UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

ASISTENTE VIRTUAL DE TELEMEDICINA CON DISPOSITIVO ELECTRÓNICO DE MEDICIÓN DE TRIAJE PARA EL DIAGNÓSTICO MÉDICO DE INFECCIONES RESPIRATORIAS UTILIZANDO INTELIGENCIA ARTIFICIAL.

Trabajo de Integración Curricular Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y redes

LÍNEA DE INVESTIGACIÓN: Programación y redes

AUTOR: Naythan Alexander Villafuerte Lozada

TUTOR: Ing. Víctor Santiago Manzano Villafuerte, Mg.

Ambato - Ecuador

marzo – 2023

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Integración Curricular con el tema: ASISTENTE VIRTUAL DE TELEMEDICINA CON DISPOSITIVO ELECTRÓNICO DE MEDICIÓN DE TRIAJE PARA EL DIAGNÓSTICO MÉDICO DE INFECCIONES RESPIRATORIAS UTILIZANDO INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Naythan Alexander Villafuerte Lozada, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la ejecución de la Unidad de Integración Curricular y la obtención del título de tercer nivel, de grado en la Universidad Técnica de Ambato y el numeral 7.4 del respectivo instructivo.

Ambato, marzo 2023.

Ing. Víctor Santiago Manzano Villafuerte**,** Mg.

TUTOR

AUTORÍA

El presente trabajo de Integración Curricular titulado: ASISTENTE VIRTUAL DE TELEMEDICINA CON DISPOSITIVO ELECTRÓNICO DE MEDICIÓN DE TRIAJE PARA EL DIAGNÓSTICO MÉDICO DE INFECCIONES RESPIRATORIAS UTILIZANDO INTELIGENCIA ARTIFICIAL es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2023.

Naythan Alexander Villafuerte Lozada

C.C. 1850039718

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Integración Curricular como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Integración Curricular en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, marzo 2023.

Naythan Villafuerte

Naythan Alexander Villafuerte Lozada

C.C. 1850039718

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Integración Curricular presentado por el señor Naythan Alexander Villafuerte Lozada, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado ASISTENTE VIRTUAL DE TELEMEDICINA CON DISPOSITIVO ELECTRÓNICO DE MEDICIÓN DE TRIAJE PARA EL DIAGNÓSTICO MÉDICO DE INFECCIONES RESPIRATORIAS UTILIZANDO INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la ejecución de la Unidad de Integración Curricular y la obtención del título de tercer nivel, de grado en la Universidad Técnica de Ambato y sus reformas y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, marzo 2023.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
JUAN PABLO PALLO NORONA

Ing. Juan Pablo Pallo, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
MARCO ANTONIO JURADO LOZADA

Ing. Marco Jurado, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

Con todo mi amor este proyecto es dedicado a mis papás, Carmita y Edisson que a pesar de la distancia siempre velaron por mi bienestar. A ustedes les debo todos mis logros, gracias por su apoyo incondicional. Simplemente resta decir que los quiero papi y mami, espero que el tiempo nos reúna una vez más.

Naythan Alexander Villafuerte Lozada

AGRADECIMIENTO

Agradezco a Dios por darme a unos padres y hermanos maravillosos, quienes creyeron en mí siempre y me brindaron su apoyo incondicional en los buenos y malos momentos de mi vida. A mis padres que son mi ejemplo de superación.

A mi familia que me aconseja y me vio crecer para formarme como una persona de bien. Un agradecimiento especial a José y Amanda por cuidarme como un hijo más en mi última etapa de carrera.

A mis amigos y compañeros de carrera Sara, Nancy, Pato y Núñez, ustedes fueron un gran apoyo en cada semestre que compartimos. Solo ustedes saben el sufrimiento de este logro.

A mis amigos de juegos Choclo, Koha, Erick y Alex, ustedes se convirtieron en los amigos que jamás imagine tener, las risas y momentos divertidos nunca faltaron. Ustedes son los mejores "mancos" del Fortnite.

A mis compañeros de casa Snowy y Koda, por hacerme compañía en mis momentos de soledad.

A mi tutor Ing. Santiago Manzano y la Dra. Verónica Córdova por su tiempo y colaboración en el desarrollo de este proyecto de grado.

Naythan Alexander Villafuerte Lozada

RESUMEN EJECUTIVO

La sintomatología del COVID-19 y otras infecciones respiratorias son muy similares, complicando el diagnóstico de estas enfermedades ya que tienen síntomas comunes. Por este motivo, se implementó un asistente virtual de telemedicina con dispositivo electrónico de medición de triaje para el diagnóstico médico de infecciones respiratorias utilizando inteligencia artificial, que proporciona una atención personalizada y oportuna a cada paciente.

El proyecto se desarrolló en colaboración con un profesional de la salud que brindo pautas para la medición de signos vitales, así como el proceso de diagnóstico de un paciente. En el caso del dispositivo electrónico, este cuenta con dos microcontroladores ESP8266 que se encargan de medir la frecuencia cardíaca, saturación de oxígeno en la sangre y temperatura corporal. Estos datos son mostrados por medio de una pantalla Oled y enviados al servidor de Google Cloud utilizando el protocolo MQTT. Dentro del servidor se encuentra alojada la aplicación web que consta de una pantalla de inicio de sesión, que verificara si el usuario se encuentra registrado. Una vez el usuario inicie sesión, accederá a una interfaz donde se muestran las mediciones de los signos vitales en tiempo real. Una vez revisado los signos vitales el usuario ingresa a la pantalla del asistente virtual (chatbot) que consta con procesamiento del lenguaje natural para simular la consulta con un doctor real e interpretar los síntomas ingresados por el usuario para ser almacenados en la base de datos MySQL. Al terminar de ingresar la sintomatología se inicia el algoritmo de inteligencia artificial con una precisión del 91%, que se encarga de determinar si el paciente tiene covid19, resfriado común o rinitis alérgica, estas tres clases fueron sugeridas por la doctora. Por último, el usuario acedera a una interfaz donde se encuentra los datos de todas las consultas realizadas al chatbot. Cabe mencionar que el aplicativo web fue desarrollado mediante el framework de Flask con las herramientas de HTML, CSS, Javascript, SQL y Bootstrap 5, para lograr que la aplicación sea responsiva, dinámica y robusta, ya que cuenta con seguridad tanto para el usuario como para el servidor.

Palabras clave: Infecciones respiratorias, covid19, Inteligencia Artificial, NPL, aplicación Web, MySQL, medición de signos vitales, MQTT, ESP8266, chatbot.

ABSTRACT

The symptoms of COVID-19 and other respiratory infections are very similar, complicating the diagnosis of these diseases since they have common symptoms. For this reason, a virtual telemedicine assistant was implemented with electronic triage measuring device for medical diagnosis of respiratory infections using artificial intelligence, which provides personalized and timely care to each patient.

The project was developed in collaboration with a health professional who provided guidelines for the measurement of vital signs, as well as the process of diagnosis of a patient. In the case of the electronic device, it has two ESP8266 microcontrollers that are responsible for measuring the heart rate, oxygen saturation in the blood and body temperature. This data is displayed by an Oled screen and sent to the Google Cloud server using the MQTT protocol. Inside the server is hosted the web application consisting of a login screen, which will verify if the user is registered. Once the user logs in, they will access to an interface where the measurements of the vital signs are shown in real time. After the review of the vital signs the user can enter to the screen of the virtual assistant (chatbot) that has natural language processing to simulate the consultation with a real doctor and interpret the symptoms entered by the user to be stored in the MySQL database. At the end of entering the symptomatology, the artificial intelligence algorithm starts with a precision of 0.91, which is responsible for determining whether the patient has covid19, common cold or allergic rhinitis, these three classes were suggested by the doctor. Finally, the user will move to an interface where the data of all the queries made to the chatbot is found. It is worth mentioning that the web application was developed using the Flask framework with the tools of HTML, CSS, Javascript, SQL and Bootstrap 5, to make the application responsive, dynamic and robust, since it has security for both the user and the server.

Keywords: Respiratory infections, covid19, Artificial Intelligence, NPL, Web application, MySQL, vital sign measurement, MQTT, ESP8266, chatbot.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de Investigación

Asistente virtual de telemedicina con dispositivo electrónico de medición de triaje para el diagnóstico médico de infecciones respiratorias utilizando inteligencia artificial.

1.1.1. Planteamiento del problema

En la última década, desde 2010 hasta 2020, la Organización Mundial de la Salud (OMS) estableció a las infecciones respiratorias como una de las principales causas de muerte de niños y adultos, estimando que alrededor de 2.4 millones de personas mueren al año a nivel mundial, representando un índice elevado de muerte de la tercera edad y niños menores a 5 años sin considerar el periodo neonatal. Según la proyección de la OMS en el año 2050 aproximadamente 2 billones de personas mayores requerirán atención en sus hogares debido a las secuelas de las enfermedades respiratorias como el deterioro de la salud pulmonar y el acrecentamiento de enfermedades respiratorias crónicas, como bronquitis y afecciones cardiovasculares[1].

En Ecuador, el Perfil Epidemiológico en 2019 confirmó que las infecciones respiratorias agudas fueron la etiología viral más común de infecciones respiratorias en pacientes menores de 2 años, según el Ministerio de la Salud Pública de Ecuador se identificó como la principal causa a los virus respiratorios representando con un 53,3%, bocavirus con 29,8%, rinovirs y eterovirus con 21,3%, adenovirus con 14,9%, parainfluenzas con 4,3% y otros [2].

En el caso de Tungurahua los pacientes que han presentado infecciones respiratorias en hospitales en su mayoría son niños, siendo las infecciones respiratorias agudas (IRA) una de las principales causas de morbilidad y mortalidad. Además, se registra un aumento de la tasa de hospitalización en invierno; donde la probabilidad de desarrollar IRA fue del 30,59% en un niño cuyo percentil de peso era menor que 3 y el promedio de estancia hospitalaria fue de 3,98 días. Según el Comercio del 21 de abril de 2020, la centralita 171 recibió 16.918 llamadas con solicitudes de telemedicina donde dependiendo de la complejidad de la consulta era atendido vía telefónica o se le asignaba un turno al centro de salud pública cercano al paciente de manera gratuita.

Este tipo de servicio se va proyectando con una mayor demanda en los últimos años por el peligro de contagio de enfermedades respiratorias. No obstante, la falta de personal capacitado que brinde el servicio a futuro será limitado, existiendo la necesidad de implementar tecnologías como la Inteligencia Artificial que cubra esta carencia ofreciendo un diagnóstico médico en menor tiempo [3].

1.2.Antecedentes Investigativos

En base al análisis de la diferente bibliografía en repositorios universitarios, al igual que base de datos de artículos científicos se logró encontrar los siguientes antecedentes:

M. Gandhi, V. Singh, V. Kumar (2019), del Instituto Sathyabama de Ciencias y Tecnología, en India, publicaron un artículo científico con el tema "Intellidoctor – AI based mecial assistant", donde proponen un asistente médico personal para brindar atención médica inteligente accesible, esta aplicación interactiva analiza síntomas para diagnosticar, predecir condiciones médicas, generar tratamientos y sugerencias basadas en la extracción de información utilizando el Procesamiento del Lenguaje Natural (NLP) mediante entradas proporcionadas por el usuario, para lo cual emplea un motor de inteligencia artificial basado en el clasificador de Naïve Bayes. Además de eso, la aplicación realiza un seguimiento de las actividades de salud del usuario, como el recuento de pasos, el seguimiento del sueño, la detección del ritmo cardíaco y otros parámetros, mostrando a los usuarios sus informes de salud periódicos. Incorpora varias actividades de acondicionamiento físico rastreadas y otros factores como su edad, sexo, ubicación, registros médicos anteriores y consumo de calorías para realizar un análisis más preciso. Realizando así un diagnóstico integral preciso, que también sirve como un dispositivo de preselección para los médicos [4].

Espinoza Sonia (2020) de la Universidad de Guayaquil, realizó un "Desarrollo e implementación de una plataforma web con chatbot para la comunicación activa entre usuario e información del portafolio de servicio de la empresa Electricsystems de la ciudad de Guayaquil", el cual indago acerca de los datos de la empresa Electrisystems para conocer su estado y determinar las necesidades que debe cubrir el Chatbot. Este chatbot se desarrolló en una página web con las herramientas de Bootstrap, HTML, CSS y JavaScript, ya para brindarle un procesamiento del lenguaje natural con la

plataforma Dialogflow que ofrece una respuesta de forma natural simulando a una persona, logrando así una mejor interacción con el cliente, ofreciendo un servicio las 24 horas del día [5].

Garibay Fabricio (2020) de INFOTEC Centro de Investigación e Innovación en Tecnologías de la información y Comunicación, realizó un "Diseño e implementación de un asistente virtual (chatbot) para ofrecer atención a los clientes de una aerolínea mexicana por medio de sus canales conversacionales" donde implemento un plan con seis pasos encargados de buscar los roles asignados a cada participante y sus obligaciones, para agregarlos al Chatbot. Este asistente se realizó empleando la herramienta "CP-Bot" que utiliza inteligencia artificial para mejorar la comunicación entre los clientes y el chatbot. Este proyecto mejoró la calidad del servicio a los clientes de la aerolínea obteniendo más usuarios afiliados ya que se agilizó el tiempo en respuesta en base a la demanda de clientes [6].

Artica Edwing (2020) de la Universidad Continental, Perú, realizó una "Implementación de un asistente virtual para la atención al cliente en Electrocentro S. A. de Huancayo", el cual implementó un asistente virtual ADA para la empresa Electrocentro S. A. con el fin de mejorar la calidad de atención al cliente. La metodología de diseño utilizada fue SCRUM, al igual que las herramientas de Dialogflow el cual emplea Procesamiento de Lenguaje Natural. El mismo que se comunican utilizando un webhooks con los servidores que cuentan con una arquitectura Serverless albergadas en Netlify. Finalmente se conecta con el endpoint del Restful API para que los usuarios puedan realizar consultas, obteniendo como resultados la optimización de los tiempos de respuesta frente a peticiones de los clientes con una valoración positiva del 77.6%, estando disponible las 24 horas del día [7].

G. Bonales, N. Pradilla, E. Citlali (2020), de la Universidad Complutense de Madrid, en España, publicaron un artículo científico con el tema "Chatbot como herramienta comunicativa durante la crisis sanitaria de la COVID-19 en España", donde se hace un análisis de la respuesta de los asistentes virtuales frente a la crisis sanitaria en España. Por este motivo se hizo un muestreo de 5 Chatbots lanzados en el mercado, para los cuales se utilizó un caso de estudio donde se realiza una encuesta a los usuarios y una entrevista a profesionales de la salud para determinar las particularidades de cada

asistente virtual. Finalmente, el resultado de este estudia determino que aproximadamente el 30% de la muestra utilizo el asistente como recurso de información acerca de los síntomas relacionados con el COVID-19 [8].

Todas las investigaciones que se han encontrado de forma bibliográfica aportan con diferentes investigaciones sobre los tipos de inteligencia artificial que podrían ser aplicables en el proyecto de investigación dependiendo de la base de datos que se desea utilizar, así como los softwares de desarrollo utilizados para crear los chatbots.

1.3. Fundamentación Teórica

1.3.1. Infecciones Respiratorias

Una infección respiratoria es una dolencia que se genera en el sistema respiratorio como consecuencia de la presencia de microorganismos como bacterias o virus. Generalmente se caracteriza por un inicio repentino y de corta duración, alrededor de dos semanas [9].

1.3.1.1. Causas más Frecuentes

La función principal de las vías respiratorias es llevar el aire a los alvéolos para que allí se produzca el intercambio gaseoso. Esta etapa constantemente es atacada por patógenos, que en su mayoría son virus. Las infecciones respiratorias virales afectan al cuerpo dependiendo de la capacidad del sistema respiratorio para responder adecuadamente y evitar el deterioro de las vías respiratorias. Entre los virus más comunes se encuentran [10]:

- Virus sincitial respiratorio
- Rinovirus
- Influenza
- Adenovirus
- Metaneumovirus

1.3.2. Triaje

El triaje es una técnica que permite la administración de la exposición clínica con el objetivo de controlar adecuadamente y de forma segura a los pacientes cuando la

demanda y las necesidades clínicas superan la capacidad del establecimiento de la salud [11].

1.3.2.1. Triaje de signos vitales

El triage de signos vitales es un sistema de selección y clasificación de pacientes en los servicios de urgencia, basado en sus necesidades terapéuticas y los recursos disponibles para atenderlo. La Resolución 5596 del 24 de diciembre de 2015 del Ministerio de Salud y Protección Social del Ecuador estipuló cinco categorías de triage como se muestra en la figura 1, con la salvedad que los tiempos establecidos de atención no aplicarán en situaciones de emergencia o desastre con múltiples víctimas [12].

Figura 1: Tipos de Triaje [12]

1.3.2.2. Métodos de adquisición de signos vitales

Los signos vitales son una serie de mediciones clínicas, entre las más esenciales para un diagnóstico médico se incluyen la temperatura (Temp), frecuencia cardíaca (BPM) y la saturación de oxígeno en la sangre (SpO2) [13]. La monitorización de signos

vitales es una actividad importante realizada por profesionales de la salud, para evaluar el progreso y detectar cualquier irregularidad en el estado físico del paciente [14]. Para la adquisición de signos vitales existen dos tipos de métodos como se observa en la figura 2 [15].

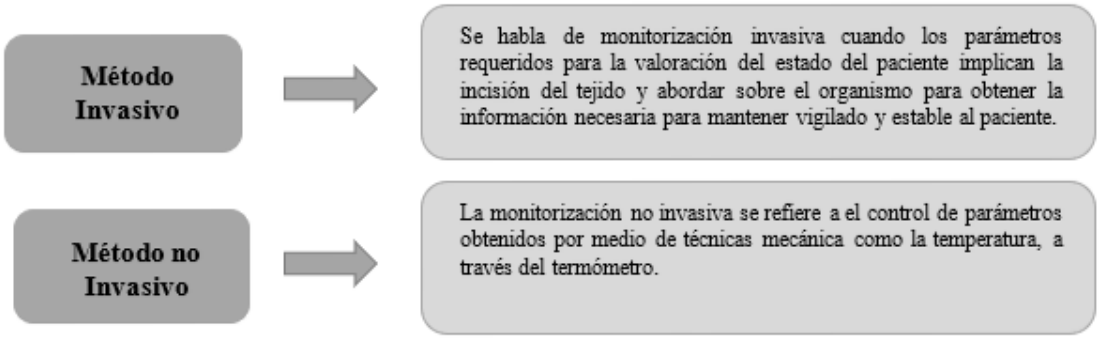


Figura 2: Métodos de adquisición de signos vitales [15]

1.3.2.3. Método no invasivos o mínimamente invasivos

Hay ventajas obvias para los dispositivos de medición biomédica que no necesitan ser insertados en el sujeto, incluyendo comodidad y seguridad. Además, no es probable que dichos dispositivos tengan una influencia negativa importante en la condición física del paciente ya que este no afectara su salud. Por lo tanto, especialmente en el ámbito clínico, ha habido una tendencia durante varias décadas para desarrollar y utilizar cuando sea posible dispositivos de medición no invasiva como muestra la tabla 1 [16].

Tabla 1: Métodos no invasivos para la medición de signos vitales

	Frecuencia Cardíaca (BPM)	**Saturación de oxígeno (SpO2)**	**Temperatura**
Método	No Invasivo	No Invasivo	No Invasivo
Adquisición	Pulso	Pulsioximetria	Temperatura corporal
Descripción	Detección de la dilatación arterial, de forma que se contabilice la cantidad de latidos que producen las arterias en un tiempo.	Método de adquisición de porcentaje de saturación de oxígeno SpO2 transportado por la hemoglobina en la sangre.	Método basado en elementos térmicos adaptables a mediciones sobre la piel.
Instrumento	Pulsómetro	Oxímetro de Pulso	Termómetro corporal

Elaborado por: El investigador basado en [17].

1.3.3. COVID-19

La enfermedad mundialmente conocida como COVID-19 o coronavirus es originada por el síndrome respiratorio agudo grave tipo 2 o más conocido como SARS-CoV-2 que se identificó por primera vez en Wuhan, provincia de Hubei de la República Popular de China, en diciembre de 2019 [18]. El COVID-19 presenta gotículas respiratorias que pueden llegar a tener entre cinco a diez micrómetros (μm) de diámetro, además sus núcleos tienen un diámetro por debajo de los cinco μm. Según la Organización Mundial de la Salud el coronavirus es transmitido esencialmente a través del contacto directo con personas infectadas. Tras el estudio de más de setenta y cinco mil casos de COVID-19 se determinó que la mayoría de estos no fue a causa de transmisión aérea [19].

1.3.3.1. Formas de transmisión del COVID-19

Según estudios epidemiológicos y virológicos, las personas infectadas pueden ser sintomáticas y asintomáticas mismas que tienen más probabilidades de transmitir el virus a otras personas, ya sea a través del contacto directo con personas infectadas o el contacto con superficies u objetos contaminados [18]. En la figura 3 se puede visualizar las formas de contagio según la Organización Mundial de la Salud (OMS).

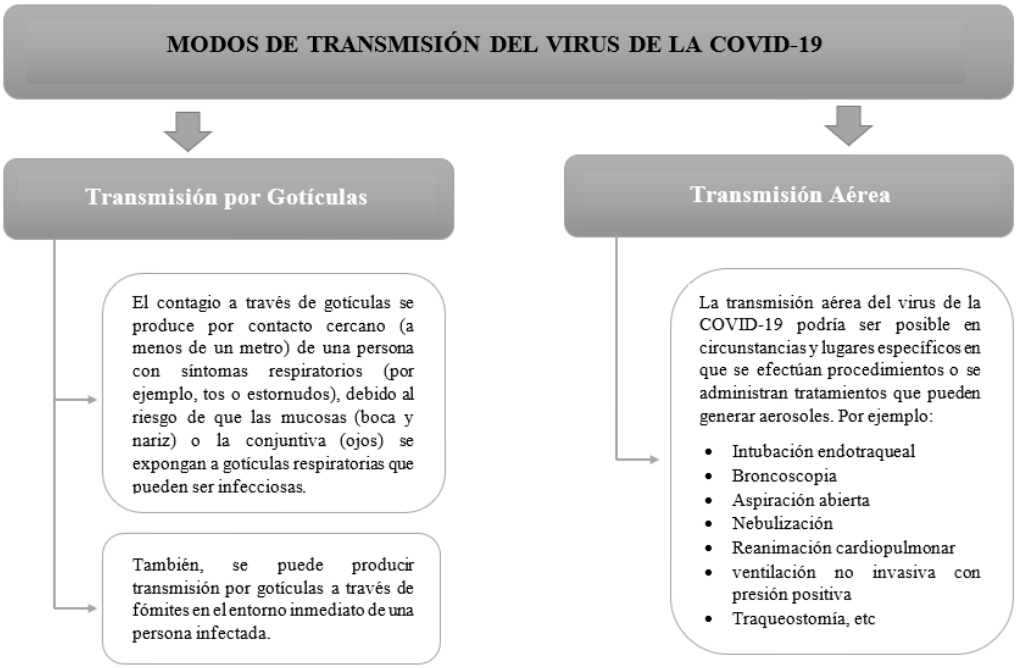


Figura 3: Transmisión del COVID-19 [18]

1.3.3.2. Síntomas del COVID-19

El COVID-19 altera de distintos modos a los pacientes infectados; sin embargo, la mayoría de ellos desarrollan un malestar entre leve y moderado sin la necesidad de ser hospitalizados [20]. Entre los síntomas más comunes desarrollados por pacientes infectados con las variantes de COVID-19 se encuentran:

- Fiebre
- Tos
- Cansancio
- Pérdida del gusto o el olfato.
- Dolor de garganta
- Dolor de cabeza
- Dolores y molestias
- Diarrea
- Dificultad para respirar o falta de aire
- Dolor en el pecho

1.3.3.3. Comportamiento de los signos vitales en pacientes con COVID-19

El triaje de los signos vitales de pacientes diagnosticados con COVID-19 determinara la gravedad del infectado como muestra la tabla 2 [21].

Tabla 2: Comportamiento de los signos vitales en pacientes con COVID-19 [21]

Estado	Leve	Moderado	Gravedad
Saturación (SpO2)	$\geq 95\%$	93% a 94%	$\leq 92\%$
Frecuencia respiratoria	≤ 20	21 a 24	≥ 25
Frecuencia Cardíaca (BPM)	≤ 90	91 a 130	≥ 131
Temperatura	$\geq 37.8\,°C$	$\geq 37.8\,°C$	$\geq 37.8\,°C$
Conducta	Considerar monitoreo remoto	Considerar internación hospitalaria / evaluación presencial.	Internación hospitalaria

1.3.4. Rinofaringitis (Resfriado común)

La rinofaringitis es un trastorno infeccioso de la mucosa nasal y faringe, también conocida como resfriado común, es la infección más común en cualquier edad. Los

virus son la principal causa del resfriado común, principalmente el rinovirus. Normalmente, el diagnóstico de esta enfermedad es leve, pero este podría empeorar y causar rinosinusitis aguda, principalmente en presencia de factores predisponentes locales, incluyendo hipertrofia de adenoides, poliposis nasal, septal desviación, promoviendo la obstrucción mecánica o factores sistémica, que incluyen alergia, disnesia ciliar e inmune deficiencia a virus respiratorios [22].

1.3.4.1. Formas de transmisión del resfriado común

El resfriado común generalmente es causado por un rinovirus que se transmite a través del contacto con la saliva o las secreciones nasales de alguien que ya está infectado por el virus. Estornudar o toser hace que el virus se propague y contamine las superficies, que otras personas pueden tocar y, por lo tanto, contraer el virus. Debido a que los virus del resfriado se transmiten con extrema rapidez, es esencial que se haga un esfuerzo para evitar que la infección se transmita a otras personas [22].

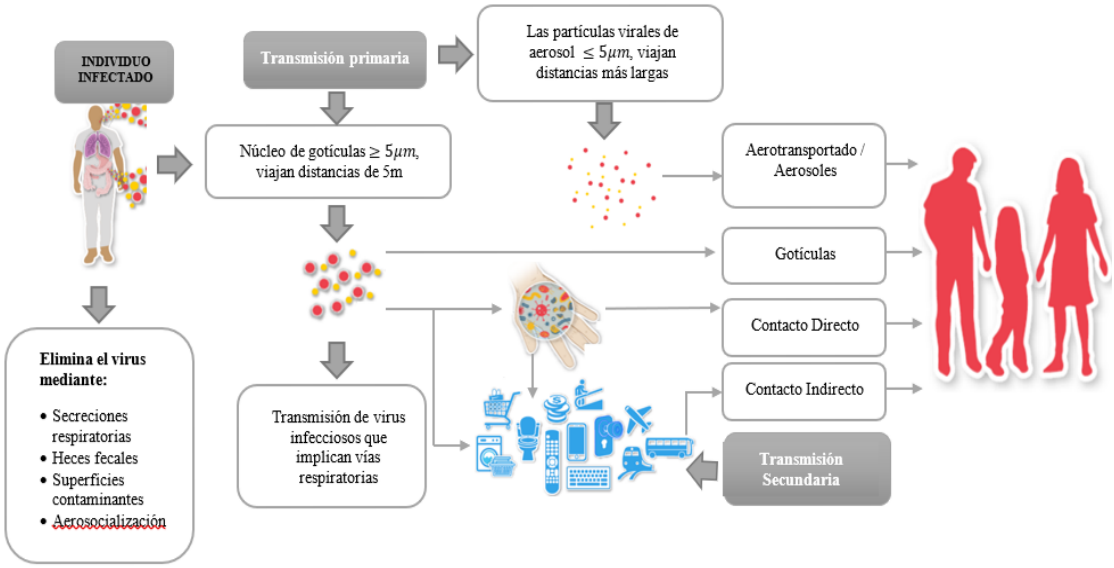


Figura 4: Transmisión del Resfriado común [22]

La transmisión primaria (directa) del resfriado común se muestra en la figura 4 en donde la principal causa es la inhalación directa de gotículas respiratorias grandes o la inhalación de pequeñas gotículas en el aire. Mientras que la transmisión secundaria (indirecta) del resfriado común también puede ocurrir a través del contacto con superficies contaminadas por gotículas afectando a las vías respiratorias. Entre los agentes contaminantes de las superficies encontramos a los aerosoles u otros fluidos corporales del paciente (esputo, mucosidad, sangre, líquido lagrimal, semen, orina o

heces). Para mitigar el contagio de la enfermedad se recomienda lavarse las manos regularmente y se debe advertir a las personas que eviten tocarse la cara en la medida de lo posible [23].

1.3.4.2. Síntomas del resfriado común

Al momento de que los patógenos externos infectan el organismo el resfriado al comienzo infectan las vías respiratorias y los senos paranasales, haciendo que la nariz produzca una mucosidad de color transparente. Esta acción es un mecanismo de defensa del organismo para tratar de eliminar los virus y aliviar los senos paranasales. En el transcurso de los días el color de la mucosidad puede cambiar, pero esto no significa que la enfermedad se haya agravado. Entre los síntomas más recurrentes se encuentran [24]:

- Nariz taponada
- Secreción nasal acuosa, que puede ir acompañada de dolor de garganta, tos,
- Fiebre
- Estornudos frecuentes, a menudo acompañados de picazón e irritación nasal
- Prurito Nasal
- Lagrimeo
- Respiración bucal
- Roncar por la noche

La obstrucción nasal puede dificultar la respiración y la lactancia y puede ser los primeros síntomas de sarampión, influenza u otra enfermedad. La condición puede infectarse secundariamente o complicarse con otitis media y aguda sinusitis en niños menores de 5 años [24].

1.3.4.3. Comportamiento de los signos vitales en pacientes con resfriado común

Según estudios realizados por la Organización mundial de la Salud en Irán, se estudió el comportamiento de los signos vitales de los pacientes diagnosticados con rinofaringitis tuvieron el siguiente comportamiento que se observan en la tabla 3 [25].

Tabla 3: Signos vitales en pacientes con Rinofaringitis [25]

Estado	Variable	Frecuencia (%)
Ritmo Cardíaco (BPM)	≥ 90	41.5
	< 90	58.5
Frecuencia respiratoria	≥ 20	56.1
	< 20	43.9
Presión arterial	$\geq 90\ mmHg$	15
	90-140 $mmHg$	74
	$< 140\ mmHg$	11
Temperatura	≥ 36.5	8
	36.6-37.7	21
	≤ 37.8	71

1.3.5. Rinitis Alérgica

La Nasofaringitis o más conocida como rinitis alérgica es una inflamación de las fosas nasales, generalmente asociada con secreción nasal acuosa y picazón en la nariz y los ojos. También es considerado un trastorno común que afecta hasta un 40% de la población [26]. Mientras que la rinitis alérgica crónica afecta del 10 al 20% de la población, y la evidencia sugiere que la prevalencia de la enfermedad va en aumento. La rinitis alérgica en la actualidad se ha asociado con deterioros significativos en la calidad de la vida, el sueño y el rendimiento laboral [27].

Esta enfermedad respiratoria, que parece inofensiva en las etapas iniciales, tiene un inmenso potencial para problemas más graves como asma, alergias persistentes no controladas y sinusitis. La Organización Mundial de la Salud (OMS) informó que alrededor de 400 millones de personas en todo el mundo sufre de rinitis alérgica. Afecta a todos los grupos de edad, entre el 10 y el 30 por ciento de los adultos. Continúa viendo un rápido aumento entre los niños y adultos jóvenes, quienes son los más afectados por este problema desatendido [27].

1.3.5.1. Causas más comunes de la rinitis alérgica

Los principales causantes de la rinitis alérgica son los alérgenos que se encargan de desencadenar reacciones alérgicas en los pacientes, como se puede observar en la figura 5, estos pueden causar que el cuerpo libere sustancias químicas que provocaran síntomas de alergia manifestados por una picazón cerca de la nariz acompañada por una inflamación en las fosas nasales [28].

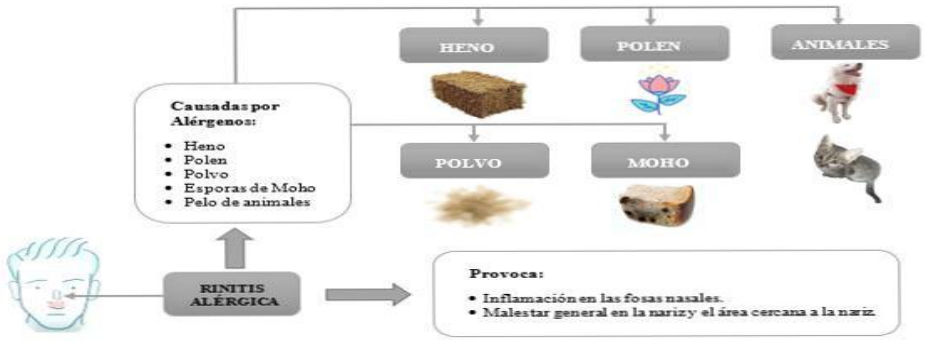


Figura 5: Alérgenos causantes de la Rinitis Alérgica [28]

1.3.5.2. Síntomas de la rinitis alérgica

La rinitis alérgica ocurre poco después de entrar en contacto con una sustancia a la que una persona es alérgica, manifestando los siguientes síntomas [24]:

- Picazón en la nariz
- Problemas con el olfato
- Moquera
- Estornudos
- Lagrimeo
- Congestión nasal
- Ronquidos
- Tos
- Dolor de garganta
- Círculos oscuros debajo de los ojos
- Hinchazón debajo de los ojos
- Malestar general
- Dolor de cabeza

1.3.5.3. Comportamiento de los signos vitales en pacientes con rinitis alérgica

Por lo general, los pacientes con nasofaringitis o rinitis alérgica aparentar presentar síntomas de resfriado sin estar enfermos con dicha enfermedad. Sin embargo, tras realizar un examen físico los pacientes con rinitis alérgica muestran signos vitales normales como muestra la tabla 4. No obstante, los síntomas como secreción nasal, mucosa nasal hiperémica y linfadenopatía cefalea leve suelen ser notables [29].

Tabla 4: Signos vitales en pacientes con rinitis alérgica [29].

Estado	Variable
Ritmo Cardíaco (BPM)	60-100
Frecuencia respiratoria	≥ 20
Presión arterial	$\leq 120\ mmHg$
Temperatura	36.1-37.2

1.3.6. Sistema Electrónico

Un sistema electrónico es una interconexión física de componentes o elementos interrelacionados entre sí, destinados a satisfacer una necesidad, dentro de este se distingue las siguientes partes para su funcionamiento como se muestra en la figura 6, que está conformado por la etapa de captación de las señales de entrada a través de los transductores (sensores de entrada) que transforma una determinada variable física en una señal eléctrica, la cual es procesada mediante un microcomputador como una Raspberry pi o una tarjeta programable como lo es Arduino. Finalmente, la señal procesada se visualiza mediante un dispositivo de salida; además, es utilizada para controlar los actuadores conectados al sistema electrónico [30].

Figura 6: Esquema básico de un sistema electrónico [30]

1.3.6.1. Microcontroladores

Un microcontrolador es un circuito integrado único que comprende varios elementos, incluidos un microprocesador, temporizadores, contadores, puertos de entrada/salida (E/S), memoria de acceso aleatorio (RAM), memoria de solo lectura (ROM) y algunos otros componentes. Estas partes trabajan juntas para ejecutar un conjunto preprogramado de tareas específicas. Así, un microcontrolador es como una pequeña computadora que procesa e incluso ejecuta el control en un dispositivo electrónico [31].

1.3.6.2. Sensores de entrada

Los sensores de entrada utilizan los puertos analógicos para producir un cambio de tensión, corriente o resistencia en respuesta a la variación de una variable física medida en el ambiente [32].

1.3.6.3. Sensores para el triaje de signos vitales

La tecnología de sensores para monitorear los signos vitales es un tema importante para diversas aplicaciones de servicios, como plataformas de entretenimiento y personalización y sistemas de Internet de las cosas (IoT), así como para fines médicos tradicionales o para la predicción de indicaciones de enfermedades mediante inteligencia artificial. Los signos vitales para el control incluyen la respiración y la frecuencia cardíaca, la temperatura corporal, la presión arterial, la saturación de oxígeno, el electrocardiograma, la concentración de glucosa en sangre, las ondas cerebrales. Los principales sensores utilizados por doctores se pueden observar en la tabla 5 [33].

Tabla 5: Sensores para medir señales vitales

Nombre	Tipo de terminal	Acción
Sensor de temperatura	Analógico	La señal de salida aumenta a medida que aumenta el nivel de temperatura.
Sensor de frecuencia cardiaca	Analógico	Funciona iluminando el dedo con un diodo led con una longitud de onda de 550 nm y midiendo la cantidad de luz reflejada con un fotosensor.
Sensor de nivel de oxigeno	Analógico	Se utiliza una luz roja con una longitud de onda de 600nm que es absorbida en mayor cantidad por la sangre que contiene más oxígeno, por lo que se puede comparar las medidas realizadas con led infrarrojo con una longitud de 950nm con las realizadas con led rojo y encontrar el porcentaje de oxígeno presente en la sangre.

Elaborado por: El investigador basado en [33].

Sensor de temperatura

Un sensor de temperatura es un dispositivo electrónico que mide la temperatura de su entorno y convierte los datos de entrada en datos electrónicos para registrar, monitorear o señalar los cambios de temperatura [34].

Sensor de frecuencia cardíaca y nivel de oxígeno

La frecuencia cardíaca y el nivel de saturación de oxígeno en la sangre son parámetros médicos esenciales que pueden medirse ópticamente mediante un led y un infrarrojo. Sin embargo, ciertos individuos presentan variabilidad en la absorción cutánea que puede afectar la cuantificación adecuada tanto del pulso como del nivel de oxígeno en sangre [34].

1.3.7. Inteligencia artificial

La inteligencia artificial (IA) es la capacidad de una computadora o un robot controlado por una computadora para realizar tareas que generalmente realizan los humanos porque requieren inteligencia y discernimiento humanos. Aunque no hay IA que puedan realizar la amplia variedad de tareas que puede hacer un humano común, algunas IA pueden igualar a los humanos en tareas específicas [35].

1.3.7.1. Inteligencia artificial y su relevancia en la medicina

En el futuro de la medicina tradicional se está dando grandes avances ya que en la actualidad un paciente puede ver su condición médica desde un computador antes de asistir con un médico. A través de los avances en inteligencia artificial (IA), parece posible que los días de diagnóstico erróneo y tratamiento de los síntomas de la enfermedad queden atrás. La acumulación de datos generados en clínicas y almacenados en registros médicos electrónicos a través de pruebas comunes e imágenes médicas permite más aplicaciones de inteligencia artificial y medicina basada en datos de alto rendimiento. Estas aplicaciones han cambiado y seguirán cambiando la forma en que los médicos y los investigadores abordan la resolución de problemas clínicos [36].

1.3.7.2. Hardware para la Inteligencia Artificial (IA)

Existen diferentes tipos de hardware para ejecutar un algoritmo dedicado a la inteligencia artificial. Sin embargo, el nivel de eficiencia de la IA depende de la capacidad del computador respecto a su RAM, almacenamiento interno, tarjeta de video, entre otras, entre los hardware especializados se encuentran [36]:

- Arrays sistólicos
- ASICs (Circuito Integrado para Aplicaciones Específicas)

- FPGA (Field Programmable Gate Arrays)
- GPUs (Graphics Processing Unit)

1.3.7.3. Machine Learning

Machine learning (Aprendizaje automático) es una ramificación de la inteligencia artificial (IA) en combinación con la informática enfocada en el uso de las bases de datos existentes para desarrollar algoritmos de inteligencia artificial que pretenden imitar el comportamiento de los seres humanos al momento de aprender, tratando de mejorar gradualmente la precisión de la toma de decisiones o predicciones [37].

1.3.7.4. Ciclo de vida de Machine Learning

Machine Learning ha marcado un gran avance a la informática ya que permite a los algoritmos la capacidad de aprendizaje automáticamente sin ser programado para una situación en específico. El ciclo de vida de Machine Learning se puede observar en la figura 7 donde se muestra que el mismo es un proceso cíclico, empezando por la adquisición de datos, preparación de datos donde se acondiciona el dataset para construir el modelo de IA que finalmente será evaluado y mejorado a medida que los datos almacenados sean mayores [38].

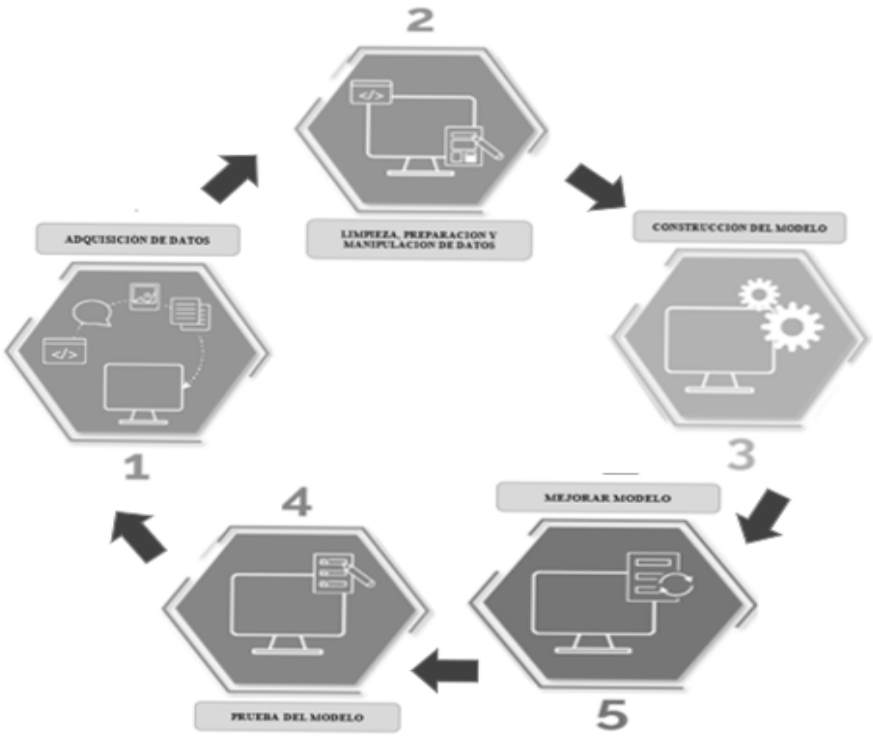


Figura 7: Ciclo de vida de Machine Learning [36]

1.3.7.5. Tipos de algoritmos de Machine learning

Los algoritmos de aprendizaje automático se organizan en taxonomía, según el resultado deseado del algoritmo. Los tipos de algoritmos comunes para machine learning se pueden ver en la tabla 6 [39]:

Tabla 6: Tipos de algoritmos para machine learning

Tipo	Definición
Aprendizaje supervisado	Sucede cuando el algoritmo genera una función que asigna entradas a las salidas deseadas. Una formulación estándar de la tarea de aprendizaje supervisado es el problema de clasificación por lo que se requiere que aprenda aproximar el comportamiento de una función que mapea un vector en una de varias clases mirando varios ejemplos de entrada-salida de la función.
Aprendizaje no supervisado	Modela el conjunto de entradas sin tener disponible los ejemplos etiquetados para dar una predicción.
Aprendizaje semisupervisado	Este tipo combina ejemplos etiquetados y no etiquetados para generar un clasificador apropiado que interprete la función.
Aprendizaje por refuerzo	El algoritmo aprende cómo actuar dependiendo de su observación del entorno. Cada acción que realice el algoritmo tiene algún impacto en el entorno. En función del resultado obtenido se dará una recompensa o una penalización, proporcionándose así una retroalimentación que guía el algoritmo de aprendizaje basándose en la evolución natural que atraviesan los seres vivos.
Transducción	Similar al aprendizaje supervisado, pero no construye explícitamente una función: en cambio, trata de predecir nuevos resultados de asignación de cada clase basados en entradas de entrenamiento.

Elaborado por: El investigador basado en [39].

1.3.7.6. Algoritmos de aprendizaje supervisado

El aprendizaje supervisado es bastante común en los problemas de clasificación porque el objetivo suele ser conseguir que el ordenador aprenda un sistema de clasificación de acuerdo con las necesidades, teniendo como ejemplo el reconocimiento de dígitos. De manera general, el aprendizaje supervisado también es llamado algoritmo de machine learning los cuales son [40]:

- Linear Classifiers
    - Logical Regression
    - Naïve Bayes Classifier
    - Support Vector Machine
- K-Means Clustering
- Boosting
- Decision Tree
    - Random Forest

1.3.7.7. Evaluación de un algoritmo de Machine Learning

La evaluación de los algoritmos de Machine Learnig es una parte de cualquier proyecto que utilice IA; sin embargo, la mayor parte de programadores en sus inicios tienden a usar solo la precisión de clasificación del modelo de aprendizaje, esto puede causar que el algoritmo arroje clasificaciones erróneas que en el área de la salud resulta peligroso. Existente diferentes métricas para evaluar el rendimiento de un modelo de Machine Learning, las cuales son[41]:

- Confusion Matrix
- Classification Accuracy
- Precision
- Recall or Sensitivity
- F1 Score

Confusion Matrix (Matriz de confusión)

La matriz de confusión es la forma más fácil de medir el desempeño de un problema de clasificación empleando Machine Learning en donde la salida puede ser de dos o más tipos de clases. Una matriz de confusión prácticamente es una tabla compuesta por dos dimensiones las cuales son Actual y Predicción, que son la comparación entre los valores reales y los valores calculados mediante predicción. Además, ambas dimensiones están compuestas a su vez de Verdaderos positivos, Verdaderos negativos, Falsos positivos, Falsos negativos como se muestra a continuación en la figura 8 [42]:

		Predicción	
		Positivos	Negativos
Actual	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos negativos (VN)

Figura 8: Matriz de confusión [42]

La explicación de los términos asociados con la matriz de confusión se explica en la siguiente tabla 7:

Tabla 7: Elementos de la matriz de confusión

Elemento	Definición
Verdaderos Positivos (VP)	Es el caso cuando tanto la clase real como la clase predicha del punto de datos es positivo.
Falsos Positivos (FP)	Es el caso cuando tanto la clase real como la clase predicha de punto de datos es negativo.
Falsos Negativos (FN)	Es el caso cuando la clase real de punto de datos es negativo y la clase predicha del punto de datos es positivo.
Verdaderos Negativos (VN)	Es el caso cuando la clase real de punto de datos es positivo y la clase predicha del punto de datos es negativa.

Elaborado por: El investigador basado en [42].

Classification Accuracy (Predicción de clasificación)

La precisión de clasificación puede definirse como la cantidad de predicciones correctas realizadas por un modelo de Machine Learning por medio de los valores que se encuentran en la diagonal principal de la matriz como muestra la figura 9 [43].

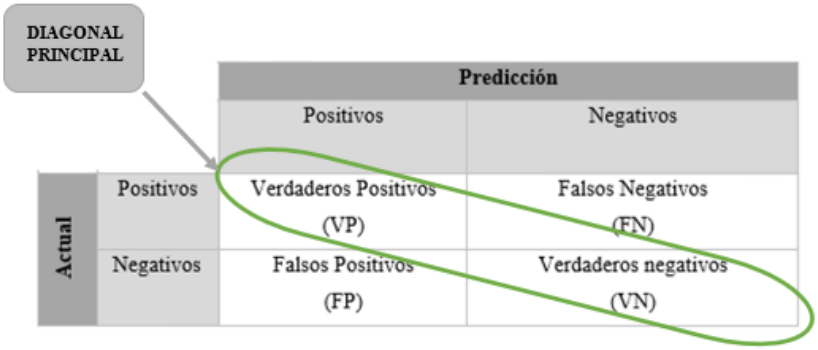


Figura 9: Diagonal principal de la matriz de confusión [43].

Fórmula para la predicción de clasificación:

La predicción de clasificación es calculable utilizando la matriz de confusión como se observa en la ecuación (1):

$$A = \frac{VP + VN}{VP + FN + FP + VN} = \frac{VP + VN}{TM} \qquad \text{Ecuación (1)}$$

Donde:

A → Accuracy (Precisión de clasificación)

VP → Verdaderos Positivos

VN → Verdaderos Negativos

FP → Falsos Positivos

FN → Verdaderos Negativos

TM → Total de Muestras

Precision (Precisión)

La precisión, es utilizada para realizar correcciones en los resultados positivos en relación con el número total de predicciones positivas realizadas por el modelo de Machine Learning [44]. Este se puede calcular por medio de la matriz de confusión utilizando la ecuación (2):

$$P = \frac{VP}{VP + FP} \qquad \text{Ecuación (2)}$$

Donde:

P → Presicion (Precisión)

VP → Verdaderos Positivos

FP → Falsos Positivos

Recall or Sensitivity (Recordatorio o Sensibilidad)

El recordatorio o sensibilidad, es utilizada para realizar correcciones en los resultados positivos en relación con el número total de positivos reales con los que fue entrenado el modelo de Machine Learning [45]. Este se puede calcular por medio de la matriz de confusión utilizando la ecuación (3):

$$R = \frac{VP}{VP + FN}$$ Ecuación (3)

Donde:

R → Recall (Sensibilidad)

VP → Verdaderos Positivos

FN → Falsos Positivos

F1 Score (Puntuación F1)

La puntuación F1 es la media armónica entre la precisión y la recuperación. El rango para F1 Score esta entre [0, 1]. Esta métrica permite verificar qué tan preciso es su clasificador es decir cuántas instancias clasifica correctamente, así como qué tan robusto es al no perder una cantidad significativa de instancias [46]. La media armónica de la puntuación F1 se puede calcular mediante la ecuación (4):

$$F1_{score} = 2\left(\frac{1}{\frac{1}{P} + \frac{1}{R}}\right) = 2\left(\frac{P * R}{P + R}\right)$$ Ecuación (4)

Donde:

$F1_{score}$ → F1 score (Puntuación F1)

P → Presicion (Precisión)

R → Recall (Sensibilidad)

1.3.8. Procesamiento de Lenguaje Natural (PLN)

El procesamiento del lenguaje natural (NLP, por sus siglas en inglés) se refiere a la rama de la informática, y más específicamente, la rama de la inteligencia artificial o IA, que se ocupa de brindar a las computadoras la capacidad de comprender textos y palabras habladas de la misma manera que los seres humanos [47].

1.3.8.1. Funcionamiento de un sistema PLN

El lenguaje humano está lleno de ambigüedades que hacen que sea increíblemente difícil escribir software que determine con precisión el significado previsto del texto o los datos de voz. Los homónimos, homófonos, sarcasmos, modismos, metáforas, excepciones gramaticales y de uso, variaciones en la estructura de las oraciones; estas son solo algunas de las irregularidades del lenguaje humano que los humanos tardan años en aprender, pero que los programadores deben enseñar a las aplicaciones impulsadas por el lenguaje natural para reconocer y entender con precisión desde el principio, si esas aplicaciones van a ser útiles [48].

1.3.8.2. Arquitectura de un sistema PLN

Los "niveles de lenguaje" son uno de los métodos más explicativos para representar el Procesamiento del Lenguaje Natural (PLN) que ayuda a generar el texto de PNL mediante la realización de las fases de planificación de contenido, planificación de oraciones y realización superficial como se muestra en la figura 10 [49].

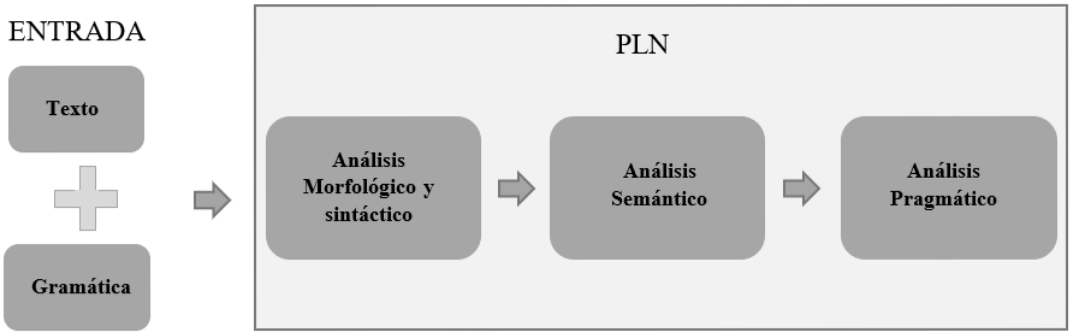


Figura 10: Arquitectura del Procesamiento del Lenguaje Natural (PLN) [49]

1.3.8.3. Niveles del lenguaje en un sistema PLN

Los niveles del lenguaje o de la lengua hace referencia al grupo Reciben el nombre de niveles de la Lengua (o del lenguaje) el conjunto de peculiaridades o la forma en que los seres humanos se comunican por medio del lenguaje. Esta denominación hace una relación entre los diversos modos de clasificación de la capacidad lingüística como se muestra en a la tabla 8 [50].

Tabla 8: Niveles del lenguaje para sistemas PLN

Nivel	Definición
Nivel Fonético	Es la parte de la Lingüística que se refiere a la disposición sistemática del sonido.
Nivel Morfológico	Se encarga de analizar las diferentes partes de la palabra que representan las unidades más pequeñas de significado conocidas como morfemas.
Nivel Sintáctico	Este nivel enfatiza en escudriñar las palabras en una oración para descubrir la estructura gramatical de la oración.
Nivel Semántico	En semántica la mayoría de la gente piensa que el significado está determinado, sin embargo, esto no es así, son todos los niveles los que le otorgan significado.
Nivel Pragmático	La pragmática se preocupa por el uso firme del lenguaje en situaciones y utiliza el nudo por encima del nudo del texto para comprender el objetivo y explicar cómo se lee el significado extra.

