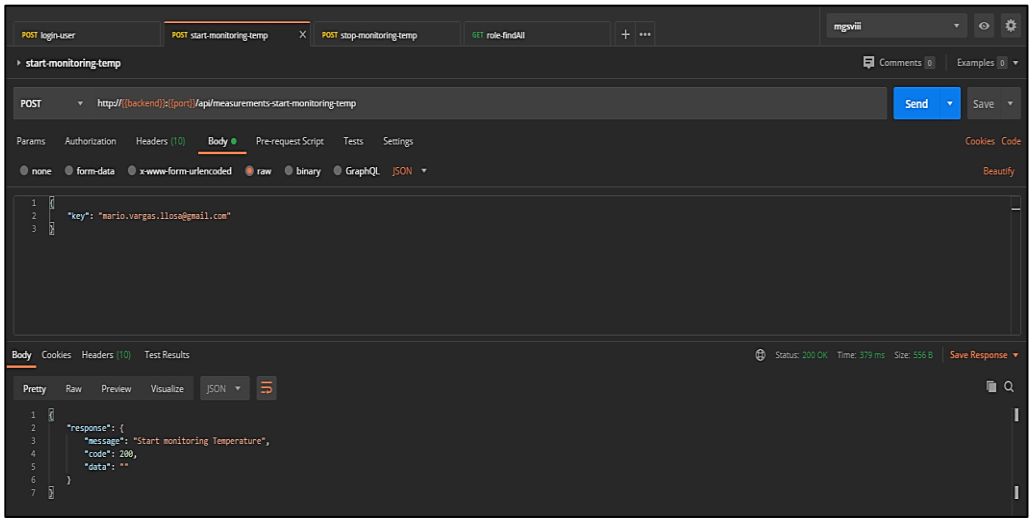
Servicio para obtener roles



En la Figura 44 se observa el servicio para inicializar el monitoreo de la temperatura, el método HTTP usado es un POST, en el body de la petición es necesario enviar un objeto JSON con una propiedad de nombre “key” con un valor único, que va a permitir distinguir la información del proceso del usuario que está ejecutando el monitoreo, una vez que se solicita el servicio este responde un status 200 OK, y en segundo plano se empieza a ejecutar el monitoreo transmitiendo información mediante el protocolo MQTT hacia los clientes subscritos.

Figura 44

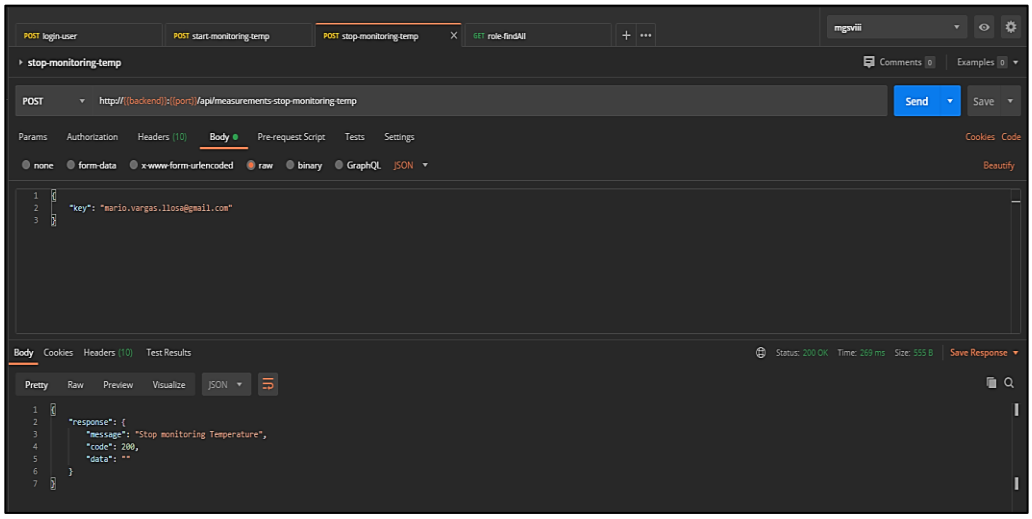
Servicio para iniciar el monitoreo de temperatura



En la Figura 45 se puede ver el servicio para finalizar el monitoreo de la temperatura, el método HTTP usado es un POST, en el body de la petición es necesario enviar un objeto JSON con una propiedad de nombre “key” con un valor único, que va a permitir distinguir la información del proceso del usuario que está ejecutando el monitoreo, una vez que se solicita el servicio este responde un status 200 OK, y a continuación se finaliza el monitoreo para el usuario que está especificado.

Figura 45

Servicio para parar el monitoreo de temperatura



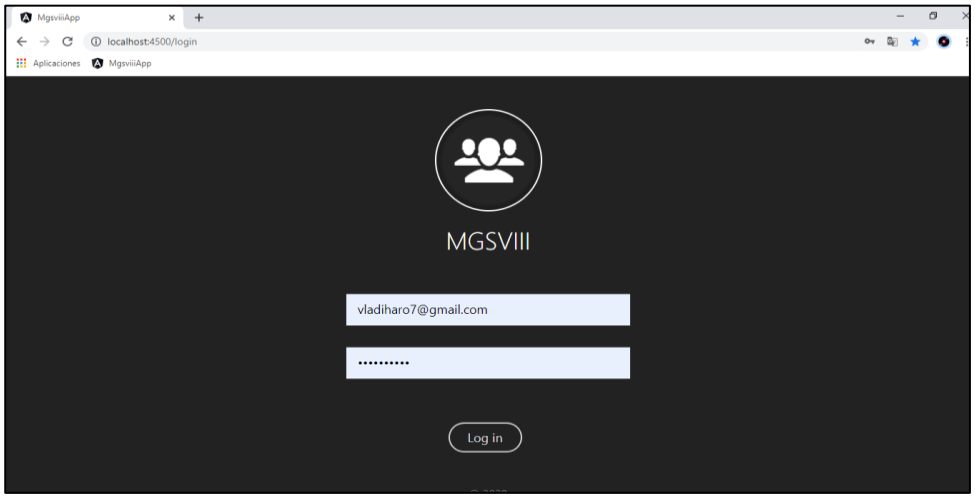
Desarrollar el Frontend

El Frontend ha sido desarrollado utilizando el programa Visual Studio Code, útil para el desarrollo de aplicaciones web, con el Framework Angular 9 distribuido en módulos y componentes según su funcionalidad; en el Anexo 4 se puede encontrar el código fuente del proyecto. A continuación, se detallan imágenes del resultado del desarrollo del Frontend.

En la Figura 46 se puede observar el resultado del desarrollo de la página de inicio de sesión, en la cuál en los campos de texto, se debe ingresar el correo y la contraseña del usuario registrado.

Figura 46

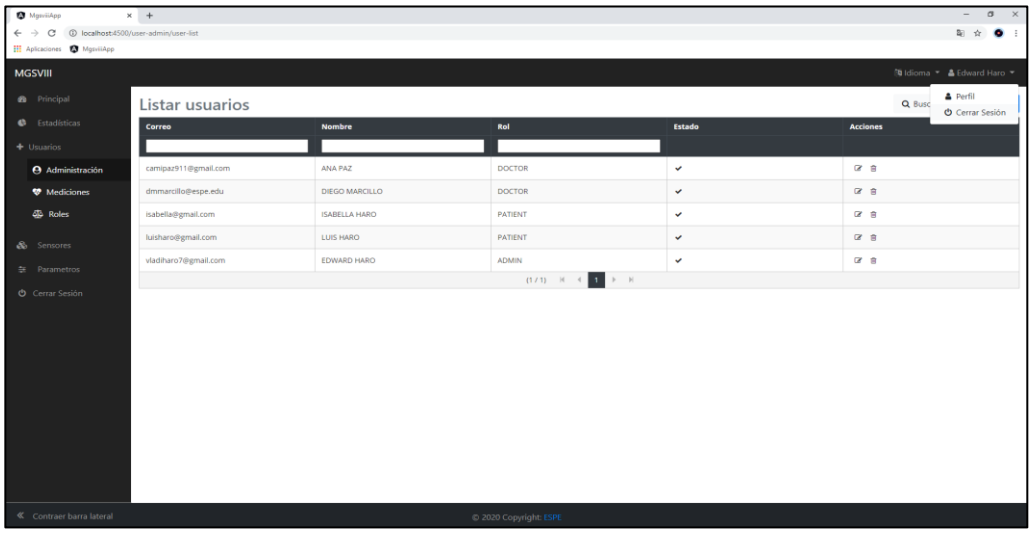
Página de inicio de sesión



Una vez que se ha ingresado al sistema, como podemos ver en la Figura 47, se habilita una opción que permite realizar una salida segura de la aplicación mediante el ítem Cerrar sesión y así poder eliminar los datos de seguridad creados para poder navegar en la aplicación web.

Figura 47

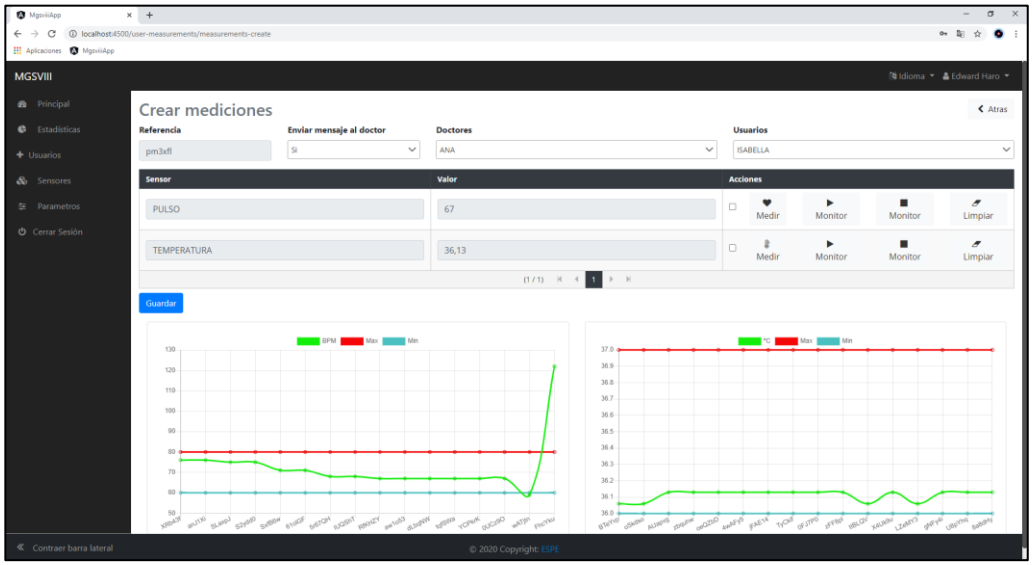
Opción para cerrar sesión



En la Figura 48, se puede observar la pantalla que permite crear mediciones, en donde se genera un código de referencia aleatorio para identificar la medición que se está realizando, un usuario administrador o doctor, puede escoger si es necesario enviar un mensaje informativo al doctor acerca de inconvenientes detectados en la medición en curso. Existe también una opción que permite seleccionar un Doctor para poder notificar una alerta en caso de que existan inconvenientes con las mediciones obtenidas del paciente que se está realizando la medición. Al final un usuario con un rol de paciente va a aparecer seleccionado automáticamente, en el caso de que el usuario tenga el rol de Doctor o de Administrador, podrá seleccionar cualquier usuario disponible en la lista. En la tabla de abajo se muestra los sensores y las acciones que cada uno puede ejecutar, el campo valor se llenara con la información recibida de los sensores, en la lista de acciones se encuentra las opciones de medir, la cual va a obtener una sola medición y al obtener esta ponerla en el campo de valor, mientras que los botones Iniciar Monitoreo y Finalizar Monitoreo, van a permitir realizar un monitoreo en tiempo real y representarlo en gráficos de líneas. Finalmente se puede observar el botón de Guardar, el cual envía toda la información de la página al Backend para guardar en la base de datos.

Figura 48

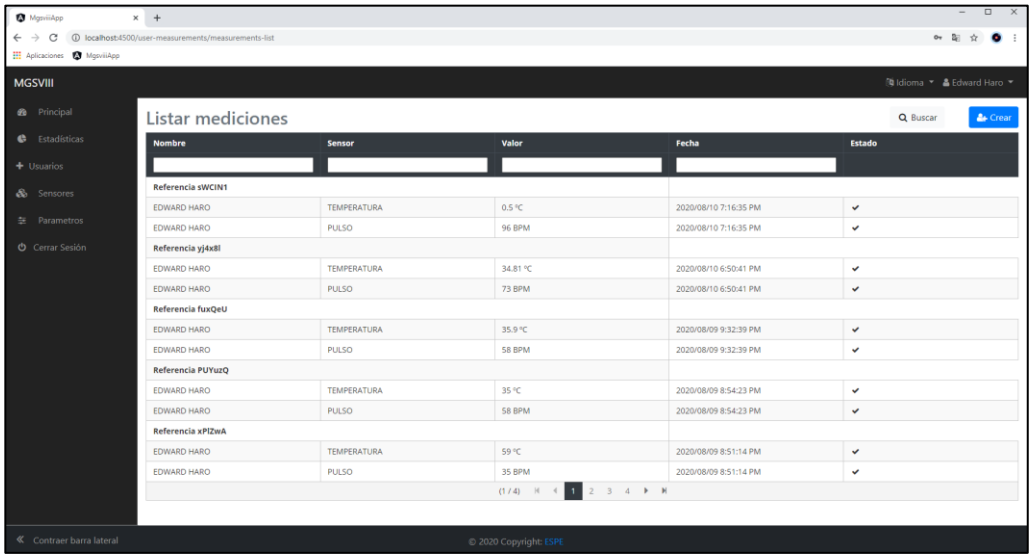
Pantalla para crear mediciones



En la Figura 49, se puede observar una tabla, con todas las mediciones que se han realizado para uno o todos los usuarios dependiendo del perfil del usuario que ha ingresado en la sesión. En la tabla se puede filtrar por cada columna que tenga un input en su cabecera. La pantalla de “Listar mediciones” tiene un botón “Crear” que permite el acceso a la pantalla de “Crear mediciones”.

Figura 49

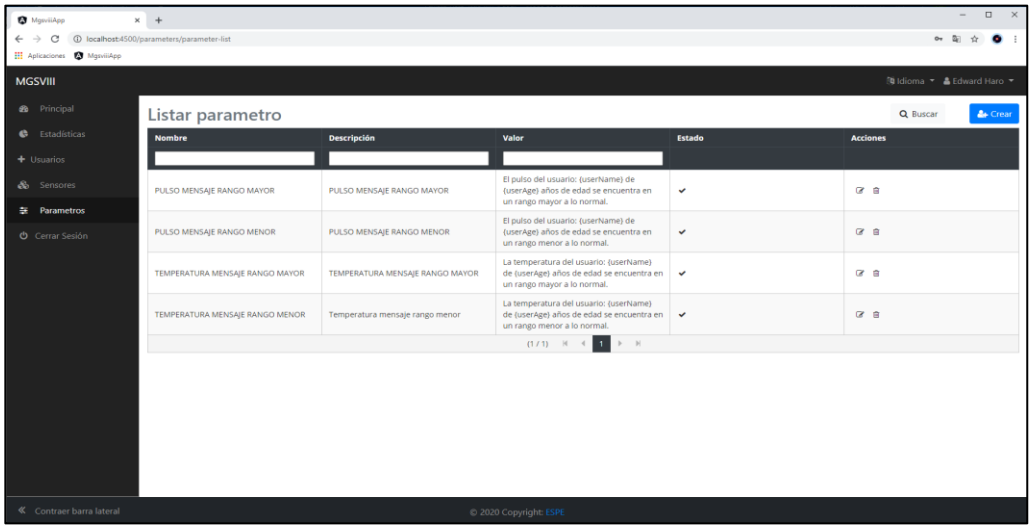
Pantalla para listar mediciones



En la Figura 50, se puede observar la pantalla desarrollada para mostrar la lista de parámetros que van a estar disponibles en la aplicación, en este caso se puedo ver parámetros creados para los mensajes de pulso y temperatura, en el caso de que el valor sea mayor o menor al rango normal, estos parámetros van a ser la base para enviar mensajes al médico seleccionado en caso de que existan novedades en las mediciones de los sensores.