Elaborado por: El investigador basado en [50].

1.3.9. Asistentes virtuales (chatbots)

Los chatbots virtuales son asesores virtuales, asistentes personales de IA (Inteligencia Artificial) o agentes virtuales inteligentes que se comunican con empresas y marcas a través de aplicaciones de mensajería. El marketing de productos, la participación de la marca, la asistencia de productos, las ventas y las discusiones de soporte son usos comunes de los bots conversacionales. El asistente virtual impulsado por inteligencia artificial utiliza procesamiento del lenguaje natural y aprendizaje automático para extraer información y datos complejos de las conversaciones para comprenderlos y procesarlos en consecuencia [51].

Los nuevos asistentes de salud utilizan algoritmos avanzados de inteligencia artificial para aprender sobre el paciente a fin de personalizar las respuestas y brindar

información precisa en tiempo real. Los asistentes de salud permiten al paciente administrar todos los aspectos de sus citas en línea, incluidas la reserva, la reprogramación y la cancelación, gracias a la integración de la base de datos backend en tiempo real [52].

1.3.9.1. Beneficios de instalar un chatbot para el cuidado de la salud

AI Chatbot en la industria de la salud marca su presencia, ya que brinda asistencia médica rápida a los clientes en el momento de la emergencia. Además, los chatbots de atención médica pueden automatizar todas las tareas repetitivas y de nivel inferior que, de lo contrario, realizaría un representante y brindar asistencia a los pacientes las 24 horas del día, los 7 días de la semana. Chatbot permite a los pacientes no esperar en una cola durante horas para que un representante revise sus consultas. Entre los beneficios más relevantes se puede destacar los siguientes [53]:

- Mantiene a los pacientes actualizados.
- Gana la confianza del paciente.
- Mayor compromiso del paciente.
- Manejar consultas frecuentes.

1.3.9.2. Tecnologías para el desarrollo de chatbots

Las grandes empresas de software ven a las interfaces conversacionales como una evolución del uso del software a través de internet, ofreciendo soluciones que proporcionen una infraestructura que facilite el desarrollo de aplicaciones basada en IA. Para la creación de Chatbots existen varias herramientas, la gran mayoría son de pago y cuentan con licencia para distintos usos [54].

1.3.10. Aplicación Web

La tecnología de Servicios Web se basa en el concepto de computación orientada a servicios. Los servicios web son estándares que integran aplicaciones basadas en la web a través de la conexión y el intercambio de procesos comerciales a través de la red donde las aplicaciones de diferentes los proveedores, los idiomas y las plataformas se comunican entre sí y con los clientes [55]..

Las aplicaciones web se refieren a aplicaciones a las que se accede a través de un navegador web con acceso a la red y que se desarrollan utilizando lenguajes

compatibles con el navegador (por ejemplo, HTML, JavaScript, CSS). Para su ejecución, las aplicaciones web dependen de los navegadores web e incluyen muchas aplicaciones familiares, como ventas minoristas en línea, subastas en línea y correo web [55].

1.3.10.1. Aplicaciones web vs Aplicaciones nativas

Las diferencias entre aplicaciones web y nativas reside esencialmente en su funcionamiento como la accesibilidad, experiencia de usuario y el desarrollo detrás de cada aplicación [56], en la tabla 9 se muestra las principales ventajas y desventajas de cada una.

Tabla 9: Comparativa entre aplicaciones web y aplicaciones nativas

	Aplicaciones Web	Aplicaciones nativas
Ventajas	No es necesario descargarla ya viene preinstalada en los dispositivos móviles.	Compatible con las funciones integradas solo dependiendo del desarrollador de la app.
	El navegador completa las actualizaciones automáticas.	Más fácil de trabajar dependiendo del rendimiento del dispositivo.
	Mantenimiento más fácil ya que son compatibles en todos los sistemas operativos.	Disponible en una tienda de apps que tienen control de calidad.
Desventajas	Al usar diferentes navegadores es difícil rastrear patrones para brindar soporte.	Es más caro ya que su desarrollo presenta mayor complejidad dependiendo de la versión de sistema operativo del teléfono móvil.
	No hay compatibilidad con funciones integradas de ciertos dispositivos.	El proceso de aprobación en las tiendas tarda mucho y no garantiza la seguridad en su totalidad.
	No existe un sistema de control de calidad regularizado como en tiendas de app.	Es necesario descargar actualizaciones para que la app continue funcionando.

Elaborado por: El investigador basado en [56].

1.3.10.2. Funcionamiento de una aplicación web

El funcionamiento básico de una aplicación web se puede observar en la figura 11, consiste en que un usuario o también conocido como cliente web realiza una petición a un servidor web a través de internet por medio de una interfaz de usuario. Una vez llega esta petición el servidor web, este envía una solicitud al servidor de la aplicación web para ejecutar la acción requerida por el usuario manejando frecuentemente una base de datos. Finalmente, estos resultados son enviados de regreso al servidor web encargado de llevar la información al dispositivo que esté utilizando el usuario y mostrarlo por medio de una interfaz amigable con el usuario [57].

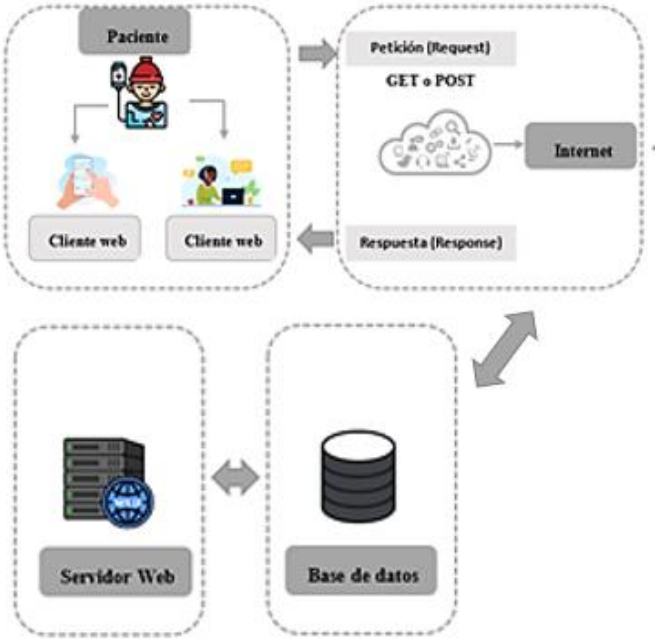


Figura 11: Funcionamiento básico de una aplicación web [57]

1.3.10.3. Arquitectura de una aplicación Web

La arquitectura básica de una aplicación Web se muestra en la figura 12 donde incluye navegadores, una red y un servidor Web. Los navegadores solicitan a las páginas Web desde el servidor. Cada página es una combinación de contenido e instrucciones de formato expresado como HTML. Algunas páginas incluyen secuencias de comandos laterales que son interpretadas por el navegador. Estos scripts definen un comportamiento dinámico adicional para la página de visualización y, a menudo, interactúan con el navegador, contenido de la página y controles adicionales de la página [58].

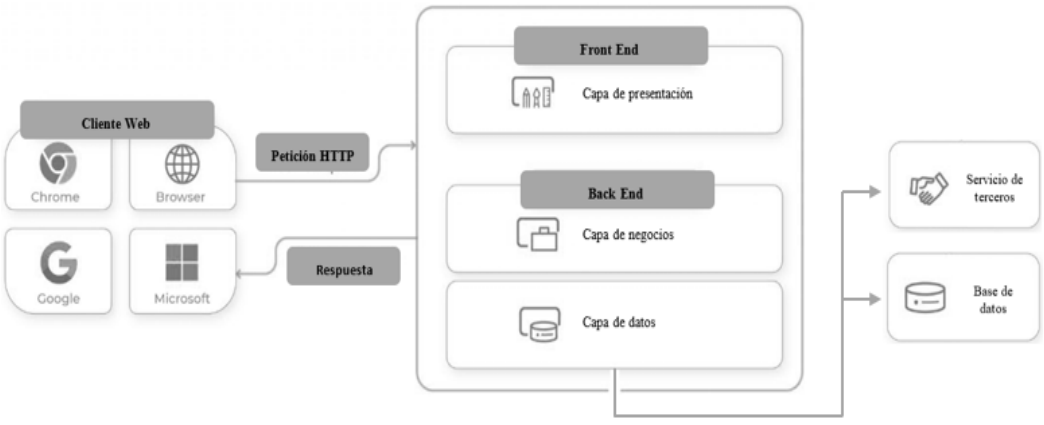


Figura 12: Arquitectura básica de una aplicación Web [58]

1.3.10.4.Frontend y Backend

En la actualidad se utiliza dos términos al momento de desplegar aplicaciones Web las cuales son Frontend y Backend. El frontend hace referencia a la interfaz visual con la que el usuario interactúa mientras que el backend se encuentra relacionado con el funcionamiento interno de la aplicación Web. Para un correcto funcionamiento se necesita que ambas partes se encuentren en armonía y cooperen entre si como un solo sistema dentro de la aplicación Web [58], los lenguajes de programación de Frontend y backend se pueden apreciar en la figura 13.

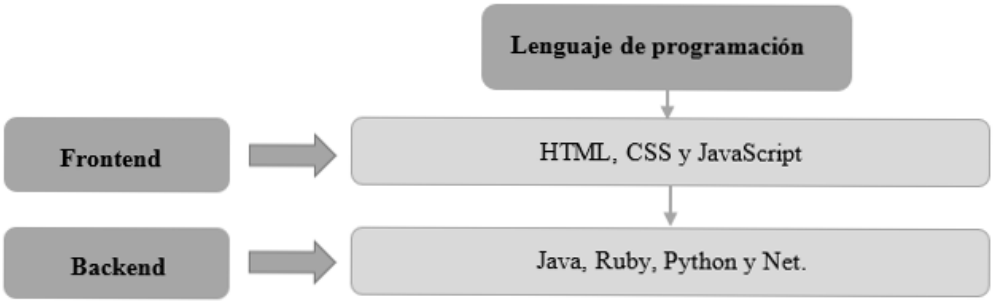


Figura 13: Lenguajes de programación de backend y frontend [58]

1.3.10.5.Frameworks para desarrollar aplicaciones Web

Los frameworks son las técnicas más utilizadas por los desarrolladores al momento de crear aplicaciones web, ya que permite el uso de módulos que facilitan la configuración de las funciones y bloques presentes en una página web. En la tabla 14 se puede ver una comparación entre los frameworks más utilizados actualmente [59]:

1.3.10.6.Base de datos

Una base de datos es una serie de información (datos) organizada y estructurada que actualmente se almacena en dispositivos electrónicos, a menudo se utiliza un sistema de administración para poder controlar los datos albergados en el mismo[60]. En la actualidad existen 2 tipos de base de datos como se muestra en la figura 14.

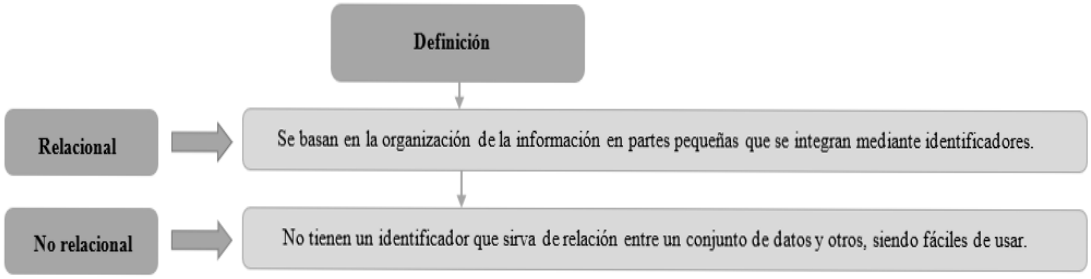


Figura 14: Tipos de base de datos [60]

1.3.10.7. Base de datos relacional

Una base de datos relacional se encarga de almacenar y brindar un acceso a datos que se encuentran relacionados entre ellos por medio de identificadores que representan los datos en tablas de forma sencilla donde cada fila es un registro con una identificación única llamada clave, mientras que las columnas contienen los atributos de los datos, y cada registro suele tener un valor para cada atributo facilitando la relación entre los datos [61].

1.3.10.8. Cloud Computing (Computación en la nube)

La computación en la nube significa almacenar y acceder a datos y programas a través de Internet en lugar del disco duro de la computadora local. Existen 3 tipos de servicios cloud computing como:

Software como servicio (SaaS), donde la empresa se suscribe a una aplicación a la que accede a través de Internet a manera de un alquiler de igual manera existe la plataforma como servicio (PaaS), donde una empresa puede crear sus propias aplicaciones personalizadas para que las usen todos en la empresa. Finalmente, la Infraestructura como servicio (IaaS), donde las empresas como Google, Microsoft y Rackspace proporcionan una columna vertebral al alquilar máquinas virtuales que dependiendo de las necesidades existe planes gratuitos durante un determinado tiempo [62].

1.4.Objetivos

1.4.1. Objetivo General

Implementar un asistente virtual de telemedicina con dispositivo electrónico de medición de triaje para el diagnóstico médico de infecciones respiratorias utilizando inteligencia artificial.

1.4.2. Objetivos Específicos

- Analizar el estado del arte de los parámetros de detección de pacientes con infecciones respiratorias.
- Diseñar un dispositivo electrónico que permita el triaje de los signos vitales.
- Desarrollar la aplicación para el diagnóstico médico de enfermedades respiratorias utilizando inteligencia artificial.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

Para llevar a cabo el presente proyecto, se utilizaron diferentes materiales electrónicos como: dos interruptores en donde el primero se encuentra destinado a prender y apagar el dispositivo, mientras que el segundo sensor se ocupa para encender el sensor MAX30100; 2 leds de 5mm de color verde y rojo que son ocupados como indicadores para saber si el dispositivo se encuentra o no conectado a una red con internet. Se emplea sensores como el MLX90614 utilizado para medir la temperatura corporal, en cambio para medir el pulso cardíaco se hizo uso del sensor MAX30100 que trabaja en base a un led rojo e infrarrojo para conocer el BPM (Latidos por Minuto) y el SpO2 (Saturación de oxígeno) del usuario a su vez este sensor se encuentra conectado a tres resistencias de 4.7$k\Omega$ pullup ya que el sensor trabaja con un nivel lógico de 3.3V, mientras que el microcontrolador trabaja con 3.7V. También se hace uso de una pantalla Oled de 0.96" con una resolución de 128X64 con cual el usuario puede observar sus signos vitales en tiempo real y la carga de batería del dispositivo. Todos estos materiales electrónicos se encuentran conectados a dos microcontroladores, el primer microcontrolador ESP8266 se encuentra conectado al sensor MAX30100 que envía las mediciones de BPM y SpO2 por comunicación serial a una segunda ESP8266 que se encarga de medir la temperatura, conectarse a internet para mostrar los datos medidos en la pantalla Oled y a la vez enviar estos a la aplicación Web por medio del protocolo Mosquito. Para la alimentación del circuito se usó una batería Lipo de 3.7V que está conectada a un TP4056 para poder cargarla, a su vez el TP4056 se encuentra conectado dos resistencias de 100 $k\Omega$ para medir la carga de la batería.

Para desarrollar la inteligencia artificial que realizó el diagnóstico médico de enfermedades como: covid19, rinitis alérgica, resfriado común; se empleó la librería de sklearn de Python. Además, para el desarrollo de la aplicación Web se utiliza una plantilla Bootstrap 5 en el Front-End. Mientras que para el Back-End se utilizó el framework Flask encargado de desplegar la aplicación Web y gestionar la base de datos en MySQL.

2.2. Métodos

2.2.1. Modalidad de la Investigación

A continuación, se describe los diferentes tipos de investigaciones que se aplicaron para la ejecución del proyecto.

El presente proyecto es un estudio aplicado, ya que se empleó los conocimientos adquiridos en el trascurso de la carrera, siendo el punto de partida para el diseño de un asistente virtual utilizando inteligencia artificial para diagnosticar infecciones respiratorias, el cual está dentro de los parámetros dados por el profesional de la salud.

Se realizó una investigación bibliográfica con el propósito de obtener información de métodos de recolección y monitoreo de signos vitales en pacientes con infecciones respiratorias. Con este fin se recolectó fuentes documentadas como revistas, artículos o trabajos de investigación dentro de repositorios universitarios.

Investigación de campo, porque se realizaron pruebas de funcionamiento en el consultorio particular de la dra. Verónica Córdova que colaboro con él proyecto de investigación para verificar el correcto diagnóstico de las enfermedades respiratorias tratadas por el asistente virtual.

Investigación experimental debido a que, para determinar la veracidad del diagnóstico médico, fue necesario realizar pruebas de funcionamiento en colaboración con el profesional en el área de salud.

2.2.2. Recolección de la información

Para la recolección de información se usó libros, revistas, fuentes online y proyectos desarrollados; así como, guías prácticas y manuales de construcción; por lo tanto, se tomó en cuenta bases de datos técnicas debidamente fundamentadas. Así mismo, se obtuvo información de profesionales especialistas en el área para identificar aspectos relevantes sobre el tema que permitieron el desarrollo del proyecto.

2.2.3. Procesamiento y Análisis de datos

El procesamiento y análisis de datos se realizó teniendo en cuenta los siguientes

pasos:

- Análisis de la información recopilada.
- Estudio de las propuestas de solución planteadas para diagnosticar a pacientes con infecciones respiratorias.
- Planteamiento de la propuesta de solución.
- Comprobación de la funcionalidad del asistente virtual, permitiendo un diagnóstico médico validado por un profesional en el área de salud.

2.2.4. Propuesta de solución

El desarrollo de un asistente virtual de telemedicina con inteligencia artificial que cuenta con un dispositivo electrónico para el triaje de los signos vitales en tiempo real como la temperatura, frecuencia cardiaca y saturación de oxígeno, mismos que son presentados en una interfaz gráfica y en una pantalla Oled en el dispositivo permitió diagnosticar de manera oportuna a los pacientes con infecciones respiratorias, logrando una optimización recursos del personal médico con respecto a los métodos de atención tradicionales. En consecuencia, el asistente virtual y el dispositivo electrónico fueron verificados y validados por la doctora que colaboró con el proyecto de investigación. Adicionalmente, el usuario puede visualizar el historial médico de cada consulta realizada al asistente virtual en la misma aplicación Web.

2.2.5. Desarrollo del Proyecto

El proceso para la implementación de un asistente virtual para el diagnóstico médico utilizando inteligencia artificial consta de las siguientes actividades:

1) Análisis del estado de las enfermedades respiratorias a nivel mundial.
2) Búsqueda del vínculo de la situación actual de las infecciones respiratorias a con la situación local.
3) Reconocimiento de los problemas que muestran los pacientes con infecciones respiratorias.
4) Análisis de los signos vitales de los pacientes con infecciones respiratorias agudas.
5) Selección de métodos para medir la saturación de oxígeno, temperatura, presión arterial y frecuencia cardiaca.
6) Selección del hardware y software necesario para el desarrollo del proyecto.

7) Diseño y construcción del dispositivo electrónico para la adquisición y procesamiento de los signos vitales.
8) Programación de los componentes necesarios para el algoritmo de inteligencia artificial empleando el software que cumpla con las necesidades del proyecto.
9) Diseño de la interfaz gráfica para que el paciente y medico pueda visualizar el historial médico generado por el asistente virtual tras la consulta.
10) Pruebas de funcionamiento del asistente virtual de telemedicina para el diagnóstico médico basado en inteligencia artificial.
11) Análisis de los resultados obtenidos de las pruebas de funcionamiento del asistente.
12) Elaboración del informe final.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de los resultados

La implementación y desarrollo del asistente médico conjunto con su dispositivo de medición de signos vitales en tiempo real instauran una evolución en las herramientas tecnológicas utilizadas en el área de la salud frente a las infecciones respiratorias que amenazan el estado físico de una persona. Este proyecto mejora la atención de los pacientes con covid19, resfriado común y rinitis alérgica; mediante el uso de inteligencia artificial (IA) ya que esta aplicación Web es de fácil acceso y compatible con cualquier dispositivo que tenga conexión a internet. Una vez, se diagnostique al paciente este debe acercarse al centro de salud más cercano para ser recetado por un profesional de la salud. Además, la aplicación web ofrece una interacción más amigable con el usuario por medio del chatbot con lenguaje Procesamiento del lenguaje Natural (NPL) que simula la interacción con un doctor.

3.2. Desarrollo de la propuesta

El proyecto de investigación consta de una metodología teórica-práctica, ya que se utilizó bases de datos debidamente verificadas de Universidades de Ecuador, así como Universidades del extranjero con temas relacionados con el proyecto de investigación, con el fin fundamentar el desarrollo. Asimismo, fue necesario hacer consultas con profesionales de la salud para compilar información útil que ayudo a conocer los procedimientos para efectuar un diagnóstico y que se encuentre dentro de las directrices estipuladas por el profesional de la salud.

3.2.1. Consideraciones para el desarrollo del proyecto

Al trabajar con una aplicación web y un sistema electrónico que conjuntamente ofrecen la medición de los signos vitales y el diagnóstico médico de enfermedades como COVID19, Resfriado común, Rinitis alérgica. Era necesario buscar con que software se desarrollaría la aplicación Web, mientras que para el sistema electrónico fue importante determinar con que componentes se va a ensamblar el mismo. Además, antes de la fabricación del prototipo fue fundamental establecer criterios de diseño como:

- Usar componentes electrónicos con dimensiones reducidas con el objetivo de crear un dispositivo de medición de signos vitales portable.
- La apariencia de la interfaz de la aplicación Web debe ser llamativa y amigable con el usuario, para que sea fácil de usar.
- La estructura del dispositivo debe ser resistente ya que el mismo al ser portable puede sufrir caídas.
- El dispositivo debe ser transparente para ser manejado por los pacientes.

3.2.2. Selección de hardware para medir los signos vitales

Para la selección de los componentes electrónicos se dividió los mismos en 4 etapas de acuerdo con la arquitectura básica de un sistema electrónico como muestra la figura 15. Para ello se realizó una comparativa entre distintos componentes electrónicos disponibles en el mercado ecuatoriano.

Figura 15: Arquitectura de un sistema electrónico

3.2.2.1. Capa de sensorización

La capa de sensorización se encargan de la adquisición de los datos medidos de las variables físicas para transformarlas en señales analógicas y enviarlas al microcontrolador para ser procesados [63].

Sensor de temperatura

El análisis de las características expuestas en la tabla 10 de cada sensor de temperatura existente en el mercado ecuatoriano, se decidió elegir el MLX90614 principalmente por sus dimensiones reducidas, consumo mínimo de corriente lo cual lo hace ideal para dispositivos portables. Finalmente, su relación precio calidad lo hace la opción más rentable ya que tiene un bajo costo y una precisión de ±0.5% lo cual lo hace accesible.

Tabla 10: Comparación entre sensores de temperatura

Características	Sensores de Temperatura		
	MLX90614	**MLX90640**	**LM35**
Rango de operación	-40°C – 170°C	-4.4°C – 85°C	-55°C – 150°C
Rango de medición	-70°C - 380°C	-70°C - 300°C	-10°C - 85°C
Voltaje de operación (CD)	3.3V – 5V	3.3V – 5V	4V – 30V
CLK	1MHz	1MHz	No usa
Consumo de corriente	2mA	18mA	60 μA
Precisión	±0.5%	±0.2%	±0.93%
Dimensión	16 x 11 x 6 mm	27 x 16 x 9 mm	4 x 3 x 3 mm
Peso	2.8 gr	5 gr	0.24 gr
I2C	si	si	no

Elaborado por: El investigador basado en [34].

Sensor de oxímetro y pulso

En base al análisis de las características expuestas en la tabla 11 de cada sensor de BPM y SpO2 existente en el mercado ecuatoriano, se decidió elegir el MAX30100 fundamentalmente por sus dimensiones reducidas, consumo mínimo de corriente lo cual lo hace ideal para dispositivos portables. Finalmente, su relación precio calidad lo hace la opción más rentable ya que tiene un bajo costo en comparación al MAX30102; además, existe más documentación de como filtrar los datos sin procesar del sensor MAX3010 mediante la programación en Arduino. Cabe mencionar que este tipo de sensores son muy delicados por lo que es importante revisar la polarización de estos y que el voltaje de alimentación sea el recomendado por el fabricante caso contrario se pueden llegar a sobrecalentar disminuyendo su vida útil.

Tabla 11: Comparación entre sensores de BPM y SpO2

Características	Sensores de BPM y SpO2	
	MAX30100	**MAX30102**
Voltaje de operación	1.8V – 3.3V	1.8V – 3.3V
Corriente de consumo	1.2mA	1.2mA
Temperatura de operación	-40℃ - 85℃	-40℃ - 85℃
Longitud de onda de Led IR	870 - 900 nm	870 - 900 nm
Longitud de onda de Led Rojo	650 - 670 nm	650 - 670 nm
I2C	si	si
CLK	4MHz	4MHz
Dimensiones	11 x 17 x 1.2 mm	20.6 x 15.5 x 1.2 mm

Elaborado por: El investigador basado en [34].

3.2.2.2. Capa de procesamiento

En la capa de procesamiento los datos son recopilados de la capa de sensores para convertirlos en información significativa que en la mayoría de los casos son almacenados en la nube con un acceso a los mismos mediante un Back-End. En esta etapa se utiliza un microcontrolador que mediante un IDE de programación permite procesar los datos para que puedan ser interpretados por los usuarios y visualizados en una interfaz Web o una pantalla conectada al sistema electrónico [63].

Mediante un análisis de las características expuestas en la tabla 12 de cada microcontrolador existente en el mercado ecuatoriano, se decidió elegir el ESP8266 NodeMCU principalmente por sus dimensiones reducidas, consumo mínimo lo cual lo hace ideal para el uso de baterías pequeñas. Finalmente, la disponibilidad de protocolos WIFI actuales y la comunicación I2C utilizada para conectar los sensores, convierte a este microcontrolador en la opción ideal para el presente proyecto.

Tabla 12: Comparativa de microcontroladores

Características	Microcontroladores		
	Arduino nano	ESP8266 NodeMCU	ESP32 NodeMCU
Microcontrolador	ATmega328p	ESP8266	ESP32
Tensión de funcionamiento	5V	3.7V	3.3V
Alimentación	7V – 12V	3.7V – 12V	7V – 12V
Consumo de corriente	$19mA$ - $180mA$	$15\mu A$ - $400\mu A$	$20mA$ - $240mA$
Consumo en reposo	$23\mu A$	$0.5\mu A$	$5\mu A$
Pines de E/S digitales	14	11 - 13	34
Pines analógicos	8	1	15
I2C	si	si	si
Clock	$16MHz$	$52MHz$	$80MHz$ - $160MHz$
Protocolo WIFI	No tiene WIFI	802.11 b/g/n a 2.4 GHz	802.11 b/ g/ n/ d/ e/ i/ k/ r a 2.4 GHz
Dimensiones	45 x 18 mm	48 x 26 mm	52 x 31 mm
Puerto USB tipo B	si	si	si

Elaborado por: El investigador basado en [31].

3.2.2.3. Capa de visualización

La capa de visualización se encarga de mostrar las mediciones de los sensores al usuario una vez que los datos fueron procesados por el microcontrolador ESP8266 NodeMCU mediante el IDE Arduino, esta información puede ser presentada por medio de indicadores como los son los diodos led y pantallas de visualización como una oled o LCD.

Indicadores

Tras el análisis de la tabla 13 se determinó el uso de dos diodos leds de baja intensidad con el color rojo y verde por su bajo consumo y dimensiones reducidas. Estos se utilizan a manera de indicadores luminosos para que el usuario pueda ver si el dispositivo se conectó a la red o caso contrario no se conectó ya sea porque se encuentra en una red nueva o existe pérdida de la señal WIFI en el lugar que se encuentre el usuario. Además, si el led rojo sigue prendido constantemente después de 10 segundos significa que no logro conectarse con la red y el ESP8266 se configuro

automáticamente como un AP para poder configurar la red y establecer una conexión con la red deseada.

Tabla 13: Comparación de diodos led

Parámetros	Diodo led de baja intensidad	Diodo led de alta intensidad
Perímetro	5mm	10mm
Ángulo de visión	30°	60°
Consumo de corriente	3mA	20mA
Voltaje de operación	Rojo: 1.9V Verde: 2.4V	Rojo: 1.9V Verde: 2.4V
Luminosidad	2 lúmenes	4 lúmenes

Elaborado por: El investigador basado en [31].

Visualización

Mediante un análisis de las características expuestas en la tabla 14 se decidió escoger una pantalla Oled de 0.96 pulgadas ya que sus dimensiones reducidas las hace ideales para prototipos portables; además, de contar con una mayor resolución que permite la inclusión de imágenes que hacen más entendible los valores mostrados en la pantalla, estas son utilizadas para que el usuario visualice el porcentaje de carga de la batería y sus signos vitales como la temperatura, BPM y SpO2 durante un lapso 2 segundos.

Tabla 14: Comparación de pantallas

Parámetros	Pantalla Oled	Pantalla LCD
Dimensiones	0.96”	4”
Ángulo de visión	160°	160°
Consumo de corriente	30mA	25mA
Voltaje de operación	3.3V – 5V	5V
Driver	SSD1306	LCD 1602
Resolución	128 x 64	16 x 2
I2C	si	si

Elaborado por: El investigador basado en [31].

3.2.2.4. Suministro de energía

En base al análisis de las características expuestas en la tabla 15 se escogió una batería Lipo Nanotech de una celda a 3.7V ya que esta tiene dimensiones reducidas, un menor peso y una capacidad de carga de 600mA suficientes para suministrar energía al sistema electrónico.

Tabla 15: Comparación de baterías

Parámetros	Batería Lipo Nanotech	Batería Lipo Megatronica
Capacidad	600mA	350mA
Voltaje	3.7V	3.7V
Dimensiones	40 x 26 x 10 mm	50 x 30 x 12 mm
Peso	17 gr	21 gr
Celdas	1	1

Elaborado por: El investigador basado en [31].

El análisis de las características expuestas en la tabla 16 de cada módulo de carga permite escoger al TP4056 tipo C por constar con dimensiones reducidas, una protección sobre corriente lo cual permite el uso de carga rápida ya que consta con una protección en contra de la sobre carga, este módulo tolera que el usuario pueda cargar el dispositivo mientras este usándolo o simplemente cargarlo en reposo.

Tabla 16: Comparación de módulos de carga

Parámetros	TP4056 tipo C	TP4056 tipo B
Voltaje de alimentación	4.5V – 5.5V	4.5V – 5.5V
Corriente de carga	0.6 - 1A	0.6 - 1A
Protección sobre corriente	3A	no
Dimensiones	2.6 x 1.7 cm	2.5 x 1.9 cm
Micro-USB	Tipo C	Tipo B
Protección sobrecarga	si	no

Elaborado por: El investigador basado en [31].

3.2.3. Selección del Software para el diagnóstico médico

Para la selección del software se consideró el lenguaje Python ya que en la actualidad es el lenguaje de vanguardia más versátil en el mundo; además, que este proporciona una licencia de código abierto, ya sea de uso personal o comercial. También se escogió

el desarrollo de una aplicación Web ya que las ventajas frente a una aplicación nativa son considerablemente mejores como se visualiza en la tabla 17.

Tabla 17: Comparativa entre aplicaciones web y aplicaciones nativas

	Aplicaciones Web	Aplicaciones nativas
Ventajas	No es necesario descargarla ya viene preinstalada en los dispositivos móviles.	Compatible con las funciones integradas solo dependiendo del desarrollador de la app.
	El navegador completa las actualizaciones automáticas.	Más fácil de trabajar dependiendo del rendimiento del dispositivo.
	Mantenimiento más fácil ya que son compatibles en todos los sistemas operativos.	Disponible en una tienda de apps que tienen control de calidad.
Desventajas	Al usar diferentes navegadores es difícil rastrear patrones para brindar soporte.	Es más caro ya que su desarrollo presenta mayor complejidad dependiendo de la versión de sistema operativo del teléfono móvil.
	No hay compatibilidad con funciones integradas de ciertos dispositivos.	El proceso de aprobación en las tiendas tarda mucho y no garantiza la seguridad en su totalidad.
	No existe un sistema de control de calidad regularizado como en tiendas de app.	Es necesario descargar actualizaciones para que la app continue funcionando.

Elaborado por: El investigador basado en [56].

3.2.3.1. Asistente virtual (Chatbot)

En el caso del desarrollo del asistente virtual se escogió la herramienta chatbot principalmente por usar lenguaje de programación Python. Además, al tener una gran comunidad activa a nivel mundial esta herramienta cuenta con documentación para la implementación de lenguaje Procesamiento del Lenguaje Natural (PLN) lo cual lo hace una opción ideal al tratar de simular la conversación con un humano. Finalmente, entre las mejores características de esta tecnología mostrada en la tabla 18 se encuentra su licencia la cual es gratuita y es compatible con aplicaciones web desarrollas en frameworks de Back-End.

Tabla 18: Comparación de las tecnologías para crear Chatbots

Tecnologías	Características				
	Lenguaje de programación	**PLN**	**Soporte lenguaje español**	**Licencia**	**Integración**
Watson Assistant	Lenguaje propio	si	si	Versión gratuita y pagada	Aplicaciones web, servicios de mensajes
Dialogflow	Lenguaje propio	si	si	Versión gratuita y pagada	Aplicaciones web, servicios de mensajes
Amazon Lex	Lenguaje propio	si	si	Versión gratuita y pagada	Aplicaciones web, servicios de mensajes
Microsoft Bot Framework	C#, Javascript, Typescript, Python	no	si	Versión gratuita y pagada	Azure
Chatfuel	Lenguaje propio	si	si	Versión gratuita y pagada	Messenger
Chatbot	Python	si	si	Gratis	Aplicaciones web, servicios de mensajes

Elaborado por: El investigador basado en [54].

3.2.3.2. Framework de desarrollo de Back-End

Para el desarrollo de la aplicación Web se utilizó el framework Flask tras el análisis de las características de este en la tabla 19, donde se destaca que tiene una alta lista de BD (Base Datos) compatibles. Sin embargo, un parámetro importante al momento de su selección fue el lenguaje de programación ya que se busca tener una compatibilidad con todo lo que se desarrolló por ende el lenguaje Python la hace ideal para este proyecto. Por último, la característica que destaca a este framework sobre Django es su nivel de dificultad ya que Flask tiende a ser más intuitivo y fácil de utilizar al momento de desarrollar aplicaciones Web dinámicas.

Tabla 19: Comparativa de Frameworks de desarrollo de aplicaciones móviles

Framework	Características					
	Dificultad	**Base de datos**	**Lenguaje**	**Seguridad**	**Bootstrap**	**Arquitectura**
Django	Alta	SQlite, MySQL, PostgreSQL y Oracle	Python	Alta	Si	MVC (Model View Controller)
CherryPy	Alta	MySQL, PostgreSQL SQLite, Microsoft SQL Server y Firebird	Python	Media	Si	MVC
Flask	Media	PostgreSQL, MySQL, Oracle, MongoDB, CouchDB, y Cassandra	Python	Alta	Si	MVC
Spring	Alta	MySQL	Java	Media	Si	MVC
Google Web Toolkit (GWT)	Media	Google Cloud Storage, SQL	Java/ Python	Alta	No	MVC

Elaborado por: El investigador basado en [59].

3.2.3.3. Base de datos

Se consideró el uso de una base de datos relacional ya que entre sus principales ventajas se encuentra su seguridad, el tener un rendimiento rápido. Otro beneficio ofrecido es su accesibilidad en comparación a las bases de datos no relacionales que necesitan predefinir rutas para su acceso, mientras que la base de datos relacional no necesita de una ruta predefinida. Por tal motivo se realizó un análisis técnico de la tabla 20, donde se destaca la base de datos MySQL por tener una licencia gratuita, ser compatible con cualquier sistema operativo y poseer una interfaz gráfica para su administración. Adicionalmente esta base de datos fue seleccionada en base a la experiencia del investigador trabajando con MySQL durante la carrera.

Tabla 20: Comparación de base de datos relacionales

DataBase (DB)	Características					
	Licencia	**Documentación**	**Sistema Operativo**	**Interfaz Gráfica**	**Tamaño Máximo**	**Lenguaje**
MySQL	Gratuita	Si	Windows, Linux, MAC, Android, iOS	Si	Depende del dispositivo	SQL
PostgeSQL	Pagada	Si	Windows, Linux, MAC, Android	Si	Ilimitado	SQL
SQLite	Gratuita	Si	Windows, Linux, MAC, Android, iOS	No	1 TB	SQL
Microsoft SQL	Pagado	Si	Windows, Linux	Si	524,272 TB	SQL

Elaborado por: El investigador basado en [61].

3.2.3.4. Cloud computing

La selección de cloud computing se llevó a cabo en base al análisis de las características expuestas en la tabla 21, donde se observan mejores prestaciones por parte de Google Cloud y Microsoft Azure. Sin embargo, Google Cloud destaca ante Azure por tener una licencia gratuita de $ 300 por tres meses donde ofrece los servicios necesarios para el presente proyecto como lo son Cloud DNS (Dominio DNS), Instance VM (Máquinas virtuales), base de datos MySQL. Finalmente, la seguridad de este servicio IaaS es alta lo cual ayudo a la configuración de los puertos utilizados de la máquina virtual Ubuntu Server.

Tabla 21: Comparativa de servicios de almacenamiento en la nube

Servicio	Características					
	Licencia	**Base de datos**	**Seguridad**	**Lenguaje**	**Memoria**	**Dominio**
Python Anywhere	Gratuita/ Pagada	MySQL	Alta	Python	512 MB	Si
Google Cloud	Gratuita/ Pagada	MySQL	Alta	Linux/ Python	Ilimitado	Si
Microsoft Azure	Pagada	Azure SQL	Alta	Linux/ Python	Ilimitado	Si

Elaborado por: El investigador basado en [62].

3.2.4. Consideraciones de funcionamiento antes de implementar el dispositivo

Se implementó un sistema electrónico que sirva para medir los signos vitales en tiempo real como temperatura corporal, frecuencia cardíaca, saturación de oxígeno y puedan ser mostrados mediante una pantalla Oled SSD1306 y a la vez estos datos sean enviados a la aplicación Web mediante el protocolo MQTT para que al mismo tiempo estos datos sean mostrados en una interfaz gráfica alojada en el servidor Google Cloud. Además, se integró una medición de carga de la batería para que el usuario sepa en qué momento debe cargar el dispositivo. Por último, se colocó dos diodos led que sirven como indicadores, mostrando si el dispositivo se conectó o no a la red. En vista de ello, se decidió utilizar la siguiente arquitectura mostrada en la figura 16, misma que está formada por 3 capas: sensorización, procesamiento y visualización.

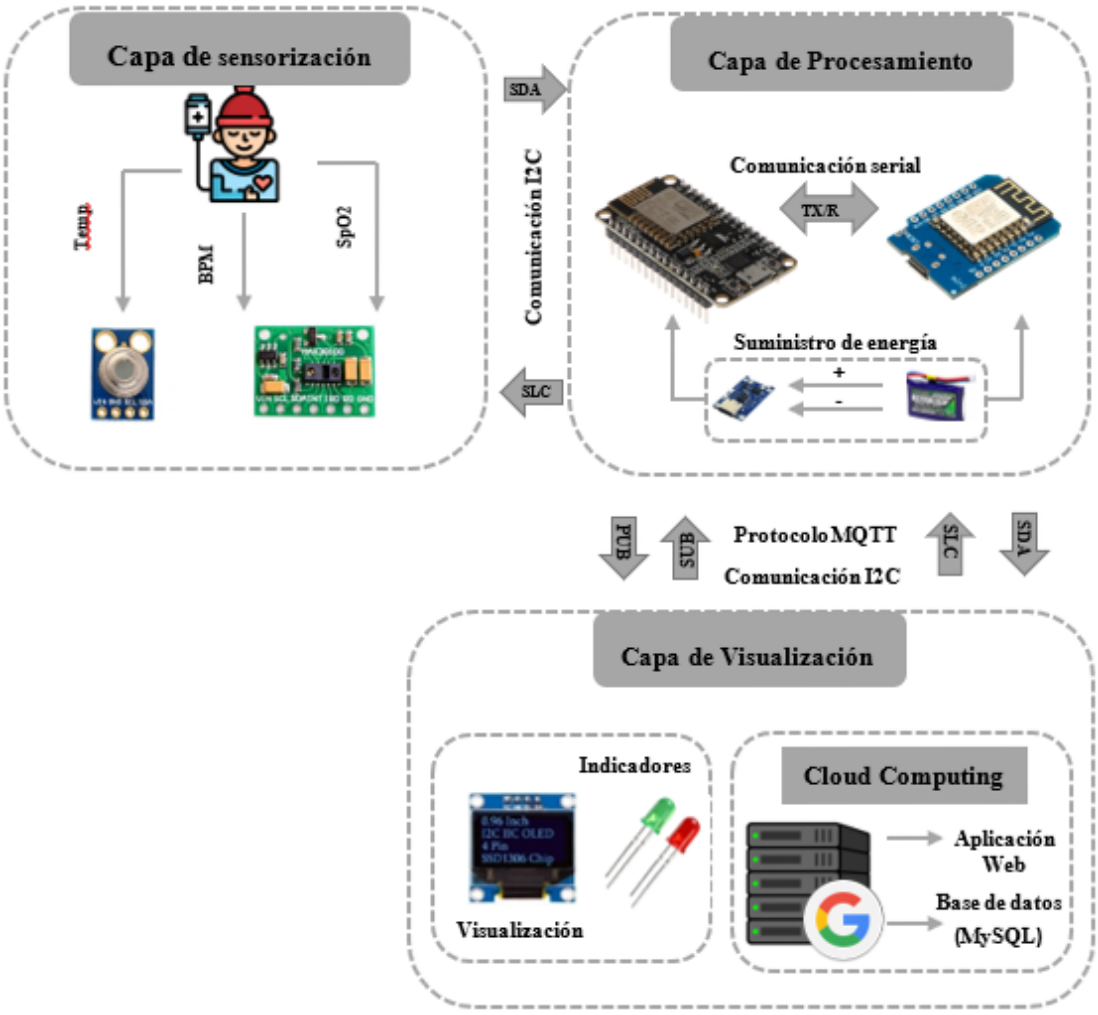


Figura 16: Arquitectura del sistema electrónico de medición de signos vitales

Elaborado por: El investigador

3.2.4.1. Funcionamiento del sensor MAX30100

El MAX30100 funciona iluminando con ambas luces (Luz roja, luz infrarroja) el dedo o cualquier otro lugar donde la piel no sea demasiado gruesa como muestra la figura 17, de modo que permita el penetrar el tejido fácilmente por ambas luces. Para posteriormente medir la cantidad de luz reflejada con un fotodetector, este método de detección de pulsos a través de la luz se denomina fotopletismografía [64].

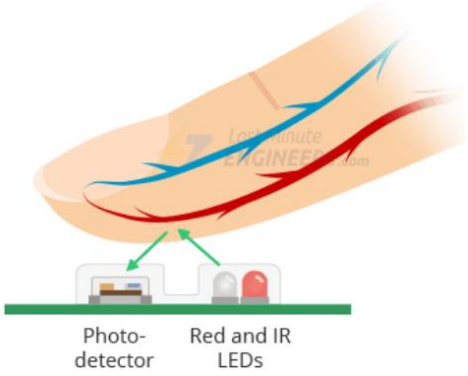


Figura 17: Funcionamiento del MAX30100 [64]

Medición de la frecuencia cardíaca

La hemoglobina oxigenada (HbO2) en la sangre arterial tiene la característica de absorber la luz IR. Cuanto más roja es la sangre (mayor concentración de hemoglobina), más luz IR se absorbe. A medida que la sangre se bombea a través del dedo con cada latido del corazón, la cantidad de luz reflejada cambia, creando una forma de onda cambiante en la salida del fotodetector como muestra la figura 18 [64].

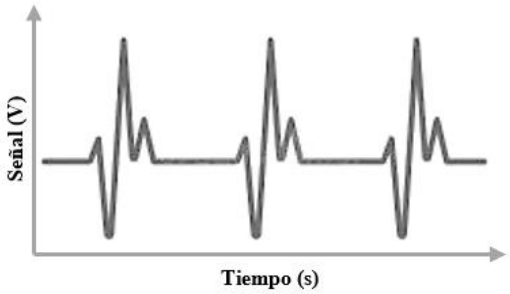


Figura 18: Salida del fotodetector [64]

Medición de la saturación de oxígeno en la sangre (Oximetría)

La oximetría de pulso se basa en el principio de la cantidad de luz roja e infrarroja absorbida varía según la cantidad de oxígeno en la sangre, como se muestra en la figura

19, donde el espectro de absorción de la hemoglobina oxigenada (HbO2) y la hemoglobina desoxigenada (Hb) [64].

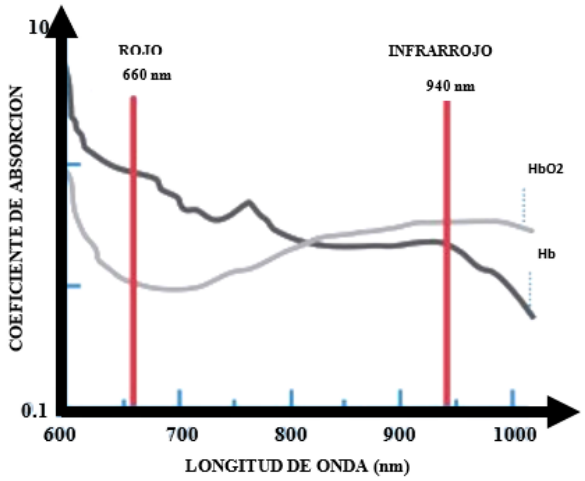


Figura 19: Espectro de absorción de la hemoglobina oxigenada y la hemoglobina desoxigenada [64]

Como se puede ver en la figura 19, la sangre desoxigenada absorbe más luz roja (660 nm), mientras que la sangre oxigenada absorbe más luz infrarroja (880 nm). Al medir la proporción de luz IR y ROJA recibida por el fotodetector, se calcula el nivel de oxígeno (SpO2) en la sangre [64].

Acondicionamiento de la señal del MAX30100

Para la lectura del sensor MAX30100 se debe importar la librería “MAX30100.h” desde el gestor de bibliotecas del IDE de Arduino. Internamente esta librería utiliza librerías como: “MAX30100 BeatDetector", “MAX3010 Filters”, “MAX30100 SpO2 Calculator”. Mediante estas librerías se puede leer los datos obtenidos del fotodiodo como se muestra la figura 20, en donde existe una ligera oscilación con un desplazamiento de 50000 unidades esto se produce por la presencia de un componente de corriente continua que causara errores en la medición de la frecuencia cardíaca y saturación de oxígeno. Por este motivo es necesario eliminar la componente CD empleando la ecuación (5) a fin de trabajar solo con la parte de corriente alterna (CA), para ello se emplea un filtro digital de primer orden (IIR) que utiliza la ecuación (6) considerando que mientras mayor sea el orden del filtro mayor será las exigencias de procesamiento causando un funcionamiento lento del microcontrolador o en el peor de los casos quemando la tarjeta del ESP8266. Las ecuaciones usadas para el filtro IIR son [17]:

$$w(t) = x(t) + \alpha * w(t-1) \quad \text{Ecuación (5)}$$

$$y(t) = w(t) - w(t-1) \quad \text{Ecuación (6)}$$

Donde:

y(t)→ Salida del filtro IIR

x(t)→ Entrada del filtro (Valor actual)

w(t)→ Historial del valor DC (Valor Intermedio)

α→ Constante de respuesta del filtro (0.95 recomendado por el fabricante)

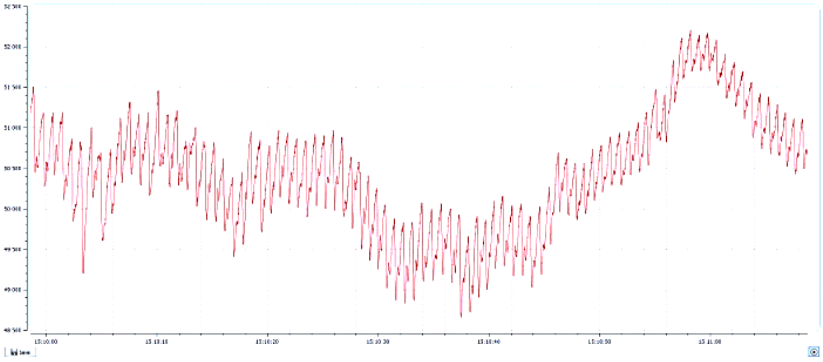


Figura 20: Datos en crudo del sensor MAX30100 con componente CD y CA [17]

Una vez aplicado el filtrado se obtiene una señal como muestra la figura 21, donde aún se observa presencia de oscilaciones. Por ello se emplea un filtro pasa bajos Butterworth de orden uno con una tasa de muestreo de 100Hz y una frecuencia de corte de 6Hz aplicados en la librería "MAX30100 Filters" [17].

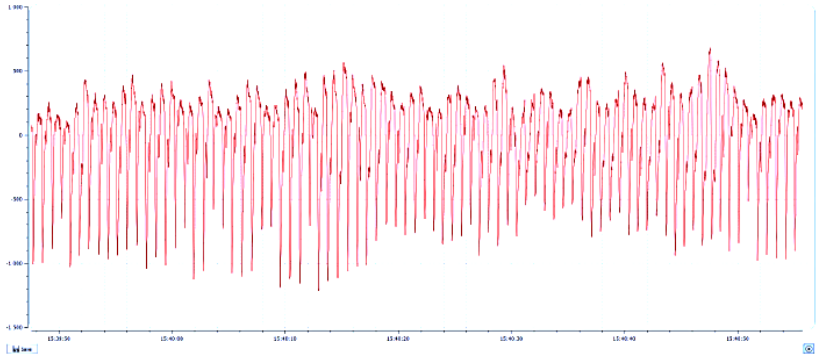


Figura 21: Datos en crudo del sensor MAX30100 con componente CD eliminado [17]

Finalmente, tras el filtrado de la señal con el filtro Butterworth se observa como la señal de figura 22 es mucho más limpia sin tanto ruido, esta señal será la que se utilice para la medición del BPM y SpO2 [17].

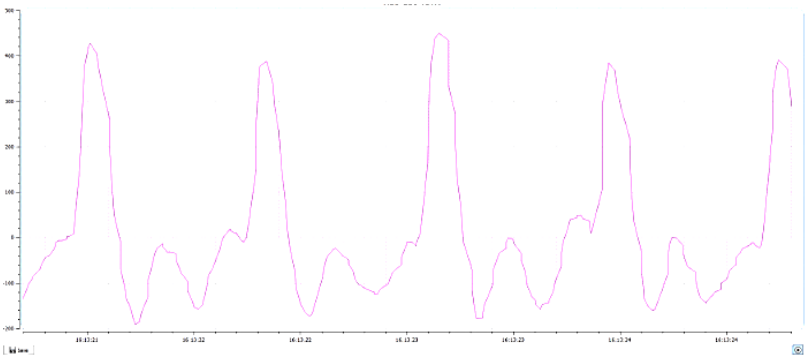


Figura 22: Señal filtrada con Butterworth de orden uno [17]

Para el cálculo de BPM se emplea la función "milis()" propia de Arduino que arroja el tiempo de funcionamiento del microcontrolador, para obtener dos marcas de tiempo con las que se puede calcular la frecuencia cardiaca haciendo usó de la ecuación (7). Mientras que para el cálculo de SpO2 se obtiene de la relación entre la luz absorbida del diodo led rojo y el led infrarrojo. La ecuación (8) es usada para obtener el cociente R que se usa para el cálculo del SpO2 en base a la relación de R con SpO2 como se muestra en la figura 23 [17].

$$\boldsymbol{BPM = \frac{6000}{T_{actual} - T_{anterior}}} \qquad \text{Ecuación (7)}$$

$$\boldsymbol{R = (100\%)\frac{log_{10}(R_{rojo})}{log_{10}(R_{infrarrojo})}} \qquad \text{Ecuación (8)}$$

Donde:

BPM→ Latidos por minuto

$\boldsymbol{T_{actual}}$→ Marca de tiempo actual (ms)

$\boldsymbol{T_{anterior}}$→ Marca de tiempo anterior (ms)

R→ Cociente de luz absorbida

$\boldsymbol{R_{rojo}}$→ Cociente de luz roja absorbida

$\boldsymbol{R_{infrarrojo}}$→ Cociente de luz infrarroja absorbida

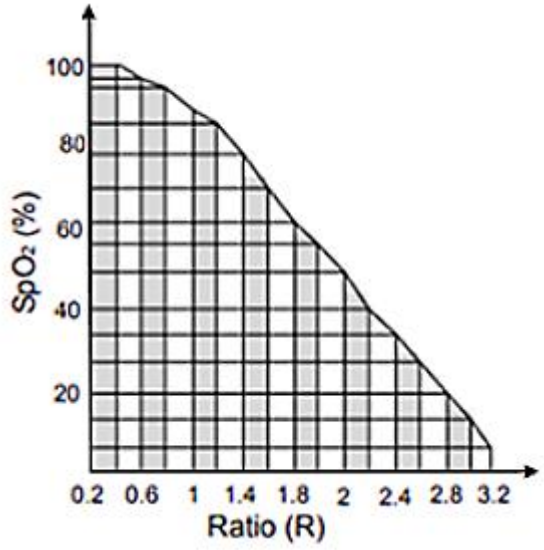


Figura 23: Relación entre SpO2 y el cociente R [17]

Acondicionamiento del circuito MAX30100

Se realizó una readecuación del circuito del MAX30100 debido a que este sensor trabaja con un nivel lógico de 1.8V en los pines SLC, SDA y INT. Por esta razón, al conectar el sensor al microcontrolador ESP8266 no aparecerá en el bus I2C debido a que el nivel lógico que está utilizando el sensor es demasiado bajo. Una vez fue detectado el problema es necesario retirar las resistencias de 4.7kΩ conectadas a 1.8V como muestra la figura 24.