Figura 50

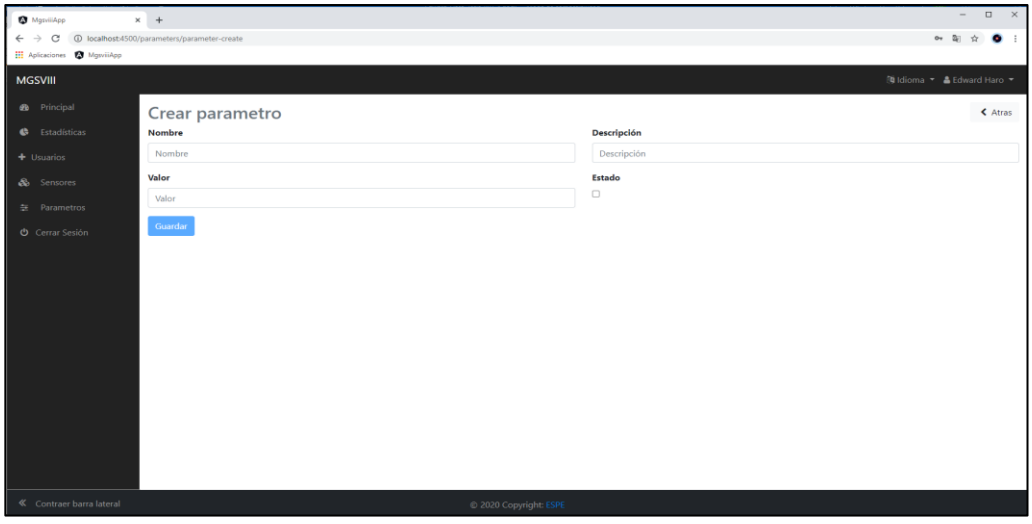
Pantalla para listar parámetros



En la Figura 51, podemos ver la pantalla desarrollada para Crear un parámetro, en donde se deben llenar los campos Nombre, Descripción, Valor y Estado para poder crear un parámetro que va a ser utilizado en la aplicación web.

Figura 51

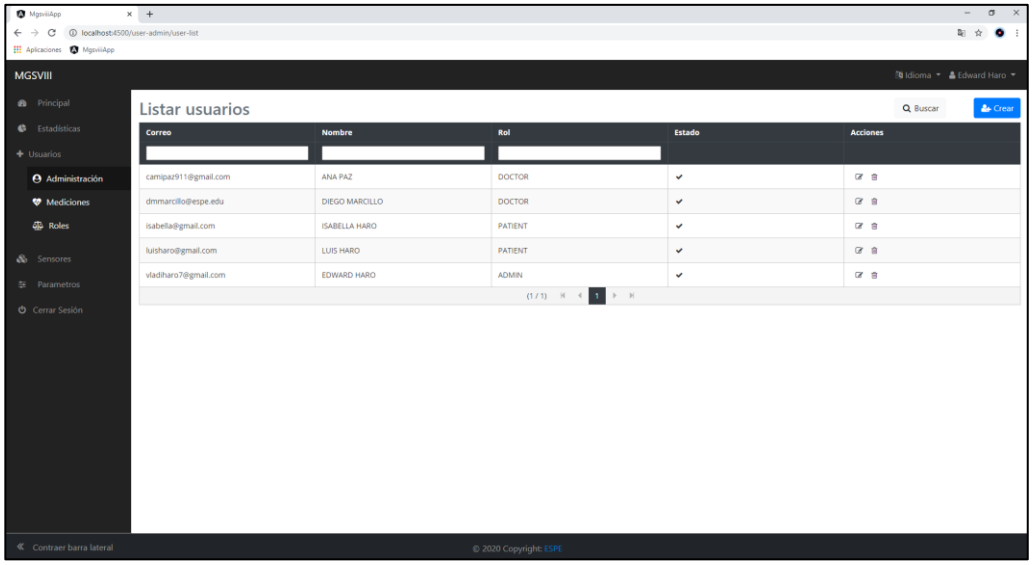
Pantalla para crear parámetros



En la Figura 52, podemos ver la pantalla desarrollada para listar los usuarios de la aplicación, en donde se muestra la aplicación principal de cada usuario como el correo, nombre, rol, estado, además se encuentra el botón que dirige a la pantalla de creación de un usuario.

Figura 52

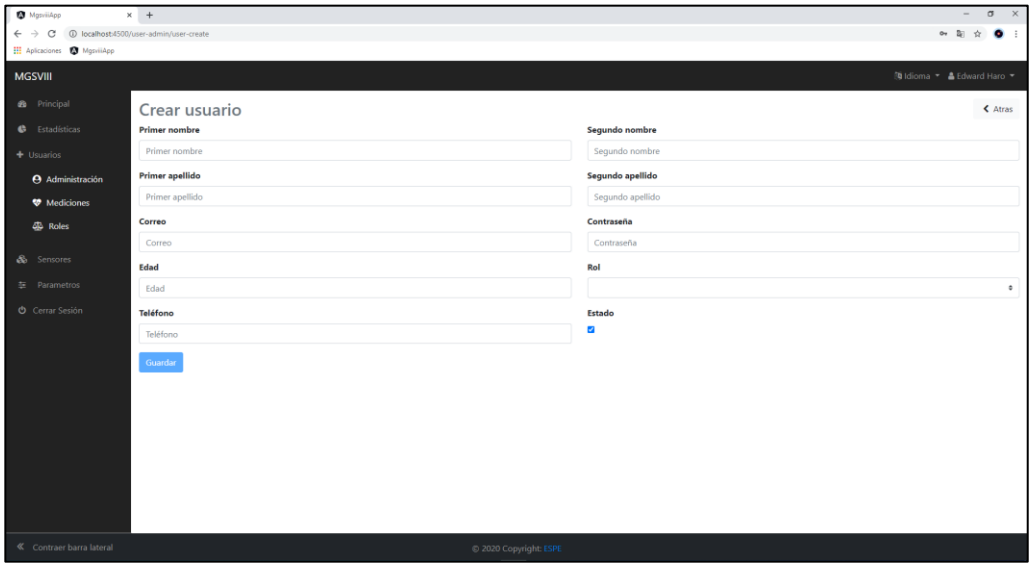
Pantalla para listar usuarios



En la Figura 53, se puede observar la pantalla para Crear un Usuario, en esta pantalla se tiene cajas de texto destinadas para el ingreso de la información principal de los usuarios.

Figura 53

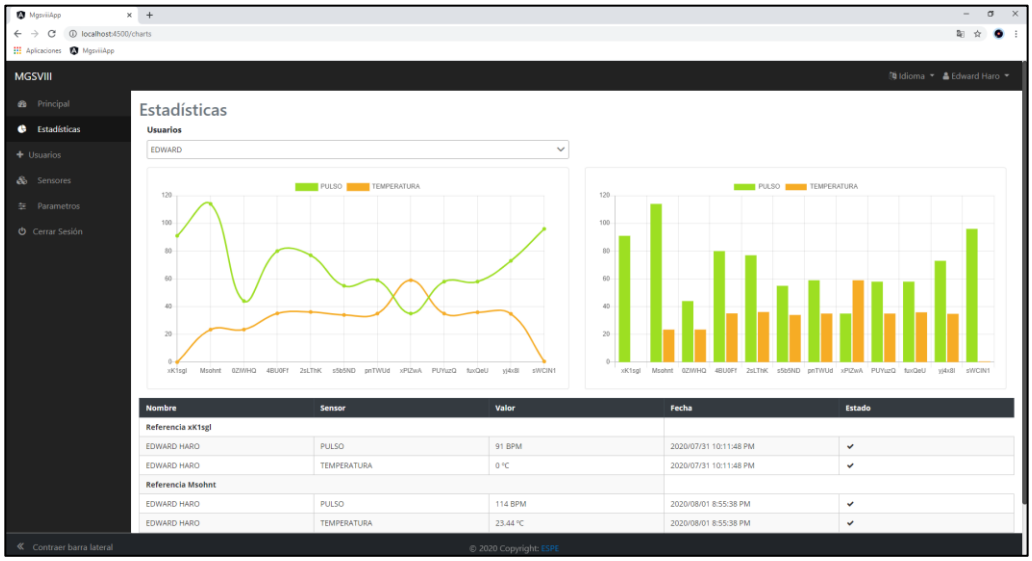
Pantalla para crear un usuario



En la Figura 54, se puede observar la pantalla de estadísticas, en donde se selecciona un usuario y se puede ver un gráfico de líneas y un gráfico de barras con la información de los valores obtenidos de los sensores en ocasiones anteriores, es decir muestra un historial de las mediciones tomadas. A demás de las gráficas se tiene una tabla con toda la información ordenada por fecha.

Figura 54

Pantalla para mostrar estadísticas



Validación del Prototipo

Para la validación del prototipo, es decir los sensores, la comunicación MQTT, los mensajes WhatsApp de alerta, la aplicación web se ha dividido en dos tipos: una encuesta a un profesional y otro tipo pruebas de funcionalidad para comprobar el funcionamiento y confiabilidad del dispositivo, así como también el envío de datos para almacenamiento y visualización de la información generada.

Pruebas de funcionamiento

Para las pruebas de funcionamiento se tomó en cuenta tres pacientes un adulto de género masculino, un adulto de género femenino y una niña de género femenino, los detalles de cada paciente se presentan a continuación:

- En la Figura 55, el paciente 1, de género masculino, edad 30 años, no presenta ninguna enfermedad, con actividad física moderada.

- En la Figura 56, el paciente 2, de género femenino, edad 23 años, no presenta ninguna enfermedad, con actividad física moderada.
- En la Figura 57, el paciente 3, de género femenino, edad 7 años, no presenta ninguna enfermedad, con actividad física moderada.

Figura 55

Paciente 1

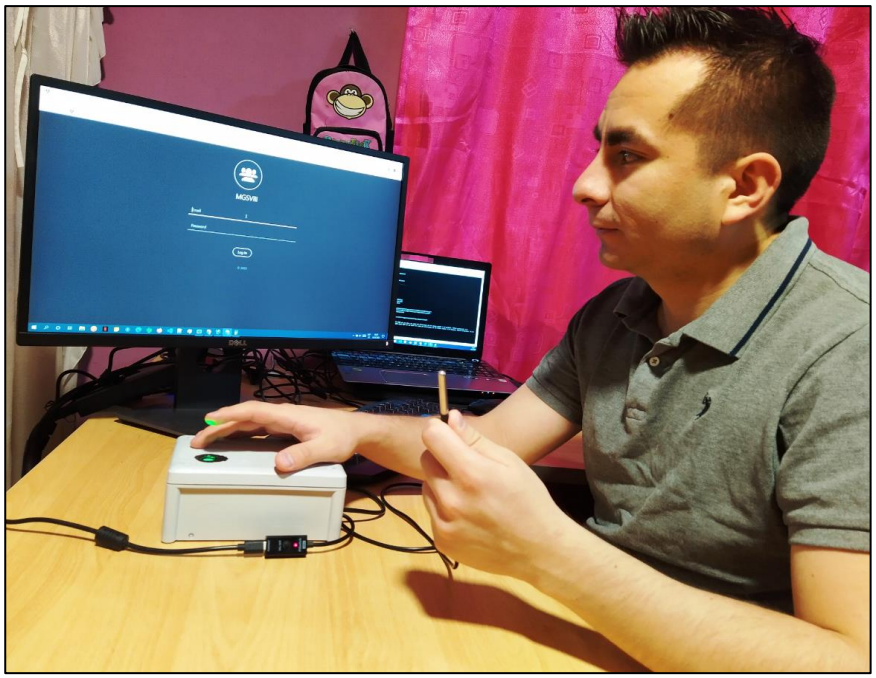


Figura 56

Paciente 2

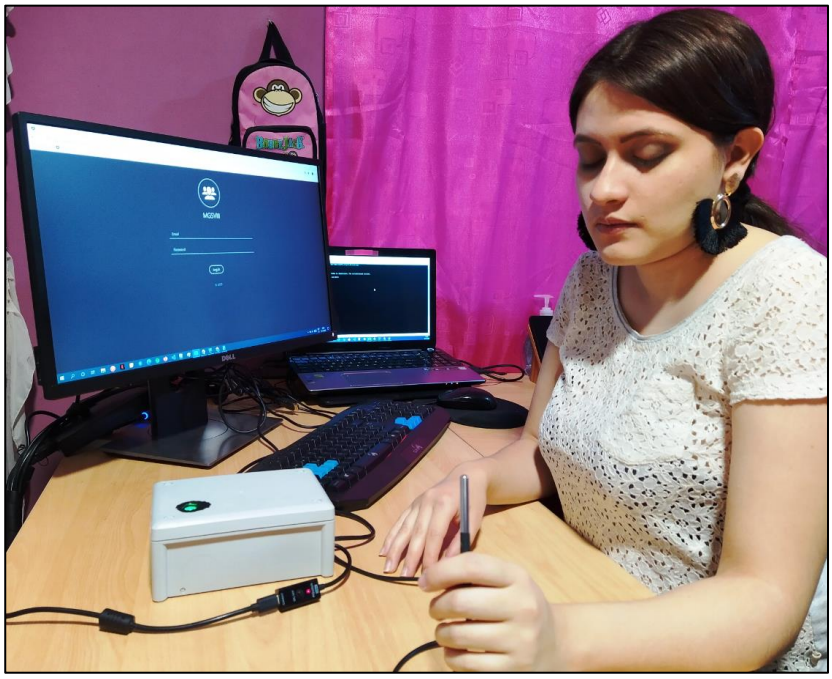


Figura 57

Paciente 3

Con respecto a la ubicación de los sensores, en los sujetos de prueba se tomará las siguientes posiciones:

- Para el sensor de temperatura, se ubicará en la axila, después de proceso de limpieza y desinfección. Como se puede observar en las Figuras 58 y 59.
- Para el sensor de pulso, en este se debe colocar el dedo índice sobre la luz que el sensor emite. Como se puede observar en la Figura 60.

Figura 58

Desinfección sensor de temperatura

Figura 59

Posición sensor de temperatura

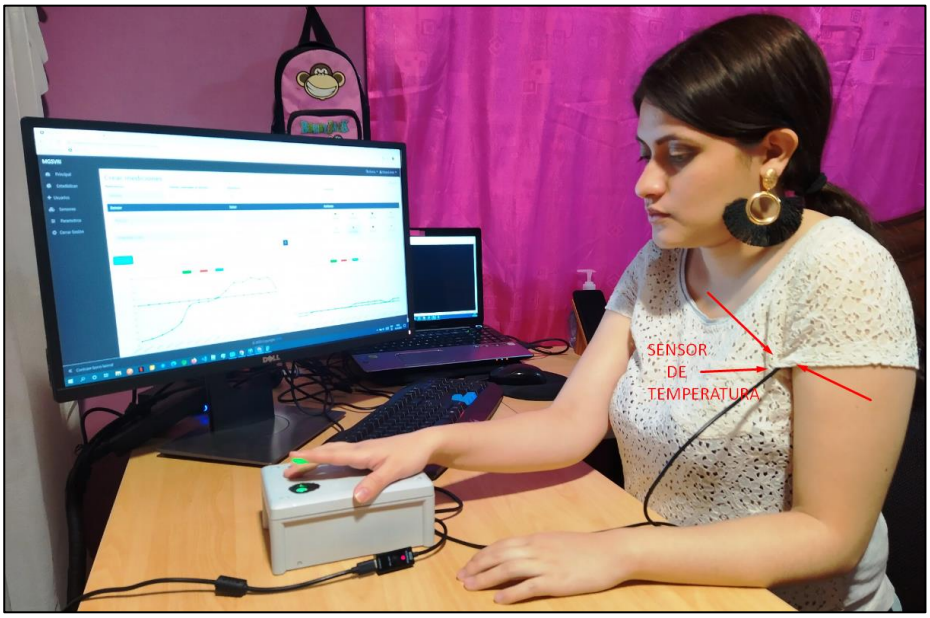
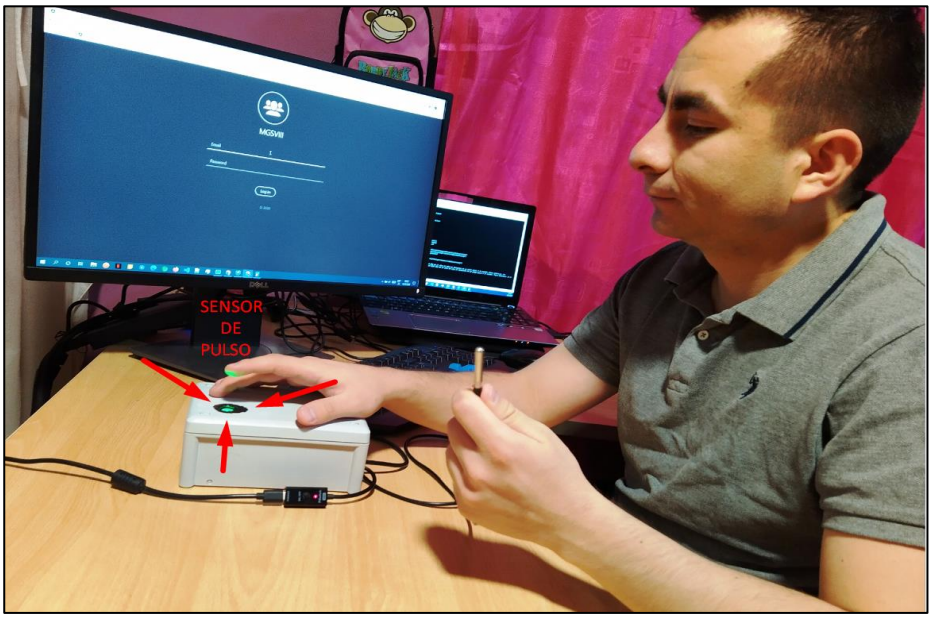