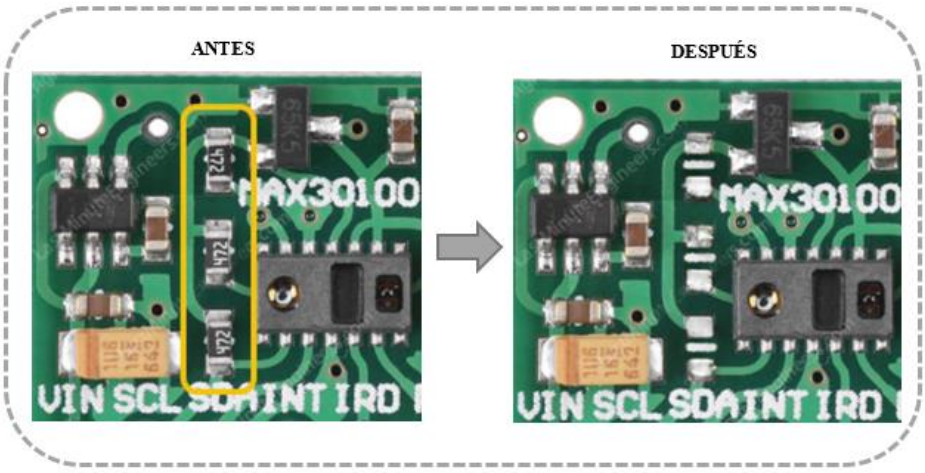


Figura 24: Acondicionamiento del sensor MAX30100

Elaborado por: El investigador

Diagrama de conexiones del MAX30100

Para el diagrama de conexiones se consideró la modificación hecha al MAX30100 ya que como se mencionó esto es necesario para que el bus I2C reconozca al sensor cuando se realice la comunicación I2C. Como se observa en la figura 25 el pin de reloj I2C (SCL) es conectado al pin D1, el pin de datos I2C (SDA) es conectado al pin D2 y finalmente el pin de interrupción (INT) con el que se puede programar una interrupción en cada pulso es conectado al pin D3 del microcontrolador. También se observa las resistencias de 4.7KΩ en modo pull-up para cambiar el nivel lógico con el que trabaja el sensor a uno de 3.7V.

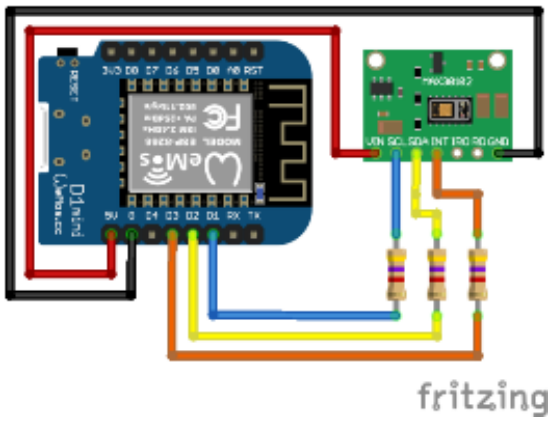


Figura 25: Diagrama de conexión del sensor MAX30100

Elaborado por: El investigador

Diagrama de flujo del MAX30100

Como se observa en la figura 26, el funcionamiento del sensor MAX30100 en conjunto con el microcontrolador ESP8266 consta de la inicialización de variables donde se coloca cada variable que se ocupa dentro del void loop. Después de la declaración de las variables se inicia el void setup que solo se ejecuta una vez, por este motivo en este apartado se estable la comunicación I2C con el sensor, en caso de que no exista la comunicación el programa entra a un ciclo for true que repetirá la inicialización del sensor MAX30100 hasta establecer una comunicación e imprimir en el monitor serial "MAX30100 funcionando". Finalmente, una vez entablada la comunicación I2C entre el sensor y la ESP8266, se establece una lectura de datos cada 3 segundos, donde se ma alrededor de 3000 datos de los cuales se hace un promedio y este es el que representa la medición de BPM y SpO2.

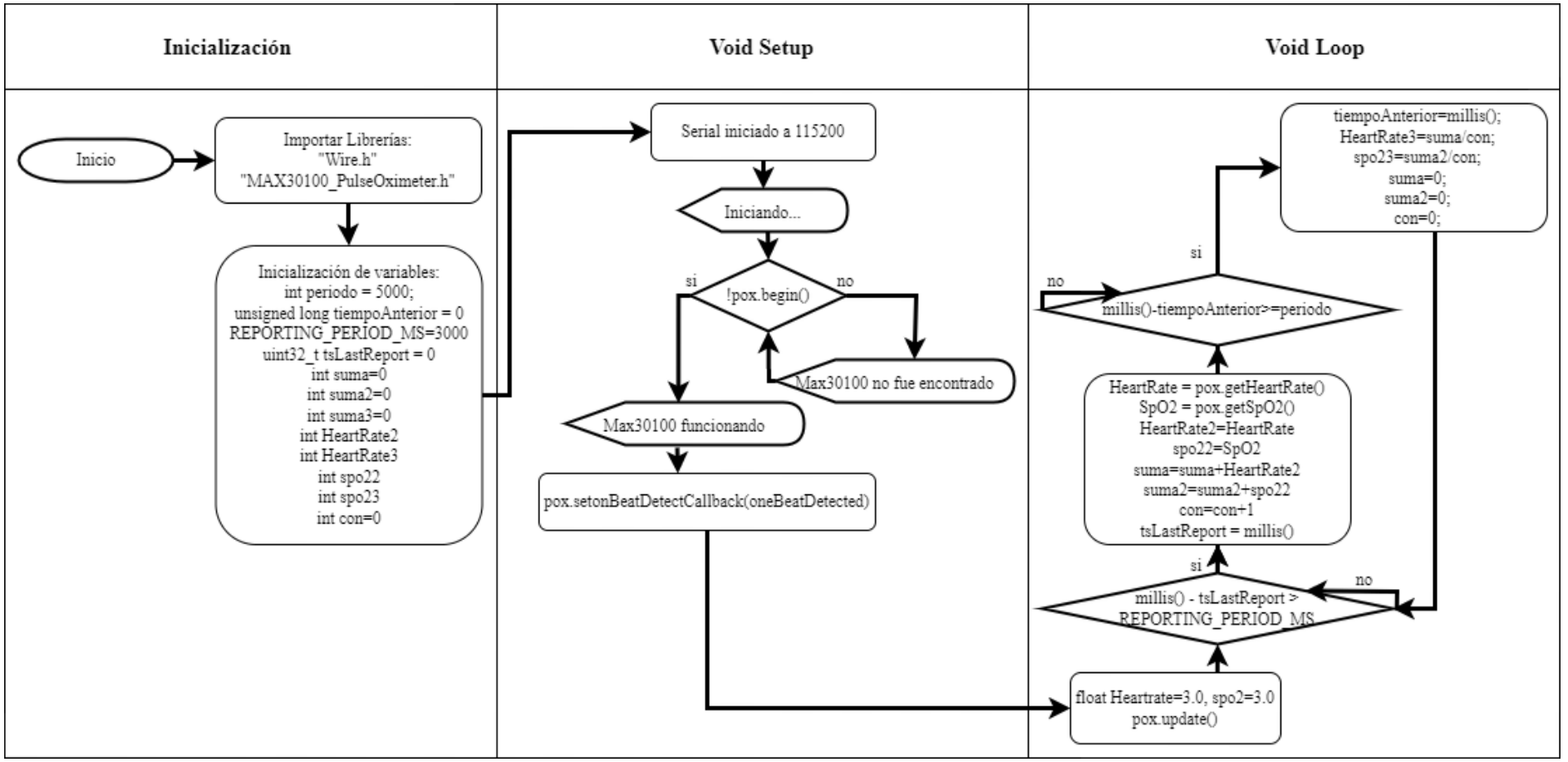


Figura 26: Diagrama de flujo del sensor MAX30100

Elaborado por: El investigador basado

3.2.4.2. Funcionamiento del sensor MLX90614

El sensor de temperatura infrarrojo MLX90614 aprovecha que cualquier tipo de objeto incluido el cuerpo humano que tenga una temperatura superior al cero absoluto (0°C o 273°K), emite una luz en el espectro infrarrojo que no visible para el ojo humano, esta taza de radiación es directamente proporcional a la temperatura del objeto de acuerdo con la ley de Stefan-Boltzmann expresado en la ecuación (9) [65]:

$$P = eA\sigma T^4 \qquad (9)$$

Donde:

P→ Taza de radiación (Watts)

A→Área de radiación (m^2)

e→ Emisividad de radiación (El cuerpo humano tiene un valor de 0.98)

σ→Constante de Boltzmann ($5.6703x10^{-8}$ [$Watts/m^2K^4$])

T→Temperatura del objeto (°K)

Campo de visión del sensor MLX90614

Como se observa en la figura 27 el campo de visión del MLX90614 tiene forma de cono y es relativamente amplio con 90°. Esto significa que por cada 1cm que se aleja de un objeto, el área de detección aumenta 2cm. Si está a 30cm (aproximadamente 1 pie) de distancia de un objeto, el área de detección será de 60cm (aproximadamente 2 pies) [65].

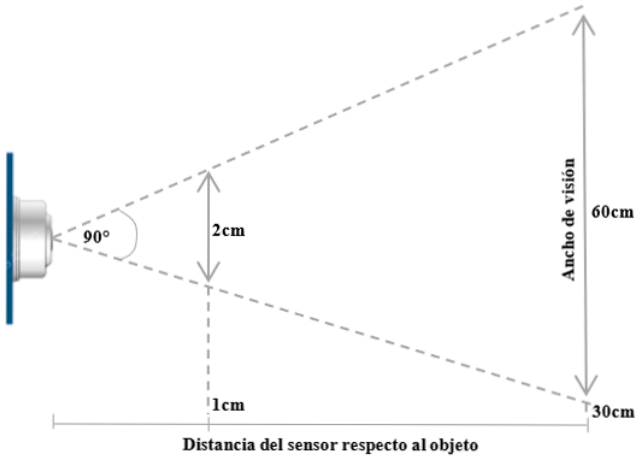


Figura 27: Campo de visión del sensor MLX90614 [65]

Diagrama de conexiones del sensor MLX90614

Para conectar el sensor MLX90614 con la ESP8266, primero se debe conectar el pin SCL al pin de reloj I2C (D1) y el pin SDA al pin de datos I2C (D2) del ESP826. Una vez se haya conectado los pines I2C se debe polarizar el componente electrónico como muestra la figura 28, donde el pin VCC se conecta al pin de alimentación de 3.3V ya que este voltaje es el nivel logico que utiliza el microcontrolador. Por ultimo, se conecta

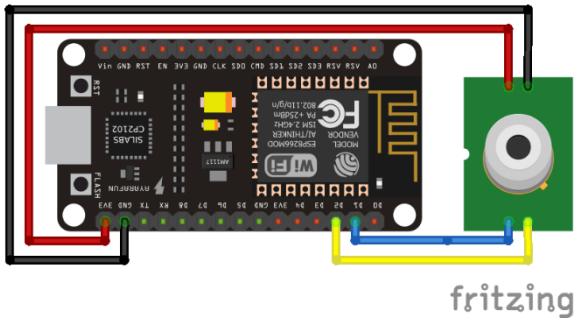


Figura 28: Diagrama de conexión del sensor MLX90614

Elaborado por: El investigador

Diagrama de flujo del sensor MLX90614

Como se observa en la figura 29, el funcionamiento del sensor MLX90614 en conjunto con el microcontrolador ESP8266 consta de la inicialización de variables donde se coloca cada variable que se ocupan dentro del void loop. Después de la declaración de las variables se inicia el void setup que solo se ejecuta una vez, por este motivo en este apartado se inicia la comunicación I2C con el sensor, en caso de que no exista la comunicación el programa entra a un ciclo while true que repetirá la inicialización del sensor MLX90614 hasta establecer una comunicación e imprimir en el monitor serial “MLX90614 funcionando”. Finalmente, una vez entablada la comunicación I2C entre el sensor y la ESP8266, se inicia la lectura de los datos, donde se toma la temperatura ambiente y del objeto en el campo de visión del sensor.

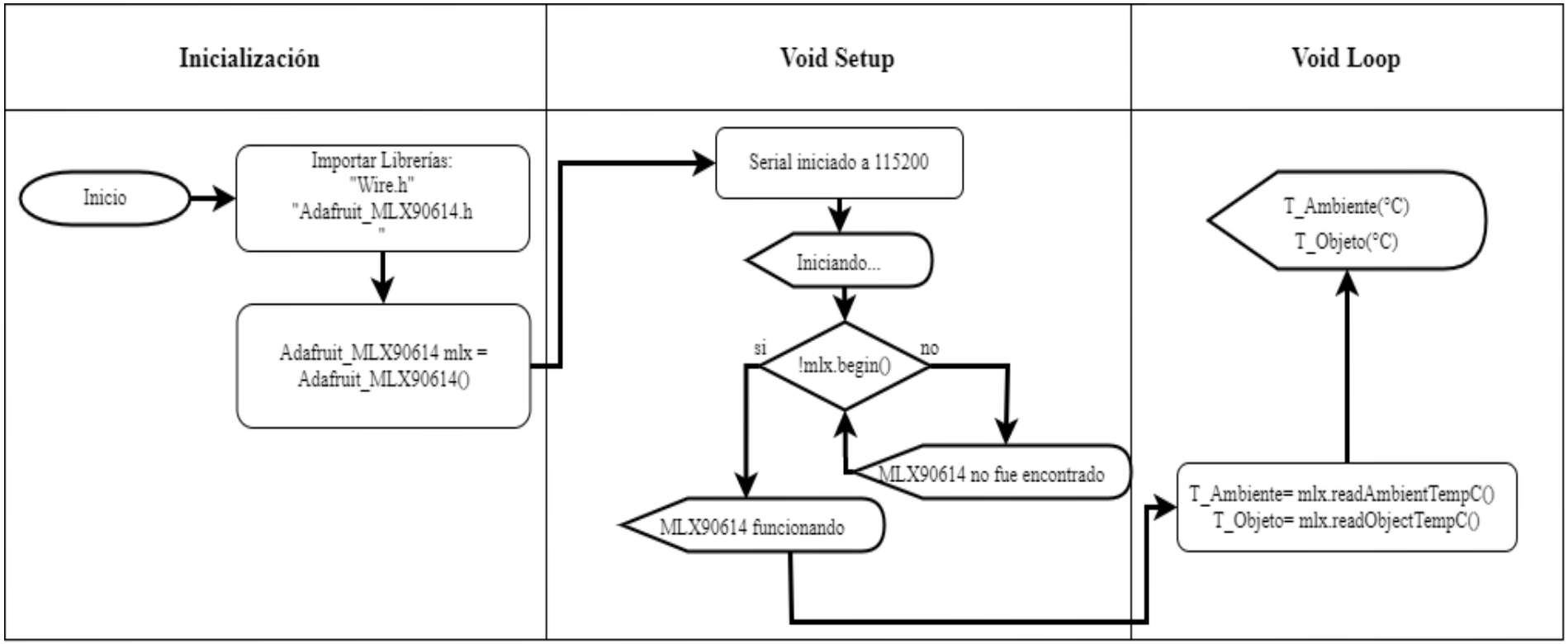


Figura 29: Diagrama de flujo del sensor MLX90614

Elaborado por: El investigador basado

3.2.4.3. Funcionamiento del módulo TP4056

El módulo TP4056 funciona suministrando energía de 5 V desde un cable micro USB tipo C o suministrando la energía desde las entradas +IN e -IN; adicionalmente, el cargador que se utilice debe tener al menos 1A para que cargue correctamente una batería Lipo conectada en los terminales de salida. Este módulo permite que el sistema electrónico sea utilizado mientras se encuentra cargando la batería; sin embargo, se aconseja no utilizar el dispositivo mientras se encuentra cargándose caso contrario se debe desconectar las terminales B+ y B-. Una vez cargada la batería se utiliza los pads OUT+ y OUT- para suministrar energía al sistema electrónico.

Diagrama de conexiones del módulo TP4056

Como se observa en la figura 30 el módulo TP4056 consta de dos métodos para el suministro de energía mediante el puerto Micro-USB tipo C o las entradas INT (+) y INT (-). No obstante, al utilizar los pines de entrada (INT) se debe tener en cuenta que la alimentación no puede sobrepasar los 5V. Los pines de la batería Lipo son conectados a los pines B (+) y B (-) del módulo. Una vez conectada la alimentación del TP4056 y la batería, los pines que alimentaran el circuito son los pines de salida (OUT (+) y OUT (-)) que permite utilizar el circuito mientras este se encuentra en modo de carga.

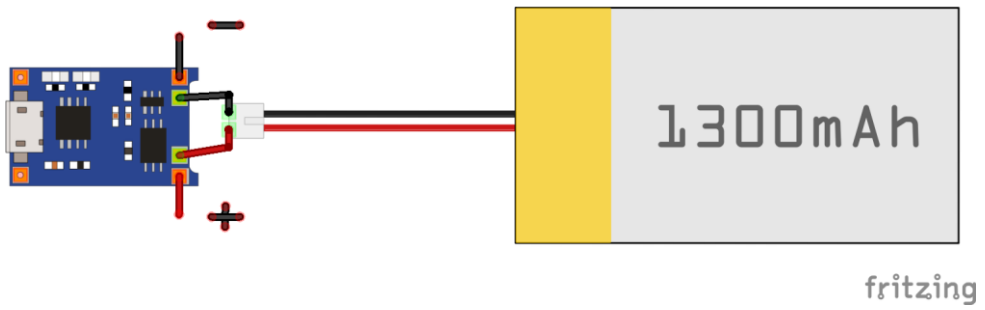


Figura 30: Diagrama de conexión del módulo TP4056

Elaborado por: El investigador

3.2.5. Diagrama del sistema electrónico de medición de signos vitales

Se utilizó 2 microcontroladores ESP8266, el primer controlador es un mini ESP8266 utilizado para la medición de BPM y SpO2, esto se realiza debido a que el sensor MAX30100 trabaja con una velocidad de reloj I2C de 4MHz, mientras que el sensor MLX90614 funciona con 1MHz esto causara que nunca se pueda activar el sensor de frecuencia cardiaca y la saturación de oxígeno en la sangre, ya que no será detectado y el microcontrolador entrara en un for true infinito. Estas conexiones se pueden observar en la figura 31 y 32 del diseño esquemático, donde el mini ESP8266 se conecta al MAX30100 previamente acondicionado para que trabaje al mismo nivel lógico del microcontrolador, además se utilizó un interruptor para controlar el encendido del sensor MAX30100.

El segundo microcontrolador ESP8266 está conectado al sensor MLX90614 a una velocidad de reloj de 1MHz para que funcione correctamente, en los mismos pines de SCL y SDA se conectó la pantalla Oled SSD1306. Sin embargo, es necesario considerar el direccionamiento I2C que ocupara cada componente para que no exista errores en el bus de datos, este direccionamiento se puede observar en la tabla 22. Adicionalmente, se colocó 2 diodos led como indicadores para observar si se estableció una conexión a la red o caso contrario no se pudo acceder a la red, donde el diodo led rojo representa que no se entablo una comunicación con la red o se quiere conectar el dispositivo a una red nueva.

En caso de que el led rojo este activo por más de 10 segundos el microcontrolador levantara una red a manera de AP para configurar de nuevo la red, por otro lado, el led verde representa que se estableció conexión con la red y se mantendrá encendido mientras esa conexión perdure, cabe recalcar que la red a la que se desee conectar debe tener acceso a Internet caso contrario el dispositivo no podrá enviar los datos a la aplicación Web. A este segundo microcontrolador se encuentra conectado un divisor de voltaje con 2 resistencias de 100KΩ para disminuir el voltaje que entra al microcontrolador, esto se utiliza con el fin de medir el nivel de carga del dispositivo y mostrarlo en la pantalla Oled. Finalmente, ambos microcontroladores ESP8266 se encuentran conectados en una comunicación serial para que el mini ESP8266 envié los datos de BPM y SpO2 al segundo microcontrolador que es el principal ya que es

el encargado de mostrar los datos en la pantalla Oled, mostrar el estado de conexión con la red, medir la carga de la batería, medir la temperatura y enviar los datos a la aplicación Web por medio del protocolo MQTT. Para el suministro de energía se empleó una batería Lipo de 3.7V a 0.6A y para poder cargar la misma se utiliza un módulo TP4056 con micro USB tipo C ya que es el cargador más común en la actualidad.

Tabla 22: Conexiones del sistema electrónico

Componente	Pin sensor	Pin ESP8266	Microcontrolador	Comunicación	Direccionamiento I2C
MAX30100	SCL	D1	Mini ESP8266	I2C	0X57
	SDA	D2			
	INT	D3			
	VCC	5V			
	GND	G			
MLX90614	SCL	D1	ESP8266	I2C	0X3C
	SDA	D2			
	VCC	3V3			
	GND	GND			
Pantalla Oled	SCL	D1	ESP8266	I2C	0X5A
	SDA	D2			
	VCC	3V3			
	GND	GND			
Diodo led rojo	VCC	D4	ESP8266	-	-
	GND	GND			
Diodo led verde	VCC	D5	ESP8266	-	-
	GND	GND			

Elaborado por: El investigador

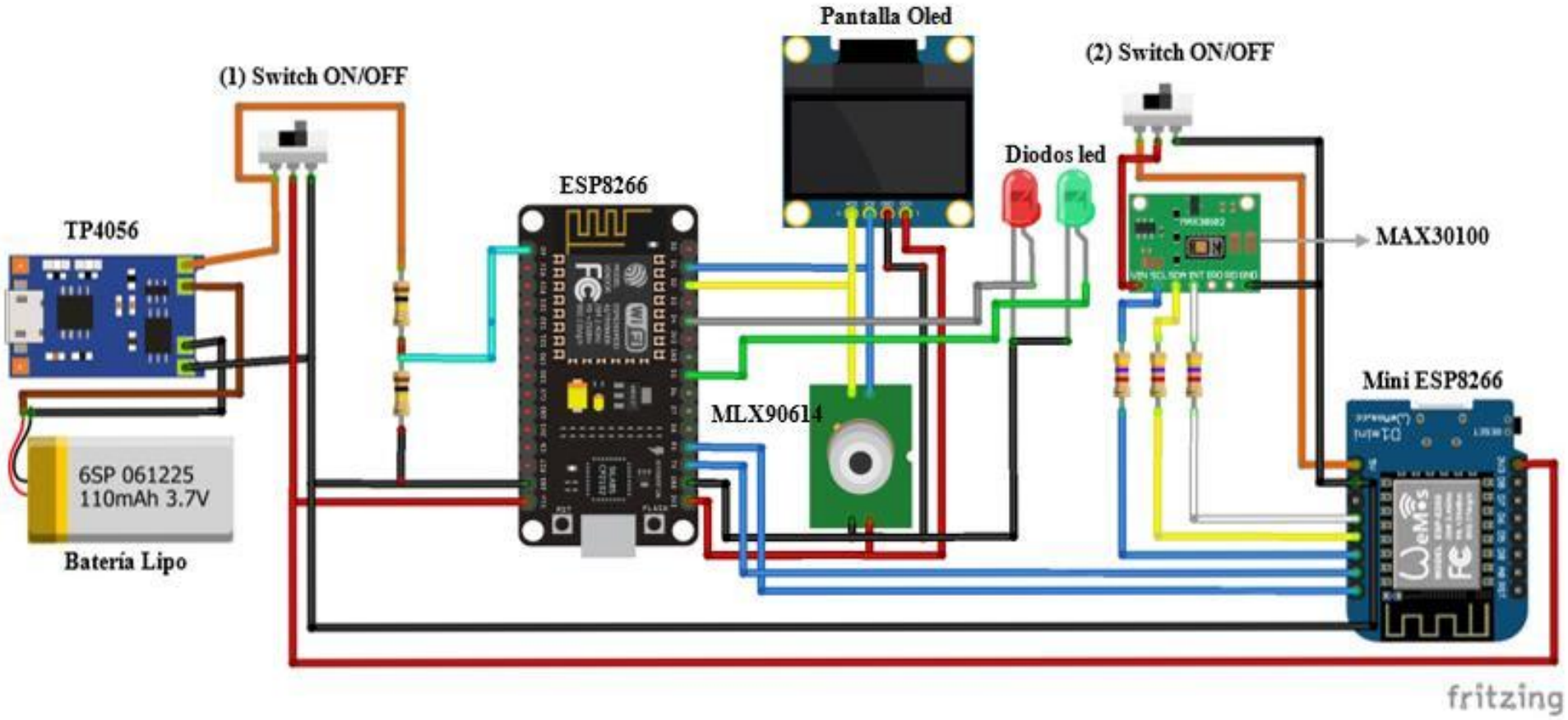


Figura 31: Diagrama de conexiones del sistema electrónico de medición de signos vitales

Elaborado por: El investigador

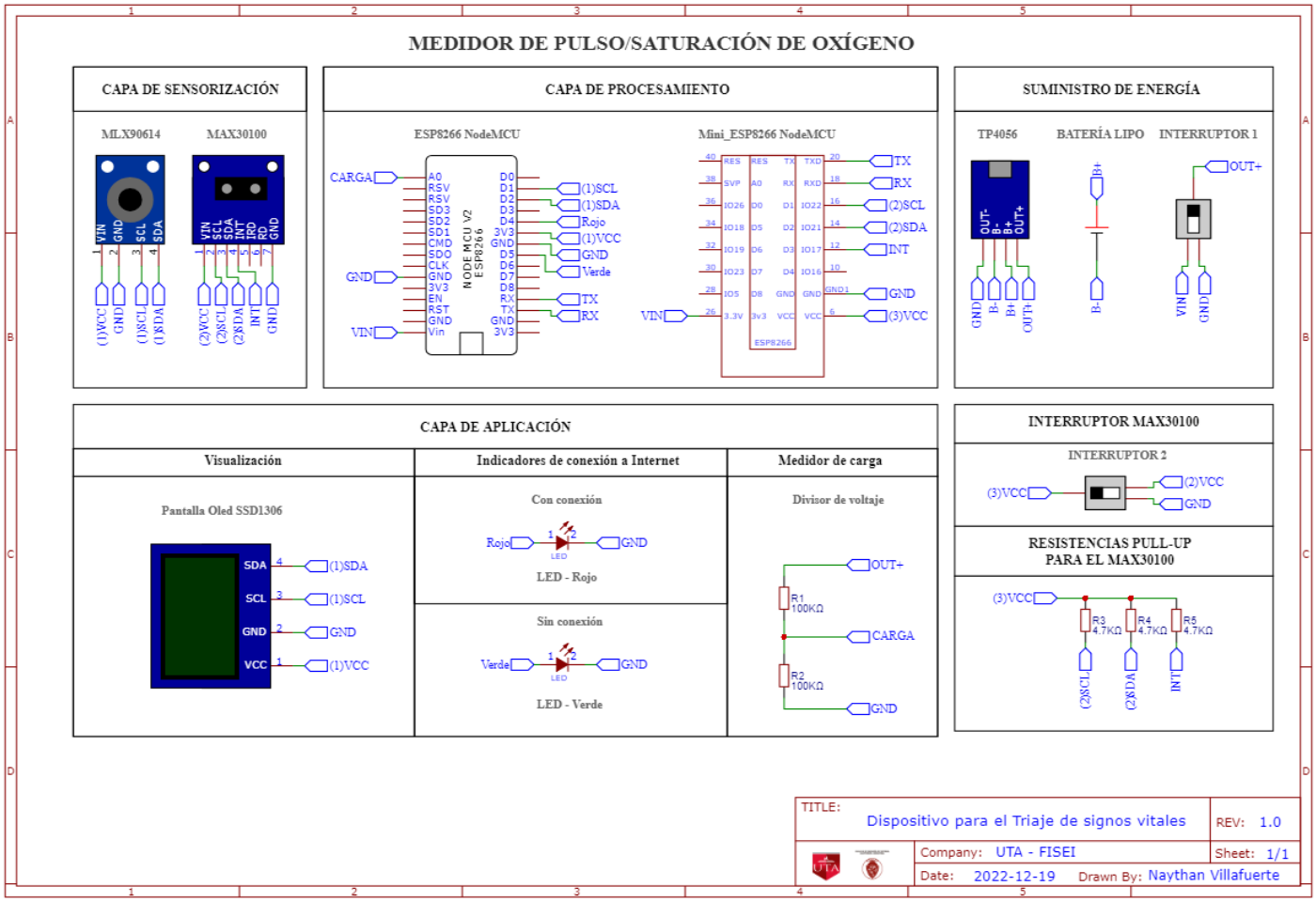


Figura 32: Diagrama de esquemático del sistema electrónico de medición de signos vitales

Elaborado por: El investigador

3.2.6. Programación de los microcontroladores ESP8266

La programación de los microcontroladores se realizó mediante Arduino IDE, en donde se configura la comunicación I2C con los sensores, la pantalla Oled, la conexión a la red, la medición del nivel de carga de la batería y el envió de los signos vitales a la aplicación web para ser visualizados en una interfaz gráfica.

3.2.6.1. Sensor MAX30100

Para programar el sensor MAX30100 se debe importar la librería "MAX30100lib" desde el gestor de bibliotecas de Arduino IDE, de donde se utilizó la biblioteca "MAX30100_PulseOximeter.h" como se visualiza en la figura 33, que se encarga de acondicionar la señal para tratar de eliminar la mayor parte de variaciones en las oscilaciones en la señal del fotodiodo al momento que este recibe la luz roja y la luz infrarrojo que rebota.

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
```

Figura 33: Inclusión de la biblioteca MAX30100 PulseOximeter

Elaborado por: El investigador

Declaración de variables para programar el MAX30100

Posteriormente se debe declarar las variables que se muestran en la figura 34 donde se define REPORTING_PERIOD_MS que es la sintaxis utilizada por la biblioteca para una variable constante que retrasa las mediciones en 3 segundos. Mientras que la variable tsLastReport almacena el tiempo en donde ocurrió el último latido del corazón, además de variables enteras que son usadas para el cálculo de BPM y SpO2 dentro del void loop. También se llama al objeto PulseOximeter como pox para mayor facilidad al momento de crear el código. Finalmente, se crea una rutina denominada onBeatDetected que se ejecuta cada vez que el sensor detecta un pulso.

```
#define REPORTING_PERIOD_MS     3000
PulseOximeter pox;
uint32_t tsLastReport = 0;

void onBeatDetected()
{
  // Serial.println("Beat!");
}

int suma=0;
int suma2=0;
int suma3=0;
int HeartRate2;
int HeartRate3;
int spo22;
int spo23;
int con=0;
```

Figura 34: Declaración de variables

Elaborado por: El investigador

Lectura de la frecuencia cardiaca y la concentración de oxígeno en la sangre

Después de declarar las variables necesarias se debe configurar el void setup que se ejecutara una sola vez, por eso se debe inicializar la comunicación serial y la comunicación I2C con el sensor max30100 como se puede contemplar en la figura 35. Adicionalmente es importante ingresar la velocidad de transferencia de datos en baudios o bits por segundo (bps) dentro de "serial.begin".

```
void setup()
{
  Serial.begin(115200);

  Serial.print("Iniciando...");
  if (!pox.begin()) {
    Serial.println("MAX30100 no fue encontrado. Chequea la alimentación.");
    for (;;);
  } else {
    Serial.println("MAX30100 en funcionamiento..");
  }
  pox.setOnBeatDetectedCallback(onBeatDetected);
}
```

Figura 35: Configuración del void setup

Elaborado por: El investigador

Una vez inicializada la comunicación serial se utiliza una sentencia "if" que mediante el operador booleano not (!) permite saber si el sensor MAX30100 está activo ("True") devolverá "MAX30100 en funcionamiento" caso contrario esto representar un "False" que hará que el microcontrolador entre en un ciclo "for" repetitivo que devolverá

“MAX30100 no fue encontrado” en el monitor serial, este comportamiento se puede observar en la tabla 23. Por último, se debe anotar la función onBeatDetected() para que registre los pulsos del corazón.

Tabla 23: Consulta de funcionamiento del sensor MAX30100

Estado del sensor (Entrada)	!pox.begin()	Monitor serial (Salida)
True	False	“MAX30100 no fue encontrado. Chequea la alimentación.”
False	True	“MAX30100 en funcionamiento...”

Elaborado por: El investigador

Una vez inicializado el sensor se configura el void loop(), para ello se declara las variables de HeartRate (BPM) y SpO2 inicializadas en 3. A continuación se emplea “pox.update” para leer los datos del sensor. Luego se obtiene las lecturas de la frecuencia cardíaca y la concentración de oxígeno en la sangre cada vez que la diferencia del tiempo transcurrido y el tiempo de funcionamiento del microcontrolador (esto se obtiene con la función “milis()”) sea mayor a 3 segundos, los datos de BPM y SpO2 son almacenadas en Heartrate2 y SpO22. Una vez almacenadas las lecturas es necesario sumar estas lecturas en las variables “suma” y “suma2” cada vez que la condición se cumpla, con el objetivo de sacar el promedio de estas lecturas y no tener una variación brusca en los resultados. Por este motivo se utiliza un contador que es la variable “con” que es utilizada en el segundo condicional “if” que tiene una lógica similar al primero por ende cada vez que se cumpla la condición de que la diferencia del tiempo transcurrido y el tiempo de funcionamiento del microcontrolador sea mayor a 5 segundos se calculara el promedio de los datos de BPM y SpO2 que serán mostrados en el monitor serial, empleando esta misma comunicación se envía estos datos al microcontrolador principal. Al final del segundo condicional “if” es necesario resetar las variables caso contrario en el promedio de los datos se incluirán mediciones anteriores que no permitirán conocer el BPM y SpO2 en tiempo real. Cabe mencionar que para obtener la lectura del ritmo cardíaco, se llama a la función getHeartRate(). De manera similar, para obtener la concentración de oxígeno en la sangre, se usó la función getSpO2() como se muestra en la figura 36.

```
  float HeartRate = 3.0, SpO2 = 3.0;
  // Inicializacion de variables de BPM y SpO2
  pox.update();

  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
      HeartRate = pox.getHeartRate();
      SpO2 = pox.getSpO2();
      HeartRate2=HeartRate;
      spo22=SpO2;
      suma=suma+HeartRate2;
      suma2=suma2+spo22;
      con=con+1;
      tsLastReport = millis();
   }

if(millis()-tiempoAnterior>=periodo){
  tiempoAnterior=millis();
  HeartRate3=suma/con;
  spo23=suma2/con;
  Serial.print(HeartRate3);
  Serial.print(",");
  Serial.println(spo23);
  suma=0;
  suma2=0;
  con=0;
  }
```

Figura 36: Configuración del void loop

Elaborado por: El investigador

Es importante mencionar que la programación para la lectura de los datos de BPM y SpO2 son desarrollados en el mini ESP8266 que se muestra en la figura 30, ya que los sensores MAX30100 y MLX90614 trabajan a una frecuencia de reloj I2C diferente. El resto de configuraciones del sistema electrónico se programaron en el ESP8266 que actúa como el microcontrolador principal que será el encargado de medir la temperatura por medio del MLX90614, mostrar la información en la pantalla Oled, configurar la conexión a la red, medir el nivel de carga de la batería, leer los datos enviado por comunicación serial del mini ESP8266 y finalmente enviar los datos de los sensores a la aplicación web para que sea mostrada en una Interfaz gráfica alojada en la nube.

3.2.6.2. Sensor MLX90614

Para programar el sensor MLX90614 se debe importar la librería “Adafruit MLX90614 Library” desde el gestor de bibliotecas de Arduino IDE, de donde se utilizó la biblioteca “Adafruit_MLX90614.h” además es necesario declarar el objetico “termometroIR” como se observa en la figura 37 para poder acceder a las funciones de la librería.

```
// Instanciar objeto (Sensor de temperatura)
Adafruit_MLX90614 termometroIR = Adafruit_MLX90614();
```

Figura 37: Instancia del objeto "termometroIR"

Elaborado por: El investigador

Declaración de variables para programar el sensor MLX90614

Una vez declarado el objeto “termometroIR” es necesario declarar las variables de temperatura como se observa en la figura 38, para obtener la lectura de la temperatura ambiente se llama a la función “termometroIR.readAmbientTempC()”. De manera similar, para obtener la temperatura del cuerpo humano, se usó la función “termometroIR.readObjectTempC()”. Cabe mencionar que se debe sumar un número entero con el objetivo de calibrar el sensor, esto se realiza de manera experimental.

```
// Obtener temperaturas grados Celsius
float temperaturaAmbiente = termometroIR.readAmbientTempC();
// se aumenta el valor de "6" para calibrar el valor de la temepratura del cuerpo
float temperaturaObjeto = (termometroIR.readObjectTempC()+6);
```

Figura 38: Variables de temperatura

Elaborado por: El investigador

Lectura de la temperatura ambiente y temperatura corporal

Después de declarar las variables necesarias se debe verificar la comunicación I2C al momento de activar el sensor MLX90614 como se puede contemplar en la figura 39.

```
if (!mlx.begin()) {
  Serial.println("Error al conectar con el MLX90614. Chequea la corriente.");
  while (1);
} else {
  Serial.println("MLX90614 en funcionamiento..");
};
```

Figura 39: Comprobación del funcionamiento del sensor MLX90614

Elaborado por: El investigador

Una vez iniciada la comunicación I2C se utiliza una sentencia “if” que mediante el operador booleano not (!) permite saber si el sensor MLX90614 está activo (“True”) devolverá “MLX90614 en funcionamiento” caso contrario esto representar un “False” que hará que el microcontrolador entre en un ciclo “while” repetitivo que devolverá “MLX90614 no fue encontrado” en el monitor serial, este comportamiento se puede observar en la tabla 24.

Tabla 24: Consulta de funcionamiento del sensor MLX90614

Estado del sensor (Entrada)	! termometroIR.begin()	Monitor serial (Salida)
True	False	"MLX90614 no fue encontrado. Chequea la corriente."
False	True	"MLX90614 en funcionamiento..."

Elaborado por: El investigador

3.2.6.3. Comunicación serial

Para la comunicación serial se establece al microcontrolador secundario como el transmisor, por otra parte, el receptor es el microcontrolador principal y su conexión se realiza de acuerdo con la figura 40 donde el pin TX es conectado al pin RX.

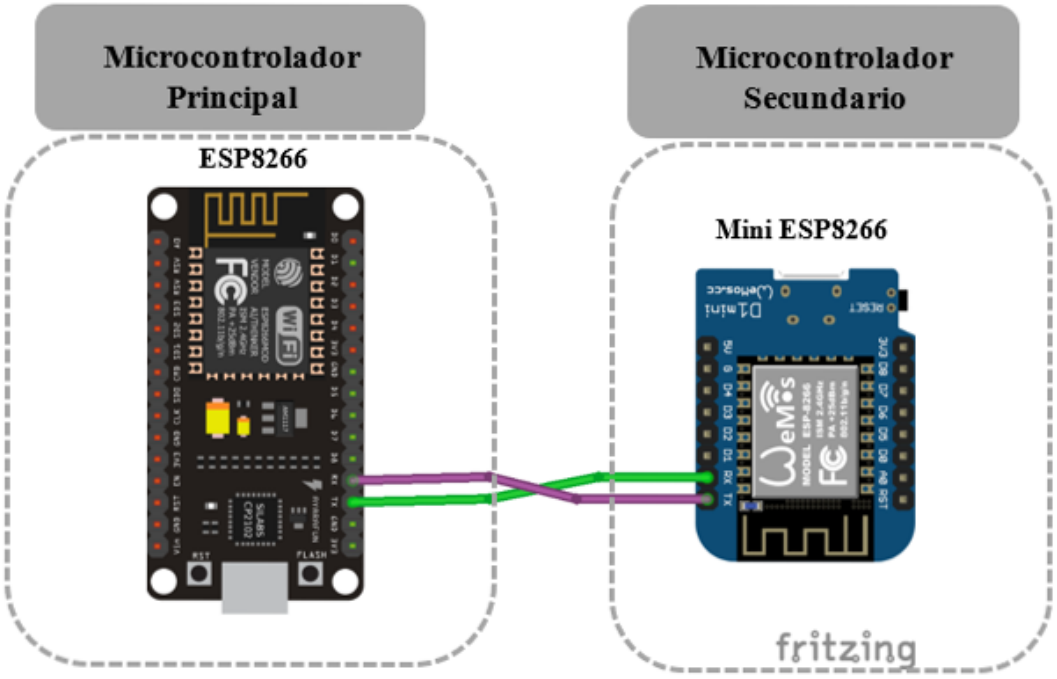


Figura 40: Comunicación serial de microcontroladores

Elaborado por: El investigador

Inicialización de la comunicación serial

Es necesario inicializar la comunicación serial de los dos microcontroladores a la misma velocidad de transferencia caso contrario los datos enviados desde el mini ESP8266 llegaran mediciones erroneas al ESP principal, por este motivo se utilizó una velocidad de 115200 baudios o 115200 bps (bits por segundo) en ambos

microcontroladores esto se puede observar en la figura 41. Esta configuración debe ser llevada a cabo dentro de “void setup()” ya que este parte del programa solo se ejecutara una sola vez, siendo ideal para activar una sola vez la comunicación serial y no activarla cada vez que se repita el loop (ciclo).

a)
```
Serial.begin(115200);
```

b)
```
Serial.begin(115200);
```

Figura 41: Configuración de la velocidad de transferencia en el: a) microcontrolador principal, b) microcontrolador secundario

Elaborado por: El investigador

Preparación del mensaje

Una vez inicializado la comunicación serial es necesario programar el mensaje que se desea enviar. Por consiguiente, fue preciso adecuar los datos del microcontrolador mini ESP8266 como se muestra en la figura 42, donde los datos de BPM (frecuencia cardíaca) y SpO2 (Saturación de oxígeno en la sangre) se encuentran concatenados en la misma línea, y a la vez divididos por el separador “,”. Esta sintaxis utilizada en el transmisor es de vital importancia ya que dependiendo a la forma cómo se ordenó el mensaje que se desea enviar se lo debe separar en el receptor.

```
Serial.print(HeartRate3);
Serial.print(",");
Serial.println(spo23);
```

Figura 42: Preparación del mensaje a enviar

Elaborado por: El investigador

Configuración del receptor

Como se mencionó anteriormente el receptor es el microcontrolador principal del sistema electrónico que se implementó. En consecuencia, es necesario declarar las variables que se utilizaran para separar la cadena de datos enviada por comunicación serial. En la figura 43 se observa la variable “separador” que fue la utilizada para distinguir los datos de BPM y SpO2 previamente concatenados dentro del transmisor.

Finalmente se utilizó las variables dataLenghth y data para almacenar la trama recibida por el microcontrolador y poder separarla.

```
//Declaracion de variables
const char separator = ','; // separador de trama
const int dataLength = 2; // el numero del incremento para los valores de la cadena
float data[dataLength]; // tipo de datos que se almacena en el vector
```

Figura 43: Declaración de variables

Elaborado por: El investigador

Finalmente, como muestra la figura 44 se separa los datos dentro del vector "data", donde el dato de BPM se encuentra en la posición 0 del vector mientras que el dato de SpO2 se encuentra en la posición 1 del vector. Una vez separados y almacenados los datos dentro de las variables se puede utilizar esta información para imprimir en la pantalla Oled y enviar el dato a la aplicación Web.

```
bpm=data[0]; // guardamos el dato en la variable BPM
spo2=data[1]; // guardamos el dato en la variable SPO2
```

Figura 44: Separación de los datos de BPM y SpO2

Elaborado por: El investigador

3.2.6.4. Configuración de la conexión a Internet

Para conectar al microcontrolador a Internet se configuró una nueva biblioteca dentro de Arduino usando como base la librería "Wifi Manager". La nueva biblioteca creada por el investigador es capaz de conectar el dispositivo de medición de signos vitales al internet haciendo uso de la memoria ROM de las placas de desarrollo de Arduino que son capaces de guardar las últimas credenciales de conexión a internet configuradas. Esta nueva librería ofrece dos modos de funcionamiento para el microcontrolador, el primero modo de funcionamiento es el esperado ya que el microcontrolador al conectarse a Internet encenderá un led verde como indicador de conexión para posteriormente después de 3 segundos accionar la pantalla Oled que muestra los datos de temperatura, BPM, SpO2 y nivel de carga. Sin embargo, el segundo modo de funcionamiento se da lugar cuando el dispositivo no puede conectarse a Internet, después de un tiempo de espera de 15 segundos, mientras tanto un led rojo estará parpadeando hasta que se establezca la conexión.

Una vez superado el tiempo de espera el microcontrolador pasa a funcionar como un AP donde levanta una red LAN interna, a la que el usuario se puede conectar para acceder a las configuraciones de Wifi Manager, donde se pide que ingrese las credenciales a fin de conectar el dispositivo a internet. En caso de que las credenciales ingresadas no sean correctas el microcontrolador seguirá funcionando como un AP hasta que el nombre de la red y contraseña sean correctas, este funcionamiento se puede apreciar en la tabla 25.

Tabla 25: Funcionamiento de la librería "Cambiar ssid"

Conexión a la red	Diodo led encendido	Modo de funcionamiento
Con conexión	Verde	Modo normal
Sin conexión	Rojo	Modo AP

Elaborado por: El investigador

Creación de la librería “cambiar ssid”

Para crear la librería “cambiar ssid” se usó como base la biblioteca de Wifi Manager, por ende, es necesario incluir las siguientes librerías dentro del programa como se visualiza en la figura 45. Todas estas librerías a excepción de “Ticker.h” son necesarias para ocupar las herramientas de Wifi Manager. La biblioteca de “Ticker” ofrece la posibilidad de crear retardos los cuales se ocuparon para crear el parpadeo del diodo led rojo usado en el sistema electrónico.

```
//Creacion de una libreria
#include <ESP8266WiFi.h>
#include <WiFiManager.h>
#include <strings_en.h>
#include <wm_consts_en.h>
#include <wm_strings_en.h>
#include <Ticker.h>
```

Figura 45: Importación de librerías de Wifi Manager

Elaborado por: El investigador

Una vez incluidas las librerías necesarias se declaró las siguientes variables pinLedWifi que es asignada al pin D4 del microcontrolador para conectar el diodo led rojo. Después se tiene a la variable pinLedWifiOFF que es asignada al pin D5 del ESP8266 donde se conectó el diodo led verde. Por último, se tiene instanciación del objeto ticker que permite el uso de las herramientas de “Ticker.h” como se observa en la figura 46.

```
#define pinLedWifi D4
#define pinLedWifiOFF D5
//Instancia de la clase Ticker
Ticker ticker;
```

Figura 46: Declaración de variables de la librería “cambiar ssid”

Elaborado por: El investigador

Fue necesario declarar una función denominada “parpadeoLedWifi” observado en la figura 47, aquí se programa el parpadeo intermitente del diodo led rojo mediante un operador booleano “!” que niega el estado del pin D4 constantemente en intervalos de 200 ms.

```
void parpadeoLedWifi(){
  //Cambiar de estado el led
  //lee el esatdo del led le en D4
  byte estado =digitalRead(pinLedWifi);
  //Despues niega el estado(Prender y apagar led)
  digitalWrite(pinLedWifi, !estado);
```

Figura 47: Función de prendido y apagado del diodo led rojo

Elaborado por: El investigador

Finalmente, en la figura 48 se declara la función “conectarWifi” donde se configuran los pines D4 y D5 como salidas, después mediante la herramienta “ticker.attach” se llama a la función “parpadeoLedWifi” para que entre en funcionamiento cada 200 ms. Adicionalmente, fue necesario instanciar el objeto wifiManager que es utilizado en una sentencia condicional “if”, que verifica si la conexión a la red se realizó con éxito o de lo contrario se levantara una red LAN con el nombre de Teledoctor y con la contraseña “12345678”. Esta comprobación de conectividad se realiza cada segundo hasta que finalmente tenga conexión a la red, una vez conectado a la red el parpadeo intermitente del diodo led rojo se desactiva y el estado del pin D5 pasa a alto, dicho de otro modo, se enciende el diodo led verde.

```
void conectarWiFi(){
  Serial.begin(115200);
  //Modo del pin del led
  pinMode (pinLedWifi, OUTPUT);
  pinMode (pinLedWifiOFF, OUTPUT);
  //Temporizador que sirve para el parpadeo el led
  ticker.attach(0.2, parpadeoLedWifi);
  // creacion de la instancia wifimanager
  WiFiManager wifiManager;
  // limpiar credencial wifi anteriores
  // wifiManager.resetSettings();
  //se trata de conectar al wifi y si no levanta una red "Teledoctor" con clave "12345678"
  if(!wifiManager.autoConnect("Teledoctor", "12345678")){
    Serial.println("Falla la conexión tiempo de espera agotado");
    ESP.reset();
    delay(1000);
  }
  Serial.println("Ya esta conectado");
  //Eliminar el temporizador
  ticker.detach();
  digitalWrite(pinLedWifi, LOW);
  digitalWrite(pinLedWifiOFF, HIGH);
}
```

Figura 48: Función de conexión a internet

Elaborado por: El investigador

Inclusión de la librería “cambiarssid.h” dentro de Arduino

Para incluir esta nueva librería es necesario dirigirse a la dirección donde se instaló el IDE Arduino, en este caso la instalación por default se realizó en la carpeta documentos. Cabe mencionar que la carpeta que contenga la librería debe tener el mismo nombre que el documento para que el microcontrolador reconozca que librerías se está utilizando esta sintaxis se puede ver en las figuras 49.

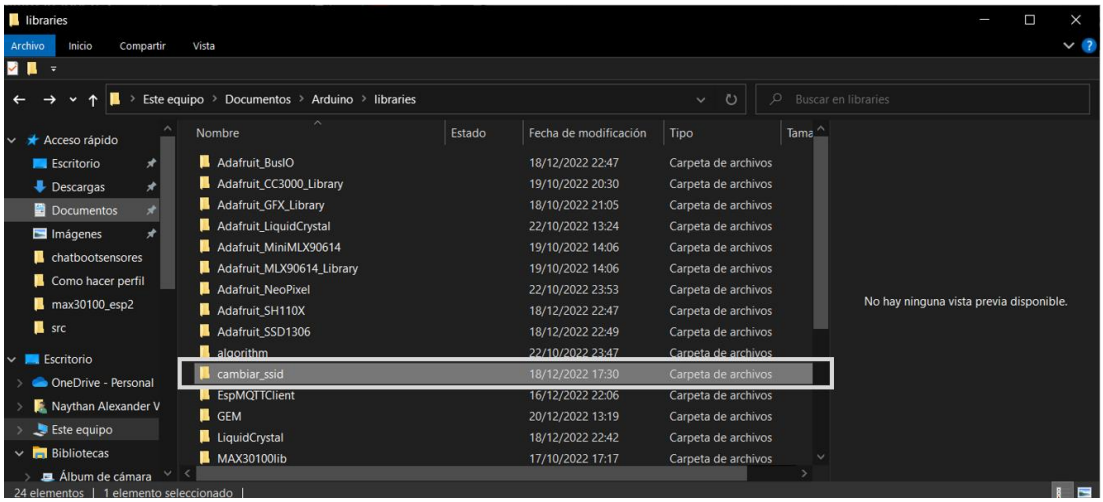


Figura 49: Inclusión de la librería Cambiar_ssid dentro de Arduino

Elaborado por: El investigador

Adición de la librería creada en el código principal

Para incluir la nueva librería solo fue necesario utilizar el comando “#include” que se observa en a figura 50 y finalmente llamar a la función “conectarwifi” vista en la figura 48. En el caso de la función llamada se debió colocarla dentro del void setup para que compruebe la conexión a la red antes de iniciar las mediciones de signos vitales.

```
//Importacion de librerias
#include <cambiar_ssid.h>
```

Figura 50: Inclusión de la librería "Cambiar_ssid"

Elaborado por: El investigador

3.2.6.5. Configuración del medidor de carga del dispositivo

Para medir el nivel de carga del dispositivo es necesario realizar un divisor de voltaje ya que el objetivo de reducir la tensión de entrada en el pin analógico debido a que un voltaje cercano o igual a 1.85V es compatible con el microcontrolador ESP8266, la conexión del divisor de voltaje se muestra en la figura 51, donde la unión de las dos resistencias es conectada al pin A0 del microcontrolador.

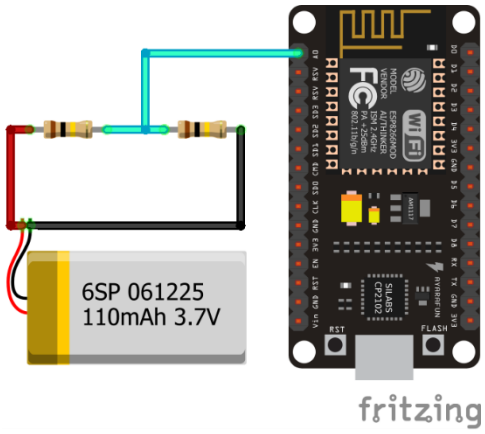


Figura 51: Conexión del divisor de voltaje

Elaborado por: El investigador

Para seleccionar las resistencias utilizadas en el divisor de voltaje se consideró el voltaje entregado por la batería, el voltaje de entrada del pin analógico recomendado por el fabricante y una resistencia de referencia R1 de 100KΩ, para obtener la resistencia R2 de acuerdo con la ecuación (10).

$$V_{out} = \frac{V_T R_2}{R_1 + R_2}$$ Ecuación (10)

Donde:

V_{out}→ Es el voltaje de salida (1.85V)

V_T→ Es el voltaje de la batería (3.7V)

R_1→ Resistencia uno (100KΩ)

R_2→ Resistencia dos (Ω)

Para obtener la R2 es necesario despejar la misma de la ecuación (10), como se muestra a continuación:

$$V_{out} = \frac{V_T R_2}{R_1 + R_2}$$

$$(R_1 + R_2)V_{out} = V_T R_2$$

$$R_1 V_{out} + R_2 V_{out} = V_T R_2$$

$$V_T R_2 - R_2 V_{out} = R_1 V_{out}$$

$$R_2(V_T - V_{out}) = R_1 V_{out}$$

$$R_2 = \frac{R_1 V_{out}}{V_T - V_{out}}$$

Una vez despejada la resistencia dos se remplaza los datos para obtener su valor, como se muestra a continuación:

$$R_2 = \frac{R_1 V_{out}}{V_T - V_{out}}$$

$$R_2 = \frac{(100K\Omega)(1.85V)}{3.7V - 1.85V} = 100K\Omega$$

Declaración de variables para medir el nivel de carga

Para medir el nivel de carga fue necesario declarar la variable analogInPin que es asignado al pin A0, sensorValue es la variable donde se almacenara el voltaje de entrada, bat_percentage almacenera el porcentaje del nivel de batería y finalmente como indica la figura 52 la variable calibration es puesta de acuerdo a la medición del voltaje de la batería en este caso la batería Lipo seleccionada es de 3.7V y el mapeo del voltaje es de 4.13V, por este motivo el valor de calibración se obtiene de la

diferencia entre el voltaje de la batería con respecto al voltaje manejado por la función "map".

```
// variables para medi nivel de carga
int analogInPin  = A0;    // Declaracion pin analogo A0
int sensorValue;          // Variable para la medicion del nivel de carga
float calibration = 0.43; // Calibrar el valor usando un multimetro de acuerdo al voltaje de la bateria
int bat_percentage;       // variable del porcentaje de la bateria
```

Figura 52: Declaración de variables

Elaborado por: El investigador

Medición del nivel de carga de la batería Lipo

El primer paso fue leer la entrada del pin analógico A0 de acuerdo con la figura 53, para posteriormente multiplicar el valor de voltaje de entrada por 2 ya que la tensión real es la suma de los voltajes de las dos resistencias de 100KΩ que tiene el mismo valor de voltaje al tener el mismo valor de resistencia. Finalmente, haciendo uso de la función "mapfloat" se hace una equivalencia entre el voltaje de la batería con respecto al porcentaje, donde el 100% representaría 4.12V. Cabe recalcar que es importante poner a cargar el dispositivo una vez que este alcance el 30% de batería según el proveedor.

```
//Medir nivel de bateria
sensorValue = analogRead(analogInPin);
float voltage = (((sensorValue * 3.3) / 1024) * 2 + calibration); //multiplique por dos ya que la red
bat_percentage = mapfloat(voltage, 2.8, 4.12, 0, 100); //2,8 V como voltaje de corte de batería y 4,12
```

Figura 53: Medición del nivel de carga de la batería

Elaborado por: El investigador

3.2.6.6. Configuración de la pantalla Oled SSD1306

Para configurar la pantalla Oled SSD1306 es necesario conectar las terminales I2C de la misma como se muestra en la figura 54, donde el pin D1 del microcontrolador se conecta a SLC y el pin D2 a SDA.

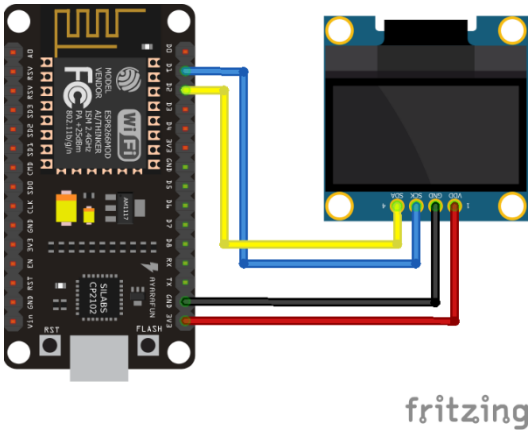


Figura 54: Conexión de pantalla Oled SSD1306

Elaborado por: El investigador

Declaración de variables para configurar la pantalla Oled

Las variables necesarias para configurar la pantalla Oled SSD1306 se muestran en la figura 55. Se debe declarar el ancho y alto de las imágenes que se desea usar para mostrarlo dentro de la pantalla oled, además es importante declarar el ancho de la pantalla que se está utilizando, estas dimensiones son proporcionadas por el proveedor del componente electrónico. Finalmente se establece un objeto "Adafruit_SSD1306" donde se establece el ancho y alto de la pantalla, que es el puntero de la clase estática de Wire (viene incluido en la librería Adafruit SSD1306), por último, se indica el pin de reinicio propio del microcontrolador.

```
// variables de pantalla oled
#define ancho_imagen 20
#define alto_imagen 20

#define ancho_imagen_2 50
#define alto_imagen_2 50
// Definir constantes
#define ANCHO_PANTALLA 128 // ancho pantalla OLED
#define ALTO_PANTALLA 64 // alto pantalla OLED

// Objeto de la clase Adafruit_SSD1306
Adafruit_SSD1306 display(ANCHO_PANTALLA, ALTO_PANTALLA, &Wire, -1);
```

Figura 55: Declaración de variables

Elaborado por: El investigador

Conversión de imágenes a matrices de bytes

Para usar cualquier imagen dentro de la pantalla Oled es necesario transformarla a una matriz de bytes debido a que la serie de pantallas Oled SSD1306 trabajan con columnas de 8 bits. Por este motivo se utilizó la herramienta online imagen2ccp que permite transformar cualquier tipo de imagen a su matriz de bytes como se ve en la figura 56.

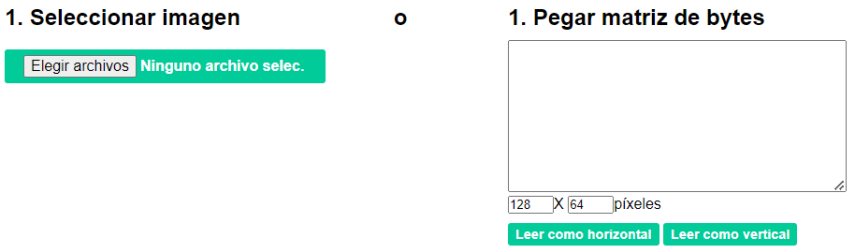


Figura 56: Subido de imagen a imagen2ccp

Elaborado por: El investigador

Una vez subida la imagen se ajusta su tamaño de la, también se recomienda escoger un fondo negro y después invertir los colores de la imagen como se indica en la figura 57 para no configurar este apartado por código dentro de Arduino.

Figura 57: Configuración de la imagen en imagen2ccp

Elaborado por: El investigador

Una de las funcionalidades de esta herramienta Web es que permite visualizar el resultado de la configuración de la imagen antes de subir el código al microcontrolador mediante una vista previa como muestra la figura 58.

Figura 58: Vista previa de la configuración de la imagen

Elaborado por: El investigador

Adicionalmente, la imagen2ccp proporciona la matriz de bytes de la imagen final para ser ocupada dentro de la programación de Arduino IDE, esta se muestra en las figuras 59.

Figura 59: Bytes generados de la imagen

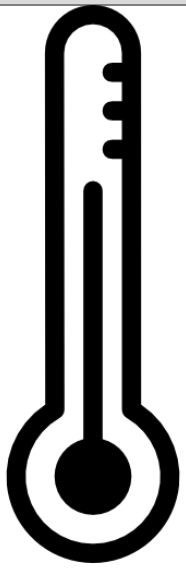
Elaborado por: El investigador

Configuración de las imágenes en Arduino IDE

Una vez se obtuvo la matriz de bytes con la ayuda de imagen2ccp se declaró cada matriz dentro del código de Arduino como muestra la tabla 26, este mismo proceso fue llevado a cabo con todas las imágenes que fueron utilizadas dentro de la presentación de la pantalla Oled SSD1306.

Tabla 26: Configuración de la matriz de bytes para la pantalla Oled

Declaración de la imagen	Imagen

`const unsigned char PROGMEM termometro [] = {`
 	
`0x00, 0x60, 0x00, 0x00, 0x90, 0x00, 0x00, 0x90, 0x00, 0x00, 0x90, 0x00, 0x00, 0x90, 0x00, 0x00,`
`0x90, 0x00, 0x00, 0x90, 0x00, 0x00, 0xb0, 0x00, 0x00, 0xb0, 0x00, 0x00, 0xb0, 0x00, 0x00, 0xb0,`
`0x00, 0x00, 0xb0, 0x00, 0x00, 0xb0, 0x00, 0x00, 0x90, 0x00, 0x01, 0x68, 0x00,`
`};`
	

Elaborado por: El investigador

Todas las imágenes utilizadas en la pantalla Oled se muestran en la tabla 27, cabe recalcar que estas imágenes son las que posteriormente son transformadas a su matriz de bytes para ser manejable en Arduino IDE.

Tabla 27: Imágenes utilizadas para la pantalla Oled

Descripción	Imagen utilizada
Es utilizada para mostrar la unidad de temperatura dentro de la pantalla Online.	
Esta imagen es utilizada para mostrar el valor de frecuencia cardíaca (BPM).	
Esta imagen es utilizada para mostrar el valor de saturación de oxígeno en la sangre (SpO2).	
Esta imagen es utilizada en una pantalla de inicio configurada en la pantalla Oled.	
Estas imágenes fueron utilizadas para representar el nivel de carga del dispositivo.	

Elaborado por: El investigador

Por último, la forma en que se usa la pantalla Oled se puede distinguir en la figura 60 donde simplemente se usa la herramienta de borrado “clearDisplay” para limpiar la pantalla después de cada transición, para poder usar la imagen transformada en matriz de bytes se empleó el comando “drawBitmap” donde las dos primeras variables corresponden a la posición. La tercera variable es la matriz de bytes, posteriormente las ultimas variables son el ancho, alto y color de la pantalla. Cabe recalcar que la gama de pantallas Oled SSD1306 solo pueden presentar las imágenes en blanco y negro.

```
///Mostrar datos en pantalla oled
display.clearDisplay();
display.setTextColor(SSD1306_WHITE); // Color del texto
display.setTextSize(1.5); // Tamaño del texto
display.setCursor(31, 0);
display.println("TEMPERATURA");
display.drawBitmap(0,15, termometro2, ancho_imagen_2, alto_imagen_2, WHITE);
display.setTextSize(2); // Tamaño del texto
display.setCursor(38, 30);
display.println(temperaturaObjeto);
display.drawBitmap(100,30, centigrados, ancho_centigrados, alto_centigrados, WHITE);
display.display(); // Enviar a pantalla
delay(2000);
```

Figura 60: Programación de la pantalla Oled SSD1306

Elaborado por: El investigador

3.2.6.7. Envío de datos a la aplicación Web

Para el envió de los datos a la aplicación Web se empleó el protocolo MQTT mismo que consta de la arquitectura que se visualiza en la figura 61, conformada por la adquisición de los datos medidos por los que son publicados en el tópico "sensores" y mediante el Broker de "Mosquitto" se publica estos datos a todos los subscritores, en este caso el único subscritor utilizado fue el servidor de "Google cloud" donde se aloja la máquina virtual.

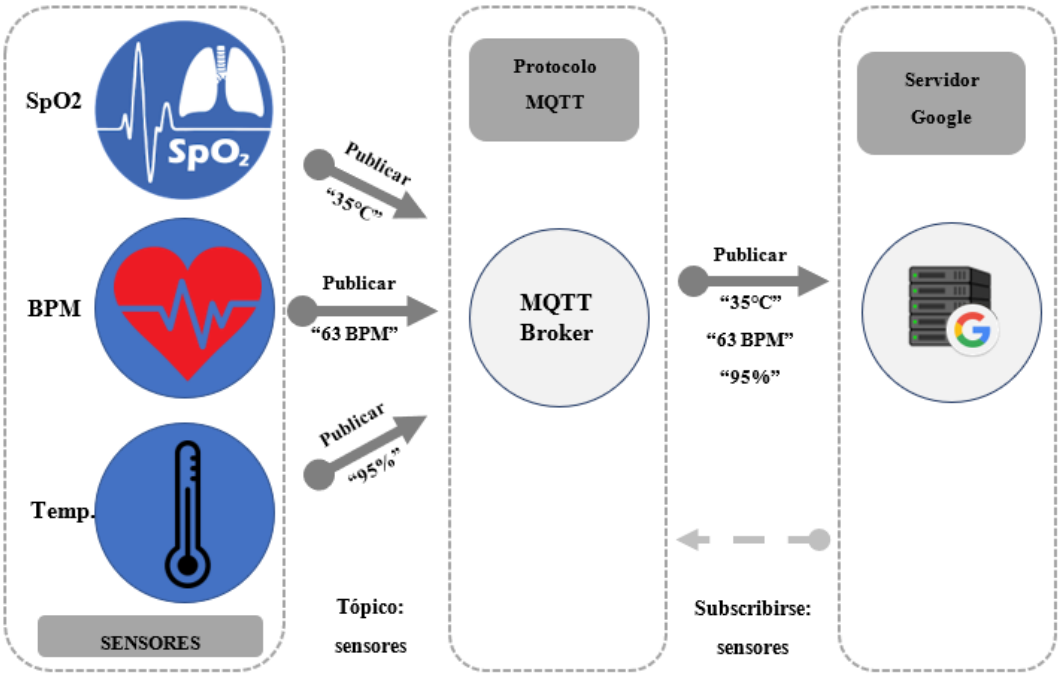


Figura 61: Arquitectura del protocolo MQTT utilizada

Elaborado por: El investigador

Declaración de variables

Para utilizar el protocolo MQTT, es necesario declarar el objeto “espClient” para poder utilizar las herramientas del bróker Mosquitto. El resto de las variables son utilizadas para transformar los datos obtenidos de los sensores de enteros a string, mientras que la variable “mqtt_server” fue donde se ingresó la dirección IP externa de la máquina virtual generada en Google cloud como se visualiza en la figura 62.

```
//Variables del broker mosquito
WiFiClient espClient;
PubSubClient client(espClient);
String str = "";
String interruptor;
const char* mqtt_server = "34.125.127.48";
//variables de sensores
int bpm, spo2;
long lastMsg = 0;
char msg[50];
int value = 0;
```

Figura 62: Declaración de variables

Elaborado por: El investigador

Cabe mencionar que para que exista conexión con la instancia virtual es necesario configurar el firewall de Google cloud ya que el puerto 1883 utilizado por el protocolo MQTT está bloqueado por defecto como se muestra en la figura 63.

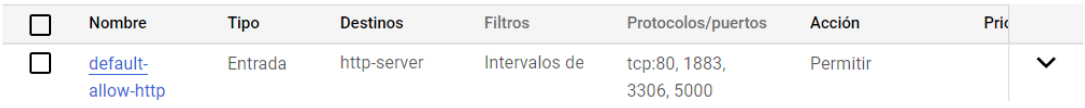


Figura 63: Configuración del Firewall de google cloud

Elaborado por: El investigador

Configuración del Broker Mosquitto

El primer paso es comprobar si existe conexión con el servidor mediante el condicional “if” donde su funcionamiento es similar al expuesto en la tabla 27. Una vez establecida la conexión se empieza a publicar los datos de los sensores en el subscritor alojado en la instancia virtual de Google cloud. En el caso de no existir conexión con el subscritor

se espera un segundo hasta tratar de reconectarse con el servidor como muestra en la figura 64.

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("chatboot/sensores", "Enviando el primer mensaje");
      // ... and resubscribe
      client.subscribe("chatboot/sensores");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

Figura 64: Configuración de la conexión con el broker Msquitto

Elaborado por: El investigador

Para poder enviar los datos haciendo uso del bróker Mosquitto es necesario transformar las variables numéricas a string para poder concatenarlas en una sola trama que será enviada y separada en el Back-End del aplicativo Web. Una vez concatenado los datos de los sensores se envió los 75 primeros caracteres dentro de la variable “msg” (mensaje) para no sobrecargar la comunicación y los datos que lleguen sean correctos, esto se logra usando la función “snprintf” que se muestra en la figura 65, esta función hace que el envío de datos no se sature tras enviar las mediciones cada segundo.

```
// Mostrar información

String temperaturaObjeto2 = String(temperaturaObjeto,2);
String bpms = String(bpm);
String spo22 = String(spo2);
String sensores = temperaturaObjeto2+","+bpms+","+spo22;

const char *sensores2= sensores.c_str();
snprintf (msg, 75, sensores2);
client.publish("chatboot/sensores", msg);
```

Figura 65: Cambiar números enteros a string

Elaborado por: El investigador

Adicionalmente, es importante considerar cual es la dirección del tópico que se esté usando en el publicador, ya que si el subscritor no tiene escrito correctamente el tópico no recibirá ningún dato.

3.2.6.8. Diagrama de flujo del sistema electrónico

El diagrama de flujo que se distingue en la figura 66 comienza con la declaración de las variables que se utilizaron dentro del microcontrolador principal, después en la etapa del void setup se inicializa la comunicación serial a una velocidad de 115200 baudios de acuerdo con lo indicado en la sección 3.2.6.3, posterior a esto se revisa la conectividad a la red. Para finalizar con la configuración del void setup se revisa la comunicación I2C del sensor MLX90614, la pantalla Oled, además se verifica si el servidor se encuentra en funcionamiento caso contrario entrara en un ciclo infinito hasta que exista conectividad y pase a las ventanas de información mostradas en la pantalla Oled. Por último, en el void loop se configuró la lectura de los datos de los sensores y la medición de la carga de batería del dispositivo, los mismos datos son mostrados en la pantalla Oled y enviados mediante el bróker Mosquitto a la aplicación Web a excepción del nivel de batería.

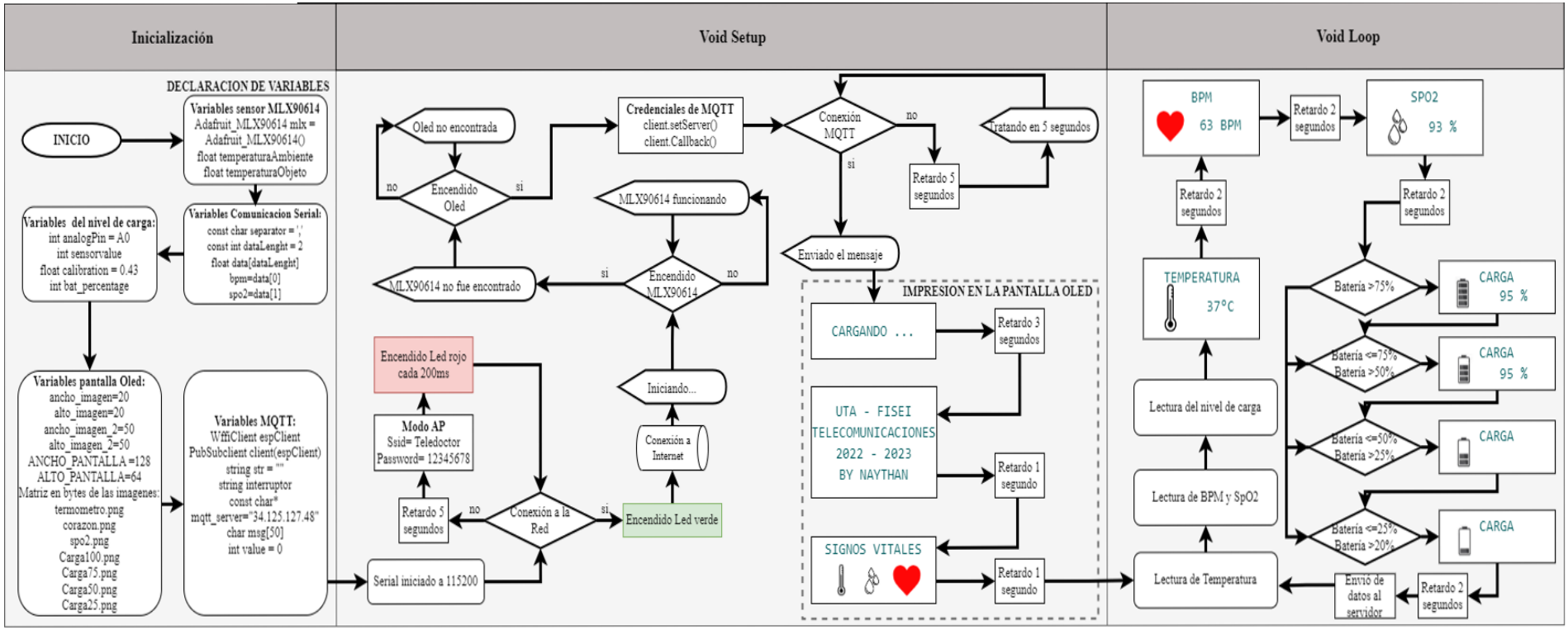


Figura 66: Diagrama de flujo del sistema electrónico de medición de Temperatura, BPM y SpO2

Elaborado por: El investigador

3.2.7. Diseño y construcción de la placa de circuito impreso

Para el diseño de la placa de circuito impreso (PCB) se utilizó el softaware de EasyEda, para el diseño como se visualiza en la figura 67, que constan únicamente de la capa inferior con color azul, donde cada pista tiene un grosor de 2.5 milímetros (mm).

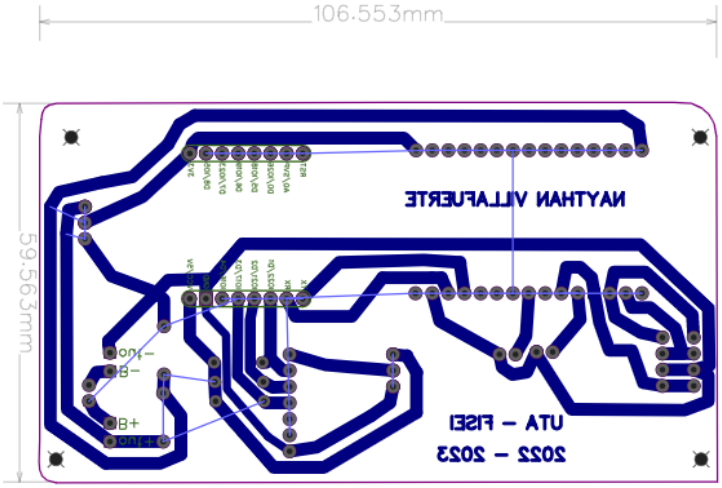


Figura 67: Diseño de la placa de circuito impreso en EasyEda

Elaborado por: El investigador

Además, esta placa fue diseñada con unas dimensiones de 106.6 x 59.6 mm, logrando así el mayor ahorro de espacio posible para tener un dispositivo electrónico de menor dimensión, la parte superior de la placa se puede visualizar en la figura 68.

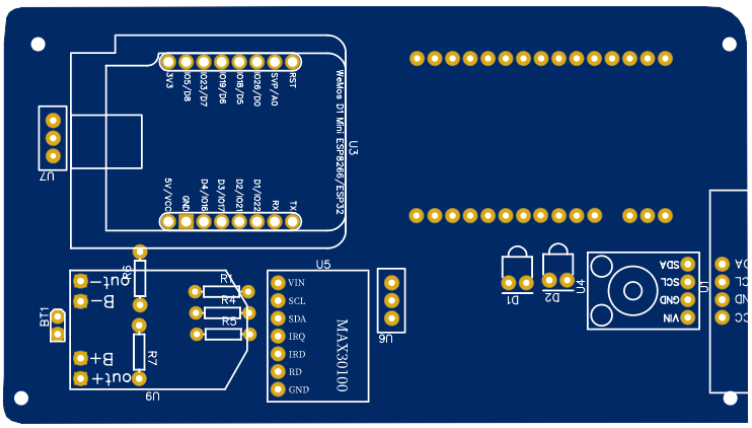


Figura 68: Vista 2D del circuito impreso en EasyEda

Elaborado por: El investigador

Para la impresión de la placa se empleó una máquina CNC, donde el resultado de la impresión se observa en la figura 69.

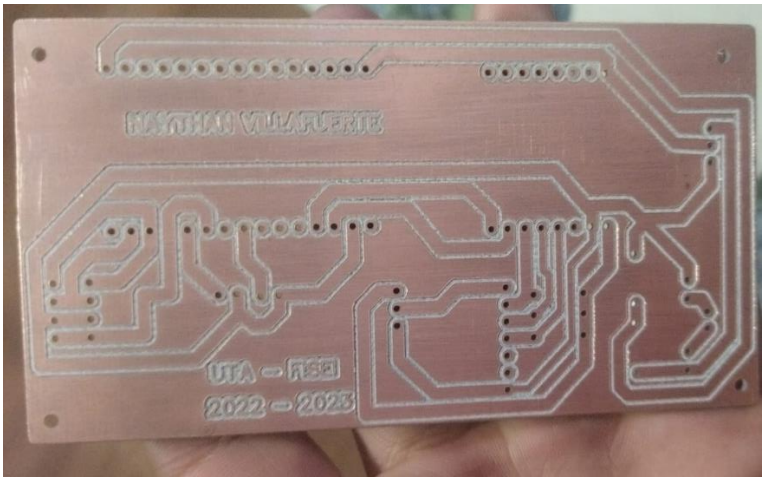


Figura 69: Circuito impreso en fisico

Elaborado por: El investigador

Implementación del circuito impreso

Para ensamblar la placa fue necesario visualizar el diagrama de la figura 30 y 31 que tienen todas las conexiones y componentes electrónicos pertinentes, con el fin de evitar conexiones erróneas que puedan causar daños irreversibles al momento de energizar el dispositivo de medición, una vez se realicen todas las soldaduras de los componentes el resultado se muestra en la figura 70.

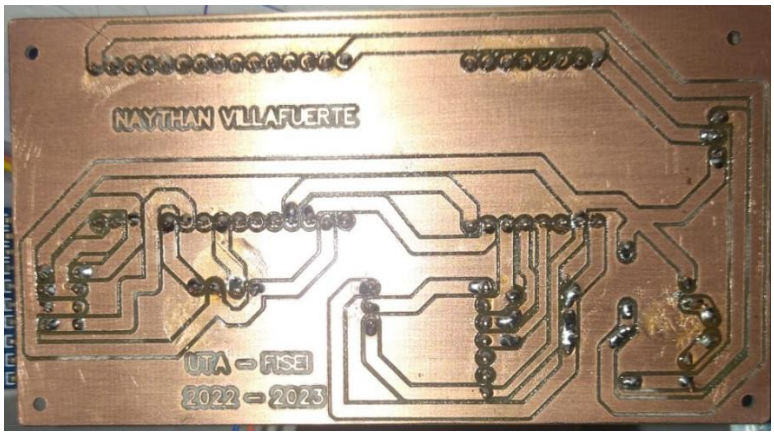


Figura 70: Soldaduras de la placa del dispositivo electrónico

Elaborado por: El investigador

Una vez se haya comprobado que las conexiones y soldaduras son correctas se conecta la alimentación del sistema electrónico para comprobar el funcionamiento, el resultado se puede visualizar en la figura 71.

Figura 71: Circuito implementado

Elaborado por: El investigador

3.2.8. Diseño y construcción de la carcasa de protección

La carcasa protectora del dispositivo fue diseñada en SolidWork, quedando como la figura 72 que conta de 7 orificios destinados para el led rojo y verde, el led rojo e infrarrojo del MAX30100, el sensor MLX90614, la pantalla Oled, el interruptor de encendido y apagado del dispositivo, y el interruptor de encendido del sensor MAX30100. Finalmente, el case cuenta con una tapa protectora para evitar golpes en la pantalla Oled en el caso de que se llegue a caer, esto hace que el diseño sea robusto y resistente para el uso diario.

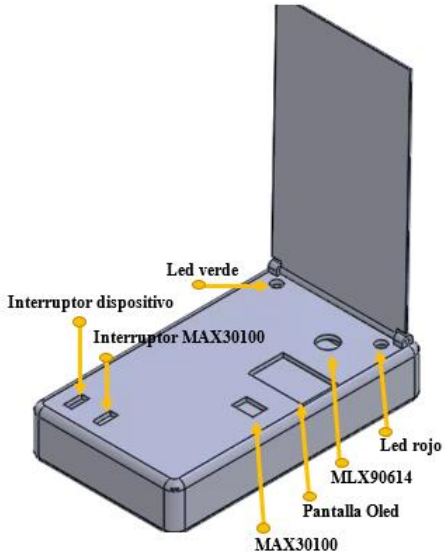


Figura 72: Diseño 3D de la carcasa protectora del dispositivo

Elaborado por: El investigador

Una vez ensamblado la placa dentro de la carcasa protectora se puede observar algo como la figura 73, que muestra la visualización correcta de los datos en la pantalla Oled.

Figura 73: Dispositivo de medición de signos vitales en funcionamiento

Elaborado por: El investigador

3.2.9. Selección del algoritmo de Inteligencia artificial (IA)

Para la selección del algoritmo de inteligencia artificial se debe trabajar de acuerdo con el ciclo de vida de IA que consiste en adquisición, filtrado, preparación de los datos para ser utilizados.

3.2.9.1. Adecuación del dataset

Para adecuar el dataset en un inicio se usa Excel con el objetivo de que todos los datos dentro de las columnas estén etiquetados y tengan relación entre sí. Por ende, el primer paso es buscar bases de datos que contengan los síntomas presentes en diagnósticos médicos de Covid19, Resfriado común y Rinitis alérgica.

Las bases de datos consultadas fueron:

- Base de datos proporcionada por la Dr. Verónica Córdova que colaboró en el desarrollo del proyecto.
- Base de datos investigadas dentro de la comunidad “Kaggle” que proporcionan varios datos de las enfermedades tratadas en el proyecto.
- Basa de datos proporcionadas por el grupo de investigación que forma parte el investigador.

El dataset unificado de las tres bases de datos utilizadas consta de 3 columnas sexo, síntomas y diagnóstico como se muestra en la figura 74. Sin embargo, la estructura de esta base de datos no es óptima para un algoritmo de clasificación de Inteligencia artificial ya que la columna “SINTOMAS” cuenta con distintos síntomas que no tienen tanta relación entre sí, además de no contar con palabras entendibles para un usuario puesto que las bases de datos consultadas cuentan con un lenguaje entendible para profesionales de la salud, de igual manera sucede con la columna “DIAGNOSTIO”.

SEXO	SINTOMAS	DIAGNOSTIO
femenino	tos que no moviliza secreciones, leve odinofagia	rinofaringitis(o resfriado comun)
femenino	CONGESTION NASAL, LEVE CEFALEA Y DOLOR OCUALAR	rinofaringitis(o resfriado comun)
femenino	CONGESTIONAL, DOLOR DE GARGANTA ACOMPAÑADA DE FIEBRE Y DOLOR ARTIULAR.	COVI19
femenino	RINORREA HIALINA, NO ALZA TERMICA	rinofaringitis(o resfriado comun)
MASCULINO	RINORREA HIALINA, RESEQUEDAD DE MUCOSA DE GARGANTA,TOS QUE NO MOVILIZA SECRECIONES	COVI19
femenino	ALZA TERMICA, RINORREA HIALINA	rinofaringitis(o resfriado comun)
femenino	ODINOFAGIA, RINORREA HIALINA	rinofaringitis(o resfriado comun)
MASCULINO	alza termica no cuantificada y cefalea, rinorrea	rinofaringitis(o resfriado comun)
FEMENINA	cefalea,se acompaña de leve odinofagia, rinorrea hialina, malestar general, tos que no moviliza secreciones	rinofaringitis(o resfriado comun)

Figura 74: Base de datos antes de procesar

Elaborado por: El investigador

Para procesar la base de datos se debe considerar que el léxico utilizado debe ser entendible por una persona promedio que no tenga conocimientos técnicos en el área de la salud. Por consiguiente, se cambia el vocabulario utilizado por profesionales de la salud a uno más entendible como muestra la tabla 28. Este cambio se realizó conjunto con la doctora que colaboro en el proyecto de investigación ya que los síntomas son los datos más importantes al momento de predecir qué tipo de enfermedad posee el usuario para que busque una atención rápida.

Tabla 28: Cambio de vocabulario técnico medico a vocabulario común

Vocabulario técnico médico	Vocabulario común
Rinorrea	Congestión nasal
Odinofagia	Dolor de garganta
Cansancio	Malestar General
Decaimiento	Malestar General
Fatiga	Malestar general
Mareo	Dolor de cabeza – leve
Dolor de costillas	Dolor de pecho
Anorexia	Perdida del apetito
Garganta reseca	Dolor de garganta - fuerte
Epigastralgia	Dolor de barriga
Artralgias	Dolor articular
Migraña	Dolor de cabeza - fuerte
Tos que no moviliza secreciones	Tos seca
Tos que moviliza secreciones	Tos con flema
Rinorrea hialina	Congestión nasal
Alza térmica	Fiebre
Alza térmica no cuantificada	Fiebre
Cefalea	Dolor de cabeza
Resequedad de mucosa de garganta	Dolor de garganta - fuerte
Astenia	Malestar general
Cefalea frontal	Dolor de cabeza – moderado
Rinofaringitis	RESFRIADO

Elaborado por: El investigador

Una vez cambiado el vocabulario a uno más entendible para los usuarios también fue necesario separar cada síntoma para que pueda ser interpretado de mejor manera al momento de adaptar el dataset al algoritmo de inteligencia artificial. Adicionalmente, se agregó una columna con el ID de los pacientes para identificar con cuantos casos positivos de COVID19, Resfriado común, Rinitis alérgica existen en la base de datos. El dataset cuenta con 24 columnas de síntomas, una columna del diagnóstico dado por los médicos, una columna del género del paciente y un ID del paciente, como se observar en el Anexo 3.

Para manipular el dataset se consideró las categorías que tendrán cada columna en este caso la mayoría de los síntomas se considera la presencia de la sintomatología

mediante un “si” caso contrario se le agrega la palabra “no”. Sin embargo, existen tres columnas que tienen las categorías “leve”, “moderado” y “fuerte”, mientras que la columna “Tos” tiene las categorías de “seca” y “con flema”. Todas las consideraciones que se muestran en la tabla 29 fueron realizadas en conjunto con la doctora Verónica Córdova.

Tabla 29: Síntomas con las categorías consideradas

Número	Síntomas	Categorías
1	Secreción nasal	si y no
2	Congestión nasal	si y no
3	Dolor de garganta	leve, moderado y fuerte
4	Lagrimeo	si y no
5	Tos	seca y con flema
6	Estornudos	si y no
7	Sensación de ahogo	si y no
8	Fiebre	si y no
9	Dolor de articulaciones	leve, moderado y fuerte
10	Malestar general	si y no
11	Dolor de cabeza	leve, moderado y fuerte
12	Picazón nasal	si y no
13	Hinchazón de los ojos	si y no
14	Ronquidos	si y no
15	Dolor muscular	si y no
16	Pedida de la voz	si y no
17	Dolor de los ojos	si y no
18	Diarrea	si y no
19	Nauseas	si y no
20	Dolor de barriga	si y no
21	Dolor de pecho	si y no
22	Perdida del apetito	si y no
23	Escalofríos	si y no

Elaborado por: El investigador

Tras adaptar el dataset utilizando Excel a lo necesitado para el proyecto, este óptimo para ser utilizado en la programación de Python, en la figura 75 se realiza una exploración de la base de datos utilizada donde se observa que no hay presencia de valores nulos en las columnas, los datos dentro de las columnas de síntomas son del

tipo object por lo que es necesario transformarlos al tipo int (entero) para ser utilizado por los algoritmos de clasificación.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6006 entries, 0 to 6005
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Paciente          6006 non-null   int64
 1   Genero            6006 non-null   object
 2   Secrecion Nasal   547 non-null    object
 3   Congestion Nasal  1694 non-null   object
 4   Dolor garganta    2887 non-null   object
 5   Lagrimeo          144 non-null    object
 6   Tos               2808 non-null   object
 7   Estornudos        486 non-null    object
 8   Sensacion ahogo   291 non-null    object
 9   Fiebre            1036 non-null   object
 10  dolor articular   1535 non-null   object
 11  Malestar general  943 non-null    object
 12  Dolor cabeza      1646 non-null   object
 13  Picazon nasal     1293 non-null   object
 14  Hinchazon ojos    265 non-null    object
 15  Ronquidos         85 non-null     object
 16  Dolor muscular    281 non-null    object
 17  Perdida voz       271 non-null    object
 18  Dolor ojos        252 non-null    object
 19  Diarrea           487 non-null    object
 20  Nauseas           457 non-null    object
 21  Dolor barriga     321 non-null    object
 22  Dolor pecho       524 non-null    object
 23  Perdida apetito   404 non-null    object
 24  Escalofrios       328 non-null    object
 25  Diagnostico       6006 non-null   object
dtypes: int64(1), object(25)
memory usage: 1.2+ MB
```

Figura 75: Exploración del dataset utilizando Python

Elaborado por: El investigador

3.2.9.2. Acondicionamiento del dataframe para la IA

Para acondicionar el dataset se hizo uso de la librería de pandas la cual permite transformar un archivo ".csv" (es el formato utilizado para la base de datos) en un dataframe que es el ocupado para entrenar a los algoritmos de clasificación de Machine learning. El formato del datadrame prácticamente es similar a una tabla ya que cuenta con filas y columnas que se pueden modificar mediante código.

Para leer el archivo ".csv" y transformarlo a dataframe se utilizó la herramienta "pandas.read" en donde se especifica que archivo se va a ocupar, que separadores usa el archivo y que lenguaje de programación se utiliza, que en el caso del proyecto se hizo uso de Python.

Eliminación de registros innecesarios y nulos

Como se vio en la figura 76 la columna "ID pacientes" no es de utilidad ya que solo es usada para saber cuántos casos positivos de las 3 enfermedades tratadas se tienen, lo que convierte los datos de la columna en información sin relevancia para el entrenamiento de la inteligencia artificial de este proyecto.

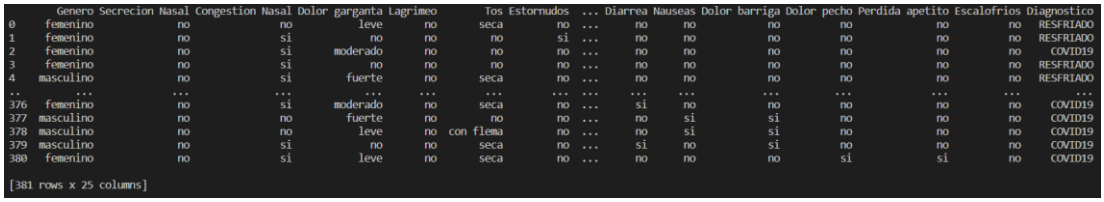
```
#Descartar registros innecesarios
cols_to_drop = ['Paciente']
datos = df.drop(cols_to_drop, axis=1)

#Rellenar datos faltantes con la palabra "no"
datos = datos.fillna('no')
print(datos)
```

Figura 76: Eliminación de columnas innecesarias

Elaborado por: El investigador

La librería panda proporciona herramientas para eliminar registros innecesarios dentro del dataframe en la figura 77 se emplea la herramienta “drop” que utiliza la etiqueta de la columna y el “axis” que representa si se desea eliminar la columna representada con “1” o la fila representada con “0”. Una vez eliminado los datos innecesarios fue preciso rellenar los datos faltantes dentro de las columnas de síntomas, para rellenar los datos del dataframe se utilizó el string “no” en todas las columnas. Tras realizar estos cambios se puede se puede observar algo como la figura 79.



	Genero	Secrecion Nasal	Congestion Nasal	Dolor garganta	Lagrimeo	Tos	Estornudos	...	Diarrea	Nauseas	Dolor barriga	Dolor pecho	Perdida apetito	Escalofrios	Diagnostico
0	femenino	no	no	leve	no	seca	no	...	no	no	no	no	no	no	RESFRIADO
1	femenino	no	si	no	no	no	si	...	no	no	no	no	no	no	RESFRIADO
2	femenino	no	si	moderado	no	no	no	...	no	no	no	no	no	no	COVID19
3	femenino	no	si	no	no	no	no	...	no	no	no	no	no	no	RESFRIADO
4	masculino	no	si	fuerte	no	seca	no	...	no	no	no	no	no	no	RESFRIADO
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
376	femenino	no	si	moderado	no	seca	no	...	si	no	no	no	no	no	COVID19
377	masculino	no	no	fuerte	no	no	no	...	no	si	si	no	no	no	COVID19
378	masculino	no	no	leve	no	con flema	no	...	no	si	si	no	no	no	COVID19
379	masculino	no	si	no	no	seca	no	...	si	no	si	no	no	no	COVID19
380	femenino	no	si	leve	no	seca	no	...	no	no	no	si	si	no	COVID19

[381 rows x 25 columns]

Figura 77: Dataframe rellenado y eliminado la información innecesaria

Elaborado por: El investigador

Balanceo de datos

Como se observa en la figura 78 los casos de covid19, resfriado común y rinitis alérgica no son iguales teniendo que los casos registrados de Covid19 son casi el doble que los casos de rinitis alérgica, esto causara que el modelo de inteligencia artificial tienda a predecir más casos de covid19 y resfriado común teniendo así un margen de error mucho más grande al momento de evaluar el modelo de machine learning.

```
Datos sin balancear-----------------------
Diagnostico
COVID19      2427
RESFRIADO    2286
RINITIS      1293
dtype: int64
```

Figura 78: Datos sin balancear

Elaborado por: El investigador

Para reducir este error se usa el balanceo de datos, en consecuencia se empleó la librería “imblearn.over_sampling” que permite igualar las clases que se usaran para la predicción del modelo como se muestra en la figura 79. Antes de balancear el dataframe es necesario separar las variables independientes y dependientes en este caso la variable dependiente es “target” ya que dependiendo de las columnas de síntomas se predice una clase u otra. Por otro lado, “data” es la variable independiente ya que esta no cambiara con respecto a otros datos por lo cual aquí se almacenan todas las columnas de síntomas dentro del dataframe.

```
#--------Separar Variable dependiente y independiente-----------------
data = datos
target = df['Diagnostico']

ros = RandomOverSampler()
bal = SMOTE()
dataros, targetros = ros.fit_resample(data, target)
```

Figura 79: Balanceo de datos del dataframe

Elaborado por: El investigador

En la figura 80 se observa cual es el proceso antes mencionado para el balanceo de datos que mediante el método “RandomOverSampler” se duplica los datos de cada columna de forma aleatoria hasta igualar en cantidad a la clase mayoritaria en casos positivos.

```
Datos balanceados  ----------------------
Diagnostico
COVID19      2427
RESFRIADO    2427
RINITIS      2427
dtype: int64
```

Figura 80: Datos balanceados

Elaborado por: El investigador

A todo esto, la clase a la cual se igualaron todas, es el “COVID19” con un total de 2427 datos. Por último, se crea dos nuevas variables denominadas dataros y targetros que son la variable dependiente e independiente respectivamente, estas almacenan los datos balanceados del dataframe.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7281 entries, 0 to 7280
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Genero            7281 non-null   object
 1   Secrecion Nasal   7281 non-null   object
 2   Congestion Nasal  7281 non-null   object
 3   Dolor garganta    7281 non-null   object
 4   Lagrimeo          7281 non-null   object
 5   Tos               7281 non-null   object
 6   Estornudos        7281 non-null   object
 7   Sensacion ahogo   7281 non-null   object
 8   Fiebre            7281 non-null   object
 9   dolor articular   7281 non-null   object
 10  Malestar general  7281 non-null   object
 11  Dolor cabeza      7281 non-null   object
 12  Picazon nasal     7281 non-null   object
 13  Hinchazon ojos    7281 non-null   object
 14  Ronquidos         7281 non-null   object
 15  Dolor muscular    7281 non-null   object
 16  Perdida voz       7281 non-null   object
 17  Dolor ojos        7281 non-null   object
 18  Diarrea           7281 non-null   object
 19  Nauseas           7281 non-null   object
 20  Dolor barriga     7281 non-null   object
 21  Dolor pecho       7281 non-null   object
 22  Perdida apetito   7281 non-null   object
 23  Escalofrios       7281 non-null   object
 24  Diagnostico       7281 non-null   object
dtypes: object(25)
memory usage: 1.4+ MB
```

Figura 81: Exploración del dataframe balanceado

Elaborado por: El investigador

Una vez balanceados los datos se puede observar en la figura 80 como las 3 clases tienen la misma cantidad de datos y si se realiza una exploración del dataframe balanceado se tiene algo como la figura 81que cuenta con un total de 7281 datos con los que se entrenó a la inteligencia artificial.

Transformación de valores del tipo string (cadenas) a int (entero)

Con el objetivo de transformar los datos del dataframe se creó una función que itere en cada una de las celdas y cambie los strings a enteros como se visualiza en la figura 82, se aplicó funciones similares para el resto de las columnas con los síntomas.

```python
def cambio0str(x):
    if x == 'femenino':
        return 1
    else:
        return 0
#Genero
```

Figura 82: Función para transformar los datos de la columna genero

Elaborado por: El investigador

Para ello es necesario crear una equivalencia como muestra la tabla 30 donde se cambia la variable string a una variable numérica ya que los modelos de clasificación que se probaron son algoritmos matemáticos por ende todas las columnas dentro de la variable independiente denominada como “dataros” son cambiados.

Tabla 30: Equivalencia para transformar de "str" a "int"

Variable en string (Cadena)	Variable en int (Entero)
femenino	1
masculino	0
si	1
no	0
leve	1
moderado	2
Fuerte	3
no	0
seca	1
con flema	0

Elaborado por: El investigador

Al finalizar con la transformación se realiza una exploración del dataframe contemplado en la figura 83, donde se observa que todas las columnas de síntomas fueron transformadas a enteros, a excepción de la columna diagnostico ya que esta contiene las clases que se predecirán (variable dependiente).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7281 entries, 0 to 7280
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Diagnostico         7281 non-null   object
 1   Genero N            7281 non-null   int64 
 2   Secrecion Nasal N   7281 non-null   int64 
 3   Congestion Nasal N  7281 non-null   int64 
 4   Lagrimeo N          7281 non-null   int64 
 5   Estornudos N        7281 non-null   int64 
 6   Sensacion ahogo N   7281 non-null   int64 
 7   Fiebre N            7281 non-null   int64 
 8   Malestar general N  7281 non-null   int64 
 9   Picazon nasal N     7281 non-null   int64 
 10  Hinchazon ojos N    7281 non-null   int64 
 11  Ronquidos N         7281 non-null   int64 
 12  Dolor muscular N    7281 non-null   int64 
 13  Perdida voz N       7281 non-null   int64 
 14  Dolor ojos N        7281 non-null   int64 
 15  Diarrea N           7281 non-null   int64 
 16  Nauseas N           7281 non-null   int64 
 17  Dolor barriga N     7281 non-null   int64 
 18  Dolor pecho N       7281 non-null   int64 
 19  Perdida apetito N   7281 non-null   int64 
 20  Escalofrios N       7281 non-null   int64 
 21  Dolor garganta N    7281 non-null   int64 
 22  Dolor articular N   7281 non-null   int64 
 23  Dolor cabeza N      7281 non-null   int64 
 24  Tos N               7281 non-null   int64 
dtypes: int64(24), object(1)
memory usage: 1.4+ MB
```

Figura 83: Columnas de síntomas transformados a enteros

Elaborado por: El investigador

3.2.9.3. Construcción y evaluación de algoritmos de IA

Para la creación del modelo de clasificación de machine learning fue necesario separar los datos de entrenamiento y testeo con el objetivo de evaluar el comportamiento de los algoritmos.

Separación de datos de entrenamiento

En la figura 84 se observa la separación del dataframe en 2, donde "x" es la variable independiente, mientras que "y" es la variable dependiente por ende se le asigna la columna "Diagnostico" que contiene las tres enfermedades que se tratan, estas enfermedades corresponderán a las 3 clases con las que se trabajara en los modelos de predicción.

```
#-------------------Datos de entrenamiento-----------------------------
# Separacion del dataframe en x and y
#x es la variable independiente
X = dataros.drop(columns='Diagnostico')
print(x)
#y es la variable dependiente
y = dataros.Diagnostico
print(y)
```

Figura 84: Separación de la variable independiente y variable dependiente

Elaborado por: El investigador

Al realizar una exploración de la variable independiente como indica la figura 85, se puede notar que esta corresponde a todas las columnas de síntomas que serán ocupadas para predecir la enfermedad.

	Genero N	Secrecion Nasal N	Congestion Nasal N	Lagrimeo N	Estornudos N	Sensacion ahogo N	...	Perdida apetito N	Escalofrios N	Dolor garganta N	Dolor articular N	Dolor cabeza N	Tos N
0	1	0	0	0	0	0	...	0	0	1	0	0	0
1	1	0	1	0	1	0	...	0	0	0	0	1	0
2	1	0	1	0	0	0	...	0	0	2	2	0	0
3	1	0	1	0	0	0	...	0	0	0	0	0	0
4	0	0	1	0	0	0	...	0	0	3	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7276	1	0	1	0	0	0	...	0	0	0	0	0	0
7277	0	0	1	0	0	0	...	0	0	0	0	0	0
7278	0	1	0	0	1	0	...	0	0	0	0	0	0
7279	0	0	1	0	0	0	...	0	0	0	0	0	0
7280	0	0	1	0	0	0	...	0	0	0	0	0	0

Figura 85: Variable independiente

Elaborado por: El investigador

Al realizar una exploración de la variable dependiente como indica la figura 86, se puede notar que esta corresponde a la columna que tiene los diagnósticos médicos de covid19, resfriado común y rinitis alérgica.

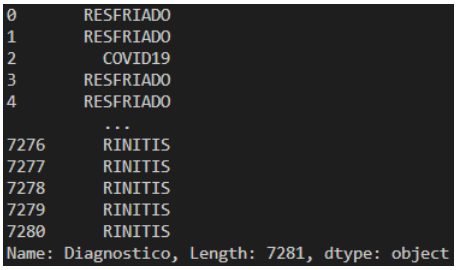


Figura 86: Variable dependiente

Elaborado por: El investigador

Una vez se tiene separado las variables es necesario destinar la cantidad de datos que son utilizados para entrenamiento y que cantidad de datos son utilizados para testear el modelo. Por lo cual se importó la herramienta "train_test_split" que tiene como argumentos las variables dependiente e independiente, la cantidad de datos de prueba que se utilizaran para el testeo y el estado aleatorio. Cabe recalcar que tras las pruebas se determinó que el valor óptimo para dividir los datos de prueba es una relación de 70% de datos de entrenamiento, mientras que el 30% es destinado al testeo de los modelos como se visualiza en la figura 87.

```
#-------------------Division datos de entrenamiento 70% /prueba 30%-------------------
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

Figura 87: División de datos de entrenamiento

Elaborado por: El investigador

Construcción de los algoritmos de IA

Realizado la división de los datos de entrenamiento y testo, solo resta probar los modelos de clasificación de machine learning en Python los cuales son:

- Árbol de decisión

El modelo de árbol de decisión se utiliza empleando la función "tree.DecisionTreeClassifier" con una profundidad de 5 nodos como se visualiza en la figura 88, en caso de no definir el número de nodos del árbol este se expandirá hasta tener el mayor porcentaje de predicción y esto puede llevar un tiempo indefinido.

```
# ARBOL DE DECISIONES-----------------------------------------------------
from sklearn import tree, ensemble
dt = tree.DecisionTreeClassifier(max_depth=5)
dt.fit(X_train, y_train)
```

Figura 88: Modelo "Desition Tree"

Elaborado por: El investigador

- Bosque aleatorio

El modelo bosque aleatorio se utiliza empleando la función "ensemble.RandomForestClassifier" que tras la investigación se determinó un total de 20 árboles dentro de random forest como muestra la figura 89, ya que al aumentar más arboles el procesamiento requerido por la maquina será mayor.

```
# RANDOM FOREST-----------------------------------------------------
print("RANDOM FOREST")
rf = ensemble.RandomForestClassifier(n_estimators=20)
rf.fit(X_train, y_train)
```

Figura 89: Modelo "Random Forest"

Elaborado por: El investigador

- Aumento de gradiente

El modelo de aumento de gradiente se utiliza empleando la función "ensemble.GradientBoostingClassifier" que tras la investigación se determinó un total de aumento de la gradiente de 40 como se visualiza en la figura 90, ya que al aumentar la gradiente a valores muy altos el procesamiento requerido por la maquina será mayor llegando a demorar meses.

```
# GRADIENT BOOSTING-----------------------------------------------------
print("GRADIENT BOOSTING")
gb = ensemble.GradientBoostingClassifier(n_estimators=40)
gb.fit(X_train, y_train)
```

Figura 90: Modelo "Gradient Boosting"

Elaborado por: El investigador

- Naive Bayes

El modelo de Naive Bayes se utiliza empleando la función “GaussianNB”, como se muestra en la figura 91.

```
# NAIVE BAYES-----------------------------------------------------
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)
```

Figura 91: Modelo "Naive Bayes"

Elaborado por: El investigador

- Vecino más cercano

El modelo de K-Nearest Neighbor se utiliza empleando la función “KNeighborsClassifier”, donde tras la experimentación se determinó un total de 3 vecinos para las consultas en la predicción como se visualiza en la figura 92.