Figura 60

Posición sensor de pulso



El tiempo de monitoreo será de 120 segundos, con la finalidad de que el tiempo de muestreo de cada sensor sea la misma.

La misma prueba se va a realizar hasta por cinco veces para reducir el margen de error.

Figura 61

Monitoreo paciente 1

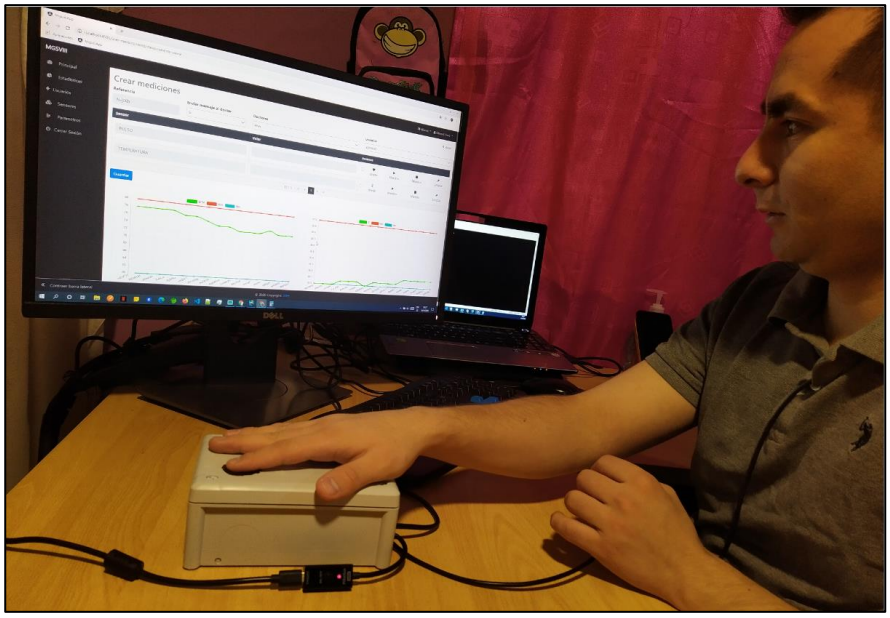


Figura 62

Monitoreo Paciente 2

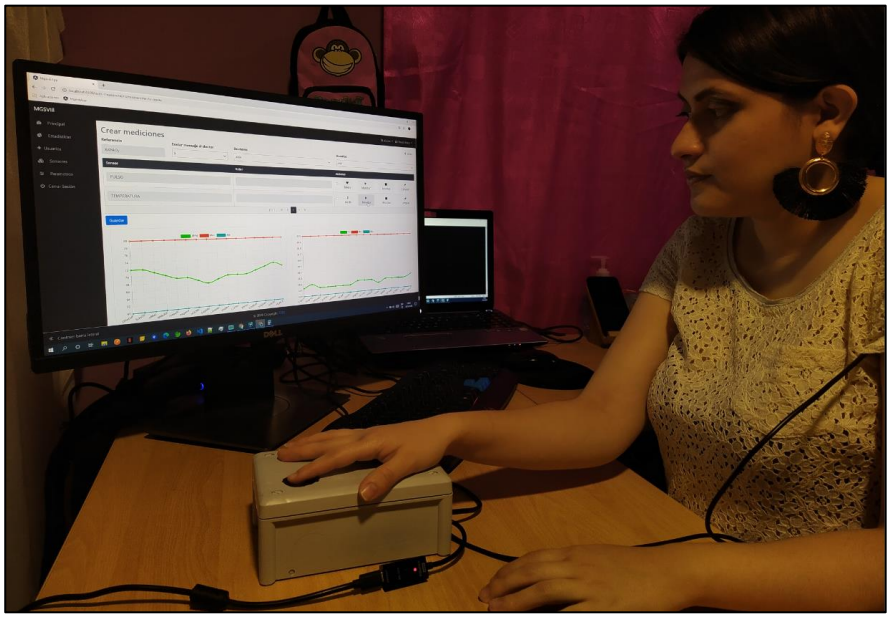


Figura 63

Monitoreo Paciente 3

Validación del sensor de temperatura

Durante esta prueba se realizó diez mediciones durante 120 segundos cada una comparando los resultados de un termómetro de mercurio y el sensor DS18B20, obteniendo los resultados de los promedios de las mediciones de los dos pacientes mayores tomados en cuenta para las pruebas que se muestran en la Tabla 13 en grados centígrados, mientras que en la Figura 64 se puede apreciar un gráfico que muestra las diferencias entre un dispositivo y otro.

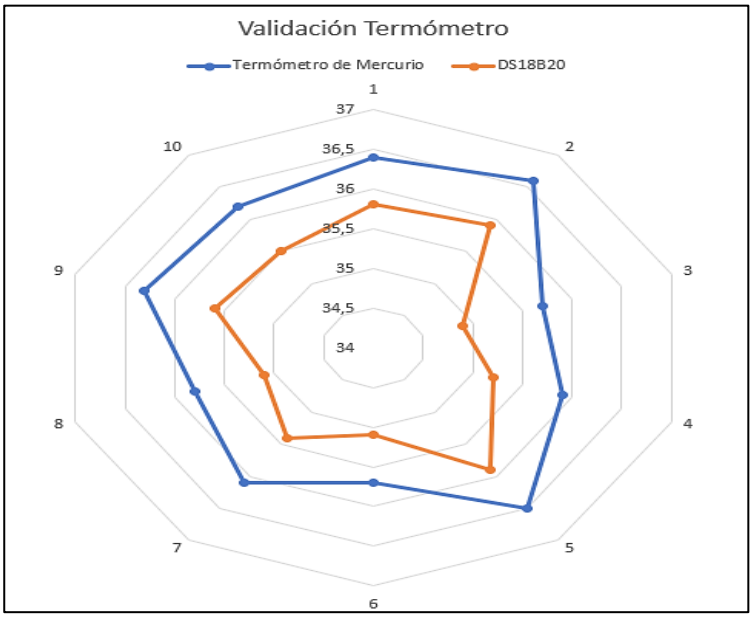
Tabla 13

Resultados comparación toma de temperatura

N.º	Termómetro de Mercurio	DS18B20	Diferencia
1	36,4	35,8	0,7
2	36,6	35,9	0,7
3	35,7	34,9	0,8
4	35,9	35,2	0,7
5	36,5	35,9	0,6
6	35,7	35,1	0,6
7	36,1	35,4	0,8
8	35,8	35,1	0,8
9	36,3	35,6	0,8
10	36,2	35,5	0,7
TOTAL	36,1	35,4	0,7

Figura 64

Resultados gráficos comparación toma de temperatura



Como se puede observar en la Tabla13 y en la Figura 64, el termómetro de mercurio marca un promedio de 0.70ºC más que el sensor DS18B20, por lo que con una compensación en programación de 0.50ºC sería suficiente para tener una diferencia de 0.20ºC que sería un error aceptable para mostrar al paciente ya que se encuentra en el rango aceptable según la Tabla 5.