```
# K-NEAREST NEIGHBOR----------------------------------------------------
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Figura 92: Modelo "K-Nearest Neighbor"

Elaborado por: El investigador

- Regresión logística

El modelo de regresión logística se utiliza empleando la función “LogisticRegresion”, como se muestra en la figura 93.

```
# LOGISTIC REGRESSION---------------------------------------------------
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Figura 93: Modelo "Logistic Regression"

Elaborado por: El investigador

- SVC (Support Vector Machine)

El modelo de Support Vector Machine se utiliza empleando la función “svm.SVC” como se visualiza en la figura 94, donde se decidió dejar la probabilidad por defecto

ya que internamente esta función utiliza métodos de validación cruzada para generar un modelo.

```
# SUPPORT VECTOR MACHINE-----------------------------------------------------
from sklearn import svm
vm = svm.SVC(probability=False)
vm.fit(X_train, y_train)
```

Figura 94: Modelo "Support Vector Machine"

Elaborado por: El investigador

Evaluación de los algoritmos de Machine Learning

Se utilizan 3 métricas para la evaluación de los modelos de clasificación, estas son; “F1 Score (F1)”, “Recall (R)”, “Precisión (P)”. Estas métricas son obtenidas gracias a la matriz de confusión generada de cada modelo de forma similar a lo visto en la sección 1.3.7.7 pero con la diferencia que al tener 3 clases para la predicción la matriz generada en Python tiene una dimensión de 3x3. Por ende, la forma de obtener Verdaderos positivos, Verdaderos negativos, Falsos positivos, Falsos negativos se muestra a continuación en la figura 95.

		VALORES ACTUALES		
		COVID19	RESFRIADO	RINITIS
VALORES PREDECIDOS	COVID19	C1	C2	C3
	RESFRIADO	C4	C5	C6
	RINITIS	C7	C8	C9

C = Celda

Figura 95: Matriz de confusión utilizada para evaluar los modelos

Elaborado por: El investigador

El Verdaderos Positivos (VP), Verdaderos Negativos (VN), Falsos positivos (FP), Falsos negativos (FN) de cada clase se calcula sumando los valores de las celdas como muestra las siguientes ecuaciones:

Clase 1→ COVID19

$$VP = C_1 \quad \text{Ecuación (11)}$$

$$FP = C_2 + C_3 \quad \text{Ecuación (12)}$$

$$VN = C_5 + C_6 + C_8 + C_9 \quad \text{Ecuación (13)}$$

$$FN = C_4 + C_9 \quad \text{Ecuación (14)}$$

Clase 2→ RESFRIADO

$$VP = C_5 \quad \text{Ecuación (15)}$$

$$FP = C_4 + C_6 \quad \text{Ecuación (16)}$$

$$VN = C_1 + C_3 + C_7 + C_9 \quad \text{Ecuación (17)}$$

$$FN = C_2 + C_8 \quad \text{Ecuación (18)}$$

Clase 3→ RINITIS

$$VP = C_9 \quad \text{Ecuación (19)}$$

$$FP = C_7 + C_8 \quad \text{Ecuación (20)}$$

$$VN = C_1 + C_2 + C_4 + C_5 \quad \text{Ecuación (21)}$$

$$FN = C_4 + C_6 \quad \text{Ecuación (22)}$$

La matriz de confusión generada en Python se realiza mediante la librería "confusión_matrix", esta librería usa como argumentos la variable de testeo y de predicción. A continuación, en la figura 96 se muestran las matrices de confusión.

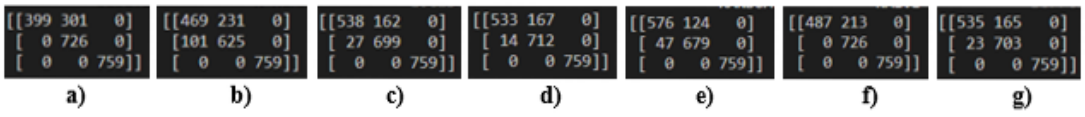


Figura 96: Matriz de confusión--> a) Desition Tree, b) K-Nearest Neighbor, c) Logistic Regresion, d) Gradiente Boosting, e) Random Forest, f) Naïve Bayes, g) Support Vector Machine

Elaborado por: El investigador

En la tabla 31se indican los valores VP, FP, VN, FN calculados usando las ecuaciones desde la (11) hasta la (22), además cuenta con la precisión de entrenamiento y testeo empleando validación cruzada. También se muestran las métricas de cada modelo empleando la librería "clasification_report" que permite obtener de manera automática

las principales métricas de clasificación, juntamente con un reporte de los promedios "macro avg" de la media no ponderada por clase, "weighted avg" de la media ponderada por cada clase que se observan en el Anexo 7.

Tabla 31: Desempeño de los modelos

DESITION TREE									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
399	301	1485	759	COVID19	1,00	0,57	0,73	0,90	0,90
726	0	1158	301	RESFRIADO	0,71	1,00	0,83	0,86	0,86
759	0	1426	0	RINITIS	1,00	1,00	1,00	0,85	0,86
Precisión de entrenamiento:					0,85989102345013				
Precisión de testeo:					0,86222358527362				
K-NEAREST NEIGHTBOR									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
469	231	1384	860	COVID19	0,82	0,67	0,74	0,85	0,85
625	101	1228	231	RESFRIADO	0,73	0,86	0,79	0,84	0,85
759	0	1426	101	RINITIS	1,00	1,00	1,00	0,84	0,85
Precisión de entrenamiento:					0,85302123492121				
Precisión de testeo:					0,82836579287197				
LOGISTIC REGRESION									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
538	162	1458	786	COVID19	0,95	0,77	0,85	0,92	0,92
699	27	1297	162	RESFRIADO	0,81	0,96	0,88	0,91	0,91
759	0	1426	27	RINITIS	1,00	1,00	1,00	0,91	0,91
Precisión de entrenamiento:					0,91012473445083				
Precisión de testeo:					0,90709085153530				
GRADIENT BOOSTING									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
538	162	1458	786	COVID19	0,97	0,76	0,85	0,93	0,93
699	27	1297	162	RESFRIADO	0,81	0,98	0,89	0,91	0,92
759	0	1426	27	RINITIS	1,00	1,00	1,00	0,91	0,92
Precisión de entrenamiento:					0,92209486954265				
Precisión de testeo:					0,90800660245105				
RANDOM FOREST									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
576	124	1438	806	COVID19	0,92	0,82	0,87	0,92	0,92
679	47	1335	124	RESFRIADO	0,85	0,94	0,89	0,92	0,92
759	0	1426	47	RINITIS	1,00	1,00	1,00	0,92	0,92
Precisión de entrenamiento:					0,91502923380644				
Precisión de testeo:					0,90708896727415				
NAIVE BAYES									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
487	213	1485	759	COVID19	1,00	0,70	0,87	0,92	0,92

726	0	1246	213	RESFRIADO	0,77	1,00	0,87	0,90	0,90
759	0	1426	0	RINITIS	1,00	1,00	1,00	0,90	0,90
Precisión de entrenamiento:					0,89972530551587				
Precisión de testeo:					0,90251398121768				

SUPPORT VECTOR MACHINE									
VP	**FP**	**VN**	**FN**	**Clase**	**P**	**R**	**F1**	**Macro avg**	**Weighted avg**
535	165	1462	782	COVID19	0,96	0,76	0,85	0,92	0,92
703	23	1294	165	RESFRIADO	0,81	0,97	0,88	0,91	0,91
759	0	1426	23	RINITIS	1,00	1,00	1,00	0,91	0,91
Precisión de entrenamiento:					0,91424642269766				
Precisión de testeo:					0,91257907615932				

Elaborado por: El investigador

3.2.9.4. Selección del algoritmo de IA

En base al análisis de las métricas de cada modelo evaluado expuestas en la tabla 34 se determinó que el mejor algoritmo de diagnóstico médico de covid19, resfriado común y rinitis alérgica, es el modelo de Support Vector Machine (Máquina de Vectores de Soporte) ya que cuenta con métricas más equilibradas entre sí, teniendo recall de 0.76 y 0.96 de precisión para la clase “COVID19”, para la clase “RESFRIADO” se tiene un recall de 0.81 y una precisión de 0.97, y para la clase “RINITIS” arroja un recall de 1 con una precisión de 1. Finalmente, la precisión de entrenamiento con 0.914 y la precisión de testeo con 0.913 conformar los valores más altos y estables entre sí, debido a que, si la precisión de entrenamiento es considerablemente más grande que la precisión de testeo, el modelo de IA no está aprendiendo si no memorizando lo cual lleva a errores al momento de probarlo con datos nuevos que nunca haya visto la inteligencia artificial. Estas características destacaron al algoritmo de Suppoort Vector Machine sobre el resto y fue el escogido para incluirlo dentro de la aplicación Web.

3.2.9.5. Creación del modelo para el diagnóstico médico

En la figura 97 se importó la librería “joblib” que es utilizada para serializar la estructura de objetos en Python que tienen un tamaño grande, creando una secuencia de bytes escritos en un archivo de extensión “. pkl”. Fue necesario crear un archivo independiente para evitar realizar una y otra vez los pasos antes expuesto cada vez que un usuario realice una consulta al asistente, porque esto puede causar que se usen más recursos y reducir la rapidez de respuesta del servidor Web.

```
# Guardar el modelo-----------------------------------------------------------

import joblib

joblib.dump(vm, 'modelo_diagnostico.pkl')

print("----------MODELO GUARDADO----------")
```

Figura 97: Creación del modelo de diagnóstico médico

Elaborado por: El investigador

Una vez se corra el programa se observa que en la carpeta donde esta alojada el script para la creación del modelo se agregó un archivo llamado “modelo_diagnostico.pkl” como se muestra en la figura 98.

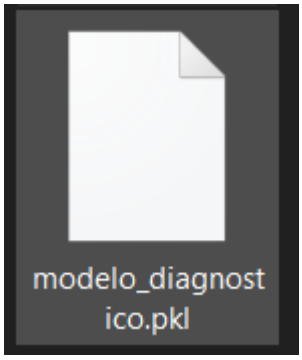


Figura 98: Modelo de diagnóstico médico creado

Elaborado por: El investigador

3.2.9.6. Importación del modelo para el diagnóstico médico

Dentro de la aplicación Web se utilizó la función “joblib.load” para llamar al archivo generado del modelo de inteligencia artificial. Esta función utiliza como argumento la dirección de alojamiento del archivo pkl, en este caso como muestra la figura 99 no es necesario introducir la dirección ya que se encuentran dentro de la misma carpeta.

```
# ------------------------------------------------------------------------

nb = joblib.load('modelo_diagnostico.pkl')
```

Figura 99: Importación del modelo para el diagnóstico médico

Elaborado por: El investigador

3.2.10. Desarrollo de la aplicación Web

El desarrollo de la aplicación web se realiza en base al diagrama de secuencias que se visualiza en la figura 100, empezando desde el inicio de sesión del usuario donde por medio de una máquina virtual alojada en el servidor de Google Cloud se comprueba si las credenciales ingresadas se encuentran registradas en la base de datos para permitir que el usuario continue a la pestaña de sensorización donde se mostraran sus signos vitales. Una vez el usuario este en la interfaz de sensorización puede visualizar mediane gráficas y cajas de texto los valores que registran los sensores del dispositivo de medición del triaje de los signos vitales. Después de la interfaz de sensores se ingresa al asistente virtual (chatbot) donde se le pide al paciente que ingrese su sintomatología que es almacenada en la base de datos para predecir su enfermedad mediante el modelo de IA creado. Finalmente, el usuario puede acceder a la venta del historial médico de la consulta.

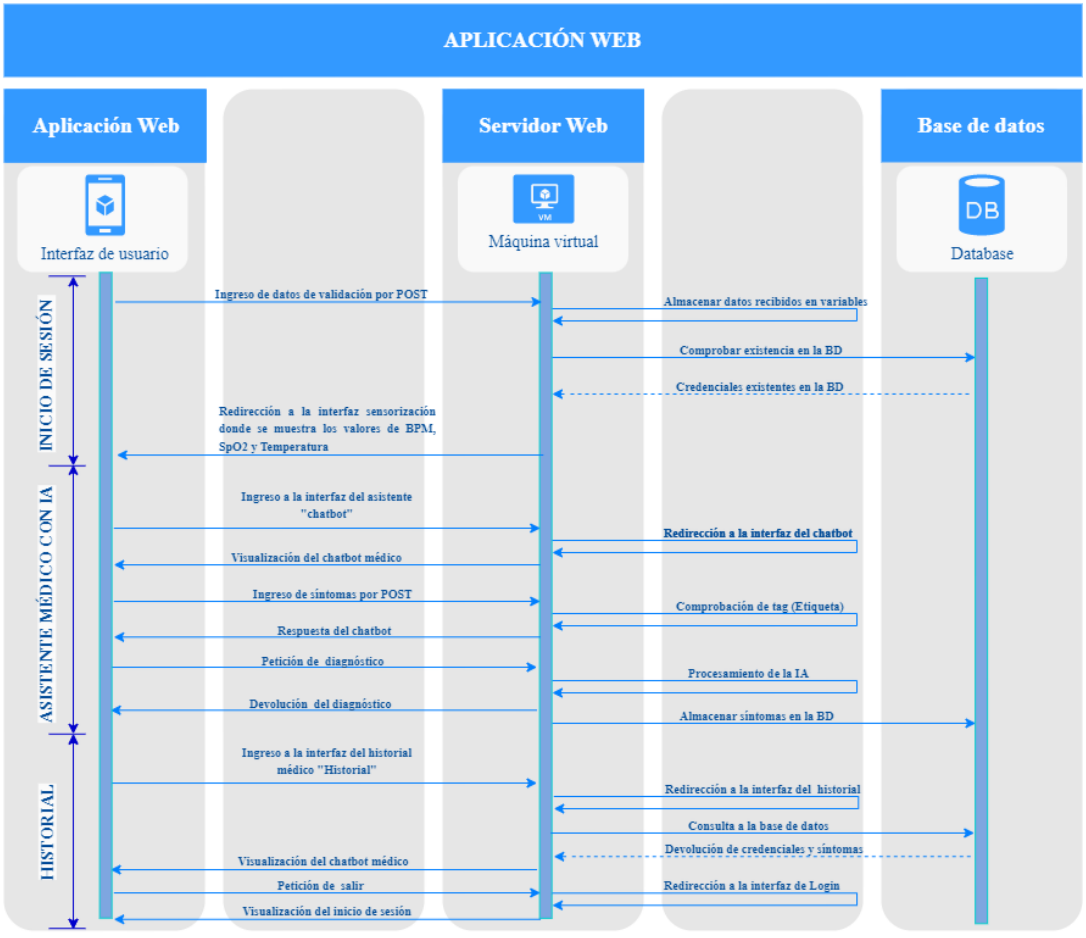


Figura 100: Diagrama de secuencias de la aplicación Web

Elaborado por: El investigador

Arquitectura de la aplicación Web

Para la arquitectura se consideró los componentes detallados en la figura 101 que consta de un Front-End que está del lado del usuario, haciendo referencia a la programación utilizada para el diseño de la aplicación utilizando lenguajes como HTML, CSS y JAVASCRIPT que hacen a la aplicación Web dinámica. Por otra parte, el Back-End corresponde a toda la programación que se encarga de interactuar internamente con el servidor; es decir, aquí se encuentran consultas a la base de datos, peticiones de cambio de ruta dentro de la aplicación Web, detección de etiqueta para respuestas del chatbot, registro en la base de datos y procesamiento de síntomas para la predicción de la enfermedad dentro de las tres clases previstas (COVD19, RESFRIADO y RINITIS).

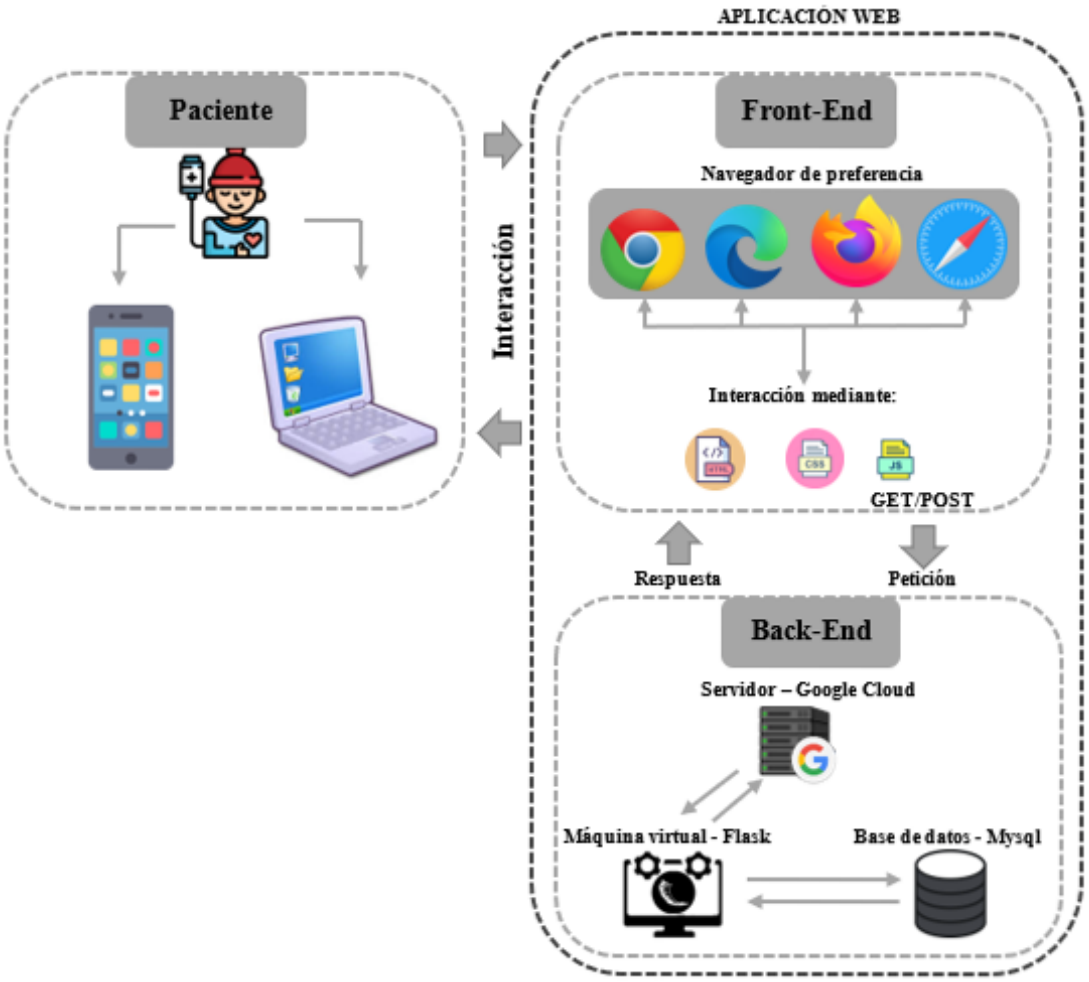


Figura 101: Arquitectura de la aplicación Web

Elaborado por: El investigador

3.2.10.1. Creación de la base de datos

Para almacenar los síntomas que ingrese el usuario se utilizó una base de datos MySQL de acuerdo con lo visto en la sección 3.2.3.3, esta se encuentra alojada dentro del servidor de Google Cloud. La base de datos contiene dos tablas donde la primera tabla denominada “síntomas” se utiliza para almacenar la sintomatología correspondiente a las enfermedades, para almacenar el diagnóstico generado por el algoritmo de inteligencia artificial y también es usada para las consultas hechas a fin de generar el historial médico de todas las consultas realizadas por el usuario. Adicionalmente, se creó otra tabla denominada “user” que se utiliza para almacenar las credenciales de cada usuario registrado en la aplicación Web, estas credenciales almacenadas sirven para comprobar si el usuario existe o la clave ingresada es incorrecta.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	ID_Paciente	int(11)			No	*Ninguna*		AUTO_INCREMENT
☐	2	Fecha	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()
☐	3	Nombre	varchar(70)	utf8mb4_general_ci		No	*Ninguna*		
☐	4	Cedula	int(11)			No	*Ninguna*		
☐	5	Genero	varchar(70)	utf8mb4_general_ci		No	*Ninguna*		
☐	6	Edad	int(11)			No	*Ninguna*		
☐	7	BPM	float			No	*Ninguna*		
☐	8	SPO2	float			No	*Ninguna*		
☐	9	Temperatura	float			No	*Ninguna*		

Figura 102: Tabla de síntomas del paciente

Elaborado por: El investigador

En la figura 102 se encuentra la tabla síntomas que cuenta con los siguientes campos:

- **“ID_Pacientes”:** El campo ID_Pacientes utiliza un auto incrementable que registra el id de cada consulta para contabilizar el número de veces que se utilizó al asistente médico.
- **“Fecha”:** El campo Fecha es del tipo timestamp, se utiliza para registrar la fecha y hora en que se realiza la consulta a fin de generar el historial médico que el usuario pueda distinguir una consulta de otra.

- **“Nombre”:** El campo Nombre es del tipo varchar con una longitud de 70 caracteres para registrar el nombre del usuario que realiza la consulta, el mismo es rellenado por medio de una consulta a la tabla “user”.
- **“Cedula”:** El campo Cedula es del tipo int con una longitud de 11 números que son utilizados para registrar la cedula del usuario, el mismo es rellenado por medio de una consulta a la tabla “user”.
- **“Genero”:** El campo Genero es del tipo varchar con una longitud de 70 caracteres para registrar el sexo del usuario, este campo es rellenado por medio de una consulta a la tabla “user”. También es una variable importante al momento de ver la probabilidad de presencia del covid19 en el paciente.
- **“Edad”:** El campo Edad es del tipo int con una longitud de 11 números que son utilizados para registrar la edad del usuario, el mismo es rellenado por medio de una consulta a la tabla “user”.
- **“BPM”:** El campo BPM es del tipo float, que almacena las mediciones de frecuencia cardíaca enviados por el dispositivo mediante el protocolo MQTT.
- **“SPO2”:** El campo SPO2 es del tipo float, que almacena las mediciones de saturación de oxígeno en la sangre enviados por el dispositivo mediante el protocolo MQTT.
- **“Temperatura”:** El campo Temperatura es del tipo float, que almacena las mediciones de temperatura corporal enviados por el dispositivo mediante el protocolo MQTT.
- **“Diagnostico”:** El campo Diagnostico es del tipo varchar con una longitud de 70 caracteres, aquí se almacena el diagnostico generado por la inteligencia artificial una vez el usuario haya solicitado su estado de salud. Este campo se llenará siempre y cuando los síntomas ingresados por el paciente se encuentren dentro de los considerados para el proyecto caso contrario el chatbot enviará un mensaje de ingreso desconocido.
- Como muestra la figura 103 los campos desde la fila 10 hasta la 32 conforman todos los síntomas vistos en la tabla 32 considerados para el diagnóstico médico. Estos campos son del tipo varchar con una longitud de 70 caracteres para el registro de cada síntoma ingresado por el usuario por medio del chatbot.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
☐	2	username	varchar(20)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	3	password	char(102)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	4	fullname	varchar(100)	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	5	cedula	int(11)			No	*Ninguna*			Cambiar Eliminar Más
☐	6	genero	text	utf8mb4_general_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	7	edad	int(11)			No	*Ninguna*			Cambiar Eliminar Más

Figura 103: Tabla de credenciales del usuario

Elaborado por: El investigador

En la figura 103 se encuentra la tabla "user" que se creó para el inicio de sesión y registro de los usuarios, esta tabla cuenta con los siguientes campos:

- **"id":** El campo id utiliza un auto incrementable que aumenta de forma automática cuando un usuario se registra, esta función es utilizada para contabilizar el número total de usuarios tiene la aplicación Web.
- **"username":** El campo username es del tipo varchar con una longitud de 20 caracteres para registrar el correo del usuario que realiza la consulta, el mismo es rellenado cuando el usuario se registra.
- **"password":** El campo password es del tipo varchar con una longitud de 102 caracteres para registrar la contraseña del usuario. Este campo tiene una longitud extensa debido a que las contraseñas son almacenadas por medio de Hashes para ofrecer seguridad al momento del registro.
- **"fullname":** El campo fullname es del tipo varchar con una longitud de 20 caracteres para registrar el nombre del usuario que realiza la consulta, el mismo es rellenado cuando el usuario se registra.
- **"cedula":** El campo cedula es del tipo int con una longitud de 11 números que son utilizados para registrar la cédula del usuario, este valor es importante para realizar las consultas a la base de datos y generar la interfaz de historial médico.
- **"genero":** El campo genero es del tipo text para registrar el sexo del usuario, este campo es rellenado al momento del registro. También es una variable importante al momento de desarrollar el algoritmo de inteligencia artificial.

- **“edad”:** El campo edad es del tipo int con una longitud de 11 números que son utilizados para registrar la edad del usuario.

3.2.10.2. Creación de la interfaz de inicio de sesión y registro

Para la creación de la interfaz de inicio de sesión y registro se utiliza el diagrama de secuencias que se visualiza en la figura 104, donde al momento que un usuario desee registrarse o iniciar sesión si y utilizo la aplicación previamente. Este diagrama se encuentra divido en 2 partes, la primera parte corresponde al inicio de sesión que comienza desde que el usuario ingresa sus credenciales los cuales son almacenados en el servidor para posterior mente realizar una consulta con la base de datos y comprobar si las credenciales que usa el usuario son correctas.

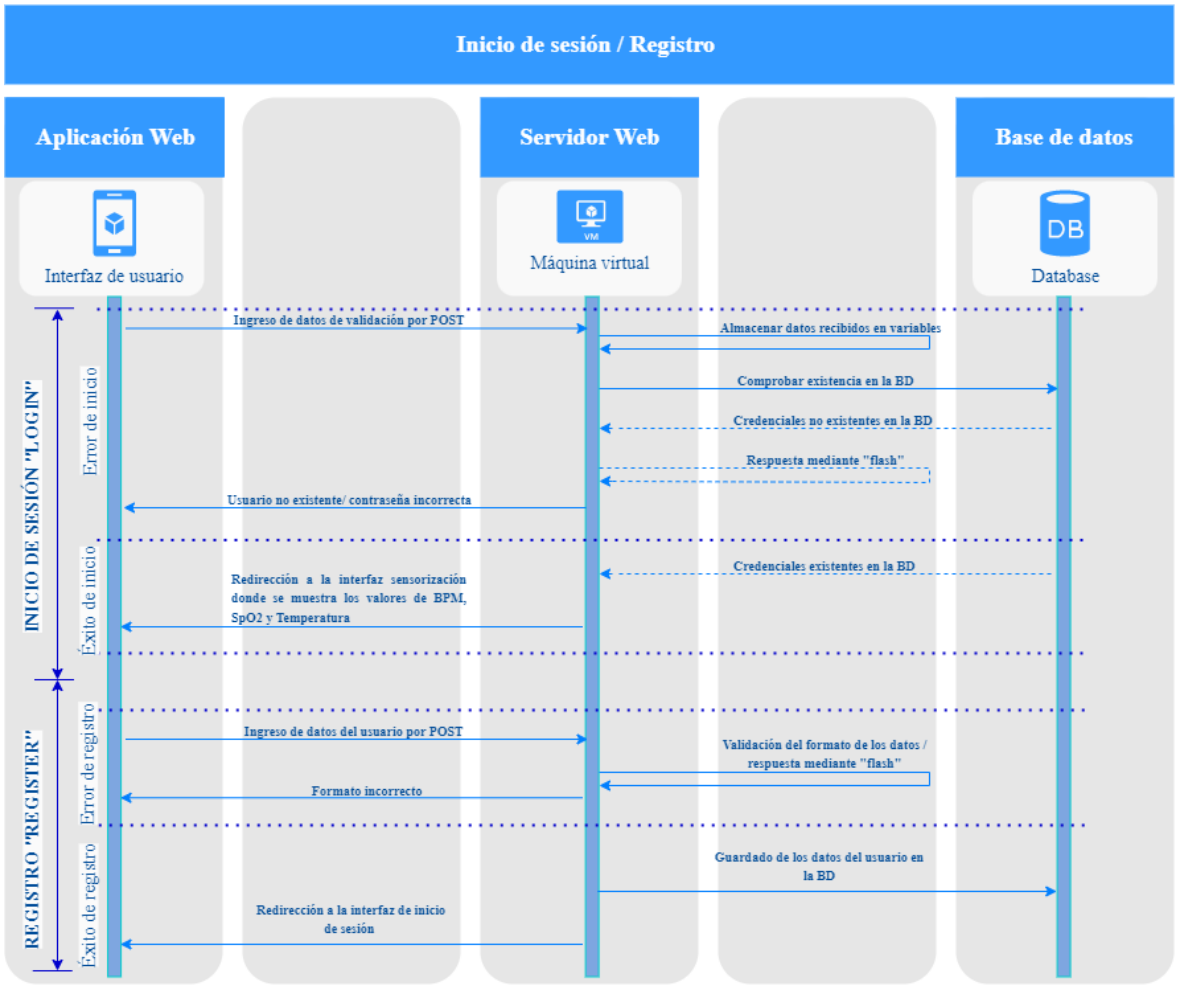


Figura 104: Diagrama de secuencias del inicio de sesión y registro de usuarios

Elaborado por: El investigador

En el caso de que los datos ingresados por el usuario sean incorrectos el servidor utilizara la herramienta “flash” que ofrece el framework de Flask para devolver una notificación de error, las notificaciones consideradas son:

- ¡Usuario no existente!
- ¡Contraseña Invalida!
- ¡Formato del correo electrónico incorrecto!

Por otra parte, para el registro del usuario se utiliza las variables vistas en la figura 105, donde el usuario debe ingresar su información que será almacenada en la base de datos del servidor. En caso de que el formato de los datos ingresados en los campos de texto no sean los adecuados el servidor devolverá una notificación de error, las notificaciones consideradas son:

- ¡Formato del correo incluye una “@example.com”!
- Al ingresar caracteres que no sean números en el campo de cédula, este no se llenara.
- Al ingresar caracteres que no sean números en el campo de edad, este no se llenara.

Configuración del inicio de sesión

Se debe configurar el inicio de sesión del usuario con el objetivo de que solo el usuario registrado pueda ver su propio historial médico y no de los demás. Para lograr esto se programó una función como se observa en la figura 105, que se encarga de verificar si el usuario existe mediante consultas a la base de datos del servidor, en el caso de existir se retornan sus credenciales, mientras que si no existe las credenciales del usuario se retorna un valor nulo que será comparado al momento de establecer la ruta en la interfaz de “login”.

```python
def login(self, db, user):
    try:
        cursor = db.connection.cursor()
        #Saber si el usuario existe en la base de datos
        sql = """SELECT id, username, password, fullname FROM user
                 WHERE username = '{}'""".format(user.username)
        cursor.execute(sql)
        row = cursor.fetchone()
        if row != None:
            #Hemos encontrado el usuario
            user = User(row[0], row[1], User.check_password(row[2], user.password), row[3])
            return user
        else:
            #si no hay retornamos no
            return None
    except Exception as ex:
        raise Exception(ex)
```

Figura 105: Consulta de la existencia del usuario

Elaborado por: El investigador

Después es necesario crear una función para obtener las credenciales del usuario existente que son mostradas en el historial médico del paciente, como se muestra en la figura 106 estas consultas se realizan por medio de comandos SQL en lenguaje de Python. Las credenciales retornadas de la consulta se encuentran en filas donde la primera fila corresponde al número de identificación (id), la segunda fila contiene el correo electrónico que registro el usuario y la última fila almacena el nombre del paciente que realiza la consulta al asistente médico.

```python
def get_by_id(self, db, id):
        try:
            cursor = db.connection.cursor()
            sql = "SELECT id, username, fullname FROM user WHERE id = {}".format(id)
            cursor.execute(sql)
            row = cursor.fetchone()
            if row != None:
                return User(row[0], row[1], None, row[2])
            else:
                return None
        except Exception as ex:
            raise Exception(ex)
```

Figura 106: Obtención de las credenciales del usuario

Elaborado por: El investigador

Por último, el framework de flask trabaja mediante la declaración de rutas que tienen designadas funciones, el nombre de estas funciones debe ser exactamente igual al nombre de la ruta que se ha creado, por lo contrario, si se accede a la ruta se devolverá un error en el servidor (error 500). En el caso de la figura 107, se declara la ruta “/login” con los métodos POST y GET para obtener los datos de ingreso, donde la función declarada como “login” utiliza las variables globales usuariologin, Nombre, Edad, Genero, Cedula que son usadas para almacenar los datos de la consulta realizada por medio de la función.

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    global usuariologin, Nombre, Edad, Genero, Cedula
    if request.method=='POST':
        #metodo post para comprobar si ya lo enviamos
        #print(request.form['email'])
        #print(request.form['password'])
        user = User(0, request.form['email'], request.form['password'])
        usuariologin=request.form['email']
        #Lo que me retorne el metodo modeluser
        logged_user = ModelUser.login(db, user)
        if logged_user != None:
            if logged_user.password:
                #Almacene como usuario logeado
                login_user(logged_user)
                print ("------------------------------")
                #print (usuariologin)
                #cONSULTA DE LOS DATOS DEL USUARIO REGISTRADO
                query = "SELECT fullname, cedula, genero, edad FROM user WHERE username =%s"
                correo_a = (usuariologin,)
                cursor1.execute(query, correo_a)
                myresult = cursor1.fetchall()
```

Figura 107: Configuración de la ruta de la interfaz de Login

Elaborado por: El investigador

Una vez se tiene almacenado el ingreso de datos del usuario, por medio del método POST se realiza la consulta de la existencia de las credenciales. Empleando el condicional "if" se utiliza el correo registrado para comprobar su existencia en el caso de ser correcto se pasa a un segundo condicional "if" encargado de comprobar si la clave es correcta para redirigir al usuario a la interfaz de sensorización. Cabe mencionar que para establecer una mayor seguridad al usuario se trabaja mediante Hash empleando la librería "werkzeug.security", en la figura 108 se genera un hash con la clave registrada.

```python
def has(passw):
    el= generate_password_hash(str(passw))
    return el
```

Figura 108: Generación de un hash

Elaborado por: El investigador

Por otro lado, en la figura 109 se descifra el hash al momento de comprobar las credenciales del usuario.

```python
def check_password(self, hashed_password, password):
    return check_password_hash(hashed_password, password)
```

Figura 109: Descifrado del hash

Elaborado por: El investigador

Diseño de la interfaz de inicio de sesión

Una vez finalizada la configuración del Back-End del inicio de sesión, es necesario configurar el Front-End que visualizara el usuario final. Para ello se emplea el lenguaje HTML que utiliza bloques, una funcionalidad de Flask para trabajar partes del archivo ".html" por separado y de esta forma reducir el código y lograr que el mismo sea más entendible, los componentes creados en el archivo "login.html" son:

- Un título "h1" con la descripción "Por favor inicia sesión"
- Dos inputs de texto que en el interior contendrán una etiqueta que indicara en donde escribir el correo y la clave.
- Un checkbox en el caso el usuario desee que se recuerden sus credenciales mediante el navegador.
- Un botón denominado "Iniciar Sesión" para redirigir al usuario a la interfaz de sensores.
- Un botón denominado "Regístrate" que redireccionara a un nuevo usuario a la interfaz de registro.
- Un párrafo con la descripción "2022-2023" que es la fecha de desarrollo del proyecto.

La programación completa del archivo HTML se puede visualizar en el Anexo 9, mientras que la configuración del cuerpo se muestra en la figura 110.

```
<h1 class="h3 mb-3 fw-normal">Porfavor inicia sesión</h1>
<input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
<div class="form-floating">
  <input name= "email" type="email" class="form-control" id="floatingInput" placeholder="name@example.com">
  <label for="floatingInput">Dirección email</label>
</div>
<div class="form-floating">
  <input name= "password" type="password" class="form-control" id="floatingPassword" placeholder="Password">
  <label for="floatingPassword">Contraseña</label>
</div>

<div class="checkbox mb-3">
  <label>
    <input type="checkbox" value="remember-me"> Recuerdame
  </label>
</div>
<button class="w-100 btn btn-lg btn-primary" type="submit">Iniciar Sesión</button>
```

Figura 110: Diseño en HTML del inicio de sesión

Elaborado por: El investigador

Para agregar un diseño estético de la interfaz se emplea CSS que permite modificar los colores, el background, los bordes, la fuente, etc. Las principales modificaciones se visualizan en la figura 111, mientras que toda la programacion del archivo ".css" se visualiza en el Anexo 11.

```css
root {
  --body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
  --msger-bg: #fff;
  --border: 2px solid #ddd;
  --left-msg-bg: #ececec;
  --right-msg-bg: #579ffb;
}

html {
  box-sizing: border-box;
}
```

Figura 111: Configuración del estilo con CSS

Elaborado por: El investigador

Adicionalmente se utilizó el framework de Bootstrap 5 para lograr que la aplicación sea "responsive" y expandir su uso más allá de un computador; es decir, que la app sea adaptable a dispositivos móviles como se muestra en la figura 112.

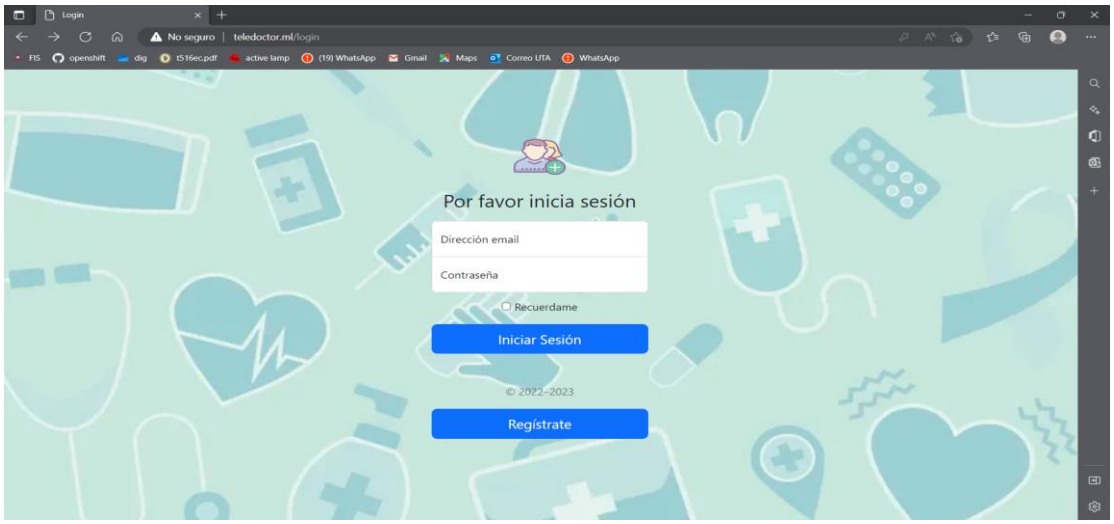


Figura 112: Interfaz de inicio de sesión vista desde un ordenador

Elaborado por: El investigador

La interfaz de inicio de sesión vista desde un celular se muestra en la figura 113, donde se aprecia como la ampliación Web se adapta correctamente a las dimensiones del dispositivo móvil.

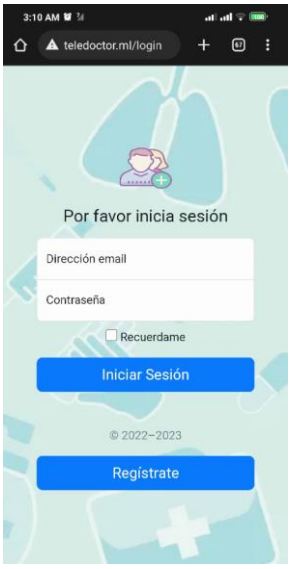


Figura 113: Interfaz de inicio de sesión vista desde un celular

Elaborado por: El investigador

Configuración del registro

La programación del registro de usuarios es necesario ya que el objetivo de la aplicación está previsto para almacenar nuevos usuarios que prueben al asistente virtual. Con esto en mente se crea una función denominada “sub” que se visualiza en la figura 114, esta función se encarga de almacenar la información del usuario en la base de datos mediante comandos SQL adaptados al lenguaje de Python.

```
def sub(db, username, password, fullname, cedula, genero, edad):
    try:
        cursor=db.connection.cursor()
        cursor.execute("INSERT INTO user (username, password, fullname, cedula, genero, edad) VALUES
        db.connection.commit()
        #return redirect(url_for)
    except Exception as ex:
        raise Exception(ex)
```

Figura 114: Ingreso de credenciales a la base de datos

Elaborado por: El investigador

En la figura 115, se declara la ruta “/Registro” con los métodos POST y GET para obtener los datos de ingresados. En la función llamada “Registro” se declaran las variables em, usr, ce, gen, eda, passw, has que son usadas para almacenar las credenciales de los usuarios y enviadas a la función creada en la figura 116. Una vez los datos de registro sean correctos mediante el botón presente en la interfaz se redireccionará al usuario al inicio de sesión.

```python
@app.route('/Registro', methods=['GET', 'POST'])
def Registro():
    if request.method=='POST':
        #metodo post para comprobar si ya lo enviamos

        em=(request.form['email'])
        usr=(request.form['username'])
        ce=(request.form['cedula'])
        gen=(request.form['genero'])
        eda=(request.form['edad'])
        passw=(request.form['password'])
        has=(hash.has(passw))
        print (em, usr, has)
        subirusuario.sub(db, em, has, usr, ce, gen, eda)
        return render_template('auth/login.html')

    else:
        return render_template('auth/registro.html')
```

Figura 115: Configuración de la ruta de la interfaz de registro

Elaborado por: El investigador

Diseño de la interfaz de registro

Tras finalizar con la configuración del Back-End del registro, es necesario configurar el Front-End como se muestra en la figura 116. Para ello se emplea el lenguaje HTML con el cual se crea un archivo llamado “registro.html” que tiene los siguientes componentes:

- Un título “h1” con la descripción “Registrate aquí”.
- Un input de texto con la etiqueta “Dirección email”.
- Un input de texto con la etiqueta “Nombre de usuario”.
- Un input de texto con la etiqueta “Cédula”.
- Un input de texto con la etiqueta “Género”.
- Un input de texto con la etiqueta “Edad”.
- Un input de texto con la etiqueta “dirección email”.
- Un input de texto con la etiqueta “Contraseña”.
- Un botón denominado “Regístrate”, si los datos de registro son correctos se redireccionará a la interfaz de inicio de sesión.
- Un párrafo con la descripción “2022-2023” que es la fecha de desarrollo del proyecto.

Cabe mencionar que la contraseña registrada por el usuario es protegida empleando un Hash como muestra la figura 108, cabe recalcar que la programación completa del archivo HTML se puede visualizar en el Anexo 10.

```
<h1 class="h3 mb-3 fw-normal">Registrate aquí</h1>
<input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
<div class="form-floating">
  <input name= "email" type="email" class="form-control" id="floatingInput" placeholder="name@example.com">
  <label for="floatingInput">Dirección email</label>
</div>
<div class="form-floating">
  <input name= "username" type="username" class="form-control" id="floatingInput" placeholder=" ">
  <label for="floatingInput">Nombre de usuario</label>
</div>
<div class="form-floating">
  <input name= "cedula" type="number" class="form-control" id="floatingInput" placeholder=" ">
  <label for="floatingInput">Cedula</label>
</div>
<div class="form-floating">
  <input name= "genero" type="text" class="form-control" id="floatingInput" placeholder=" ">
  <label for="floatingInput">Genero</label>
</div>
```

Figura 116: Diseño en HTML del registro

Elaborado por: El investigador

Adicionalmente se utilizó el framework de Bootstrap 5 para lograr que la aplicación sea “responsive”; además, de brindarle un apartado estético al mismo como se observa en la figura 117.

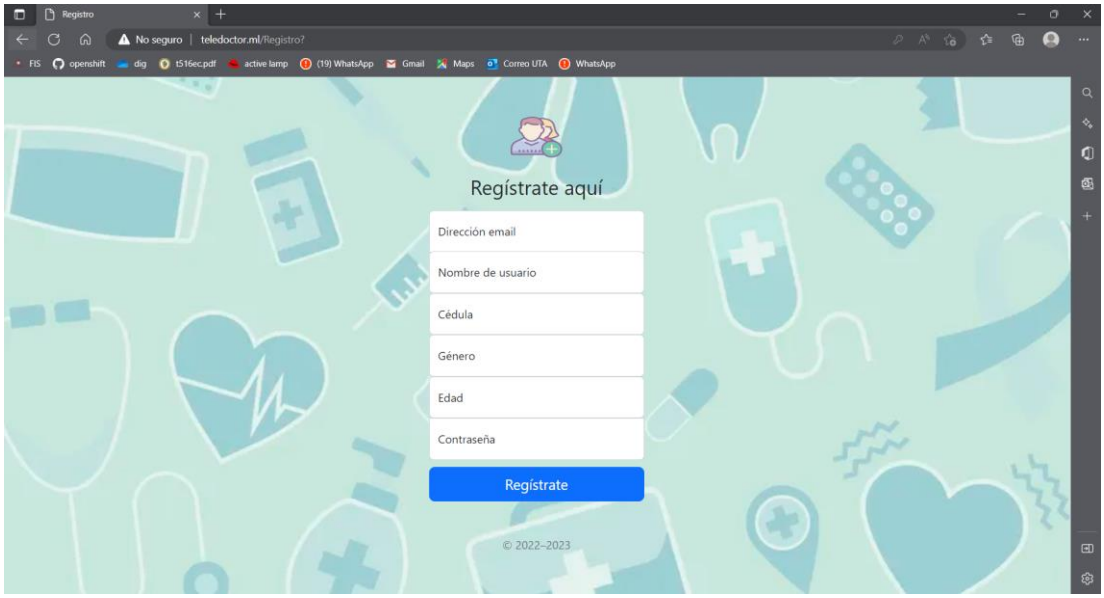


Figura 117: Interfaz de registro vista desde un ordenador

Elaborado por: El investigador

La interfaz de registro vista desde un celular se muestra en la figura 118, donde se aprecia como la ampliación Web se adapta correctamente a las dimensiones del dispositivo móvil.

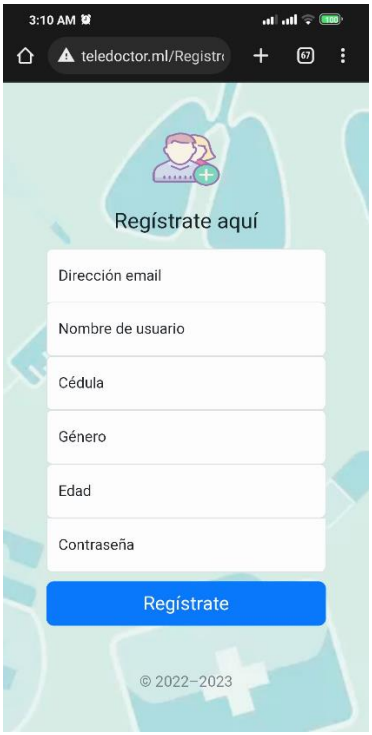


Figura 118: Interfaz de registro vista desde un celular

Elaborado por: El investigador

3.2.10.3. Configuración de la seguridad y errores para el inicio de sesión

Es importante configurar la protección del aplicativo Web e impedir inicios de sesión no autorizados, así como la seguridad para el usuario. Por este motivo se implementó seguridades como “Hash” vistas en la figura 108 y 109 que encriptaba la clave del usuario para impedir el robo de contraseñas de usuarios registrados. Por otro lado, se bloquea el uso de las funcionalidades de la aplicación Web si el usuario no está registrado o no inicia sesión. El comando de la figura 119 es el encargado de proteger el ingreso a cada ruta de la aplicación Web que sean accesibles después del “login”.

```
@login_required
```

Figura 119: Petición de inicio de sesión para acceder a las funcionalidades de la aplicación Web

Elaborado por: El investigador

También se configuró los errores dentro del Front-End que se visualizaran por medio de notificaciones en la interfaz. En consecuencia, se emplea dos funciones que se muestran en la figura 120, la primera función es denominada “estatus_401” que se encarga de redirigir al usuario en el caso de que al iniciar sesión sus credenciales sean incorrectas o la dirección a partir de la ruta raíz que desea ingresar no existe. La segunda función es denominada “error_404” es utilizada en caso el servidor no logre encontrar el recurso requerido.

```
def estatus_401(error):
    return redirect(url_for('login'))


def estatus_404(error):
    return "<h1>Pagina no encontrada</h1>", 404
```

Figura 120: Configuración de errores de petición

Elaborado por: El investigador

Una vez configuradas las funciones de error es necesario llamarlas dentro del “main” que ejecuta la aplicación, esto se hace con las líneas de código mostradas en la figura 121.

```
app.register_error_handler(401, estatus_401)
app.register_error_handler(404, estatus_404)
```

Figura 121: Llamado de las funciones de error entro del "main"

Elaborado por: El investigador

Implementación de CSRF Token

Adicionalmente se implanta una seguridad en contra de la falsificación de solicitudes entre sitios o CSRF por sus siglas en ingles. El funcionamiento de CSRF token se visualiza en la figura 124, donde el usuario por medio de un navegador Web al entrar a la interfaz de usuario el servidor genera un token de manera aleatoria y es guardado usando la librería de “CSRFProtect”. El token generado es enviado de vuelta al navegador y este se almacena de forma oculta dentro del formulario de validación al iniciar sesión, donde se comprueba las credenciales del usuario incluida el token generado por el servidor. Si las credenciales y el token son correctos el servidor redirige al usuario a la interfaz de sensorización, mientras que si el token es incorrecto o la aplicación Web se encuentre en inactividad por más de 10 minutos el token caducara y al tratar de ingresar de nuevo arrojara una notificación de error como la figura 122, obligando a iniciar sesión al usuario nuevamente.

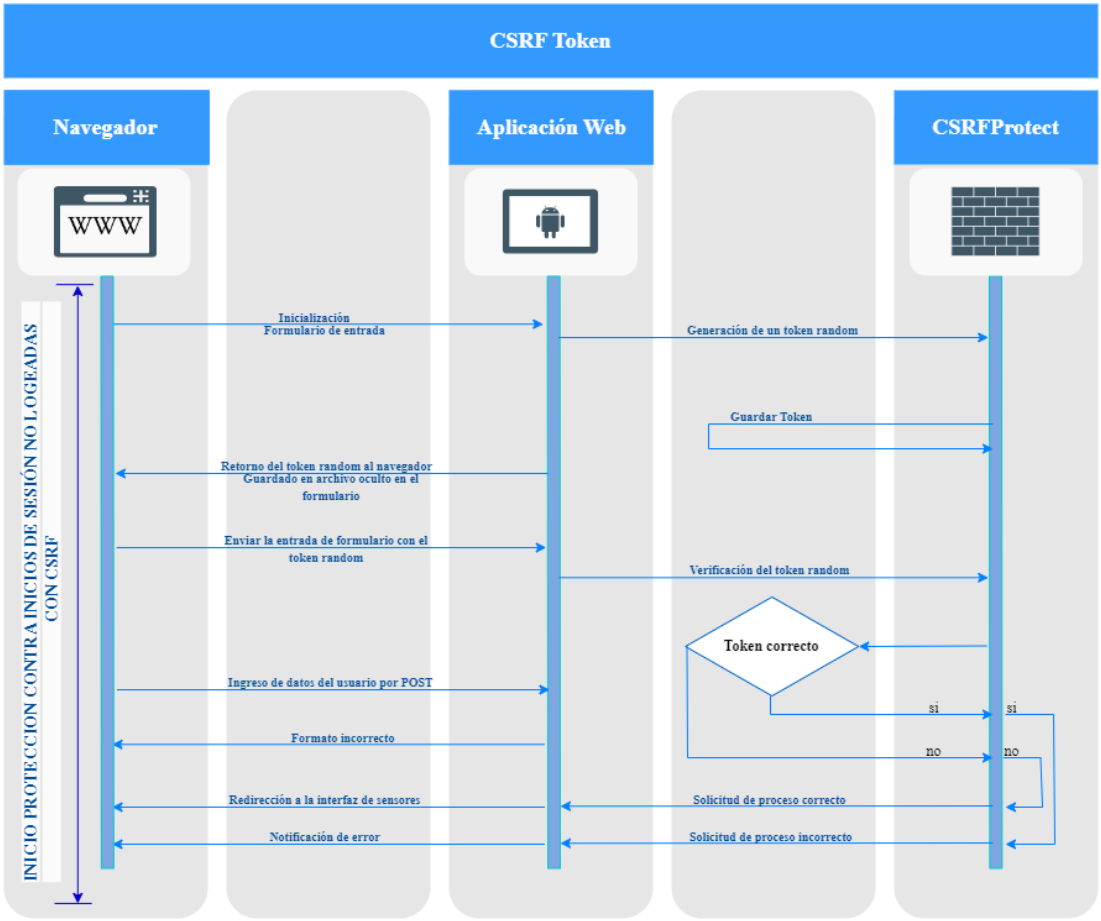


Figura 122: Diagrama de secuencias para evitar los ataques CSRF

Elaborado por: El investigador

Configuración de CSRF token

Para implementar la seguridad en contra de falsificación de inicios de sesión el framework Flask utiliza la librería “CSRFProtect” con el cual se inicializa un objeto denominado “csrf” que se encarga de producir token de forma aleatoria y a su vez verificar que el token ingresado por el formulario sea el correcto. Adicionalmente, es necesario inicializar esta seguridad dentro del “main” para que arranque al mismo tiempo que la aplicación Web, todas estas configuraciones se distinguen en la figura 123.

```
#Importacion de librerias para proteccion contra peticiones no logeadas
from flask_wtf.csrf import CSRFProtect
 #Proteccion de peticiones no logeadas
 csrf = CSRFProtect()
csrf.init_app(app)
```

Figura 123: Implementación de seguridad en contra de CSRF

Elaborado por: El investigador

Finalmente se debe agregar el token generado al formulario de inicio de sesión de forma oculta como muestra las líneas de código de la figura 124. Esta programación corresponde al archivo de “login.html” que se puede observar en el Anexo 9.

```
<h1 class="h3 mb-3 fw-normal">Porfavor inicia sesión</h1>
<input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
```

Figura 124: Inclusión del token generado en el formulario

Elaborado por: El investigador

Una vez se haya configurado la protección contra falsificación de inicios de sesión, se arranca la aplicación web y se verifica si el token ha sido generado mediante los siguientes pasos:

- Click derecho sobre la interfaz de “login”.
- Seleccionar la opción ver código fuente.
- Dentro del código fuente visualizado se puede observar el token como muestra la figura 125.