Validaciones Sensor de Pulso

Durante esta prueba, se realizó diez mediciones durante 120 segundos cada una, comparando los resultados de un pulsómetro digital y el sensor de pulso obteniendo los resultados de los promedios de las mediciones de los dos pacientes mayores tomados en cuenta para las pruebas que se muestran en la Tabla 14 en BPM.

Tabla 14

Resultados comparación toma de pulso

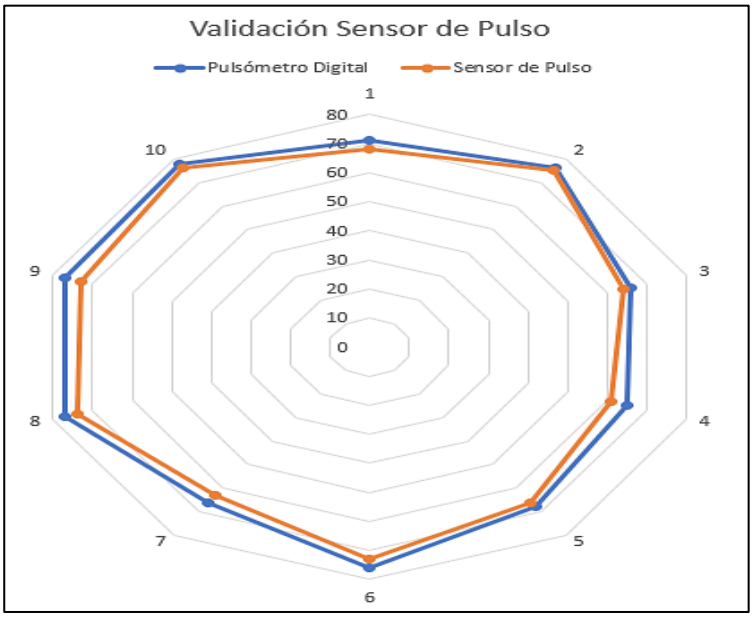
N.º	Pulsómetro Digital	Sensor de Pulso	Diferencia
1	71	68	3
2	76	75	1
3	66	64	2
4	65	61	4
5	68	66	2
6	76	73	3
7	66	63	3
8	77	74	3
9	77	73	4
10	78	76	2
TOTAL	**72**	**69**	**3**

Como se puede ver en la Tabla 14, el Pulsómetro Digital marca un promedio de 3 BPM más que el Sensor de Pulso, por lo que con una compensación en programación de 2 BPM sería suficiente para tener una diferencia de 1 BPM que sería una que sería un error aceptable para mostrar al paciente ya que se encuentra en el rango aceptable según la Tabla 1.

Mientras que en la Figura 65 se puede apreciar la comparación de valores en un gráfico que muestra las diferencias entre un dispositivo y otro.

Figura 65

Resultados gráficos comparación toma de temperatura

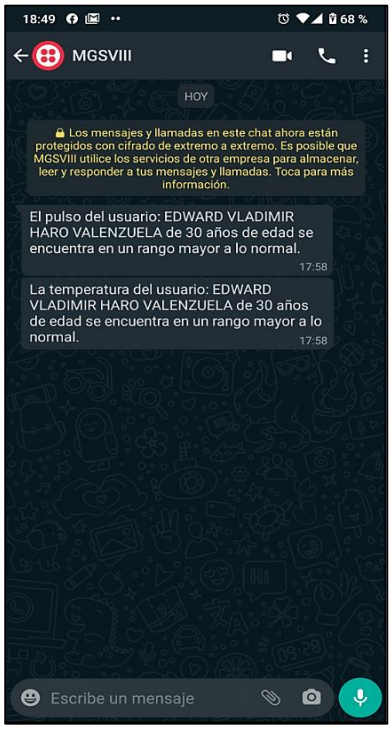


Validación Mensaje WhatsApp

Una vez realizados los monitoreos, se procede a guardar la información obtenida y si algún registro se encuentra fuera del rango normal se envía un mensaje de alerta al médico que se ha seleccionado en la aplicación antes del monitoreo, como se puede observar en la Figura 65, el mensaje de alerta recibido en la aplicación WhatsApp, muestra una alerta al médico con el nombre del paciente, la edad y la notificación.

Figura 66

Mensaje de Alerta WhatsApp



Encuesta a Profesionales

Además de las pruebas de funcionamiento, se realiza una encuesta a dos profesionales de la salud, en estos casos a una Licenciada Tecnóloga Medica En Laboratorio Clínico e Histopatológico y por otra parte a una Doctora En Medicina y Cirugía, haciendo una presentación del prototipo con el fin de dar a conocer la funcionalidad de este. En el Anexo 5 se encuentra la plantilla de la encuesta para realizar al profesional, mientras que en el Anexo 6 se muestra el resultado de la encuesta con las respuestas de la Licenciada y en el Anexo 7 se muestra el resultado de la encuesta con las respuestas de la Doctora.

La encuesta está elaborada con las siguientes preguntas:

1. ¿El prototipo de monitoreo planteado, le parece un dispositivo innovador?
2. ¿Ha sido fácil para usted el uso del dispositivo?
3. ¿Pudo hacer el monitoreo de los signos vitales?
4. ¿Los resultados obtenidos en la medición de los signos vitales son aceptables y confiables?
5. ¿La pantalla de monitoreo le pareció amigable para su uso?
6. ¿El valor de $130.00, le parece un precio bajo para un dispositivo de monitoreo de signos vitales?
7. ¿Qué le parece el envío de alertas al WhatsApp?
8. ¿Piensa usted que el historial de mediciones es un servicio complementario que le da un valor extra a la funcionalidad del prototipo?
9. ¿Qué se podría mejorar?
10. ¿Cuál es su opinión con respecto al prototipo propuesto?

Una vez realizada la encuesta a los dos profesionales, se tiene los siguientes resultados:

Tabla 15

Calificación de las encuestas

PREGUNTA	DOCTORA	LICENCIADA	TOTAL
1	5	5	5
2	5	5	5
3	5	4	4.5
4	5	5	5
5	5	5	5
6	5	5	5
7	5	5	5
8	5	5	5
9	5	4	4.5
10	5	5	5
TOTALES	**5**	**4.8**	**4.9**

Como resultado se tiene un promedio de 4.9 puntos sobre 5 de aceptación de la calificación propuesta en la encuesta de aceptación realizada a los profesionales, como se puede comprobar en la Tabla 15 el puntaje asignado a cada pregunta por cada profesional.

Evaluación De Los Costos Del Prototipo Mediante El Modelo Low Cost

El modelo de low cost, al ser un modelo de negocio basado principalmente en la reducción de costos ofreciendo un producto básico, funcional con una calidad comparable con un producto comercial equivalente, rebajar los precios de un

producto o servicio con el objetivo de aumentar los consumidores y así vender más servicios complementarios.

En el caso del prototipo desarrollado, es un dispositivo que permite monitorear varios signos vitales desde un mismo equipo y además se relaciona con una aplicación web como un servicio complementario, que va a permitir dar un seguimiento a los pacientes a través de un historial y reportes que se presentan a los interesados con el fin de poder tener un diagnóstico.

Con respecto a los dispositivos usados para armar el dispositivo, se utilizan sensores electrónicos, capaces de medir signos vitales, comparados entre varios del mismo tipo con el objetivo de identificar al dispositivo que tenga menos margen de error pero que sea muy confiable en cuanto a la función que tenga que desempeñar.

Mientras que con respecto a la aplicación web, al tratar las funcionalidades de esta como servicios complementarios, se utiliza software libre de código abierto, el cual no genera costos debido a que es de uso bajo licencias gratuitas ahorrando así una gran cantidad de costos al no comprar licencias de software.