Figura 125: Token generado

Elaborado por: El investigador

Cuando el usuario haya estado inactivo durante más de 10 minutos el token se caducará y mostrara una pantalla igual a la figura 126.

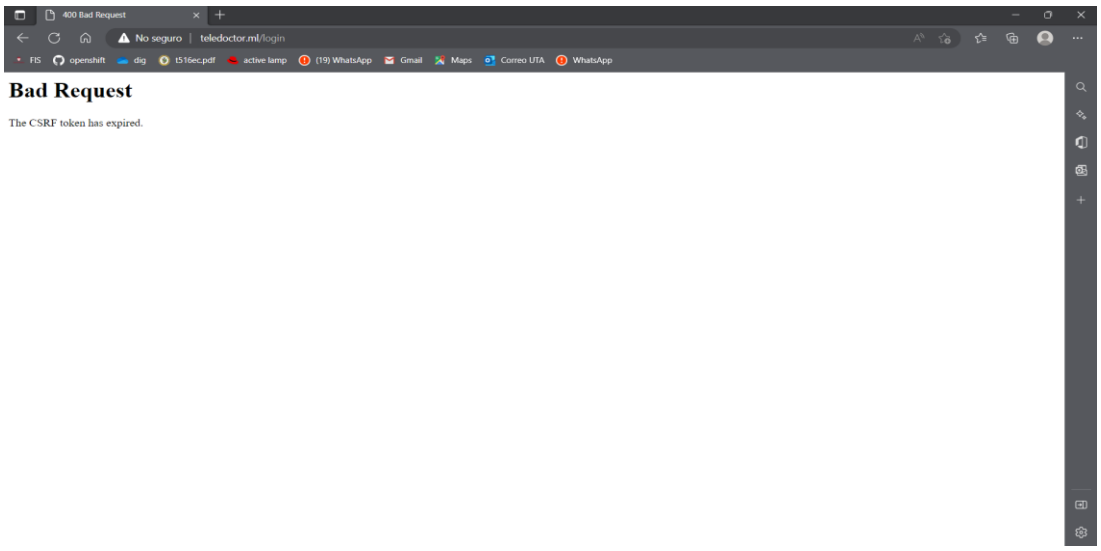


Figura 126: Notificación de error si el token es incorrecto

Elaborado por: El investigador

3.2.10.4. Implementación de la interfaz de sensorización

Para la implementación de la interfaz de sensorización se utiliza el diagrama de secuencia que se muestra en la figura 127, este proceso inicia después que el usuario haya iniciado sesión correctamente.

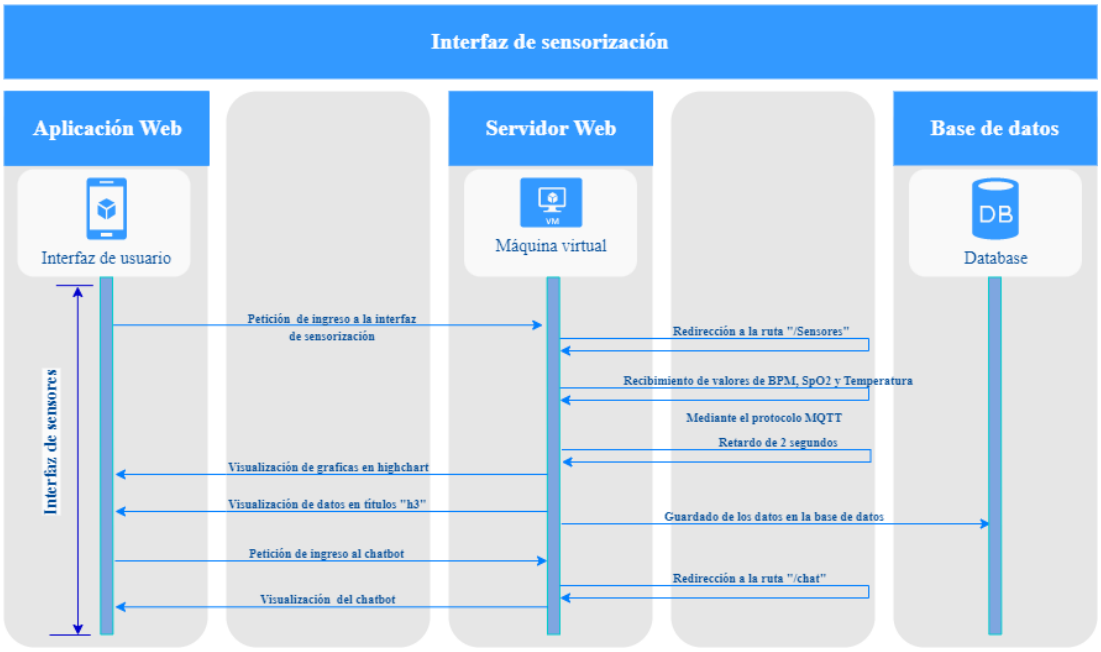


Figura 127: Diagrama de secuencias de la interfaz de sensores

Elaborado por: El investigador

Una vez la petición de ingreso a la interfaz de sensores sea ingresada al servidor, este se encarga de procesar los datos de BPM, SpO2 y temperatura recibidos por el protocolo MQTT para actualizar las gráficas de Highchart cada 2 segundos, con el objetivo de que el usuario observe sus signos vitales en tiempo real. Los datos recibidos del dispositivo de medición de signos vitales son almacenados para que puedan ser utilizados en el historial médico o mediante consultas al chatbot.

Configuración de la comunicación por medio de MQTT

Para levantar el suscriptor MQTT primero fue necesario habilitar el puerto 1883 dentro del firewall como se indica en la figura 128.

Nombre	Tipo	Destinos	Filtros	Protocolos/puertos	Acción	Pric
default-allow-http	Entrada	http-server	Intervalos de	tcp:80, 1883, 3306, 5000	Permitir	

Figura 128: Habilitación del puerto 1883

Elaborado por: El investigador

Dentro del archivo de Flask se debe declarar dos funciones, la primera función denominada “on_connect” se encarga de conectar a la aplicación web al bróker Mosquitto. Cabe mencionar que el tópico que se ingrese dentro del suscriptor debe ser el mismo del publicador, los comandos de código utilizados se muestran en la figura 129. La segunda función denominada “on_message” es utilizada para leer los datos recibidos mediante el tópico, donde toda la información es juntada en una lista. La posición “0” de la lista corresponde al valor de temperatura corporal, la posición “1” es la frecuencia cardiaca y la posición “2” corresponde a la saturación de oxígeno en la sangre.

```
#funcion para conectarse al broker
def on_connect(client, userdata, flags, rc):
    print("Se conecto con mqtt " + str(rc))
    client.subscribe("chatboot/sensores")

#funcion para leer mediante el topic
def on_message(client, userdata, msg):
    global temp, bpm, spo2
    efe=(str(msg.payload)).replace("'","")
    efe=efe.replace("b","")
    print (efe)
    g=efe.split(",")
    temp=g[0]
    bpm=g[1]
    spo2=g[2]
```

Figura 129: Funciones para configurar el tópico y los datos de bpm, spo2 y temp

Elaborado por: El investigador

Una vez realizadas las funciones de suscripción y lectura del tópico se instancia el objeto “client” que se encarga de utilizar las funciones ante mencionadas; además, es necesario ingresar las credenciales del servidor en este caso se escribe la dirección IP externa de la máquina virtual y el puerto por el cual se va a conectar como se muestra en la figura 130.

```
client = mqtt.Client()
client.on_connect= on_connect
client.on_message = on_message
client.connect("34.125.127.48", 1883, 60)
client.loop_start()
```

Figura 130: Objeto client para suscribirse con MQTT

Elaborado por: El investigador

Diseño de la interfaz de sensores

Una vez finalizada la configuración del Back-End de la interfaz de sensores, es necesario configurar el Front-End como se muestra en la figura 131, cabe mencionar la programación completa del archivo HTML se puede visualizar en el Anexo 12.

```
<div class="jumbotron jumbotron-fluid", style="left: 500cm;">
  <div class="container">
    <h1 class="display-4">Signos Vitales</h1>
    <br>
    <br>
    <h3 class="Tem">Temperatura <strong> {{temperatura}} </strong> C </h3>
    <h3 class="Hum">Bpm <strong> {{bpm}} </strong> </h3>
    <h3 class="Tem">Spo2 <strong> {{spo2}} </strong>  </h3>
    <br>
    <br>
  </div>
</div>
<form action="/chat">
  <button class="w-100 btn btn-lg btn-primary" type="submit">Chatboot</button>
</form>
```

Figura 131: Diseño en HTML de la interfaz de sensores

Elaborado por: El investigador

Para esto se emplea el lenguaje HTML con el cual se crea un archivo llamado “sensores.html” que tiene los siguientes componentes:

- Un título “h1” con la descripción “Signos Vitales”.
- Un título “h3” con la descripción “Temperatura”.
- Un título “h3” con la descripción “Bpm”.
- Un título “h3” con la descripción “Spo2”.
- Un botón denominado “Chatbot”, que redirige al usuario al asistente de telemedicina para obtener su diagnóstico médico.

Adicionalmente se utilizó el framework de Bootstrap 5 para lograr que las gráficas utilizadas sean “responsive” y se puedan adaptar a cualquier dispositivo ya sea ordenador o smartphone como muestra en la figura 132.

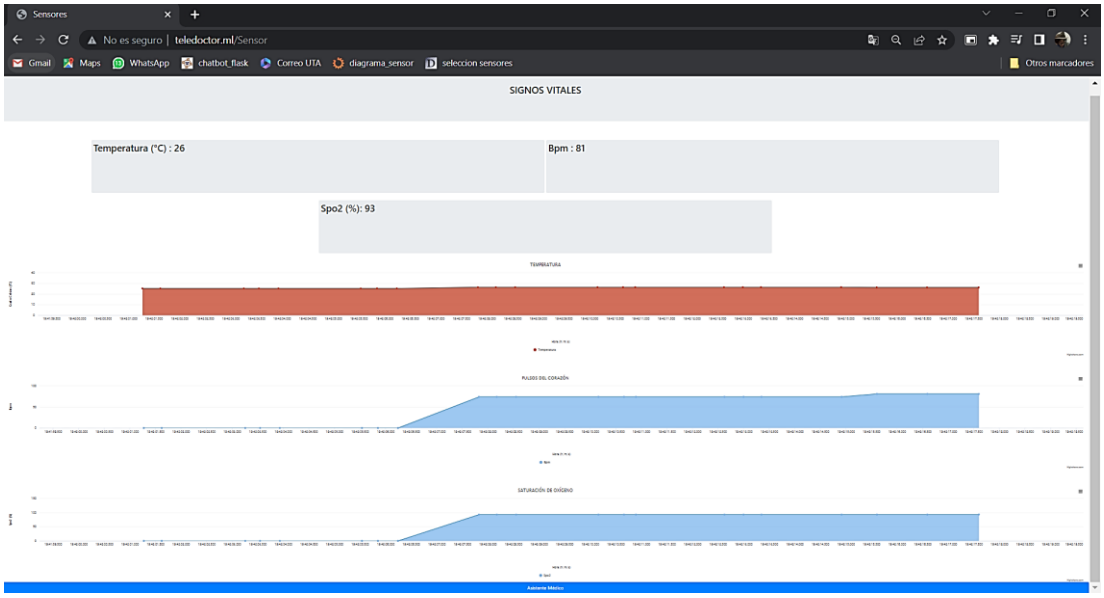


Figura 132: Interfaz de sensores vista desde un ordenador

Elaborado por: El investigador

La interfaz de inicio de sesión vista desde un celular se muestra en la figura 133, donde se aprecia como la ampliación Web se adapta correctamente a las dimensiones del dispositivo móvil.

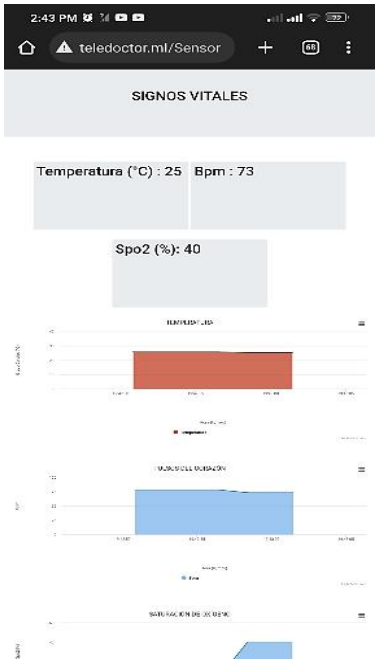


Figura 133: Interfaz de sensores vista desde un celular

Elaborado por: El investigador

3.2.10.5.Implementación del asistente virtual (chatbot)

La implementación del asistente virtual es explicada en la figura 134 que comienza cuando el usuario haya accedido a la interfaz del chatbot este contara con un mensaje de entrada en donde explicara de forma concisa las 3 enfermedades tratadas por el chatbot y pedirá al usuario que ingrese su sintomatología. Una vez el usuario comience a ingresar sus síntomas el chatbot mediante el uso de Procesamiento del Lenguaje Natural identificara la gramática usada y detectara cual es el síntoma al que se refiere el usuario, detectara la probabilidad de que una etiqueta sea la indicada y devolverá una respuesta si el síntoma está dentro de los considerados para el proyecto vistos en la tabla 29. Al finalizar el ingreso de síntomas el usuario pedirá su diagnóstico, por lo cual el servidor consultará los síntomas registrados para diagnosticar al paciente y almacenar el diagnóstico de la app en la base de datos. Adicionalmente se ingresó una opción que permite que el paciente consulte sus signos vitales por medio del cahtbot donde se detalla los valores de temperatura, frecuencia cardíaca y saturación del oxígeno con sus respectivas unidades.

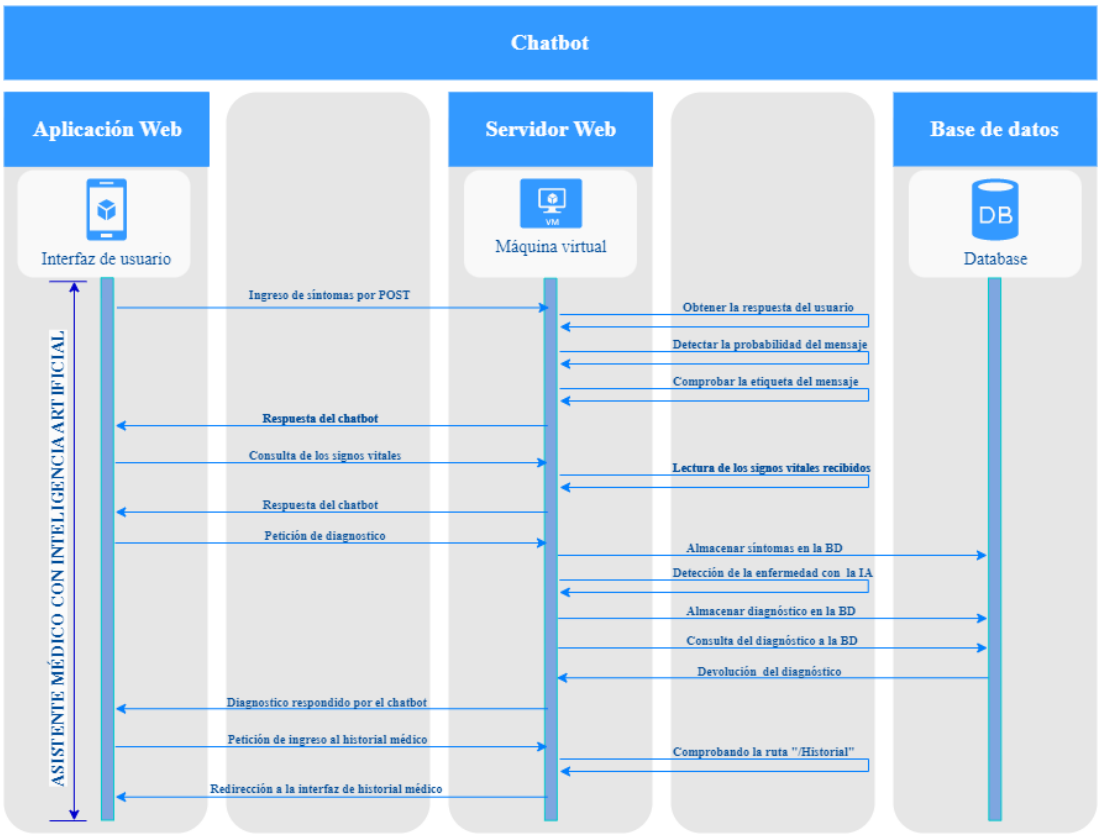


Figura 134: Diagrama de secuencias del asistente de telemedicina

Elaborado por: El investigador

Configuración del chatbot

Primero se creó una función denominada “mensaje_probabilidad” para obtener la probabilidad del síntoma ingresado por el paciente se encuentre dentro de la base de datos del servidor, para esto se separa el mensaje en palabras, las cuales son analizadas si pertenecen a una etiqueta. Para esto se calcula el porcentaje de cuantas palabras pertenecen a cada clase como se observa en la figura 135.

```
def mensaje_probabilidad(mensaje_usuario, palabras_reconocidas, respuesta_unica=False, palabras_requeridas=[]):
    mensaje_certeza = 0
    tiene_palabras_requeridas = True

    # Cuenta cuántas palabras están presentes en cada mensaje predefinido
    for palabra in mensaje_usuario:
        if palabra in palabras_reconocidas:
            mensaje_certeza += 1

    # Calcula el porcentaje de palabras reconocidas en el mensaje del usuario
    porcentaje = float(mensaje_certeza) / float(len(palabras_reconocidas))

    # Comprueba que las palabras requeridas están en la cadena
    for palabra in palabras_requeridas:
        if palabra not in mensaje_usuario:
            tiene_palabras_requeridas = False
            break
```

Figura 135: Detección de la probabilidad de un mensaje

Después se crea una función denominada “comprobar_todos_los_mensajes” que por medio de la probabilidad más alta conseguida del análisis de cada palabra como se observa en la figura 136, se determina a que tag corresponde y cuál es la respuesta que debe arrojar el chatbot al usuario. Los comandos utilizados se muestran en la figura 138, donde se hace un llamado a la función “mensaje_probabilidad” para obtener el porcentaje de cada palabra analizada.

```
def comprobar_todos_los_mensajes(mensaje):
    lista_de_probabilidad_más_alta = {}

    # Simplifica la creación de respuestas por medio del tag
    def respuesta(bot_respuesta,lista_de_palabras, respuesta_unica=False, palabras_requeridas=[]):
        #Usa la variable de la funcion "comprobar_todos_los_mensajes" dentro de la funcion "respuesta"
        nonlocal lista_de_probabilidad_más_alta
        lista_de_probabilidad_más_alta[bot_respuesta] = mensaje_probabilidad(mensaje,lista_de_palabras, respuesta_unica, palabras_requeridas)
```

Figura 136: Detección de la etiqueta del mensaje

Elaborado por: El investigador

Además, los tags utilizados para determinar a qué etiqueta corresponden cada entrada de texto del usuario se visualiza en la tabla 32, en el caso de la etiqueta de diagnóstico hace referencia que la aplicación Web iniciara el algoritmo de inteligencia artificial para diagnosticar al paciente.

Tabla 32: Etiquetas creadas para respuestas del chatbot

Uso	Tag	Respuesta	Palabras clave
Síntomas	1	¿Algún otro síntoma? Listo, ¿Qué más tienes? ¿Algún otro dolor? ¿Alguna otra molestia?	'moquera', 'secrecion', 'nasal', 'secreción', 'mocos'
	2		'congestion', 'congestión', 'nasal', 'obstrucción', 'obstruccion'
	3		'leve', 'levemente', 'poco'
	4		'moderado', 'normal', 'regular'
	5		'fuerte', 'mucho'
	6		‘lagrimeo’, ‘llorosos’, ‘lagrimea’
	7		'tos', 'seca', 'sin', 'flema'
	8		'tos', 'con', 'flema'
	9		'estornudar', 'estornudo', 'estornudos', 'estornudando’
	10		'respirando', 'respiro', 'respirar', 'falta', 'aire'
	11		'fiebre', 'caliente', 'calentura'
	12		'malestar', 'fatiga', 'dolor', 'cuerpo', 'cansado', 'cansancio', 'perdida', 'animo', 'pérdida', 'ánimo'
	13		'pica', 'picazón', 'picazon', 'hormigeo', 'nariz', 'comezón', 'comezon'
	14		'hinchazon', 'hinchazón', 'hinchados', 'bolsas', 'ojeras'
	15		'ronca', 'ronco', 'ronquidos', 'roncando'
	16		'muscular', 'músculos', 'músculo', 'musculos', 'musculo', 'mialgia'
	17		'voz', 'diafonia', 'diafonía'
	18		'ocular', 'ojos'
	19		'diarrea'
	20		'nauseas', 'arcadas'
	21		'barriga', 'abdomen'
	22		'pecho', 'tórax', 'torax'
	23		'apetito', 'comer', 'hambre', 'anorexia'
	24		'escalofríos', 'escalofrios', 'temblor', 'temblores', 'sacudida', 'sacudidas', 'espasmo', 'espasmos'
	25		'diagnostico', 'diagnóstico', 'no', ‘nada’

Conversación convencional	40	¡Hola de nuevo! Cuéntame, ¿Cuáles son tus síntomas? 😄	'hola', 'encantado', 'hello', 'hi', 'hey', 'sup', 'heyo'
	41	Cuídate 👌🏻, recuerda todo queda entre nosotros 😇	'hasta', 'chao', 'luego', 'adios', 'adiós', 'cuidate', 'cuídate', 'ir', 'despues', 'después', 'bai', 'chau', 'chaitos', 'bye', 'goodbye'
	42	Estoy muy bien gracias. Dime, ¿Cuáles son tus síntomas?	'como', 'cómo', 'estas'
	43	De nada estoy aquí para servirte, espero verte de nuevo	'gracias', 'thank', 'thanks'
Palabras desconocidas	44	¿Puedes escribirlo de otra forma?, no lo he captado	Palabras desconocidas
	45	No lo he entendido ...	
	46	Suena bien si lo entendiera jaja. Escríbelo de otra forma	
	47	¿Qué significa eso?, escríbelo de otra manera	
Respuestas largas	48	¡Estoy aquí para servirte! Dime, ¿Cuáles son tus síntomas?	'haces', 'opciones', 'ayudarme', 'ayudame'
	49	Mi papá es Naythan, ¿Lo conoces?, el me creo para ayudar a los pacientes con infecciones respiratorias. Es un buen tipo 😊. Así que, ¿Cuáles son tus síntomas?	'creo', 'creador', 'papá', 'papa', 'padre'
Respuesta a síntomas	100	El dolor de garganta es, ¿Leve, moderado o fuerte?	'garganta'
	101	Con flema o sin flema	'tos'
	102	El dolor de las articulaciones es, ¿Leve, moderado o fuerte?	'articular', 'articulaciones'
	103	El dolor de la cabeza es, ¿Leve, moderado o fuerte?	'cabeza', 'migraña', 'cefalea', 'jaqueca'

Elaborado por: El investigador

En la figura 137 se muestra que a la variable “mejor_tag” se le asigna la etiqueta con la mayor probabilidad, para ingresar a un condicional “if” que analiza si esta probabilidad obtenida es mayor a 1 se entregara la etiqueta establecida, caso contrario se retornara una etiqueta de respuesta desconocida.

```
mejor_tag = max(lista_de_probabilidad_más_alta, key=lista_de_probabilidad_más_alta.get)
return largas.desconocido() if lista_de_probabilidad_más_alta[mejor_tag] < 1 else mejor_tag
```

Figura 137: Selección de la etiqueta con mayor probabilidad

Elaborado por: El investigador

Para acondicionar la entrada de texto se elimina caracteres especiales que no son de relevancia para este proyecto, en la figura 138 se logra este cometido utilizando la herramienta "re.split" que envía el mensaje adaptado al resto de funciones para poder retornar la respuesta procesada del asistente virtual.

```
# Funcion para tener la respuesta del bot
def get_response(user_input):
    split_message = re.split(r'\s+|[,;?!.-]\s*', user_input.lower())
    response0 = check_all_messages(split_message)
    global response
    response = response0.translate({ord(letter): None for letter in '12345678910*'})
    base_numeros(response0)
    return response
```

Figura 138: Respuesta del chatbot

Elaborado por: El investigador

Diseño de la Interfaz del chatbot

Una vez finalizada la configuración del Back-End del chatbot, es necesario configurar el Front-End como se muestra en la figura 139. Para ello se emplea el lenguaje de programación HTML con el cual se crea un archivo denominado "index.html" que tiene los siguientes componentes:

- Un encabezado "header" con la descripción "Tu Asistente Médico".
- Una etiqueta "label" con la leyenda "Baymax", que se utiliza para nombrar al asistente de telemedicina creado.
- Una etiqueta "label" con la leyenda "24:00", que se utiliza para mostrar la hora en el mensaje predeterminado, este número hace referencia que el chatbot es utilizable las 24 horas del día.
- Una etiqueta "label" con la leyenda "¡Hola!, Yo soy Baymax tu asistente personal de salud. Mi trabajo es diagnosticarte una enfermedad (Covid19, Rinitis alérgica y Resfriado común) 😟, dependiendo de tus síntomas. Si tu enfermedad no está dentro de mi base de datos espero ayudarte en un futuro 😕. Dicho esto, cuéntame, ¿Cuáles son tus síntomas? Al terminar de escribir

tus síntomas por favor escribe "Diagnóstico" para ver tus resultados.", Este es el mensaje predeterminad que se muestra al usuario.

- Un botón denominado "Enviar", que ingresa el mensaje para ser procesado por la aplicación Web.
- Un botón denominado "Historia Clínica", que redirige al usuario a la última interfaz donde se muestra los detalles de todas sus consultas.

Toda la programación del archivo HTML se puede visualizar en el Anexo 13.

```
<header class="msger-header">
  <div class="msger-header-title">
    <i class="fa-solid fa-user-doctor"></i> Tu Asistente Médico <i class="fa-solid fa-stethoscope"></i>
  </div>
</header>

<main class="msger-chat">
  <div class="msg left-msg">
    <div class="msg-img" style="background-image: url(https://cdn-icons-png.flaticon.com/512/4660/4660583.png)"></div>

    <div class="msg-bubble">
      <div class="msg-info">
        <div class="msg-info-name">Baymax</div>
        <div class="msg-info-time">24:00</div>
      </div>
```

Figura 139: Diseño HTML del chatbot

Elaborado por: El investigador

Para diseñar el apartado estético de la interfaz se emplea CSS que permite modificar los colores, el background, los bordes, la fuente, etc. Las principales modificaciones se visualizan en la figura 140; sin embargo, toda la programación del archivo "style.css" se visualiza en el Anexo 14.

```
:root {
    --body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
    --msger-bg: #fff;
    --border: 2px solid #ddd;
    --left-msg-bg: #ececec;
    --right-msg-bg: #579ffb;
  }

  html {
    box-sizing: border-box;
  }
```

Figura 140: Configuración del estilo del chatbot con CSS

Elaborado por: El investigador

Adicionalmente se utilizó el framework de Bootstrap 5 para lograr que la aplicación sea "responsive" y expandir su compatibilidad con ordenadores de escritorio y dispositivos móviles como se muestra en la figura 141.

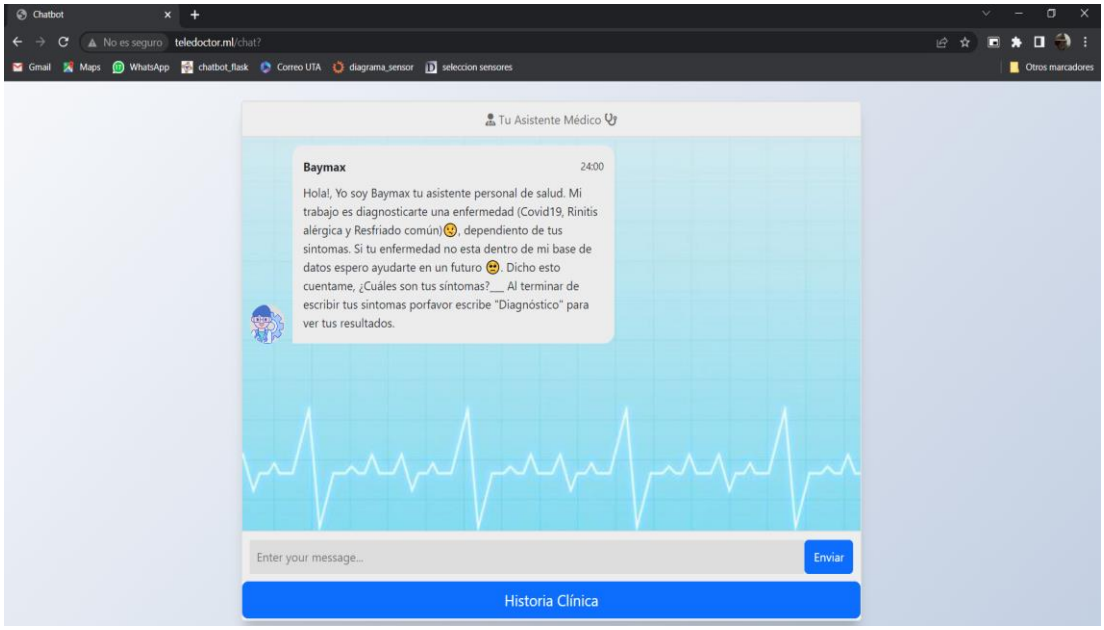


Figura 141: Interfaz del chatbot vista desde un ordenador

La interfaz de inicio de sesión vista desde un celular se muestra en la figura 142, donde se aprecia como la ampliación Web se adapta correctamente a las dimensiones del dispositivo móvil.

Elaborado por: El investigador

Figura 142: Interfaz del chatbot vista desde un celular

Elaborado por: El investigador

3.2.10.6. Implementación de la interfaz para el historial médico

Para la implementación de la interfaz del historial médico se utiliza el diagrama de secuencias de la figura 143, donde se explica el funcionamiento de la interfaz del historial médico implementado, una vez el usuario envié una petición para acceder a la interfaz, el servidor se encarga de redirigir al navegador a la ruta "/historial". Después realiza consultas a la base de datos para definir los encabezados de cada columna, después de obtener los datos del usuario junta todo en una tupla para generar la tabla con el historial de consultas realizadas por el paciente. Tras revisar el resultado de su consulta el usuario puede salir de la aplicación web y será redirigido a la pantalla de inicio de sesión.

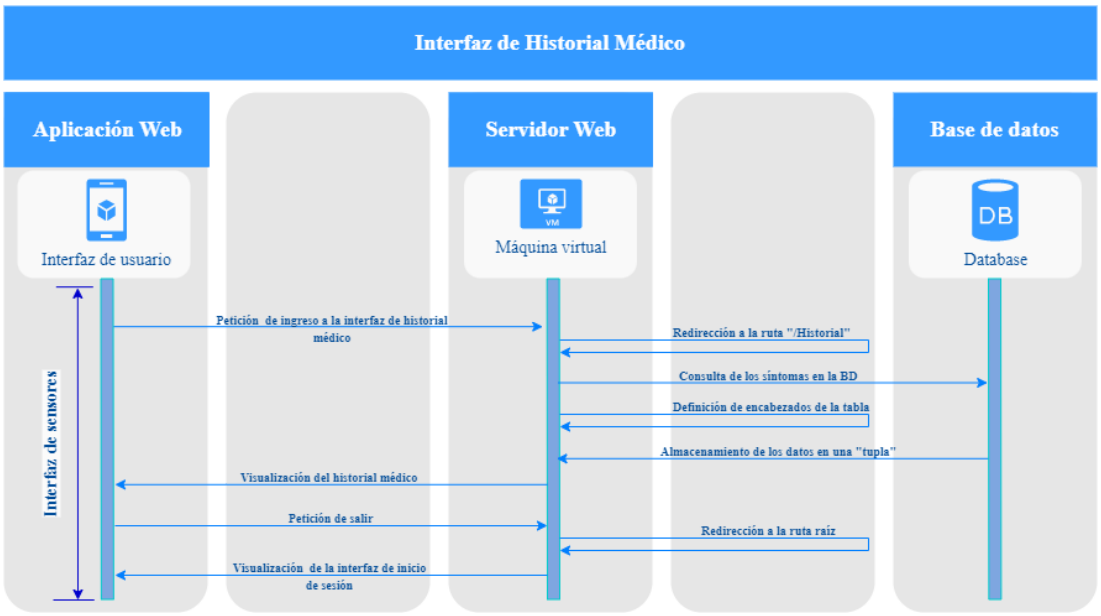


Figura 143: Diagrama de secuencias de la interfaz de historial médico

Elaborado por: El investigador

Configuración del historial médico

Como se observa en la figura 144 la tabla del historial es generada mediante comandos SQL, donde los encabezados de cada columna corresponden a la fecha y hora de la consulta, los datos del usuario que inicio sesión, los síntomas que este presento y por último, el diagnostico dado por el algoritmo de inteligencia artificial.

```
#-------------------------------------------------------------------------------------------------------------------------
#-----------------------------------------------Creacion del historial medico-------------------------------------------
#Titulos de la tabla para el historial
headings = ("Fecha - Hora", "Nombre", "Cédula", "Género", "Edad", "BPM", "SPO2", "Temperatura", "Síntomas", "Diagnóstico")
#CONSULTA DE LOS usuarios repetidos por cedula
cedula= Cedula
#cursor1.execute("SELECT Nombre, Genero, COUNT(C) FROM pacientes GROUP BY Nombre HAVING COUNT(Nombre) > 1")
sql="SELECT Fecha, Nombre, Cedula, Genero, Edad, BPM, SPO2, Temperatura, Secrecion_Nasal, Congestion_Nasal, Dolor_garganta,
cedula1 = (cedula,)
cursor1.execute(sql,cedula1)
#Guardar consulta en una variable
a= cursor1.fetchall()
```

Figura 144: Declaración de encabezados para la tabla historial

Elaborado por: El investigador

Una vez se haya realizado la consulta a la base de datos es necesario adecuar la información para ser mostrada en la tabla. Por ende, se crea una lista denominada "list_sintomas" que juntara todos los síntomas del paciente; sin embargo, es necesario discernir que síntomas si tubo el paciente, por lo cual se elimina los valores nulos de la lista. Para finalizar con el adecuamiento de la información se una la lista de síntomas a una lista general que es transformada en tupla, ya que este es el formato con el que trabaja las tablas de Bootstrap 5, los comandos utilizados para adecuar la información de las tablas se muestran en la figura 145.

```
#Creacion de lista de sintomas
list_sintomas= [s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23]
#Eliminar valores vacios de la lista
for i in range(len(list_sintomas)-1, -1, -1):
    if not list_sintomas[i]:
        del list_sintomas[i]
#Concatenacion de sintomas
C_sintomas= ", ".join(list_sintomas)
#print(C_sintomas)
#Obtencion de diagnostico
d= row[31]
#print(d)
#------------------------------Creacion de una lista completa para el historial-------------------------
list_completa=[n1,n2,n3,n4,n5,n6,n7,n8,C_sintomas,d]
```

Figura 145: Adecuación de la información de la tabla historial médico

Elaborado por: El investigador

Diseño de la Interfaz

Una vez finalizada la configuración del Back-End del historial médico, es necesario configurar el Front-End como se muestra en la figura 146, para ello se emplea el lenguaje de programación HTML con el cual se crea un archivo llamado "tabla_bootstrap.html" que tiene los siguientes componentes:

- Un título “h2” con la leyenda “Historia Clínica”.
- Un párrafo “p” con la leyenda “A continuación se muestra de forma detallada tus consultas con su respectivo diagnóstico:”.
- Una tabla “table” que es usada para mostrar los datos de todas las consultas realizadas por el usuario.
- ¡Un párrafo “p” con la leyenda “Gracias por visitarnos!”.

Además, la programación del archivo HTML cuenta con la inclusión de un pie de página con información acerca de la facultad y de lugares de atención en caso el diagnóstico sea covid19, resfriado común o rinitis alérgica, este código se puede visualizar en el Anexo 15.

```
<div class="table-responsive">
  <table class="table table-bordered">
    <thead class="table-dark">
      <tr>
        {% for header in headings %}
        <th class="text-center">{{ header }}</th>
        {% endfor %}
      </tr>
    </thead>
    <tbody>
      {% for row in data %}
      <tr>
        {% for cell in row %}
        <td class="text-center">{{ cell }}</td>
        {% endfor %}
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
```

Figura 146: Diseño HTML del historial médico

Elaborado por: El investigador

Cabe mencionar que se utilizó el framework de Bootstrap 5 con el objetivo que la tabla que muestre el historial sea “responsive” y sea compatibilidad con ordenadores de escritorio y dispositivos móviles como se muestra en la figura 147.

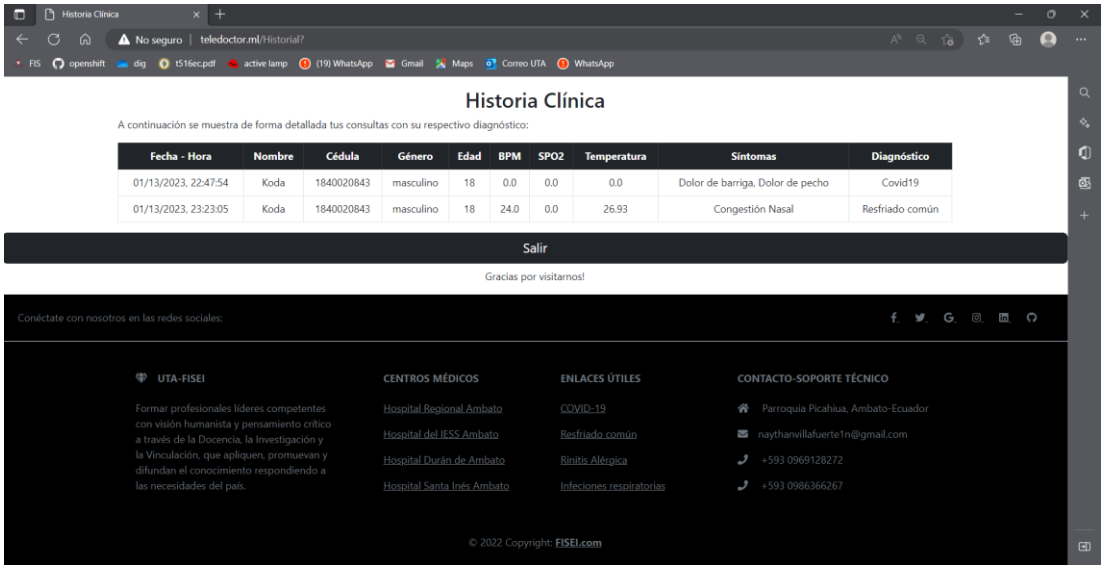


Figura 147: Interfaz del historial médico visto desde un ordenador

Elaborado por: El investigador

La interfaz de inicio de sesión vista desde un celular se muestra en la figura 148, donde se aprecia como la ampliación Web se adapta correctamente a las dimensiones del dispositivo móvil.

Figura 148: Interfaz del historial médico visto desde un celular

Elaborado por: El investigador

3.2.10.7. Creación de la máquina virtual

Levantar una máquina virtual dentro de Google Cloud resulta relativamente fácil, ya que solo es necesario elegir la imagen del sistema operativo con el que se desea trabajar. Cabe recalcar que todos los sistemas operativos disponibles son basados en Linux; sin embargo, el que tiene más prestaciones es el sistema de Ubuntu Server. Una vez seleccionado el sistema opera, el disco de arranque y su capacidad, se asignó el nombre de "ubuntuserver" a la instancia VM.

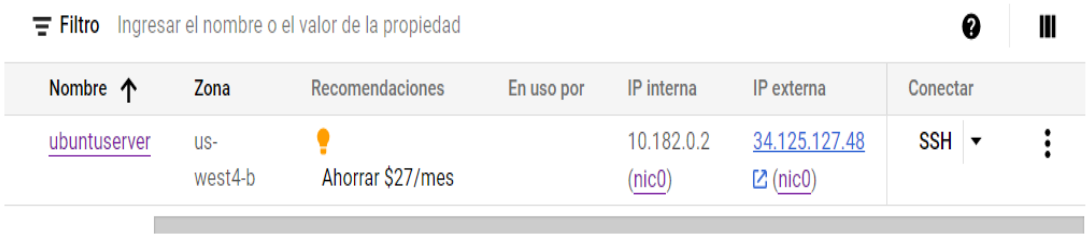


Figura 149: Instancia VM creada en Google Cloud

Elaborado por: El investigador

En la figura 149, se observa la máquina virtual donde se puede destacar la IP externa "34.125.127.48" que es la utilizada para conectar se la base de datos, levantar el cliente MQTT y arrancar la aplicación Web.

3.2.10.8. Configuración de un dominio DNS

Para configurar el DNS dentro del servidor primeramente es necesario conseguir un dominio, para esto se utilizó a la página Web "Freenom" que distribuye dominios gratuitos durante un determinado tiempo. Dentro de la página Web se selección el dominio "teledoctor.ml" que es gratuito durante un año como se indica en la figura 150.

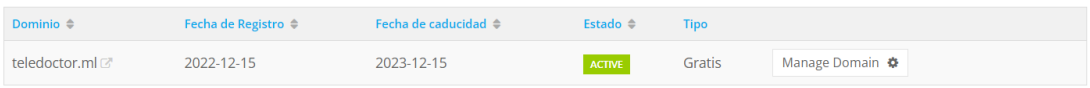


Figura 150: Obtención de dominio DNS en Freenom

Elaborado por: El investigador

Una vez se obtenga un dominio DNS se lo debe vincular con la máquina virtual creada. Por ende, se crea 2 conjuntos de registros, el primer registro es del tipo A y contiene la dirección IP externa de la instancia VM. El segundo registro es del tipo CNAME utilizado para agregar un alias al dominio obtenido de Freenom. Al terminar la

configuración del DNS dentro de la pestaña Cloud DNS de Google se observa una configuración similar a la figura 151, que tiene los dos conjuntos de registros creados más otros dos conjuntos de registros predeterminados para correr en el servidor de Google Cloud.

Figura 151: Levantamiento de un dominio DNS en Google Cloud

Elaborado por: El investigador

Por último, se debe registrar los datos de enrutamiento como se ve en la figura 152, estos datos son proporcionados por el servidor y es necesario escribir este enrutamiento ya que la pagina Freenom que funciona como registrador de la zona para tener noción de en donde se está utilizando el DNS y permitir su uso.

Figura 152: Ingreso de datos de enrutamiento dentro de Freenom

Elaborado por: El investigador

3.2.11. Pruebas de funcionamiento y análisis de resultados

Una vez finalizado la implementación de la propuesta del proyecto de investigación, es necesario verificar la exactitud del dispositivo electrónico ya que los valores medidos de frecuencia cardíaca, saturación de oxígeno en la sangre y temperatura corporal deben cumplir niveles tolerables según la doctora que colaboro con el proyecto. En consecuencia, se comparan los valores medidos de BPM y SpO2 con un pulsioxímetro comercial X004C, mientras que el valor medido de temperatura corporal es comparado con un termómetro infrarrojo comercial TWQ-01. Cabe mencionar que los dispositivos para la comparativa fueron seleccionados por la recomendación del profesional de la salud.

Para verificar cual es la exactitud del dispositivo se debe calcular 3 métricas de calidad, las cuales son error absoluto, error relativo y fiabilidad. En el caso de error absoluto se debe trabajar con el valor medido por el dispositivo, mientras que el valor real es la medición del pulsioxímetro comercial, de igual manera sucede con la temperatura corporal donde el valor real es el medido por el termómetro digital comercial. El cálculo del error absoluto se obtiene empleando la ecuación (23). La segunda métrica de calidad es el error relativo que determina cual es el porcentaje de mediciones erróneas que arroja el dispositivo en comparación al comercial, este se calcula utilizando la ecuación (24). Por último, se calcula la fiabilidad del dispositivo que determina el porcentaje de mediciones adecuadas cotejadas con el dispositivo comercial, la forma de obtener esta métrica es por medio de la ecuación (25).

$$E_{absoluto} = |V_{medido} - V_{real}| \quad \text{Ecuación (23)}$$

$$E_{relativo}\,(\%) = \frac{E_{absoluto}}{V_{medido}} * 100\% \quad \text{Ecuación (24)}$$

$$F = 100\% - E_{relativo} \quad \text{Ecuación (25)}$$

Donde:

$E_{absoluto}$ → Error absoluto

$E_{relativo}$ → Error relativo

F → Fiabilidad

V_{medido} → Valor medido

V_{real} → Valor real

3.2.11.1. Medición de BPM (Frecuencia cardíaca)

Para determinar si la medición de frecuencia cardíaca es correcta se realiza una comparativa de las medidas del dispositivo construido con respecto a un dispositivo de uso comercial. En la tabla 33 se puede observar que se trabajó con un total de 20 personas que acudieron al consultorio de la doctora que ayudo en el proyecto. Los resultados de las mediciones de "BPM" tuvieron una variación de 3.1 bpm con respecto a un pulsioxímetro de uso comercial, dando lugar a un error de medición promedio de 4.52%. Este porcentaje de error arroja una fiabilidad de 95.48%, lo cual lo hace funcional al momento de tratar con un paciente ya que el profesional de la salud hizo énfasis en que se revisa el estado de los signos vitales en busca de una anomalía.

Tabla 33: Fiabilidad del dispositivo al medir la frecuencia cardíaca

Nº Paciente	Medición dispositivo (BPM)	Medición X004C (BPM)	$E_{absoluto}$	$E_{relativo}$	$E_{relativo}$ (%)
1	72	77	5	0,069	6,944
2	75	76	1	0,013	1,333
3	63	64	1	0,016	1,587
4	74	78	4	0,054	5,405
5	63	65	2	0,032	3,175
6	75	78	3	0,040	4,000
7	78	81	3	0,038	3,846
8	62	68	6	0,097	9,677
9	74	77	3	0,041	4,054
10	62	65	3	0,048	4,839
11	68	70	2	0,029	2,941
12	63	67	4	0,063	6,349
13	72	73	1	0,014	1,389
14	72	74	2	0,028	2,778
15	74	76	2	0,027	2,703
16	66	70	4	0,061	6,061
17	77	80	3	0,039	3,896
18	64	72	8	0,125	12,500
19	69	71	2	0,029	2,899
20	74	77	3	0,041	4,054
Promedio	69,85	72,95	3,1	0,045	4,522
Fiabilidad (%)					95,478

Elaborado por: El investigador

En la figura 153 se puede visualizar el comportamiento de las mediciones de BPM en donde existe una tendencia de mayor precisión mientras el rango de frecuencia cardiaca se encuentre entre 60bpm y 65bpm, ya que cuando el valor real es superior a los 70bpm el dispositivo presenta cierto grado de dificultad al medir. También es importante mencionar que el sensor MAX30100 utilizado es muy sensible a la posición del dedo sobre los leds, por lo que un mal posicionamiento del dedo sobre el sensor provocara una medición errónea.

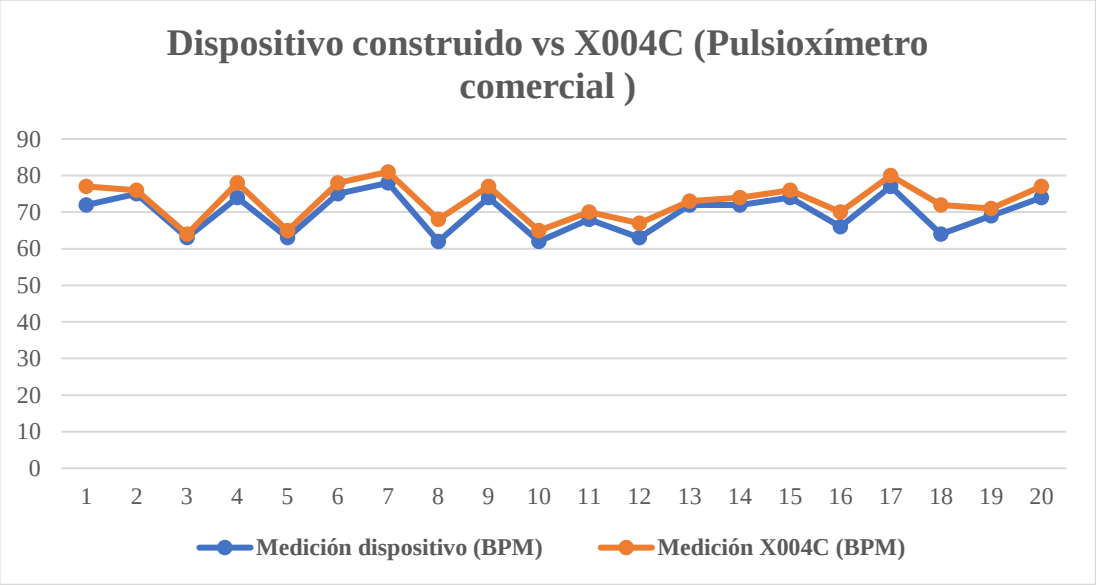


Figura 153: Comportamiento de la medición de BPM

Elaborado por: El investigador

3.2.11.2.Medición de SpO2 (Saturación de oxígeno en la sangre)

Para establecer la fiabilidad al medir la saturación de oxígeno en la sangre se coteja al dispositivo implementado con un dispositivo de uso comercial, en la tabla 34 se puede observar que la muestra utilizada son 20 personas que arrojaron una variación de 2.25% de SpO2 con respecto a un pulsioxímetro de uso comercial. El error de medición promedio que se registro fue de 2.45% lo que da como resultado una fiabilidad de 97.55%. Estos resultados están bajo las pautas del médico con el cual se consultó ya que permite saber al paciente si existe una anomalía en la cantidad de hemoglobinas oxigenadas que posee su cuerpo.

Tabla 34: Fiabilidad del dispositivo al medir el nivel de saturación de oxígeno

Nº Paciente	Medición dispositiva (SpO2)	Medición X004C (SpO2)	$E_{absoluto}$	$E_{relativo}$	$E_{relativo}$ (%)
1	93	93	0	0,000	0,000
2	93	92	1	0,011	1,075
3	95	97	2	0,021	2,105
4	93	94	1	0,011	1,075
5	90	93	3	0,033	3,333
6	90	93	3	0,033	3,333
7	90	95	5	0,056	5,556
8	93	94	1	0,011	1,075
9	93	91	2	0,022	2,151
10	91	96	5	0,055	5,495
11	91	94	3	0,033	3,297
12	91	93	2	0,022	2,198
13	93	92	1	0,011	1,075
14	93	91	2	0,022	2,151
15	92	95	3	0,033	3,261
16	93	95	2	0,022	2,151
17	93	91	2	0,022	2,151
18	92	92	0	0,000	0,000
19	93	97	4	0,043	4,301
20	94	91	3	0,032	3,191
Promedio	92,3	93,45	2,25	0,024	2,449
Fiabilidad					97,551

Elaborado por: El investigador

En la figura 154 se puede visualizar el comportamiento de las mediciones de SpO2 en donde existe una tendencia de mayor precisión mientras el rango del nivel de saturación se encuentre entre 90% y 93%, ya que cuando el valor real es superior a los 97% el dispositivo comienza a tener un ligero incremento de error de medición. Cabe recalcar que la exactitud que presente la medición dependerá de cómo el usuario posicione su dedo sobre el dispositivo ya que un mal posicionamiento del dedo sobre el sensor provocara una medición errónea.