El financiamiento para el desarrollo del proyecto está valorado en $130.00, mientras que para la implementación de este el valor es de $0.00, en la Tabla 16 se detallan los costos de todos los dispositivos tecnológicos que se utilizaron para el desarrollo del prototipo.

Tabla 16

Costos de los dispositivos utilizados en el prototipo

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	Tarjeta Arduino UNO	$11.00	$11.00
1	Sensor de Pulso	$6.00	$6.00
1	Sensor de Temperatura	$5.00	$5.00
20	Cables Jumper	$0.10	$2.00
1	Raspberry Pi 4	$85.00	$85.00
1	Otros	$21.00	$21.00
	TOTAL	**$128.10**	**$130.00**

En la Tabla 17 se detallan los costos de implementación tanto en recursos humanos, hardware y software. Al utilizar herramientas de software de código abierto, estas tienen costo $0.00.

Tabla 17

Costos de implementación

Recurso	Costo Nominal	Costo Real
1. Recursos Humanos		
Desarrollador Web	$1500.00	$0.00
Desarrollador Arduino	$500.00	$0.00
2. Hardware		
Computadora con Windows	$1000.00	$0.00
3. Software		
Arduino IDE 1.8.13	$0.00	$0.00
Notepad++	$0.00	$0.00
MongoDB	$0.00	$0.00
Visual Studio Code	$0.00	$0.00

Recurso	Costo Nominal	Costo Real
Node.js	$0.00	$0.00
Java 8	$0.00	$0.00
Angular 9	$0.00	$0.00
Git	$0.00	$0.00
TOTAL	$3000.00	$0.00

Los costos de estos recursos se los ha separado en dos columnas, un costo nominal, en donde se especifican los precios más bajos ofrecidos en el mercado por el tiempo y el trabajo desempeñado y el costo real que se refiere al dinero que se ha pagado para el desarrollo del prototipo.

Para los recursos humanos en el valor nominal se coloca el valor mínimo para un desarrollador web por tres meses de desarrollo de la aplicación web. Mientras que para el desarrollo Arduino solo se ha necesitado un mes de trabajo.

Para el hardware se ha utilizado un PC ya existente por lo que no ha sido necesaria una compra de esta para el desarrollo del proyecto.

Para el software, como se ha mencionado anteriormente, el uso de software libre y aplicaciones open source, han permitido que el costo de uso de estas sea de $0.00.

En la Tabla 18 se puede ver una lista de dispositivos comerciales para monitoreo de signos vitales en donde se puede ver la imagen y los precios de cada uno permitiendo así conocer los valores de tres dispositivos con un precio medio de acuerdo con otros disponibles en el mercado. A demás estos dispositivos no cuentan con una aplicación web capaz de mostrar un reporte con gráficos de barras y/o de líneas con el historial de mediciones de los pacientes.

Por lo que mediante los datos comparados se puede deducir que el dispositivo si cumple con el modelo low cost propuesto para el desarrollo del mismo.

Tabla 18

Dispositivos comerciales para monitoreo de signos vitales

Dispositivo	Precio	Referencia
Zoncare – PM7000	$1496.00	
Advanced - VSM-300C	$1390.00	
Biolight - M1000	$1250.00	

En la Tabla 19 se muestra un detalle del porcentaje de ahorro que se obtiene comparando cada uno de los dispositivos comerciales con el prototipo low cost desarrollado.

Tabla 19

Comparación de precios entre dispositivos y el prototipo

Dispositivo	Precio	Precio Prototipo	% Ahorro
Zoncare – PM7000	$1496.00	$130.00	91.36%
Advanced-VSM-300C	$1390.00	$130.00	90.65%
Biolight - M1000	$1250.00	$130.00	89.6%

Capítulo 4: Conclusiones y Líneas de Trabajo Futuro

Conclusiones

Se evaluaron las herramientas tecnológicas de hardware y software para monitorear y evaluar el estado homeostático del paciente, determinando los sensores para la temperatura y la frecuencia cardiaca y componentes hardware y software basados en sistemas empotrados, que permitieron implementar el prototipo e-health.

El diseño del prototipo e-health se facilitó mediante el uso de sistemas empotrados, software de código libre y API's de uso gratuito, permitiendo minimizar los costos y manteniendo el rendimiento.

Dado que el prototipo e-health, fue implementado mediante sistemas empotrados, se utilizó para el protocolo de comunicación MQTT, por ser de fácil implementación y ligero para la transmisión de datos; para el almacenamiento de los datos se configuró una base de datos MongoDB alojada en la nube, que permitió manejar la data sin restricciones relacionales; para los servicios REST, integración con la base de datos, comunicación con API de mensajería y comunicación con los sensores se implementó el servidor de aplicación basado en Node.js por ser de fácil configuración e integración con los requerimientos requeridos por el prototipo e-health, lo cual permitió implementar un prototipo e-health viable para su uso.

La validación del prototipo e-health, se desarrolló de dos maneras, una validación funcional mediante una comparación con dispositivos comerciales donde se obtuvo un promedio de 0.7ºC de temperatura de diferencia y con respecto a las pulsaciones una diferencia de 3BPM entre los dispositivos; y una evaluación mediante encuestas a profesionales médicos arrojando una puntuación promedio de 4.9 sobre 5 puntos permitiendo demostrar que el dispositivo es funcional y cumple con las expectativas de personas calificadas para usarlo.

Se evaluó los costos del prototipo e-health, considerando los valores de los dispositivos y sensores utilizados obteniendo un valor de $130 cumpliendo con el modelo low cost; al comparar con dispositivos profesionales existentes en el mercado se ha obtenido un ahorro promedio de 90.48%.

Líneas de Trabajo Futuro

Diseñado el prototipo, que al momento sus servicios corren en servidores locales, se propone como línea de trabajo futuro que estos servicios se implementen mediante el modelo de ejecución de computación sin servidor; lo que permitirá agregar sensores para una mayor variedad de signos vitales.

Bibliografía

Araujo Mena, E. M. (2015). *Implementación de un sistema de video vigilancia para los exteriores de la UPS, mediante mini computadores y cámaras Raspberry PI.*

Back-End, R. (01 de 08 de 2020). *The Robotics Back-End*. Obtenido de The Robotics Back-End: https://roboticsbackend.com/

Balamba Camacho, B. D., & Sacristán Vargas, J. E. (2019). Prototipo funcional de un servicio e-Health para monitorear, transmitir y almacenar el estado de la presión arterial de pacientes crónicos-hipertensos.

Barillaro, S., De Luca, G., Valiente, W., Carnuccio, E., García, G., Volker, M., . . . Pérez, M. (2016). Diseño de sistema IoT de monitoreo y alarma para personas mayores. *XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016, Entre Ríos, Argentina).*

Bermúdez-Ortega, J., Besada-Portas, E., López-Orozco, J., Bonache-Seco, J., & De la Cruz, J. (2015). Remote web-based control laboratory for mobile devices based on EJsS, Raspberry Pi and Node. js. *IFAC-PapersOnLine*, *48*, págs. 158-163.

Blaya, J. A., Fraser, H. S., & Holt, B. (2010). E-health technologies show promise in developing countries. *Health Affairs, 29*, 244-251.

Camargo Bareño, C. I. (2011). Transferencia tecnológica y de conocimientos en el diseño de sistemas embebidos. *Universidad Nacional de Colombia*.

Castillo, J. N., Garcés, J. R., Navas, M. P., Segovia Jácome, D. F., & Naranjo Armas, J. E. (2017). Base de Datos NoSQL: MongoDB vs. Cassandra en operaciones CRUD (Create, Read, Update, Delete). *Revista Publicando, 4*, 79-107.

Castro Navarro, J. H. (2019). Diseño e implementación de un prototipo de automonitoreo preventivo de signos vitales para los trabajadores del área de producción–Copeinca.

Catalán Cantero, C., & Blesa Gascón, A. (2016). Enseñanza de sistemas empotrados: de Arduino a Raspberry Pi. *Actas de las XXII JENUI*, (págs. 351-354).

Cobo, D., & Daza, P. (2011). Signos vitales en pediatría. *Gastrohnup*, *13*, págs. A58-A58.

Collina, M. (2020). *Mosca*. Recuperado el 01 de 08 de 2020, de https://github.com/moscajs/mosca

Davis, J. C., Williamson, E. R., & Lee, D. (2018). A sense of time for JavaScript and Node. js: first-class timeouts as a cure for event handler poisoning. *27th {USENIX} Security Symposium ({USENIX} Security 18)*, (págs. 343-359).

Express. (2020). *Express*. Recuperado el 01 de 08 de 2020, de https://expressjs.com/es/

Eysenbach, G. (2001). What is e-health? *Journal of medical Internet research*, *3*, pág. e20.

Fritzing. (2020). *Fritzing*. Recuperado el 19 de 08 de 2020, de https://fritzing.org/

García Salvatierra, A. (2012). El Internet de las Cosas y los nuevos riesgos para la privacidad. *E_Telecomunicacion*.

Goel[1], V., Srivastava, Sharad, Pandit, Dharmendra, Tripathi[4], D., & Goel[5], P. (2018). Heart rate monitoring system using finger tip through IoT. *Heart, 5*(03).

Gomez Blazquez, A. (2016). Sistema sensor autònom sense fils de baix consum per a aplicacions industrials. *Universitat Politècnica de Catalunya*.

Gonzalez Mejia, C. M., & Rodríguez Sarmiento, L. A. (s.f.). Diseño e Implementación de una Red de Sensores para el Monitoreo de Señales Biomédicas Utilizando Requerimientos de IoT con el Grupo de Investigación Integra.

Haro Valenzuela, E. V., Guarda, T., Peñaherrera Zambrano, A. O., & Ninahualpa Quiña, G. (2019). Desarrollo backend para aplicaciones web, Servicios Web Restful: Node. js vs Spring Boot. *Revista Ibérica de Sistemas e Tecnologias de Informação*(E17), 309-321.

IEBS. (2020). *IEBS*. Recuperado el 01 de 08 de 2020, de https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/#:~:text=Metodolog%C3%ADas%20%C3%A1giles%20m%C3%A1s%20utilizadas,-Pero%2C%20%C2%BFcu%C3%A1les%20son&text=Existen%20diferentes%20opciones%20pero%20las,12%20principios%20del%20software%2

Kumbhar, P., Mulla, A., Kanagi, P., & Shah, R. (2018). Smart mirror using Raspberry PI. *International Journal For Research In Emerging Science And Technology*, *5.*

Luna, F., Peña, CM, & Iacono, M. (2014). Programador Web Full Stack.

Mafla Flores, S. M. (2019). Comparativa de los frameworks angular y primefaces para el desarrollo del aplicativo control de materia prima en la empresa Mastercubox SA, utilizando la metodología Scrum.

Maksimović, M., Vujović, V., Davidović, N., Milošević, V., & Perišić, B. (2014). Raspberry Pi as Internet of things hardware: performances and constraints. *Design issues, 3*(8).

Martín, A. E., Chávez, Susana Beatriz, Murazzo, María Antonia, Rodríguez, Nelson R, & Valenzuela, Adriana. (2015). MongoDB en ambiente cloud híbrido con OpenStack. *XVII Workshop de Investigadores en Ciencias de la Computación (Salta, 2015).*

Martínez Jimeno, J. (2018). Desarrollo de un sistema de monitorización de temperaturas en tiempo real para intercambiador de calor de doble tubo con sondas sumergibles de temperatura DS18B20 usando el miconcontrolador Arduino.

MongoDB. (2020). *MongoDB Atlas*. Recuperado el 01 de 08 de 2020, de https://www.mongodb.com/cloud/atlas

Moreno Cerdà, F. (2018). Demostrador arquitectura publish/subscribe con MQTT. *Universitat Politècnica de Catalunya*.

Narváez, M. E., Grefa, Pablo Ronny Calapucha, Caisa, Marco Vinicio Tarco, & Guisñan, Pamela Alexandra Buñay. (2020). Análisis de Desempeño entre MONGODB y COUCHDB utilizando Norma ISO/IEC 25000. *Revista Perspectivas, 2*(2), 13-20.

Núñez, C. V., Peña, Jairo Cardona, & Antolinez, Iván Saavedra. (2014). Telemonitoreo de datos cardiacos y respiratorios a través de un sistema Web con JSP. *Ingeniería y Desarrollo, 2*(1), 102-114.

Ortega Checa, J. S. (2019). Desarrollo del sistema backend web para el proceso de precontratación y contratación de la Empresa Mastercubox SA con el Framework Spring.

Pardo Agudelo, D. A., & Guacaneme Valbuena, Gerardo. (2016). Diseño e Implementación de un Sistema de Medición de Consumo de Energía Eléctrica y Agua Potable Remoto con Interacción al Usuario Basado en el Concepto "Internet de las Cosas".

Pedrera, A. C. (2017). Arduino para Principiantes: 2ª Edición. *IT Campus Academy*.

Penagos, S. P., Salazar, Luz Dary, & Vera, Fanny E. (2005). Control de signos vitales. *Guias para manejo de Urgencias. Bogotá, Colombia: Fundación Cardioinfantil*, p1465a1473.

Pérez, D. (2009). Sistemas embebidos y sistemas operativos embedidos. *Lecturas en ciencias de la computación. Universidad Central de Venezuela, Vols.% i de% 2ISSN*, 1316-6239.

Rahadian, H., & Arifin, Zaenal. (2016). Pemrosesan Data Pulse Sensor Amped pada Rancangan Sistem Informasi Dokter dan Pasien. *Semarang: Skripsi, Universitas Dian Nuswantoro Semarang*.

Raspberrypi. (2020). *Teach, Learn, and Make with Raspberry Pi – Raspberry Pi*. Recuperado el 08 de 02 de 2020, de https://www.raspberrypi.org

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University-Computer and Information Sciences, 30*(3), 291-319.

Ruiz Sánchez, W. R. (2016). Redes de sensores inalámbricos enfocadas a la medicina con énfasis en control de los signos vitales en pacientes adultos mayores. *PUCE*.

Sánchez, E. (2012). Diseño de un sistema de control domótico basado en la plataforma Arduino. *Master's thesis. Escuela Técnica Superior de Ingeniería Informática. Universidad Politécnica de Valencia*.

Schmiedehausen, K. (2018). Single page application architecture with angular.

Semle, A., & eFalcom, K. (2016). Protocolos IIoT para considerar. *Revista AADECA*.

Silberschatz, A., Korth, Henry F, Sudarshan, S, Pérez, Fernando Sáenz, Santiago, Antonio Ibarra, & Sánchez, Antonio Vaquero. (2002). Fundamentos de bases de datos.

Simon, D. E. (1999). An embedded software primer. *Addison-Wesley Professional, 1*.

Tejada Pardo, R. L. (2019). Implementacion De Un Sistema De Seguridad Iot. *Universidad Industrial de Santander, Escuela de Ingeniería Eléctrica*.

Terán Flores, E. d. (2019). Sistema de monitoreo remoto y visualización para dispositivos de análisis de signos vitales orientados a e-health. *Universidad de las Fuerzas Armadas ESPE*.

Twilio. (2020). *Twilio*. Recuperado el 03 de 08 de 2020, de https://www.twilio.com/

Valdivia-Caballero, J. J. (2016). Modelo de procesos para el desarrollo del front-end de aplicaciones web. *Interfases*(009), 187-208.

Valencia Zambrano, W. A. (2018). Diseño de prototipo doctor Pi para la medición y monitorización de signos vitales en adultos mayores utilizando sensores biométricos y médicos acoplados a raspberry Pi.

Vele Zhingri, C. A. (2016). Análisis de rendimiento entre la base de datos relacional: MySQL y una base de datos no relacional: MongoDB. *Universidad del Azuay*.

Wilson, E. V., & Lankton, Nancy K. (2004). Modeling patients' acceptance of provider-delivered e-health. *Journal of the American Medical Informatics Association, 11*(4), 241-248.

Wolf, M. (2012). Computers as components: principles of embedded computing system design. *Elsevier*.

Zárate-Ocaña, G., Ramos-Cuevas, MD, Contreras-Cariño, LB, Bélen-Luna, JC, & González-Morán, CO. (2018). Diseño y construcción de un prototipo biomédico para la adquisición vía remota de signos vitales utilizando tecnologías del internet de las cosas (IoT). *Memorias del Congreso Nacional de Ingeniería Biomédica, 5*(1), 470-473.

Anexos