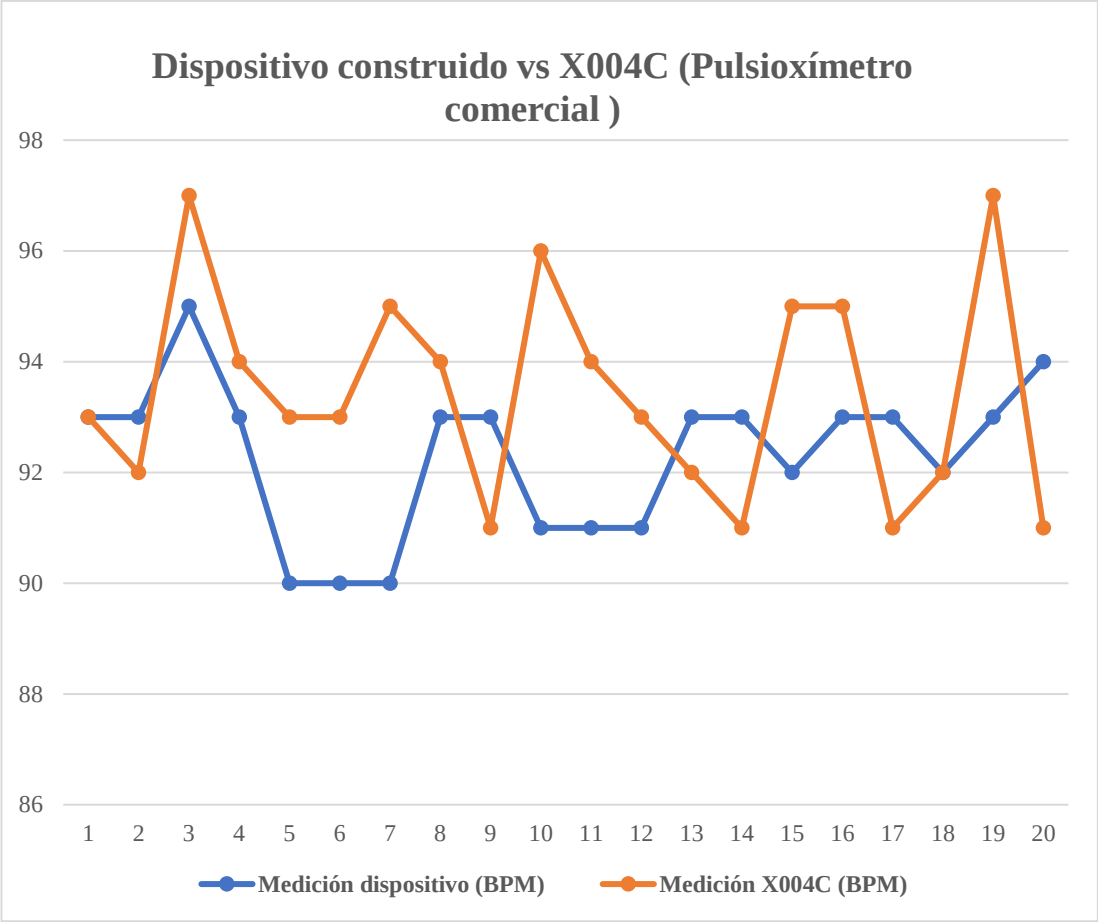


Figura 154: Comportamiento de la medición de SpO2

Elaborado por: El investigador

3.2.11.3. Medición de temperatura corporal

Una vez terminada la implementación del dispositivo es necesario comparar los valores de medición de temperatura para establecer la fiabilidad del dispositivo creado. En la tabla 35 se realizó pruebas con un total de 20 personas que dio como resultado una variación de 0.39°C con respecto a un termómetro digital infrarrojo. Las mediciones del dispositivo tienen un error de medición promedio de 1.07% siendo la medición más precisa y estable ya que tiene una fiabilidad de 98.93%. Este ligero error permite detectar si la persona tiene un alza térmica (fiebre) que será registrado en la base de datos para que el algoritmo de inteligencia artificial pueda dar un diagnóstico.

Tabla 35: Fiabilidad del dispositivo al medir la temperatura corporal

N° Paciente	Medición dispositivo (°C)	Medición TWQ-01 (°C)	$E_{absoluto}$	$E_{relativo}$	$E_{relativo}$ (%)
1	36,54	37,10	0,56	0,015	1,533
2	36,36	36,70	0,34	0,009	0,935
3	36,10	36,32	0,22	0,006	0,609
4	36,32	36,84	0,52	0,014	1,432
5	37,20	37,64	0,44	0,012	1,183
6	36,10	36,60	0,5	0,014	1,385
7	36,20	36,70	0,5	0,014	1,381
8	37,02	36,24	0,78	0,021	2,107
9	37,80	37,68	0,12	0,003	0,317
10	37,20	37,37	0,17	0,005	0,457
11	37,39	37,71	0,32	0,009	0,856
12	36,15	36,95	0,8	0,022	2,213
13	36,39	36,13	0,26	0,007	0,714
14	37,05	37,33	0,28	0,008	0,756
15	36,23	36,52	0,29	0,008	0,800
16	36,03	36,28	0,25	0,007	0,694
17	36,06	36,68	0,62	0,017	1,719
18	36,37	36,71	0,34	0,009	0,935
19	36,04	36,28	0,24	0,007	0,666
20	36,90	36,62	0,28	0,008	0,759
Promedio	36,57	36,82	0,3915	0,011	1,073
Fiabilidad					98,927

Elaborado por: El investigador

En la figura 155 se muestra el comportamiento de las mediciones de temperatura que presenta una tendencia de mayor precisión mientras el rango del nivel de saturación se encuentre entre 36.0°C y 36.4°C, ya que cuando el valor real es superior a los 37°C el dispositivo empieza a mostrar un pequeño aumento en el error de medición. La exactitud de la medida de temperatura dependerá de la zona a la que apunte el dispositivo, por esta razón se optó por colocar el sensor cerca de la palma del usuario.

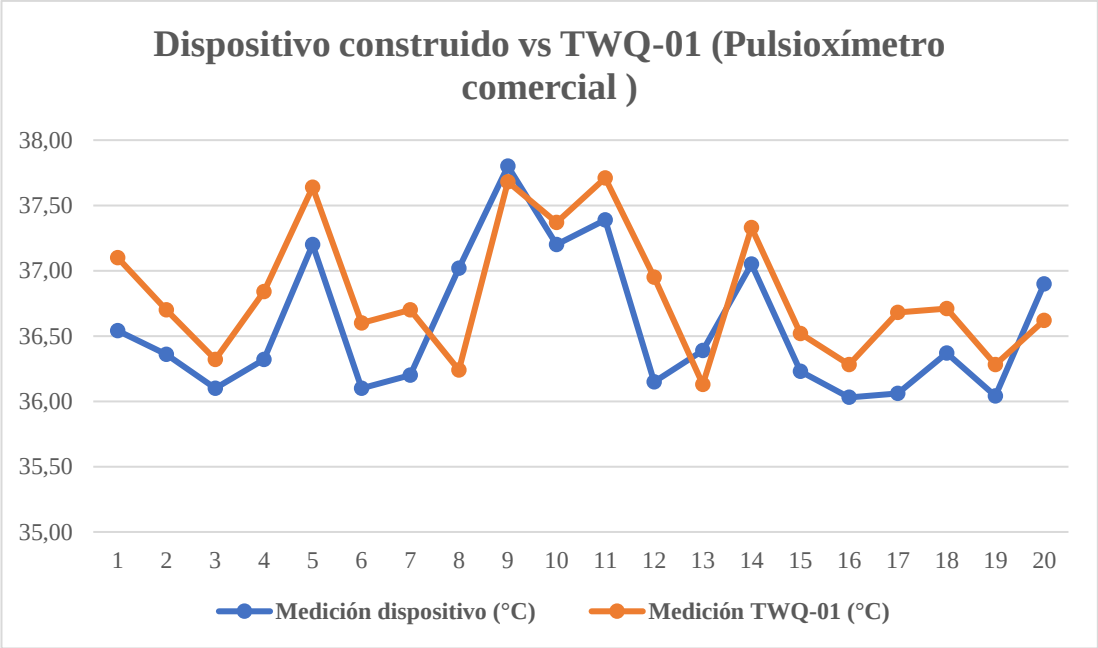


Figura 155: Comportamiento de la medición de temperatura corporal

Elaborado por: El investigador

3.2.11.4. Pruebas de funcionamiento del asistente virtual de telemedicina

Una vez verificado el funcionamiento del dispositivo electrónico de acuerdo con las pautas de la doctora, se realiza las pruebas de funcionamiento del asistente virtual de telemedicina para el diagnóstico médico de covid19, resfriado común y rinitis alérgica. La identidad de las personas es reservada para respetar la privacidad de los pacientes con los que trabaja la doctora, en la tabla 36 se muestra los resultados obtenidos de los 20 pacientes con los que se trató, de los cuales solo dos diagnósticos fueron erróneos, el primer diagnóstico incorrecto fue un caso de resfriado común y covid19 que se confundió con resfriado común. Los resultados se encuentran mejor detallados en la figura 156, donde se observa que la clase rinitis alérgica muestra 3 aciertos con 0 errores, la clase resfriado común obtuvo 9 aciertos con 1 error y la clase covid19 tuvo 6 aciertos con 1 error. Estos resultados están de acuerdo con la precisión del modelo seleccionado en la sección 3.2.9.4, en donde la clase rinitis tenía el más alto porcentaje de precisión al momento de clasificar los síntomas de la enfermedad.

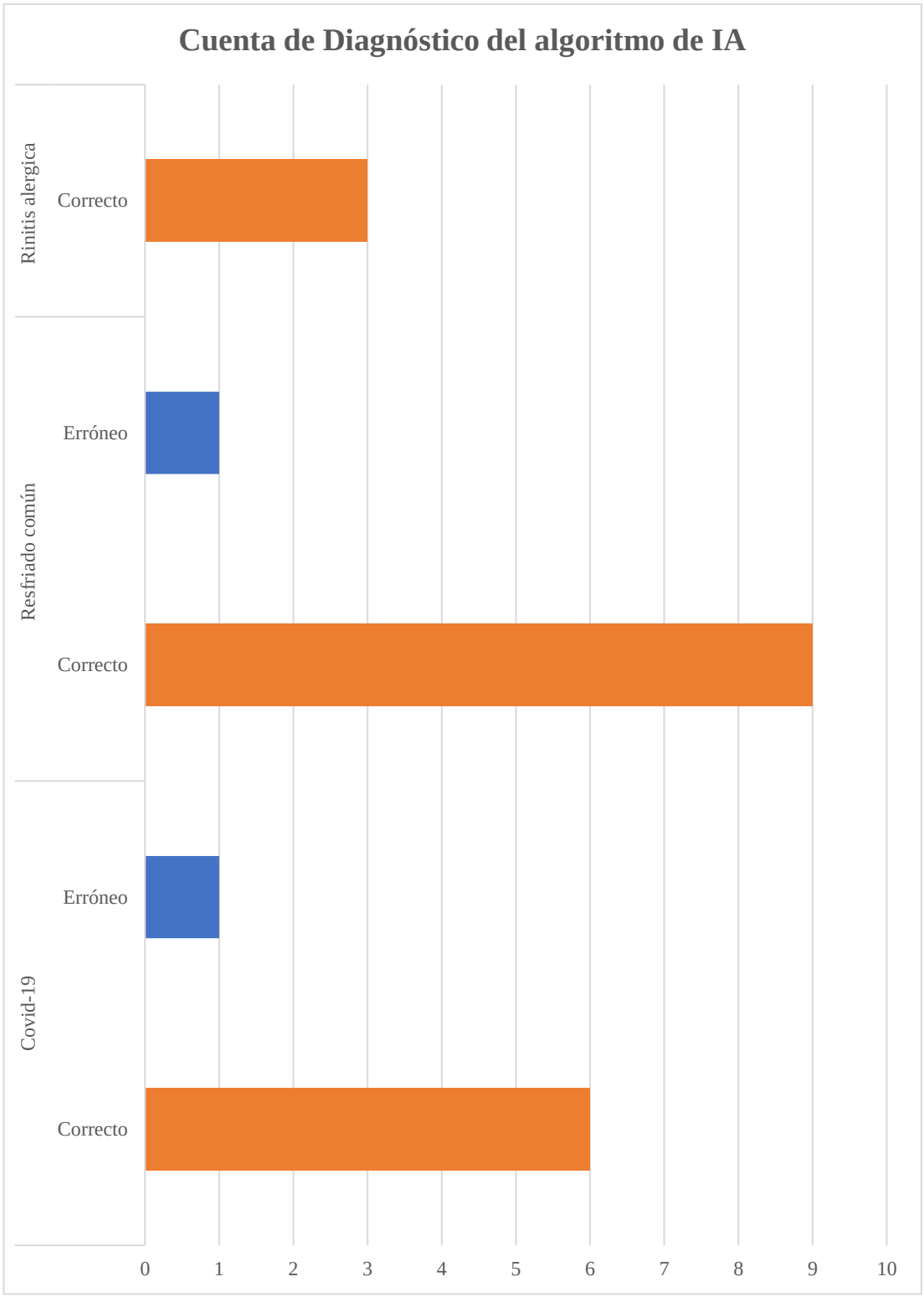


Figura 156: Diagnóstico del algoritmo de inteligencia artificial verificados por la doctora

Elaborado por: El investigador

Tabla 36: Pruebas realizadas del asistente virtual de telemedicina

Nº	Género	Edad	Temperatura (°C)	Frecuencia cardíaca (BPM)	Saturación de oxígeno (SpO2)	Síntomas	Diagnóstico del algoritmo de IA	Revisión de la doctora
1	femenino	33	36,54	72	93	Dolor de garganta leve y tos seca	Resfriado común	Correcto
2	masculino	29	36,36	75	93	Congestión nasal, Dolor de garganta moderado, tos con flema y Malestar general	Resfriado común	Erróneo
3	masculino	47	36,10	63	95	Dolor de garganta moderado, Dolor articular moderado, Malestar general, Dolor de barriga y Vomito	Covid-19	Correcto
4	femenino	29	36,32	74	93	Congestión nasal, Estornudos, Malestar general, Dolor de cabeza leve	Resfriado común	Correcto
5	femenino	36	38,03	63	90	Congestión nasal, Tos seca, Fiebre, Dolor de cabeza fuerte, Diarrea y Escalofríos	Covid-19	Correcto
6	femenino	37	36,10	75	90	Dolor de garganta moderado, Tos con flema, Sensación de ahogo, Dolor de cabeza leve, Dolor de pecho y Escalofríos	Covid-19	Correcto
7	masculino	25	36,20	78	90	Congestión nasal, Dolor de garganta leve, Tos seca y Malestar general	Resfriado común	Correcto

8	femenino	35	37,02	62	93	Dolor de garganta leve, Tos seca, Malestar general y Dolor de cabeza moderado	Resfriado común	Correcto
9	masculino	45	37,80	74	93	Secreción nasal, Congestión nasal, Picazón nasal y Hinchazón de los ojos	Rinitis alérgica	Correcto
10	masculino	44	38,40	62	91	Congestión nasal y Fiebre	Resfriado común	Correcto
11	masculino	49	37,96	68	91	Dolor de garganta fuerte y Fiebre	Covid-19	Erróneo
12	masculino	51	36,15	63	91	Tos seca, Dolor muscular y Lagrimeo	Resfriado común	Correcto
13	femenino	51	36,39	72	93	Dolor de garganta leve, Tos con flema, Nauseas, Pérdida de apetito y Escalofríos	Covid-19	Correcto
14	femenino	49	37,05	72	93	Congestión nasal, Tos seca, Picazón nasal e Hinchazón de los ojos	Rinitis alérgica	Correcto
15	femenino	42	36,23	74	92	Congestión nasal, Tos con flema y Malestar general	Resfriado común	Correcto
16	masculino	46	36,03	66	93	Congestión nasal, Tos seca, Dolor de barriga y Diarrea	Covid-19	Correcto
17	femenino	25	38,90	77	93	Tos seca, Fiebre, Malestar general y Dolor de cabeza leve	Resfriado común	Correcto
18	masculino	48	36,37	64	92	Congestión nasal, Dolor de cabeza leve, Malestar general y Dolor de cabeza leve	Resfriado común	Correcto

19	femenino	50	38,00	69	93	Tos con flema, Fiebre, Dolor articular leve, Malestar general y Diarrea	Covid-19	Correcto
20	masculino	49	36,90	74	94	Congestión nasal, Estornudos y Picazón nasal	Rinitis alérgica	Correcto

Elaborado por: El investigador

La identidad de las personas es reservada para respetar la privacidad de los pacientes con los que trabaja la doctora, las evidencias de las pruebas se observan en la figura 157.

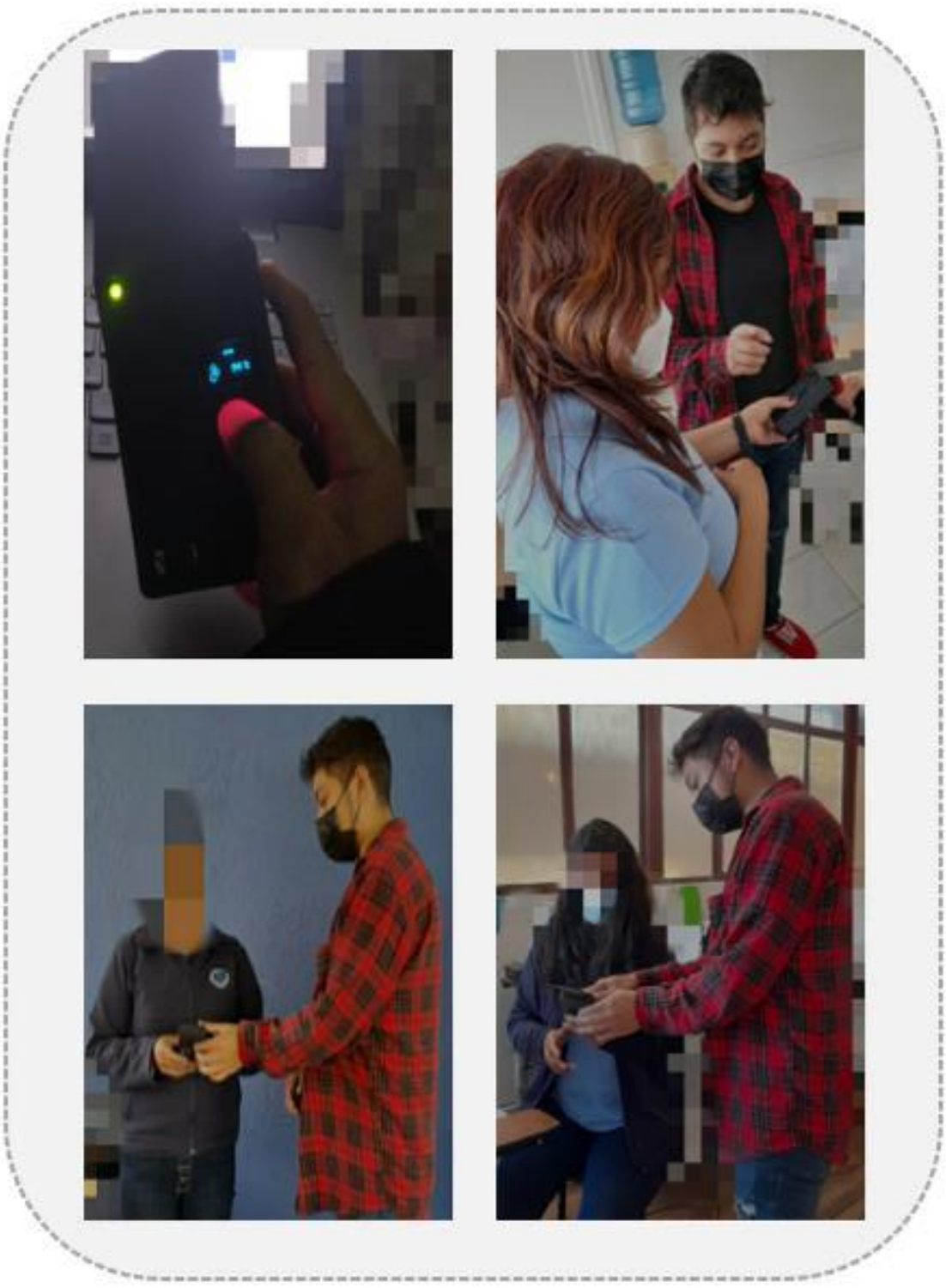


Figura 157: Pruebas del asistente virtual de telemedicina en pacientes del consultorio de la doctora

Elaborado por: El investigador

3.2.12. Presupuesto del proyecto de investigación

El precio de implementación del dispositivo electrónico de medición de triaje de signos vitales en tiempo real se encuentra en la tabla 37, donde se especificó todos los componentes electrónicos, la construcción del PCB con su case, materiales de oficina y materiales necesarios para desarrollar el proyecto como lo son las impresiones, internet, almuerzos, transporte y capital extra en caso de imprevistos como daño de sensores.

Tabla 37: Presupuesto de implementación del proyecto de investigación

Ítem	Material	Unidad	Cantidad	Valor unidad ($)	Valor total ($)
1	MLX90614	c/u	1	15,00	15,00
2	MAX30100	c/u	1	7,50	7,50
3	ESP8266	c/u	2	8,99	17,98
4	Diodo led rojo	c/u	1	0,10	0,10
5	Diodo les verde	c/u	1	0,10	0,10
6	Pantalla Oledd de 0,96"	c/u	1	8,00	8,00
7	Bateria Lipo de una celda	c/u	1	13,00	13,00
8	TP4056	c/u	1	2,50	2,50
9	Impresión PCB con CNC	c/u	1	10,00	10,00
10	Estaño	c/u	1	1,50	1,50
11	Malla	c/u	1	2,50	2,50
12	Cables arduino Macho-Hembra (kit)	c/u	1	3,20	3,20
13	Espadines Hembra (kit de 40)	c/u	1	0,60	0,60
14	Espadines Macho (kit de 40)	c/u	1	0,40	0,40
15	Tornillos	c/u	4	0,15	0,60
16	Cautín de madera	c/u	1	3,00	3,00
17	Pasta de soldar	c/u	1	2,50	2,50
18	Carcasa protectora (Horas)	c/u	11	2,00	22,00
19	Impresiones	c/u	200	0,05	10,00
20	Internet	meses	6	22,00	132,00
21	Pistola de silicona	c/u	1	4,00	4,00
22	Barras de silicona	c/u	4	0,25	1,00
23	Transporte urbano	c/u	160	0,30	48,00
24	Almuerzos	c/u	50	2,00	100,00
25	Flash Memory	c/u	1	8,00	8,00
26	Libros	c/u	2	15,00	30,00
27	Curso en “Udemy”	c/u	1	15,00	15,00

Subtotal	458,48
Imprevistos (15%)	68.77
Total	527,25

Elaborado por: El investigador

En cuanto al presupuesto utilizado para desarrollar la aplicación Web del asistente virtual de telemedicina, se empleó software libre y gratuito como lo son:

- Python 3.10
- HTML, CSS, Javascript
- Bootstrap 5
- Base de datos MySQL
- Servidor Google Cloud con 3 meses gratuitos

Por otra parte, el precio de mano de obra se basa de acuerdo con el sueldo básico de un ingeniero en Telecomunicaciones es de $ 700.00 según ley orgánica de transferencia y acceso a la información en su artículo 7. Para encontrar la remuneración por hora se debe considerar los 22 días laborales del mes a 8 horas de trabajo.

$$Remuneración_{diaria} = \frac{Remuneración_{mensual}}{Días\ laborales} = \frac{\$\ 700}{22} = \$\ 31.82$$

$$Remuneración_{hora} = \frac{Remuneración_{diaria}}{Horas\ laborales} = \frac{\$\ 31.82}{8} = \$\ 3.98$$

Por consiguiente, el precio de mano obra es de $ 477.6 ya que la implementación de la propuesta tomo un total de 120 horas laborales; adicionalmente, el costo de diseño tiene un valor de $ 43.78 ya que el tiempo empleado fue de 11 horas.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Las pruebas de funcionamiento ratificaron el cumplimiento del algoritmo de inteligencia artificial, el modelo Máquina de Soporte de Vectores fue la mejor opción ya que manejo adecuadamente la clase “covid19”, “resfriado” y “rinitis” con una precisión de 0.85, 0.88 y 1 respectivamente, donde la clase que mejor se desempeña en el modelo es la clase “rinitis”. Lo que le da como resultado una precisión promedio de 0.91, haciendo al modelo apto para el uso médico dentro de las pautas establecidas por la doctora colaboradora.
- Los resultados obtenidos del dispositivo electrónico en la medición de frecuencia cardiaca (BPM), saturación de oxígeno en la sangre (SpO2) y temperatura corporal (°C) arrojaron una fiabilidad del 95.48%, 97.55% y 98.93% respectivamente. Esto hace que el dispositivo pueda ser empleado por el paciente para verificar sus signos vitales en tiempo real y detectar si existe alguna anomalía en sus medicines, estos datos son enviados al servidor de Google Cloud con un retardo de 2 segundos, logrando así una reducción de la latencia del sistema y la sobrecarga del servidor.
- La aplicación Web llamada “Baymax” utilizada para el diagnosticar a pacientes con covid19, resfriado común y rinitis alérgica, fue alojada en el servidor de Google Cloud para lograr que los usuarios puedan acceder a la misma desde cualquier dispositivo con conexión a internet sin requerir ninguna instalación previa. Al utilizar el framework Flask juntamente con Bootstrap 5 se logró una aplicación responsiva, dinámica y robusta ya que cuenta con seguridades tanto para el usuario como para la aplicación Web. Además, las herramientas de HTML, CSS, Javascript, Python y MySQL utilizadas en el desarrollo del proyecto permiten que la aplicación web sea escalable dependiendo de las necesidades que presenten los pacientes.

4.2. Recomendaciones

- Con un mayor presupuesto dependiendo el costo de los nuevos componentes se puede lograr que la medición de signos vitales tenga una mayor fiabilidad, utilizando sensores de grado médico, debido a que el sensor MAX30100 dependen en gran medida del posicionamiento del mismos, resultando en que si el dedo del paciente está mal posicionado puede provocar un ligero aumento de error en las mediciones, de igual manera sucede con el sensor MLX90614 que depende de la dirección en la que se apunte el infrarrojo ya que si el lugar donde apunta es más alejado del centro del cuerpo la medición tiende a disminuir.
- Se recomienda acondicionar los datos del dataset de acuerdo con la utilidad que se dará a la inteligencia artificial, por lo que el primer paso es balancear los datos caso contrario el modelo de inteligencia artificial presentará un sesgo que tendera a predecir en mayor medida las clases que tengan más cantidad de datos. Esto se realiza para lograr que la precisión del modelo aumente; no obstante, si la forma de ordenar los datos es errónea puede conducir a una mala interpretación del investigador al momento de seleccionar el algoritmo que más se adecue al proyecto.
- Para ampliar el alcance del proyecto de investigación es recomendable indagar más bases de datos de otras enfermedades con el objetivo de aumentar los diagnósticos que pueda realizar el algoritmo de inteligencia artificial, asemejándose más a las consultas hecha por un médico general.

BIBLIOGRAFÍA

[1] E. R. Society, The global impact of respiratory disease, Sherri Damlo, 2021.

[2] L. Silva, D. Callejas, C. Silva y G. Silva, «Perfil epidemiológico de infecciones respiratorias agudas en pacientes pediátricosen ecuador,» *Universidad Técnica de Ambato,* vol. 7, nº 2, pp. 1-6, 2022.

[3] P. Paredes, G. Celis, I. Toapanta y L. Bravo, «Perfil epidemiológico del Servicio de Pediatría del Hospital General Ambato,» *Revista Médica Científica CAMbios,* vol. 18, nº 2, pp. 18-23, 2019.

[4] M. Gandhi, V. Singh y V. Kumar, «IntelliDoctor – AI based Medical Assistant,» *IEEE,* pp. 162-168, 2019.

[5] S. Espinoza, «Desarrollo e implementación de una plataforma web con chatbot para la comunicación activa entre usuario e información del portafolio de servicio de la empresa Electricsystems de la ciudad de Guayaquil,» Universidad de Guayaquil, Guayaquil, 2020.

[6] F. Garibay, «Diseño e implementación de un asistente virtual (chatbot) para ofrecer atención a los clientes de una aerolínea mexicana por medio de sus canales conversacionales,» INFOTEC, Ciudad de México, 2020.

[7] E. Artica, «Implementación de un asistente virtual para la atención al cliente en Electrocentro S. A. de Huancayo,» Universidad Continental, Huancayo, 2020.

[8] G. Bonales, N. Pradilla y E. Citlati, «Chatbot como herramienta comunicativa durante la crisis sanitaria de la COVID-19 en España,» Universidad Complutense de Madrid, Madrid, 2020.

[9] M. d. S. d. Ecuador, «Minsalud,» Ministerio de Salud de Ecuador, 2021 Marzo 20. [En línea]. Available: https://www.minsalud.gov.co/salud/Paginas/Infecciones-Respiratorias-Agudas-(IRA).aspx#:~:text=La%20Infecci%C3%B3n%20Respiratoria%20Aguda%20(IRA,duran%20menos%20de%202%20semanas.. [Último acceso: 2022 Junio 14].

[10] Y. Bayona y J. Niederbacher, «Infecciones respiratorias virales en pediatría: generalidades sobre fisiopatogenia, diagnóstico y algunos desenlaces clínicos,» *Revista de los estudiantes de medicina de la universidad industrial de santander,* vol. 28, nº 1, pp. 133-141, 2014.

[11] W. Soler, M. Gómez, E. Bragulat y A. Álvarez, «El triaje: herramienta fundamental en urgencias y emergencias,» *Scielo*, vol. 33, nº 1, pp. 55-68, 2010.

[12] M. d. S. d. Ecuador, «Minsalud,» Ministerio de Salud de Ecuador, 14 Junio 2022. [En línea]. Available: https://www.minsalud.gov.co/salud/PServicios/Paginas/triage.aspx. . [Último acceso: 20 Noviembre 2022].

[13] D. Evans, B. Hodgkinson y J. Berry, «Vital signs in hospital patients: a systematic review,» *Int J Nurs Stud*, vol. 38, nº 6, pp. 643-650, 2001.

[14] S. Kumara y C. Krishnamoorthib, «Desarrollo de sensores táctiles portátiles basados en transducción eléctrica para monitores de signos vitales humanos: Fundamentos, metodologías y aplicaciones,» *ELSEVIER*, vol. 321, nº 1, pp. 11-18, 2021.

[15] G. Moreno, J. Vélez, M. Campuzano, J. Zambrano y R. Vera, «Monitorización invasiva y no invasiva en pacientes ingresados a UCI,» *RECIMUNDO,* vol. 5, nº 2, pp. 280-292, 2021.

[16] P. Rolfe, Z. Yan y S. Jinwei, «Invasive and non-invasive measurement in medicine and biology: Calibration issues,» *SPIE*, vol. 7544, nº 12, pp. 1-8, 2010.

[17] A. Carrillo, «Sistema de telemedicina basado en IOT para monitoreo de pacientes con enfermedades respiratorias,» Universidad Técnica de Ambato, Ambato, 2022.

[18] O. Organización Mundial de la Salud, Manejo clínico de la COVID-19, Ginebra: OMS, 2021.

[19] L. J, L. X y Q. S, «WHO,» Organización Mundial de la Salud, 29 Marzo 2020. [En línea]. Available: https://www.who.int/es/news-room/commentaries/detail/modes-of-transmission-of-virus-causing-covid-19-implications-for-ipc-precaution-recommendations. [Último acceso: 19 Noviembre 2022].

[20] OMS, «WHO,» Organización Mundial de la Salud, 16 Julio 2020. [En línea]. Available: https://www.who.int/es/health-topics/coronavirus#tab=tab_3. [Último acceso: 20 Noviembre 2022].

[21] O. P. d. l. Salud, «PAHO,» 7 Agosto 2020. [En línea]. Available: https://iris.paho.org/handle/10665.2/52551. [Último acceso: 20 Noviembre 2022].

[22] A. Varricchio, I. La Mantia, F. Paolo Brunese y G. Ciprandi, «Inflammation, infection, and allergy of upper airways: new insights from national and real-world studies,» *Italian Journal of Pediatrics,* vol. 46, nº 18, pp. 1-10, 2020.

[23] I. Khalid, S. Sattar, R. Nims y J. Rubino, «Combating SARS-CoV-2: leveraging microbicidal experiences with other emerging/re-emerging viruses,» *PeerJ,* vol. 8, nº 15, pp. 1-24, 2020.

[24] I. Saha, Standard Treatment Guidelines for Primary Health Care, Guyana: Ministerio de Salud Pública, 2020.

[25] M. Javanian, A. Babazadeh, S. Ebrahimpour, M. Shokri y M. Bayani, «Clinical and laboratory findings of patients with the possible diagnosis of influenza hospitalized in affiliated hospitals of Babol University of Medical Sciences,» *Sciendo,* vol. 31, nº 3, pp. 113-116, 2018.

[26] M. Dykewicz y D. Hamilos, «Rinitis y Sinusitis,» *The Journal of allergy and clinical immunology,* vol. 125, nº 2, pp. 103-115, 209.

[27] A. Bourdin, D. Gras, I. Vachier y P. Chanez, «Vía aérea superior x 1: rinitis alérgica y asma: enfermedad unida a través de células epiteliales,» *BMJ,* vol. 64, nº 11, pp. 999-1004, 2009.

[28] D. Wallace, M. Dykewicz, J. Oppenheimer, J. Portnoy y D. Lang, «Tratamiento farmacológico de la rinitis alérgica estacional: sinopsis de la orientación del grupo de trabajo conjunto de 2017 sobre parámetros de práctica,» *Annals of Internal Medicine,* vol. 167, nº 2, pp. 876-881, 2017.

[29] M. Gibson y A. Younes, «Comprender los síntomas del resfriado común y la influenza,» *The Lancet,* vol. 5, nº 11, pp. 718-725, 2005.

[30] I. Guasch, Electrotecnia, Madrid: McGraw-Hill Interamericana de España S.L., 2006.

[31] D. N. Peña, «Microcontroladores - Arquitectura, programación y aplicación,» Atlantic Internacional University Honolulu, Hawai , 2008.

[32] M. Bates, PIC Microcontrollers, Amsterdam: Newnes, 2011.

[33] Y. Jong-Ryul, H. Eugin y S. Kwon-Kim, Sensores para Monitoreo de Signos Vitales, Beijing: MDPI, 2021.

[34] Naylamp, «Naylamp Mechatronics,» Naylamp Mechatronics SAC, 6 Junio 2021. [En línea]. Available: https://naylampmechatronics.com/. [Último acceso: 4 Enero 2022].

[35] D. Greenfield, Artificial Intelligence in Medicine: Applications, implications, and limitations, Harvard University, 2019.

[36] P. Campbell, «5 industrias con mayor probabilidad de transformarse con el aprendizaje,» Learnwoo, 4 Enero 2022. [En línea]. Available: https://learnwoo.com/machine-learning/. [Último acceso: 28 Noviembre 2022].

[37] N. Balparda, «Introducción a Machine Learning,» Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento de Uruguay, 24 Noviembre 2020. [En línea]. Available: https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/sites/agencia-gobierno-electronico-sociedad-informacion-conocimiento/files/2020-11/20201124%20-%20Introducci%C3%B3n%20a%20Machine%20Learning.pdf. [Último acceso: 28 Noviembre 2022].

[38] P. Ríos y C. Chávez, «Repositorio Universidad Técnica de Ambato,» 30 Febrero 2021. [En línea]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/34043. [Último acceso: 28 Noviembre 2022].

[39] Y. Zhang, New Advances in Machine Learning, Portsmouth: InTech, 2010.

[40] N. Andreas y T. Fredrik, Supervised Machine Learning, Uppsala: Universidad de Uppsala, 2019.

[41] A. Mishra, «Towards Data Science,» TDS, 24 febrero 2018. [En línea]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234. [Último acceso: 28 Noviembre 2022].

[42] M. Mohtashim, Machine Learning with python, Hyderabad: Tutorials Point Pvt. Ltd., 2019.

[43] S. Walkera, W. Khana, K. Katica, W. Maassenab y W. Zeilera, «Accuracy of different machine learning algorithms and added-value of predicting aggregated-level energy performance of commercial buildings,» *ELSEVIER*, vol. 209, nº 50, pp. 1-15, 2020.

[44] J. Barrios, «Health Big Data,» BIG DATA, 26 Julio 2019. [En línea]. Available: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/. [Último acceso: 30 Noviembre 2022].

[45] B. Zach, «Statology,» Bobbitt, 8 Septiembre 2021. [En línea]. Available: https://www.statology.org/f1-score-in-r/. [Último acceso: 30 Noviembre 2022].

[46] Y. Arias, Sistema Web de reconocimiento y clasificación de patologías a través de imágenes médicas basado en técnicas de aprendisaje de máquina, Bucaramanga: Universidad Autónoma de Bucaramanga, 2019.

[47] I. C. Education, «International Business Machines,» IBM, 2 Julio 2020. [En línea]. Available: https://www.ibm.com/cloud/learn/natural-language-processing. [Último acceso: 30 Noviembre 2022].

[48] M. Borja, «An´alisis de las herramientas de procesamiento de lenguaje natural para estructurar textos médicos,» Tecnum, San Sebasti´an, 2020.

[49] D. Khurana, A. Koli, K. Khatter y S. Singh, «Procesamiento del lenguaje natural: estado del arte, tendencias y desafíos actuales,» *Multimedia Tools and Applications,* vol. 22, nº 4, pp. 1-32, 2022.

[50] A. Cortez, H. Vega y J. Quispe, «Procesamiento de lenguaje natural,» *Revista de Ingeniería de Sistemas e Informática,* vol. 6, nº 2, pp. 45-54, 2009.

[51] S. Lyer, «AISERA,» AISX, 5 Enero 2022. [En línea]. Available: https://aisera.com/chatbots-virtual-assistants-conversational-ai/. [Último acceso: 2 Diciembre 2022].

[52] S. Stumpf, «Omnia Health,» Informa Market, 13 Enero 2020. [En línea]. Available: https://insights.omnia-health.com/technology/chatbots-healthcare-ai-assistants-rescue. [Último acceso: 2 Diciembre 2022].

[53] S. Bansal, «Healthcare chatbot,» Ameyo, 23 Marzo 2022. [En línea]. Available: https://addevice.io/blog/how-to-make-a-chatbot-from-scratch/. [Último acceso: 2 Diciembre 2022].

[54] Addevice, «How to Make a Chatbot: Best Practices, Technologies & Business Benefit,» Stonestep, 1 Marzo 2022. [En línea]. Available: https://addevice.io/blog/how-to-make-a-chatbot-from-scratch/. [Último acceso: 2 Diciembre 2022].

[55] S. Fedaghi, «Developing Web Applications,» *International Journal of Software Engineering and Its Applications,* vol. 5, nº 2, pp. 57-68, 2011.

[56] C. Smith, «Forty Creates,» F8C, 23 Marzo 2021. [En línea]. Available: https://forty8creates.com/web-app-versus-native-app-whats-difference/. [Último acceso: 3 Diciembre 2022].

[57] C. Mateu, Desarrollo de aplicaciones web, Barcelona: FUOC, 2004.

[58] J. Conallen, «Modeling Web aplication architectures with UML,» *COMMUNICATIONS OF THE ACM,* vol. 42, nº 10, pp. 63-70, 1999.

[59] H. Mejia y V. Monteza, «Análisis comparativo de frameworks para el desarrollo de aplicaciones web en java,» *Rev. Ingeniería: Ciencia, Tecnología e Innovación,* vol. 1, nº 2, pp. 60-72, 2015.

[60] H. Molina, J. Ullman y J. Widom, Database Systems: The Complete Book, Pearson Prentice Hall, 2009.

[61] C. Győrödi, R. Győrödi y R. Sotoc, «Un estudio comparativo de modelos de bases de datos relacionales y no relacionales en una aplicación Web,» *International Journal of Advanced Computer Science and Applications,* vol. 6, nº 11, pp. 78-83, 2015.

[62] C. Penga y Z. Jiang, «Building a Cloud Storage Service System,» *ELSEVIER*, vol. 10, nº 1, p. 691–696, 2011.

[63] G. Chanchí y M. Ospina, «Arquitectura IoT para el desarrollo de sistemas de monitorización y análisis de variables fisiológicas en el área de asistencia médica,» *Investigación e Innovación en Ingenierías*, vol. 8, nº 3, pp. 1-13, 2020.

[64] LastMinuteEngineers, «LME,» 20 Febrero 2020. [En línea]. Available: https://lastminuteengineers.com/max30100-pulse-oximeter-heart-rate-sensor-arduino-tutorial/. [Último acceso: 7 Enero 2022].

[65] J. Zhang, «Development of a Non-contact Infrared Thermometer,» *International Conference Advanced Engineering and Technology Research*, vol. 153, nº 4, pp. 308-312, 2017.

ANEXOS

Anexo 1: Certificado de los datos

Dra. Maritza Paola Córdova Vivas
Dra. Verónica Alexandra Córdova Vivas
NOUMEDICINE
Consultorio Médico
MEDICINA GENERAL

CERTIFICACIÓN

Ingeniera, Mg
Pilar Urrutia
DECANA
Facultad de Ingeniería en Sistemas, Electrónica e Industrial
Presente
De mi consideración:

Por medio de la presente, yo Dra. Verónica Alexandra Córdova Vivas con cedula de identidad N° 1804173449 con registro del Senescyt N° 1804173449, Medico General, certifico que el estudiante Naythan Alexander Villafuerte Lozada de 23 años de edad con cédula de identidad N° 1850039718, acude a nuestra casa de salud donde nos solicita información de nuestras historias clínicas para realizar la tesis de grado con el tema **"Asistente virtual de telemedicina con dispositivo electrónico de medición de triaje para el diagnóstico médico de infecciones respiratorias utilizando inteligencia artificial.".** Lo cual se le brida parte de la información de nuestras historias clínicas y no en su totalidad, recalcando que es un documento médico legal, por lo que se guarda la confidencialidad de nuestros pacientes.

Se extiende el presente documento para los fines legales que al interesado le convenga

Atentamente:

Dra. Verónica Córdova
Médico General
C.I: 1804173449

0988 71 87 40 / 0984 71 72 54
noumedicine@gmail.com
Av. Pitágoras y José Ingenieros, Parroquia Picaihua
Ambato - Ecuador

Anexo 2: Certificado de la doctora

Dra. Martha Paola Córdova Vivas
Dra. Verónica Alexandra Córdova Vivas
NOUMEDICINE
Consultorio Médico
MEDICINA GENERAL

CERTIFICACIÓN

Ingeniero, Mg.
Pilar Urrutia
DECANA
Facultad de Ingeniería en Sistemas Electrónica e Industrial
Presente

Señora Decana:

Por medio de la presente, yo Dra. VERONICA ALEXANDRA CORDOVA VIVAS con cédula de identidad *N°*1804173449 con registro del Senescyt *N°* 1804173449, Médico General., certifico que he revisado el proyecto técnico "ASISTENTE VISRTUAL DE TELEMEDICINA CON DISPOSITIVO ELECTRONICO DE MEDICINA DE TRIAJE PARA EL DIAGNOSTICO MEDICO DE INFECCIONES RESPIRATORIAS UTILIZANDO INTELIGENCIA ARTIFICIAL", el mismo que fue diseñado en su total autoria por el Sr. NAYTHAN ALEXANDER VILLAFUERTE LOZADA, con cédula de identidad N° 1850039718 , estudiante de la carrera de Telecomunicaciones de la Universidad Técnica de Ambato.

Después de haber realizado las pruebas médicas necesarias me permito verificar la funcionabilidad del equipo y puedo validar que los resultados obtenidos hasta la fecha son aceptables y de gran ayuda en el área de detección y prevención contra el COVID. 19,RESFRÍO COMÚN, RINITIS ALERGICA, además indico que los sensores se visualizan dentro de los rangos médicos normales.

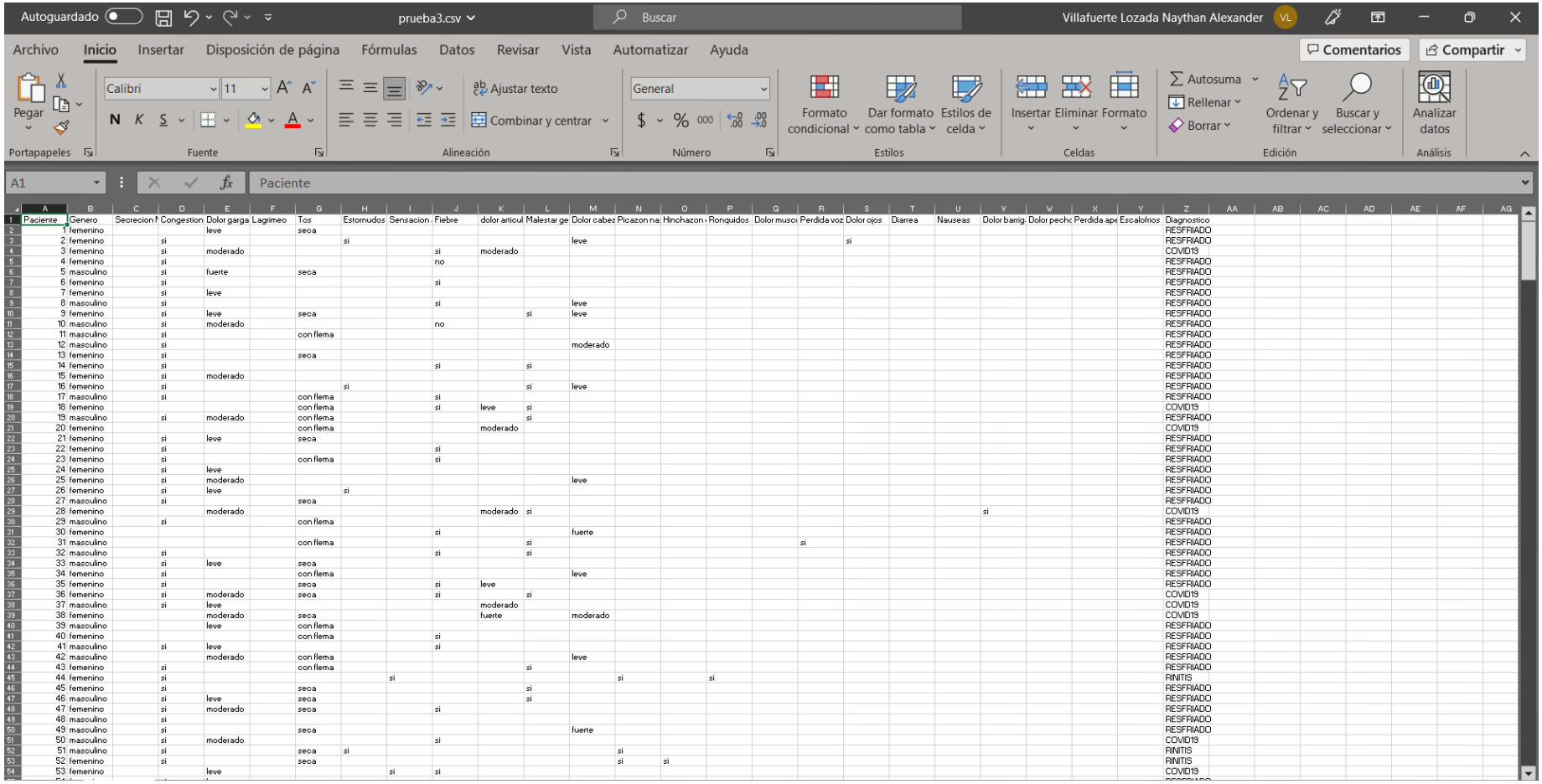
Saludos cordiales,
Atentamente,

Dra. Verónica Córdova
MEDICINA GENERAL
C.I. 1804173449

Dra.Verónica Alexandra Córdova Vivas
Médico General
C.I.: 1804173449

0988 71 87 40 / 0984 71 72 54
Av. Pitágoras y José Ingenieros, Parroquia Picaihua
Ambato - Ecuador

Anexo 3: Manipulación de los datos de entrenamiento de la IA utilizando Excel



Anexo 4: Programación del microcontrolador ESP8266 principal

```
//Importacion de librerias
#include <cambiar_ssid.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h>
//Librerias de pantalla oled
#define __DEBUG__
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
////////////////////////////////////////////////////////////////////////////
//Declaracion de variables
#define DEBUG_ARRAY(a) {for (int index = 0; index < sizeof(a) / sizeof(a[0]); index++)
{Serial.print(a[index]); Serial.print('\t');} Serial.println();};
const char separator = ','; // separador de trama
const int dataLength = 2; // el numero del incremento para los valores de la cadena
//Variables del broker mosquito
WiFiClient espClient;
String str = "";
String interruptor;
const char* mqtt_server = "0.0.0.0";
//variables de sensores
int bpm, spo2;
// Instanciar objeto (Sensor de temperatura)
Adafruit_MLX90614 termometroIR = Adafruit_MLX90614();
// variables para medi nivel de carga
int analogInPin  = A0;    // Declaracion pin analogo A0
int sensorValue;          // Variable para la medicion del nivel de carga
float calibration = 0.43; // Calibrar el valor usando un multimetro
int bat_percentage;       // variable del porcentaje de la bateria
// variables de pantalla oled
#define ancho_imagen 20
#define alto_imagen 20

#define ancho_imagen_2 50
#define alto_imagen_2 50


const unsigned char PROGMEM termometro[] = {
  0x00, 0x60, 0x00, 0x00, 0x90, 0x00, 0x00, 0x90, 0x00, 0x00, 0x90, 0x00, 0x00,
  0x00, 0x00, 0xb0, 0x00, 0x00, 0xb0, 0x00, 0x00, 0xb0, 0x00, 0x00, 0x90, 0x00, 0x01, 0x68, 0x00,
  0x01, 0x68, 0x00, 0x01, 0x68, 0x00, 0x01, 0x08, 0x00, 0x00, 0xf0, 0x00
};

const unsigned char PROGMEM corazon[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x0f, 0x00, 0x3f, 0x9f, 0xc0, 0x3f, 0xff, 0xc0, 0x7f,
  0xff, 0xe0, 0x7f, 0xff, 0xe0, 0x7f, 0xff, 0xe0, 0x3f, 0xff, 0xc0, 0x3f, 0xff, 0xc0, 0x3f, 0xff,
  0x00, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

const unsigned char PROGMEM saturacion[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0xe0, 0x00, 0x00, 0xa0, 0x00, 0x01,
  0xb0, 0x00, 0x01, 0x98, 0x00, 0x01, 0xfc, 0x00, 0x00, 0xf4, 0x00, 0x01, 0xf6, 0x00, 0x03, 0x72,
  0x03, 0xe0, 0x00, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};


const unsigned char PROGMEM termometro2[] = {
  0x00, 0x00, 0x01, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x06, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x08,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x38, 0x00, 0x00,
};

```

```
63  const unsigned char PROGMEM corazon2[] = {
64    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
65    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0xff, 0xe0, 0x01,
66    0xff, 0xc0, 0x00, 0x01, 0xff, 0xf0, 0x03, 0xff, 0xe0, 0x00, 0x03, 0xff, 0xfc, 0x07, 0xff, 0xf0,
67    0x00, 0x0f, 0xff, 0xfc, 0x0f, 0xff, 0xfc, 0x00, 0x0f, 0xff, 0xfe, 0x1f, 0xff, 0xfe, 0x00, 0x1f,
68    0x00, 0x00, 0x00, 0x3f, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00, 0x00, 0x00, 0x00,
69    0x00, 0x07, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03,
70    0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00,
71    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
72    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
73  };
74
75  const unsigned char PROGMEM saturacion2[] = {
76    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
77    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
78    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
79    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00,
80    0x00, 0x02, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04,
81    0x00, 0x00, 0x01, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x8e, 0x40, 0x00, 0x00, 0x00, 0x00,
82    0x00, 0x61, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
83    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
84    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
85  };
86
87  const unsigned char PROGMEM carga100[] = {
88    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f,
89    0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00,
90    0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0x00, 0x00, 0x00,
91    0x00, 0x0f, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x1c,
92    0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,
93    0x60, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x18, 0x7f, 0xf8, 0x60, 0x00,
94    0x00, 0x00, 0x00, 0x18, 0xff, 0xfc, 0x60, 0x00, 0x00, 0x00, 0x00, 0x18, 0xff, 0xfc, 0x60, 0x00, 0x00, 0x00,
95    0x00, 0x18, 0xff, 0xfc, 0x60, 0x00, 0x00, 0x00, 0x00, 0x18, 0xff, 0xfc, 0x60, 0x00, 0x00, 0x00, 0x00, 0x18, 0xff,
96    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
97  };
98

99  const unsigned char PROGMEM carga75[] = {
100   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
101   0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x01, 0x7f, 0xf0, 0x00, 0x00, 0x00,
102   0x00, 0x0f, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x1c,
103   0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00,
104   0xe0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xe0, 0x00,
105   0x00, 0x00, 0x38, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00,
106   0x38, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x38, 0x00,
107   0x00, 0xe0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x38, 0x7f, 0xf0, 0xe0,
108   0x00, 0x00, 0x00, 0x38, 0xff, 0xf8, 0xe0, 0x00, 0x00, 0x00, 0x38, 0xff, 0xf8, 0xe0, 0x00, 0x00,
109   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
110 };
111
112 const unsigned char PROGMEM carga50[] = {
113   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
114   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00,
115   0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00, 0x00,
116   0x00, 0x0f, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x1c,
117   0x00, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,
118   0x60, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00,
119   0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00,
120   0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x18, 0x00,
121   0x03, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
122 };
123

124 const unsigned char PROGMEM carga25[] = {
125   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
126   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0, 0x00,
127   0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0, 0x00, 0x00, 0x00,
128   0x00, 0x0f, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x3c,
129   0x00, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00,
130   0xc0, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0xc0, 0x00,
131   0xc0, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x3c, 0x00, 0x01, 0xc0, 0x00,
132   0x00, 0x00, 0x1f, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
133   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
134 };
```

```
135
136 const unsigned char PROGMEM carga0[] = {
137   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
138   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x80, 0x00,
139   0x00, 0x00, 0x00, 0x02, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xfc, 0x00, 0x00, 0x00,
140   0x00, 0x3f, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x30,
141   0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x07,
142   0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xfe, 0x00, 0x00,
143   0x00, 0x00, 0x1f, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00,
144   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
145 };
146
147 #define ancho_centigrados 15
148 #define alto_centigrados 15
149
150 const unsigned char PROGMEM centigrados[] = {
151   0x00, 0x00, 0x00, 0x00, 0x78, 0x1c, 0x88, 0x7e, 0x88, 0xc0, 0x49, 0x80, 0x01, 0x80, 0x01, 0x00,
152   0x01, 0x00, 0x01, 0x80, 0x01, 0x80, 0x00, 0xe0, 0x00, 0x7e, 0x00, 0x00, 0x00, 0x00
153 };
154 // Definir constantes
155 #define ANCHO_PANTALLA 128 // ancho pantalla OLED
156 #define ALTO_PANTALLA 64 // alto pantalla OLED
157
158 // Objeto de la clase Adafruit_SSD1306
159 Adafruit_SSD1306 display(ANCHO_PANTALLA, ALTO_PANTALLA, &Wire, -1);
160 ////////////////////////////////////////////////////////////////////////////
161 void setup() {
162  conectarWiFi();
163  oled();
164  client.setServer(mqtt_server, 1883);
165  client.setCallback(callback);
166   // Iniciar termómetro infrarrojo con Arduino
167   termometroIR.begin();
168 }

169
170 void loop() {
171   if (!client.connected()) {
172     reconnect();
173   }
174   client.loop();
175   while (Serial.available() > 0) {
176  envioCondicion();
177  }
178  bpm=data[0]; // guardamos el dato en la variable BPM
179  spo2=data[1]; // guardamos el dato en la variable SPO2
180   // Obtener temperaturas grados Celsius
181   float temperaturaAmbiente = termometroIR.readAmbientTempC();
182  // se aumenta el valor de "6" para calibrar el valor de la temepratura del cuerpo
183   float temperaturaObjeto = (termometroIR.readObjectTempC()+6);
184
185   // Mostrar información
186
187   String temperaturaObjeto2 = String(temperaturaObjeto,2);
188   String bpms = String(bpm);
189   String spo22 = String(spo2);
190   String sensores = temperaturaObjeto2+","+bpms+","+spo22;
191
192   const char *sensores2= sensores.c_str();
193   snprintf (msg, 75, sensores2);
194   client.publish("chatboot/sensores", msg);

195
196   delay(500);
197   ///Mostrar datos en pantalla oled
198   display.clearDisplay();
199   display.setTextColor(SSD1306_WHITE); // Color del texto
200   display.setTextSize(1.5); // Tamaño del texto
201   display.setCursor(31, 0);
202   display.println("TEMPERATURA");
203   display.drawBitmap(0,15, termometro2, ancho_imagen_2, alto_imagen_2, WHITE);
204   display.setTextSize(2); // Tamaño del texto
205   display.setCursor(38, 30);
206   display.println(temperaturaObjeto);
207   display.drawBitmap(100,30, centigrados, ancho_centigrados, alto_centigrados, WHITE);
208   display.display(); // Enviar a pantalla
209   delay(2000);
```

```
210
211   display.clearDisplay();
212   display.setTextColor(SSD1306_WHITE); // Color del texto
213   display.setTextSize(1.5); // Tamaño del texto
214   display.setCursor(57, 0);
215   display.println("BPM");
216   display.drawBitmap(0,15, corazon2, ancho_imagen_2, alto_imagen_2, WHITE);
217   display.setTextSize(2); // Tamaño del texto
218   display.setCursor(70, 30);
219   display.println(bpm);
220   display.display(); // Enviar a pantalla
221   delay(2000);
222
223   display.clearDisplay();
224   display.setTextColor(SSD1306_WHITE); // Color del texto
225   display.setTextSize(1.5); // Tamaño del texto
226   display.setCursor(55, 0);
227   display.println("SPO2");
228   display.drawBitmap(0,15, saturacion2, ancho_imagen_2, alto_imagen_2, WHITE);
229   display.setTextSize(2); // Tamaño del texto
230   display.setCursor(70, 30);
231   display.println(spo2);
232   display.setCursor(100, 30);
233   display.println("%");
234   display.display(); // Enviar a pantalla
235   delay(2000);
236
237   //Medir nivel de bateria
238   sensorValue = analogRead(analogInPin);
239   float voltage = (((sensorValue * 3.3) / 1024) * 2 + calibration);
240   bat_percentage = voltaje * 100;
241   if (bat_percentage>75){
242     display.clearDisplay();
243     display.setTextColor(SSD1306_WHITE); // Color del texto
244     display.setTextSize(1.5); // Tamaño del texto
245     display.setCursor(2, 0);
246     display.println("CARGA DEL DISPOSITIVO");
247     display.drawBitmap(0,10, carga100, ancho_imagen_2, alto_imagen_2, WHITE);
248     display.setTextSize(4); // Tamaño del texto
249     display.setCursor(50, 30);
250     display.println(bat_percentage);
251     display.setTextSize(1); // Tamaño del texto
252     display.setCursor(120, 30);
253     display.println("%");
254     display.display(); // Enviar a pantalla
255     delay(2000);
256   }
257   else if (bat_percentage>50 && bat_percentage<=75){
258     display.clearDisplay();
259     display.setTextColor(SSD1306_WHITE); // Color del texto
260     display.setTextSize(1.5); // Tamaño del texto
261     display.setCursor(2, 0);
262     display.println("CARGA DEL DISPOSITIVO");
263     display.drawBitmap(0,10, carga75, ancho_imagen_2, alto_imagen_2, WHITE);
264     display.setTextSize(4); // Tamaño del texto
265     display.setCursor(50, 30);
266     display.println(bat_percentage);
267     display.setTextSize(1); // Tamaño del texto
268     display.setCursor(120, 30);
269     display.println("%");
270     display.display(); // Enviar a pantalla
271     delay(2000);
272   }
```

```
  else if (bat_percentage>25 && bat_percentage<=50){
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE); // Color del texto
    display.setTextSize(1.5); // Tamaño del texto
    display.setCursor(2, 0);
    display.println("CARGA DEL DISPOSITIVO");
    display.drawBitmap(0,10, carga50, ancho_imagen_2, alto_imagen_2, WHITE);
    display.setTextSize(4); // Tamaño del texto
    display.setCursor(50, 30);
    display.println(bat_percentage);
    display.setTextSize(1); // Tamaño del texto
    display.setCursor(120, 30);
    display.println("%");
    display.display(); // Enviar a pantalla
    delay(2000);
  }
  else if (bat_percentage>20 && bat_percentage<=25){
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE); // Color del texto
    display.setTextSize(1.5); // Tamaño del texto
    display.setCursor(2, 0);
    display.println("CARGA DEL DISPOSITIVO");
    display.drawBitmap(0,10, carga25, ancho_imagen_2, alto_imagen_2, WHITE);
    display.setTextSize(4); // Tamaño del texto
    display.setCursor(50, 30);
    display.println(bat_percentage);
    display.setTextSize(1); // Tamaño del texto
    display.setCursor(120, 30);
    display.println("%");
    display.display(); // Enviar a pantalla
    delay(2000);
  }
```

```
  else {
    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE); // Color del texto
    display.setTextSize(1.5); // Tamaño del texto
    display.setCursor(2, 0);
    display.println("CARGA DEL DISPOSITIVO");
    display.drawBitmap(0,10, carga0, ancho_imagen_2, alto_imagen_2, WHITE);
    display.setTextSize(4); // Tamaño del texto
    display.setCursor(50, 30);
    display.println(bat_percentage);
    display.setTextSize(1.5); // Tamaño del texto
    display.setCursor(120, 30);
    display.println("%");
    display.display(); // Enviar a pantalla
    delay(2000);
  }

}
void oled() {
    // Iniciar pantalla OLED en la dirección 0x3C
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
#ifdef __DEBUG__
    Serial.println("No se encuentra la pantalla OLED");
#endif
    while (true);
  }
```

```
  display.clearDisplay(); // Limpiar buffer
  display.setTextSize(1); // Tamaño del texto
  display.setTextColor(SSD1306_WHITE); // Color del texto
  display.setCursor(10, 32); // Posición del texto
  display.println("CARGANDO ..."); // Escribir texto
  display.display(); // Enviar a pantalla
  ////
  delay(3000);
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE); // Color del texto
  display.setCursor(30, 12);
  display.println("UTA - FISEI");
```

```
  display.setCursor(10, 28);
  display.println("TELECOMUNICACIONES");
  display.setCursor(30, 44);
  display.println("2022 - 2023");
  display.setCursor(30, 55);
  display.setTextSize(0.5); // Tamaño del texto
  display.println("BY NAYTHAN");
  display.display(); // Enviar a pantalla
  delay(1000);
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE); // Color del texto
  display.setTextSize(1); // Tamaño del texto
  display.setCursor(23, 5);
  display.println("SIGNOS VITALES");
  display.drawBitmap(33,20, termometro, ancho_imagen, alto_imagen, WHITE);
  display.drawBitmap(55,20, corazon, ancho_imagen, alto_imagen, WHITE);
  display.drawBitmap(75,20, saturacion, ancho_imagen, alto_imagen, WHITE);
  display.display(); // Enviar a pantalla
  delay(2000);
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is acive low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("casa/comedor/temperatura", "Enviando el primer mensaje");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
    }
  }
}
void envioCondicion(){
  str = Serial.readStringUntil('\n'); // leemos la cadena
      for (int i = 0; i < dataLength ; i++) // inicializamos el for
      {
        int index = str.indexOf(separator); // tomamos valores en funcion del separador
        str = str.substring(index + 1); // aumentamos el index
      }
}
```

Anexo 5: Programación del segundo microcontrolador ESP8266

```
// tiempo para letura de datos y envio
int periodo = 1000;
unsigned long tiempoAnterior = 0;
  #include <Wire.h>
  #include "MAX30100_PulseOximeter.h"
  #define REPORTING_PERIOD_MS     1000

  PulseOximeter pox;
  uint32_t tsLastReport = 0;
  void onBeatDetected()
  {
    Serial.println("Beat!");
  }
int suma=0;
int suma2=0;
int suma3=0;
int HeartRate2;
int HeartRate3;
int spo22;
int spo23;
int con=0;
float adc_filtrado = 0;
int adc_filtrado2 = 0;
int adc_raw1 = 0;
int adc_raw2 = 0;
#define alpha 0.05

void setup()
{
  Serial.begin(115200);

  Serial.print("Iniciando...");
  if (!pox.begin()) {
    Serial.println("MAX30100 no fue encontrado. Chequea la alimentación.");
    for (;;);
  } else {
    Serial.println("MAX30100 en funcionamiento..");
  }
  pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop()
{
  pox.update();

  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
      HeartRate = pox.getHeartRate();
      SpO2 = pox.getSpO2();
      HeartRate2=HeartRate;
      spo22=SpO2;
      suma=suma+HeartRate2;
      suma2=suma2+spo22;
      con=con+1;
      tsLastReport = millis();
   }

if(millis()-tiempoAnterior>=periodo){
  tiempoAnterior=millis();
  HeartRate3=suma/con;
  spo23=suma2/con;
  Serial.print(HeartRate3);
  Serial.print(",");
  Serial.println(spo23);
  suma=0;
  suma2=0;
  con=0;
  }

}
```

Anexo 6: Programación del algoritmo de IA (SVM)

```python
#Librerias para Naive Bayes
import pandas as pd
# import numpy as np
# import seaborn as sb #sirve para mapear

df = pd.read_csv(r"datosIA6.csv", sep=',', engine='python')

#Descartar registros innecesarios
cols_to_drop = ['Paciente']
datos = df.drop(cols_to_drop, axis=1)

#Rellenar datos faltantes con la palabra "no"
datos = datos.fillna('no')

#----------------------Balanceo de datos-----------------------------
#para no condicionar el modelo y que no mida mas casos de covid
from imblearn.over_sampling import RandomOverSampler, SMOTE
#--------Separar Variable dependiente y independiente----------------
data = datos
target = df['Diagnostico']

ros = RandomOverSampler()
bal = SMOTE()
dataros, targetros = ros.fit_resample(data, target)

#---------Funciones para transformar str a numeric-------------------------------------
def cambio0str(x):
    if x == 'femenino':
        return 1
    else:
        return 0
#Genero
dataros['Genero N'] = dataros['Genero'].apply(cambio0str)
dataros = dataros.drop('Genero', axis=1)

def cambio1str(x):
    if x == 'si':
        return 1
    else:
        return 0
```

```python
dataros['Secrecion Nasal N'] = dataros['Secrecion Nasal'].apply(cambio2str)
dataros['Congestion Nasal N'] = dataros['Congestion Nasal'].apply(cambio2str)
dataros['Lagrimeo N'] = dataros['Lagrimeo'].apply(cambio2str)
dataros['Estornudos N'] = dataros['Estornudos'].apply(cambio2str)
dataros['Sensacion ahogo N'] = dataros['Sensacion ahogo'].apply(cambio2str)
dataros['Fiebre N'] = dataros['Fiebre'].apply(cambio2str)
dataros['Malestar general N'] = dataros['Malestar general'].apply(cambio2str)
dataros['Picazon nasal N'] = dataros['Picazon nasal'].apply(cambio2str)
dataros['Hinchazon ojos N'] = dataros['Hinchazon ojos'].apply(cambio2str)
dataros['Ronquidos N'] = dataros['Ronquidos'].apply(cambio2str)
dataros['Dolor muscular N'] = dataros['Dolor muscular'].apply(cambio2str)
dataros['Perdida voz N'] = dataros['Perdida voz'].apply(cambio2str)
dataros['Dolor ojos N'] = dataros['Dolor ojos'].apply(cambio2str)
dataros['Diarrea N'] = dataros['Diarrea'].apply(cambio2str)
dataros['Nauseas N'] = dataros['Nauseas'].apply(cambio2str)
dataros['Dolor barriga N'] = dataros['Dolor barriga'].apply(cambio2strr)
dataros['Dolor pecho N'] = dataros['Dolor pecho'].apply(cambio2str)
dataros['Perdida apetito N'] = dataros['Perdida apetito'].apply(cambio2str)
dataros['Escalofrios N'] = dataros['Escalofrios'].apply(cambio2str)

#-------------------Datos de entrenamiento----------------------------
X = dataros.drop(columns='Diagnostico')
#y es la variable dependiente
y = dataros.Diagnostico

#-------------------Division datos de entrenamiento 70% /prueba 30%-----------------
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)

```

```python
#---------------------------------------------------------------------
#--------------------Construccion de lo algoritmos de IA--------------
#---------------------------------------------------------------------

# SUPPORT VECTOR MACHINE-----------------------------------------------
from sklearn import svm
vm = svm.SVC(probability=False)
vm.fit(X_train, y_train)

#---------------------------------------------------------------------
#--------------------Evaluacion del algoritmos de IA------------------
#---------------------------------------------------------------------

# Validacion Cruzada
from sklearn.model_selection import cross_val_score

print("---------------------------------------------------------------")
print("Validacion Cruzada - SUPPORT VECTOR MACHINE")
print("Precision de entrenamiento: " + str(cross_val_score(vm, X_train, y_train, cv=3).mean()))
print("Precision de testeo: " + str(cross_val_score(vm, X_test, y_test, cv=3).mean()))


# Matriz de confuSion
from sklearn.metrics import confusion_matrix, classification_report
print("---------------------------------------------------------------")
print("-----------------------MATRIZ DE CONFUSION---------------------")

print("----------SUPPORT VECTOR MACHINE----------")
y_pred = vm.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print("---------------------------------------------------------------")
print(classification_report(y_test, y_pred))
print("---------------------------------------------------------------")

# Guardar el modelo----------------------------------------------------
import sklearn.externals
import joblib
import pickle
joblib.dump(vm, 'modelo_diagnostico.pkl')


print("----------MODELO GUARDADO----------")
```

Anexo 7: Resultado de los modelos de IA probados

```
------------------------------------------------------------
Validacion Cruzada - DESITION TREE
Precision de entrenamiento: 0.8598910234501322
Precision de testeo: 0.8622385273619843
```

```
----------DESITION TREE----------
[[399 301   0]
 [  0 726   0]
 [  0   0 759]]
------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       1.00      0.57      0.73       700
   RESFRIADO       0.71      1.00      0.83       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.86      2185
   macro avg       0.90      0.86      0.85      2185
weighted avg       0.90      0.86      0.86      2185
```

```
------------------------------------------------------------
Validacion Cruzada - K-NEAREST NEIGHBOR
Precision de entrenamiento: 0.8530212349212093
Precision de testeo: 0.8283657928719658
```

```
----------K-NEAREST NEIGHBOR----------
[[469 231   0]
 [101 625   0]
 [  0   0 759]]
------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       0.82      0.67      0.74       700
   RESFRIADO       0.73      0.86      0.79       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.85      2185
   macro avg       0.85      0.84      0.84      2185
weighted avg       0.85      0.85      0.85      2185
```

```
------------------------------------------------------------
Validacion Cruzada - LOGISTIC REGRESSION
Precision de entrenamiento: 0.9101247344508293
Precision de testeo: 0.9070908515352959
```

```
----------LOGISTIC REGRESSION----------
[[538 162   0]
 [ 27 699   0]
 [  0   0 759]]
---------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       0.95      0.77      0.85       700
   RESFRIADO       0.81      0.96      0.88       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.91      2185
   macro avg       0.92      0.91      0.91      2185
weighted avg       0.92      0.91      0.91      2185
```

```
---------------------------------------------------------------
Validacion Cruzada - GRADIENT BOOSTING
Precision de entrenamiento: 0.9220948695426511
Precision de testeo: 0.9080066024510468
```

```
----------GRADIENT BOOSTING----------
[[533 167   0]
 [ 14 712   0]
 [  0   0 759]]
---------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       0.97      0.76      0.85       700
   RESFRIADO       0.81      0.98      0.89       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.92      2185
   macro avg       0.93      0.91      0.91      2185
weighted avg       0.93      0.92      0.92      2185
```

```
---------------------------------------------------------------
Validacion Cruzada - RANDOM FOREST
Precision de entrenamiento: 0.9150292338064402
Precision de testeo: 0.9070889672741526
```

```
----------RANDOM FOREST----------
[[576 124   0]
 [ 47 679   0]
 [  0   0 759]]
---------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       0.92      0.82      0.87       700
   RESFRIADO       0.85      0.94      0.89       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.92      2185
   macro avg       0.92      0.92      0.92      2185
weighted avg       0.92      0.92      0.92      2185
```

```
----------------------------------------------------------------
Validacion Cruzada - NAIVE BAYES
Precision de entrenamiento: 0.899725305515866
Precision de testeo: 0.9025139812176849
```

```
 ----------NAIVE BAYES----------
 [[487 213   0]
  [  0 726   0]
  [  0   0 759]]
 ----------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       1.00      0.70      0.82       700
   RESFRIADO       0.77      1.00      0.87       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.90      2185
   macro avg       0.92      0.90      0.90      2185
weighted avg       0.92      0.90      0.90      2185
```

```
----------------------------------------------------------------
Validacion Cruzada - SUPPORT VECTOR MACHINE
Precision de entrenamiento: 0.914246422697663
Precision de testeo: 0.912579076159323
```

```
------------------------MATRIZ DE CONFUSION----------------------
----------SUPPORT VECTOR MACHINE----------
[[535 165   0]
 [ 23 703   0]
 [  0   0 759]]
----------------------------------------------------------------
              precision    recall  f1-score   support

     COVID19       0.96      0.76      0.85       700
   RESFRIADO       0.81      0.97      0.88       726
     RINITIS       1.00      1.00      1.00       759

    accuracy                           0.91      2185
   macro avg       0.92      0.91      0.91      2185
weighted avg       0.92      0.91      0.91      2185
```

Anexo 8: Programación de la aplicación Web en Flask

```
#-------------------------------------------------------------------------
#//////////////////////////IMPORTAR LIBRERAS////////////////////////////////
#Importacion de librerias para flask
from flask import Flask, render_template, redirect, url_for, request, flash
#Importacion de librerias para proteccion contra peticiones no logeadas
from flask_wtf.csrf import CSRFProtect
from config import config
import subprocess
#Importacion de librerias para el registro y login
from flask_login import LoginManager, login_user, logout_user, login_required
from models.ModelUser import ModelUser, subirusuario
from models.entities.User import hash
from models.entities.User import User
from werkzeug.security import check_password_hash, generate_password_hash
#Importacion de libreria para activar la clase usuario
from flask_login import UserMixin
#Importacion de libreria para la base de datos
from flask_mysqldb import MySQL
import mysql.connector
#Importacion de librerias charbot
import re
import long_responses as long
import random
#librerias para mqtt
import paho.mqtt.client as mqtt
#////////////////////////////////////////////////////////////////////////////
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#//////////////////////////DECLARACION DE VARIABLES//////////////////////////
#inicializacion de variables de la base de datos
resultado_diagnostico=""
Nombre=""
Cedula=""
Genero=""
Edad=""
Temperatura=""
Secrecion_Nasal=""
Congestion_Nasal=""
Dolor_garganta=""
Lagrimeo=""
Tos=""
Estornudos=""
Sensacion_ahogo=""
Fiebre=""
dolor_articular=""
Malestar_general=""
Dolor_cabeza=""
Picazon_nasal=""
Hinchazon_ojos=""
Ronquidos=""
Dolor_muscular=""
Perdida_voz=""
Dolor_ojos=""
Diarrea=""
Nauseas=""
```

```
Dolor_barriga=""
Dolor_pecho=""
Perdida_apetito=""
Escalofrios=""
Diagnostico=""
temp="10"
bpm="20"
spo2="30"
temp2=""
bpm2=""
spo22=""
#guardar respuestas del chatbot para ver la gravedad del dolor y la tos
g=1
h=1
i=1
j=1
p=1
sensor=1
jp=1
resultado_diagnostico_str=''
```

```python
#variables para conectar a la base de datos
conexion1=mysql.connector.connect(host="0.0.0.0", user="", passwd="", database="")
cursor1=conexion1.cursor()
#Variable para crear entorno de flask
app = Flask(__name__)
#////////////////////////////////////////////////////////////////////////////
#----------------------------------------------------------------------------
#----------------------------------------------------------------------------
#////////////////////////////DECLARACION DE FUNCIONES//////////////////////////
#----------------------------Funciones chatbot-------------------------------
#----------------------------------------------------------------------------
#Funcion para calcular la pobabilidad del bot
def message_probability(user_message, recognised_words, single_response=False, required_words=[]):
    message_certainty = 0
    has_required_words = True

    # Cuenta cuántas palabras están en cada mensaje predefinido
    for word in user_message:
        if word in recognised_words:
            message_certainty += 1

    # Calcula el porcentaje de palabras reconocidas ingresadas por el paciente
    percentage = float(message_certainty) / float(len(recognised_words))

    # Revisar que tiene la palabra requerida
    for word in required_words:
        if word not in user_message:
            has_required_words = False
            break

    # Must either have the required words, or be a single response
    if has_required_words or single_response:
        return int(percentage * 100)
    else:
        return 0
#funcion para conectarse al broker
def on_connect(client, userdata, flags, rc):
    print("Se conecto con mqtt " + str(rc))
    client.subscribe("chatboot/sensores")

#funcion para leer mediante el topic
def on_message(client, userdata, msg):
    global temp, bpm, spo2
    efe=(str(msg.payload)).replace("'","")
    efe=efe.replace("b","")
    print (efe)
    g=efe.split(",")
    temp=g[0]
    bpm=g[1]
    spo2=g[2]
```

```python

client = mqtt.Client()
client.on_connect= on_connect
client.on_message = on_message
client.connect("0.0.0.0", 1883)
def respuesta_sintomas():
    res_sintoma = ["¿Algún otro sintoma?",
                   "Listo, ¿Qué más tienes?",
                   "¿Algún otro dolor?",
                   "¿Alguna otra molestia?"][
        random.randrange(4)]
    return res_sintoma

#Funcion para determina la respuesta del bot que tiene mas alta probabilidad
def check_all_messages(message):
    highest_prob_list = {}

    # Simplifica la creación de respuestas y las agrega al directorio
    def response(bot_response, list_of_words, single_response=False, required_words=[]):
        nonlocal highest_prob_list
        highest_prob_list[bot_response] = message_probability(message, list_of_words, single_response,
        required_words)

```

```
    # Respuestas  ----------------------------------------------------------------------------------------------------
    #-----------------------------------------------------Conversacion-----------------------------------------------
    response('Hola de nuevo!. Cuentame, ¿Cuáles son tus síntomas?😷'+ '*40', ['hola', 'encantado', 'hello',
    'hi', 'hey', 'sup', 'heyo'], single_response=True)
    response('Cuídate 👋, recuerda todo queda entre nostros 😇'+ '*41', ['hasta', 'chao', 'luego', 'adios',
     'adiós', 'cuidate', 'cuídate', 'ir', 'despues', 'después', 'bai', 'chau', 'chaitos', 'bye', 'goodbye'],
      single_response=True)
    response('Estoy muy bien gracias. Dime, ¿Cuáles son tus síntomas?'+ '*42', ['como', 'cómo', 'estas'],
    required_words=['como', 'cómo'])
    response('De nada estoy aquí para servirte, espero verte de nuevo 😁'+ '*43', ['gracias', 'thank',
    'thanks'], single_response=True)
    response(long.R_AYUDAR, ['haces', 'opciones', 'ayudarme', 'ayudame'], single_response=True)
    response(long.R_PAPA, ['creo', 'creador', 'papá', 'papa', 'padre'], single_response=True)
    response(respuesta_sintomas()+ '*1', ['moquera', 'secrecion', 'nasal', 'secreción', 'mocos'],
    single_response=True)
    response(respuesta_sintomas()+ '*2', ['congestion', 'congestión', 'nasal', 'obstrucción', 'obstruccion']
    , single_response=True)
    response('El dolor de garganta es, ¿Leve, moderado o fuerte?'+'*100', ['garganta'],
    required_words=['garganta'])
    response(respuesta_sintomas()+ '*3', ['leve', 'levemente', 'poco'], single_response=True)
    response(respuesta_sintomas()+ '*4', ['moderado', 'normal', 'regular'], single_response=True)
    response(respuesta_sintomas()+ '*5', ['fuerte', 'mucho'], single_response=True)
    response(respuesta_sintomas()+ '*6', ['lagrimea', 'llorosos', 'lagrimeo'], single_response=True)
    response(respuesta_sintomas()+ '*7', ['tos', 'seca', 'sin', 'flema'], single_response=True)
    response('Con flema o sin flema'+'*101', ['tos'], required_words=['tos'])
    response(respuesta_sintomas()+ '*8', ['tos', 'con', 'flema'], single_response=True)
    response(respuesta_sintomas()+ '*9', ['estornudar', 'estornudo', 'estornudos', 'estornudando'],
    single_response=True)
    response(respuesta_sintomas()+ '*10', ['respirando', 'respiro', 'respirar', 'falta', 'aire'],
    single_response=True)
    response(respuesta_sintomas()+ '*11', ['fiebre', 'caliente', 'calentura'], single_response=True)
    response('El dolor de las articulaciones es, ¿Leve, moderado o fuerte?'+'*102', ['articular',
    'articulaciones'], single_response=True)
    response(respuesta_sintomas()+ '*12', ['malestar', 'fatiga', 'dolor', 'cuerpo', 'cansado', 'cansancio',
    'perdida', 'animo', 'pérdida', 'ánimo'], single_response=True)
    response('El dolor de la cabeza es, ¿Leve, moderado o fuerte?'+'*103', ['cabeza', 'migraña', 'cefalea',
    'jaqueca'], single_response=True)
    response(respuesta_sintomas()+ '*13', ['pica', 'picazón', 'picazon', 'hormigeo', 'nariz', 'comezón',
    'comezon'], single_response=True)
    response(respuesta_sintomas()+ '*14', ['hinchazon', 'hinchazón', 'hinchados', 'bolsas', 'ojeras'],
    single_response=True)
    response(respuesta_sintomas()+ '*15', ['ronca', 'ronco', 'ronquidos', 'roncando'], single_response=True)
    response(respuesta_sintomas()+ '*16', ['muscular', 'músculos', 'músculo', 'musculos', 'musculo',
    'mialgia'], single_response=True)
    response(respuesta_sintomas()+ '*17', ['voz', 'diafonia', 'diafonía'], single_response=True)
    response(respuesta_sintomas()+ '*18', ['ocular', 'ojos'], single_response=True) #revisar OJOS
    response(respuesta_sintomas()+ '*19', ['diarrea'], single_response=True)
    response(respuesta_sintomas()+ '*20', ['nauseas', 'arcadas'], single_response=True)
    response(respuesta_sintomas()+ '*21', ['barriga', 'abdomen'], single_response=True)
    response(respuesta_sintomas()+ '*22', ['pecho', 'tórax', 'torax'], single_response=True)
    response(respuesta_sintomas()+ '*23', ['apetito', 'comer', 'hambre', 'anorexia'], single_response=True)
    response(respuesta_sintomas()+ '*24', ['escalofríos', 'escalofrios', 'temblor', 'temblores', 'sacudida',
    'sacudidas', 'espasmo', 'espasmos'], single_response=True)
```

```
    response('En breves momentos te dare tu diagnóstico'+ '*25', ['diagnostico', 'diagnóstico', 'no'],
    single_response=True)
    #-----------------------------------------------------------------------------------------------------------------
    #Respuesta con la probabilidad mas alta
    best_match = max(highest_prob_list, key=highest_prob_list.get)
    #si la entrada no se entiende se pide que escriba de nuevo
    return long.unknown() if highest_prob_list[best_match] < 1 else best_match

# Funcion para tener la respuesta del bot
def get_response(user_input):
    split_message = re.split(r'\s+|[,;?!.-]\s*', user_input.lower())
    response = check_all_messages(split_message)
    return response

def base_numeros(respuesta_num):
    respuesta_num= respuesta_num.split('*')
    #print(respuesta_num[1])
    global headings, data, resultado_diagnostico_str, g, p, h, i, j, sensor, jp, Nombre, Cedula, Genero,
    Edad, Temperatura, Secrecion_Nasal, Congestion_Nasal, Dolor_garganta, Lagrimeo, Tos, Estornudos,
    Sensacion_ahogo, Fiebre, dolor_articular, Malestar_general, Dolor_cabeza, Picazon_nasal, Hinchazon_ojos,
    Ronquidos, Dolor_muscular, Perdida_voz, Dolor_ojos, Diarrea, Nauseas, Dolor_barriga, Dolor_pecho,
    Perdida_apetito, Escalofrios
```

```
    #------------------------dolor garganta gravedad
    if (respuesta_num[1]== "100"):
        g=0
    if (g==0 and respuesta_num[1]=='3'):
        respuesta_num[1]='1003'
        g=1 #resetear g=1, si no siempre va a estar en 100
    elif (g==0 and respuesta_num[1]=='4'):
        respuesta_num[1]='1004'
        g=1
    elif (g==0 and respuesta_num[1]=='5'):
        respuesta_num[1]='1005'
        g=1
    #------------------------dolor garganta articular
    if (respuesta_num[1]== "102"):
        h=0
    if (h==0 and respuesta_num[1]=='3'):
        respuesta_num[1]='1023'
        h=1
    elif (h==0 and respuesta_num[1]=='4'):
        respuesta_num[1]='1024'
        h=1
    elif (h==0 and respuesta_num[1]=='5'):
        respuesta_num[1]='1025'
        h=1
    #------------------------dolor garganta cabeza
    if (respuesta_num[1]== "103"):
        i=0
    if (i==0 and respuesta_num[1]=='3'):
        respuesta_num[1]='1033'
        i=1
    elif (i==0 and respuesta_num[1]=='4'):
        respuesta_num[1]='1034'
        i=1
    elif (i==0 and respuesta_num[1]=='5'):
        respuesta_num[1]='1035'
        i=1
    #------------------------tos
    if (respuesta_num[1]== "101"):
        j=0
    if (j==0 and respuesta_num[1]=='7'):
        respuesta_num[1]='1017'
        j=1
    elif (j==0 and respuesta_num[1]=='8'):
        respuesta_num[1]='1018'
        j=1
    #print(respuesta_num[1])
    #-----------------------------------------------
    if (respuesta_num[1]=="1"):
```

```
        Secrecion_Nasal="si"
    if (respuesta_num[1]=="2"):
        Congestion_Nasal="si"
    if (respuesta_num[1]=="1003"):
        Dolor_garganta="leve"
    if (respuesta_num[1]=="1004"):
        Dolor_garganta="moderado"
    if (respuesta_num[1]=="1005"):
        Dolor_garganta="fuerte"
    if (respuesta_num[1]=="6"):
        Lagrimeo="si"
    if (respuesta_num[1]=="1017"):
        Tos="seca"
    if (respuesta_num[1]=="1018"):
        Tos="con flema"
    if (respuesta_num[1]=="9"):
        Estornudos="si"
    if (respuesta_num[1]=="10"):
        Sensacion_ahogo="si"
    if (respuesta_num[1]=="11"):
        Fiebre="si"
    if (respuesta_num[1]=="1023"):
        dolor_articular="leve"
    if (respuesta_num[1]=="1024"):
        dolor_articular="moderado"
    if (respuesta_num[1]=="1025"):
        dolor_articular="fuerte"
    if (respuesta_num[1]=="12"):
        Malestar_general="si"
```

```
    if (respuesta_num[1]=="1033"):
        Dolor_cabeza="leve"
    if (respuesta_num[1]=="1034"):
        Dolor_cabeza="moderado"
    if (respuesta_num[1]=="1035"):
        Dolor_cabeza="fuerte"
    if (respuesta_num[1]=="13"):
        Picazon_nasal="si"
    if (respuesta_num[1]=="14"):
        Hinchazon_ojos="si"
    if (respuesta_num[1]=="15"):
        Ronquidos="si"
    if (respuesta_num[1]=="16"):
        Dolor_muscular="si"
    if (respuesta_num[1]=="17"):
        Perdida_voz="si"
    if (respuesta_num[1]=="18"):
        Dolor_ojos="si"
    if (respuesta_num[1]=="19"):
        Diarrea="si"
    if (respuesta_num[1]=="20"):
        Nauseas="si"
    if (respuesta_num[1]=="21"):
        Dolor_barriga="si"
    if (respuesta_num[1]=="22"):
        Dolor_pecho="si"
    if (respuesta_num[1]=="23"):
        Perdida_apetito="si"
    if (respuesta_num[1]=="24"):
        Escalofrios="si"
    if (float(temp2)>20):
        Fiebre="si"
    else :
        Fiebre="no"
    if (respuesta_num[1]=="25"):
        sql=("INSERT INTO pacientes (Nombre, Cedula, Genero, Edad, BPM, SPO2, Temperatura, Secrecion_Nasal,
        Congestion_Nasal, Dolor_garganta, Lagrimeo, Tos, Estornudos, Sensacion_ahogo, Fiebre, dolor_articular
        , Malestar_general, Dolor_cabeza, Picazon_nasal, Hinchazon_ojos, Ronquidos, Dolor_muscular,
        Perdida_voz, Dolor_ojos, Diarrea, Nauseas, Dolor_barriga, Dolor_pecho, Perdida_apetito, Escalofrios)
        VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)")
        datos=(Nombre, Cedula, Genero, Edad, bpm2, spo22,  temp2, Secrecion_Nasal, Congestion_Nasal,
        Dolor_garganta, Lagrimeo, Tos, Estornudos, Sensacion_ahogo, Fiebre, dolor_articular, Malestar_general
        , Dolor_cabeza, Picazon_nasal, Hinchazon_ojos, Ronquidos, Dolor_muscular, Perdida_voz, Dolor_ojos,
        Diarrea, Nauseas, Dolor_barriga, Dolor_pecho, Perdida_apetito, Escalofrios)
        cursor1.execute(sql, datos)
        conexion1.commit()
        ejecutar_Script()
        #Respuesta de diagnostico para el chatbot
        query = 'SELECT Diagnostico FROM pacientes ORDER BY ID_Paciente DESC LIMIT 1'
        cursor1.execute(query)
        resultado_diagnostico = cursor1.fetchall()
        #Trasformar tuple to string
        resultado_diagnostico_str = ''.join(resultado_diagnostico[0])
```

```

def ejecutar_Script():
    result=subprocess.getoutput(r"import_modelo.py")
#Proteccion de peticiones no logeadas
csrf = CSRFProtect()
#Conexion con la base de datos
db = MySQL(app)
#Control de login
login_manager_app = LoginManager(app)
@login_manager_app.user_loader
def load_user(id):
    return ModelUser.get_by_id(db, id)

#Ruta raiz, redirigirme hacia el LOGIN
@app.route('/')
def index():
    return redirect(url_for('login'))

#Permita metodo get y post
@app.route('/login', methods=['GET', 'POST'])
def login():
    global usuariologin, Nombre, Edad, Genero, Cedula
    if request.method=='POST':
```

```
        user = User(0, request.form['email'], request.form['password'])
        usuariologin=request.form['email']
        logged_user = ModelUser.login(db, user)
        if logged_user != None:
            if logged_user.password:
                login_user(logged_user)
                print ("-----------------------------")
                query = "SELECT fullname, cedula, genero, edad FROM user WHERE username =%s"
                correo_a = (usuariologin,)
                cursor1.execute(query, correo_a)
                myresult = cursor1.fetchall()
                for x in myresult:
                    #print(x)
                    #print(x[1])
                    Nombre= x[0]
                    Cedula= x[1]
                    Genero= x[2]
                    Edad= x[3]
                print(Nombre, Cedula, Genero, Edad)
                return redirect(url_for('Sensor'))
                #return redirect(url_for('home'))
            else:
                flash("¡Contraseña Invalida!")
                return render_template('auth/login.html')
        else:
            flash("¡Usuario no existente!")
            print("¡Usuario no existente amigo")
            return render_template('auth/login.html')
    else:
        #metodo get si recien vamos a ingresar los datos
        return render_template('auth/login.html')

@app.route('/logout')
def logout():
    logout_user()
    return redirect(url_for('login'))

@app.route('/Registro', methods=['GET', 'POST'])
def Registro():
    if request.method=='POST':
        #metodo post para comprobar si ya lo enviamos

        em=(request.form['email'])
        usr=(request.form['username'])
        ce=(request.form['cedula'])
        gen=(request.form['genero'])
        eda=(request.form['edad'])
        passw=(request.form['password'])
        has=(hash.has(passw))
        print (em, usr, has)
        subirusuario.sub(db, em, has, usr, ce, gen, eda)
        return render_template('auth/login.html')
```

```

    else:
        return render_template('auth/registro.html')

@app.route('/home')
@login_required
def home():
    return render_template('home.html')

@app.route('/chat')
@login_required
def chat():
    global temp2,spo22,bpm2,temp, spo2, jp, bpm, resultado_diagnostico, sensor
    print (sensor, jp)
    if (sensor==0 and jp==0):
        temp2=temp
        spo22=spo2
        bpm2=bpm
        jp=2
    #Respuesta de diagnostico para el chatbot
    query = 'SELECT Diagnostico FROM pacientes ORDER BY ID_Paciente DESC LIMIT 1'
    cursor1.execute(query)
    resultado_diagnostico = cursor1.fetchall()
    #Trasformar tuple to string
    resultado_diagnostico_str = ''.join(resultado_diagnostico[0])
    return render_template("index.html", lol=resultado_diagnostico_str)
```

```python


def estatus_401(error):
    return redirect(url_for('login'))


def estatus_404(error):
    return "<h1>Pagina no encontrada</h1>", 404

@app.route('/safe')
@login_required
def safe():
    return "<h1>Vista protegida por login</h1>"

@app.route("/get")
@login_required
def get_bot_response():
    global p
    userText = request.args.get('msg')
    if (p==1):
        return get_response(userText)

    elif (p==0):
        p=1
        return 'Usted tiene '+resultado_diagnostico_str+' se recomienda acudir a un médico
        para ser recetado'



@app.route('/Sensor')
@login_required
def Sensor():
    global temp, bpm, spo2, jp, sensor
    sensor=0
    jp=0
    return render_template("sen/sensor.html",temperatura=temp, bpm=bpm, spo2=spo2)

@app.route("/Historial")
@login_required
def component():
    #-------------------------------------------------------------------------------------------------------------
    #-----------------------------------------------------Creacion del historial medico-------------------------
    #Titulos de la tabla para el historial
    headings = ("Fecha - Hora", "Nombre", "Cédula", "Género", "Edad", "BPM", "SPO2", "Temperatura",
    "Síntomas", "Diagnóstico")
    cedula= Cedula
    sql="SELECT Fecha, Nombre, Cedula, Genero, Edad, BPM, SPO2, Temperatura, Secrecion_Nasal,
    Congestion_Nasal, Dolor_garganta, Lagrimeo, Tos, Estornudos, Sensacion_ahogo, Fiebre, dolor_articular,
    Malestar_general, Dolor_cabeza, Picazon_nasal, Hinchazon_ojos, Ronquidos, Dolor_muscular, Perdida_voz,
    Dolor_ojos, Diarrea, Nauseas, Dolor_barriga, Dolor_pecho, Perdida_apetito, Escalofrios, Diagnostico
    FROM pacientes WHERE Cedula=%s"
    cedula1 = (cedula,)
    cursor1.execute(sql,cedula1)
    #Guardar consulta en una variable
    a= cursor1.fetchall()
```

```python
    #print(a)
    #print("-////////////////////////////////////////////////////////////////////////////////////////////////////")
    lol = []
    for row in a:
        #print(row)
        #print(type(row))
        n0= row[0]
        n1= n0.strftime("%m/%d/%Y, %H:%M:%S")
        #print(type(n0.strftime("%m/%d/%Y, %H:%M:%S")))
        n2= row[1]
        #print(n2)
        n3= row[2]
        #print(n3)
        n4= row[3]
        #print(n4)
        n5= row[4]
        #print(n5)
        n6= row[5]
        #print(n6)
        n7= row[6]
        #print(n7)
        n8= row[7]
        #print(n8)
```

```
            #-------------------------------------------
            s1=row[8]
            s3=row[8]
            s4=row[8]
            s5=row[8]
            s6=row[8]
            s7=row[8]
            s8=row[8]
            s9=row[8]
            s10=row[8]
            s11=row[9]
            s12=row[10]
            s13=row[11]
            s14=row[12]
            s15=row[13]
            s16=row[14]
            s17=row[15]
            s18=row[16]
            s19=row[17]
            s20=row[18]
            s21=row[19]
            s22=row[20]
            s23=row[21]
            #Creacion de lista de sintomas
            list_sintomas= [s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23]
            #Eliminar valores vacios de la lista
            for i in range(len(list_sintomas)-1, -1, -1):
                if not list_sintomas[i]:
                    del list_sintomas[i]
            #Concatenacion de sintomas
            C_sintomas= ", ".join(list_sintomas)
            #print(C_sintomas)
            #Obtencion de diagnostico
            d= row[31]
            #print(d)
            #----------------------------Creacion de una lista completa para el historial----------------------
            list_completa=[n1,n2,n3,n4,n5,n6,n7,n8,C_sintomas,d]
            #Guardado de consultas en una lista llamda lol
            lol.append(list_completa)
        #Transformar lista a tupla para trabajar con la tabla de Bootstrap
        data=tuple(lol)
        #////////////////////////////////////////////////////////////////////////////
        #Redirecionar a la pagina Historial
        return render_template("auth/tabla_bootstrap.html", headings=headings, data=data)

#Ejecutar flask
if __name__ == '__main__':
    app.config.from_object(config['development'])
    csrf.init_app(app)
    app.register_error_handler(401, estatus_401)
    app.register_error_handler(404, estatus_404)
    app.run("0.0.0.0")
```

Anexo 9: Programación HTML de la interfaz de inicio de sesión

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>{% block title %}{% endblock %}</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <!--Bootstrap-->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" int
    <!--Custom css-->
    {% block customcss %}
    {% endblock %}
</head>
<body class="text-center">
    {% block body %}
    {% endblock %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha

</body>
</html>
```

```html
{% extends './base.html' %}

{% block title %}Login{% endblock %}

{% block customcss %}
<link rel="stylesheet" href="{{ url_for('static', filename='css/login.css')}}">
{% endblock %}

{% block body %}
<main class="form-signin w-100 m-auto">
  <form action="/login" method="POST">
    <img class="mb-4" src="{{ url_for('static', filename='img/usuario.png')}}" alt="" width="72"
    height="57">

    <h1 class="h3 mb-3 fw-normal">Porfavor inicia sesión</h1>
    <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
    <div class="form-floating">
      <input name= "email" type="email" class="form-control" id="floatingInput"
      placeholder="name@example.com">
      <label for="floatingInput">Dirección email</label>
    </div>
    <div class="form-floating">
      <input name= "password" type="password" class="form-control" id="floatingPassword"
      placeholder="Password">
      <label for="floatingPassword">Contraseña</label>
    </div>

    <div class="checkbox mb-3">
      <label>
        <input type="checkbox" value="remember-me"> Recuerdame
      </label>
    </div>
    <button class="w-100 btn btn-lg btn-primary" type="submit">Iniciar Sesión</button>

    {% with messages = get_flashed_messages() %}
    {% if messages %}
    <br/>
    {% for messages in messages %}
    <div class="alert alert-primary alert-dismissible role="alert">
      <strong>{{ messages }}</strong>
      <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="close"></button>
    </div>
    {% endfor %}
    {% endif %}
    {% endwith %}
    <p class="mt-5 mb-3 text-muted">&copy; 2017-2022</p>
  </form>
  <form action="/Registro">
    <button class="w-100 btn btn-lg btn-primary" type="submit">Registrate</button>
  </form>
</main>
{% endblock %}
```

Anexo 10: Programación HTML de la interfaz de registro

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>{% block title %}{% endblock %}</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <!--Bootstrap-->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" int
    <!--Custom css-->
    {% block customcss %}
    {% endblock %}
</head>
<body class="text-center">
    {% block body %}
    {% endblock %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha

</body>
</html>
```

```html
{% extends './base.html' %}

{% block title %}Registro{% endblock %}

{% block customcss %}
<link rel="stylesheet" href="{{ url_for('static', filename='css/login.css')}}">
{% endblock %}

{% block body %}

<main class="form-signin w-100 m-auto">
    <form action="/Registro" method="POST">

      <img class="mb-4" src="{{ url_for('static', filename='img/usuario.png')}}" alt="" width="72"
      height="57">

      <h1 class="h3 mb-3 fw-normal">Registrate aquí</h1>
      <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
      <div class="form-floating">
        <input name= "email" type="email" class="form-control" id="floatingInput"
        placeholder="name@example.com">
        <label for="floatingInput">Dirección email</label>
      </div>
      <div class="form-floating">
        <input name= "username" type="username" class="form-control" id="floatingInput" placeholder=" ">
        <label for="floatingInput">Nombre de usuario</label>
      </div>
      <div class="form-floating">
        <input name= "cedula" type="number" class="form-control" id="floatingInput" placeholder=" ">
        <label for="floatingInput">Cedula</label>
      </div>
      <div class="form-floating">
        <input name= "genero" type="text" class="form-control" id="floatingInput" placeholder=" ">
        <label for="floatingInput">Genero</label>
      </div>
      <div class="form-floating">
        <input name= "edad" type="number" class="form-control" id="floatingInput" placeholder=" ">
        <label for="floatingInput">Edad</label>
      </div>
      <div class="form-floating">
        <input name= "password" type="password" class="form-control" id="floatingPassword"
        placeholder="Password">
        <label for="floatingPassword">Contraseña</label>
      </div>

      <button class="w-100 btn btn-lg btn-primary" type="submit">Registrate</button>

      <p class="mt-5 mb-3 text-muted">&copy; 2017–2022</p>
    </form>

  </main>
{% endblock %}
```

Anexo 11: Configuración CSS de la interfaz de inicio de sesión y registro

```css
html,
body {
  height: 100%;
  background-color: #fcfcfe;
  background-image:
  url("https://images.rawpixel.com/image_800/czNmcy1wcml2YXRlL3Jhd3BpeGVsX2ltYWdlcy93ZWJzaXRlX2NvbnRlbnQvbHIv
  height: 700px; /* You must set a specified height */
  background-position: center; /* Center the image */
  background-repeat: no-repeat; /* Do not repeat the image */
  background-size: cover; /* Resize the background image to cover the entire container */
}

body {
  display: flex;
  align-items: center;
  padding-top: 40px;
  padding-bottom: 40px;

}

.form-signin {
  max-width: 330px;
  padding: 15px;
}

.form-signin .form-floating:focus-within {
  z-index: 2;
}

.form-signin input[type="email"] {
  margin-bottom: -1px;
  border-bottom-right-radius: 0;
  border-bottom-left-radius: 0;
}

.form-signin input[type="password"] {
  margin-bottom: 10px;
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}
```

Anexo 12: Programación HTML de la interfaz de sensores

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta http-equiv="refresh" content="5" >

    <title>Sensores</title>
    <html>
    <head>
        <meta charset='utf-8'>
        <meta http-equiv='X-UA-Compatible' content='IE=edge'>
        <title>{% block title %}{% endblock %}</title>
        <meta name='viewport' content='width=device-width, initial-scale=1'>
        <!--Bootstrap-->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
        rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
        crossorigin="anonymous">
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>


    </script>
  <body>
    {% block content %}

    {% endblock %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3" c
    rossorigin="anonymous"></script>
  </body>
</html>
```

```
{% extends 'sen/base.html' %}
{% block content %}
<link rel="stylesheet" href="{{ url_for('static', filename='css/login.css')}}", style="left: 500cm;">
<div class="jumbotron jumbotron-fluid", style="left: 500cm;">
  <div class="container">
    <h1 class="display-4">Signos Vitales</h1>
    <br>
    <br>
    <h3 class="Tem">Temperatura <strong> {{temperatura}} </strong> C </h3>
    <h3 class="Hum">Bpm <strong> {{bpm}} </strong> </h3>
    <h3 class="Tem">Spo2 <strong> {{spo2}} </strong>  </h3>
    <br>
    <br>
  </div>
</div>
<form action="/chat">
  <button class="w-100 btn btn-lg btn-primary" type="submit">Chatboot</button>
</form>

{%  endblock  %}

```

Anexo 13: Programación HTML de la interfaz del asistente virtual (chatbot)

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Chatbot</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <script src="https://kit.fontawesome.com/8dcc82df50.js" crossorigin="anonymous"></script>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body>
  <!-- partial:index.partial.html -->

  <section class="msger">
    <header class="msger-header">
      <div class="msger-header-title">
        <i class="fa-solid fa-user-doctor"></i> Tu Asistente Médico <i class="fa-solid fa-stethoscope"></i>
      </div>
    </header>

    <main class="msger-chat">
      <div class="msg left-msg">
        <div class="msg-img" style="background-image:
        url(https://cdn-icons-png.flaticon.com/512/4660/4660583.png)"></div>

        <div class="msg-bubble">
          <div class="msg-info">
            <div class="msg-info-name">Baymax</div>
            <div class="msg-info-time">24:00</div>
          </div>

          <div class="msg-text">
            Hola!, Yo soy Baymax tu asistente personal de salud. Mi trabajo es diagnosticarte
            una enfermedad (Covid19, Rinitis alérgica y Resfriado común)😲,
            dependiento de tus sintomas. Si tu enfermedad no esta dentro de mi base de datos
            espero ayudarte en un futuro 🙂. Dicho esto cuentame, ¿Cuáles son tus síntomas?___
            Al terminar de escribir tus sintomas porfavor escribe "Diagnóstico" para ver tus resultados.
          </div>
        </div>
      </div>

    </main>

    <form class="msger-inputarea">
      <input type="text" class="msger-input" id="textInput" placeholder="Enter your message...">
      <button type="submit" class="btn btn-primary">Enviar</button>
    </form>
    <form action="/Historial">
      <button class="w-100 btn btn-lg btn-primary align-text-bottom" type="submit">Historia Clínica</button>
```

```
    </form>
  </section>
  <!-- partial -->
  <script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>

  <script>

    const msgerForm = get(".msger-inputarea");
    const msgerInput = get(".msger-input");
    const msgerChat = get(".msger-chat");


    // Icons made by Freepik from www.flaticon.com
    const BOT_IMG = "https://cdn-icons-png.flaticon.com/512/4660/4660583.png";
    const PERSON_IMG = "https://cdn-icons-png.flaticon.com/512/4660/4660603.png";
    const BOT_NAME = "    Baymax";
    const PERSON_NAME = "Tu";

    msgerForm.addEventListener("submit", event => {
      event.preventDefault();

      const msgText = msgerInput.value;
      if (!msgText) return;
```

```
      appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
      msgerInput.value = "";
      botResponse(msgText);
    });

    function appendMessage(name, img, side, text) {
      //   Simple solution for small apps
      const msgHTML = `
<div class="msg ${side}-msg">
  <div class="msg-img" style="background-image: url(${img})"></div>

  <div class="msg-bubble">
    <div class="msg-info">
      <div class="msg-info-name">${name}</div>
      <div class="msg-info-time">${formatDate(new Date())}</div>
    </div>

    <div class="msg-text">${text}</div>
  </div>
</div>
`;

      msgerChat.insertAdjacentHTML("beforeend", msgHTML);
      msgerChat.scrollTop += 500;
    }

    function botResponse(rawText) {

      // Bot Response
      $.get("/get", { msg: rawText }).done(function (data) {
        console.log(rawText);
        console.log(data);
        const msgText = data;
        appendMessage(BOT_NAME, BOT_IMG, "left", msgText);

      });

    }


    // Utils
    function get(selector, root = document) {
      return root.querySelector(selector);
    }

    function formatDate(date) {
      const h = "0" + date.getHours();
      const m = "0" + date.getMinutes();

      return `${h.slice(-2)}:${m.slice(-2)}`;
    }
  </script>
</body>

</html>
```

Anexo 14: Configuración CSS de la interfaz del asistente virtual (chatbot)

```css
:root {
  --body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
  --msger-bg: #fff;
  --border: 2px solid #ddd;
  --left-msg-bg: #ececec;
  --right-msg-bg: #579ffb;
}

html {
  box-sizing: border-box;
}

*,
*:before,
*:after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-image: var(--body-bg);
  font-family: Helvetica, sans-serif;
}

.msger {
  display: flex;
  flex-flow: column wrap;
  justify-content: space-between;
  width: 100%;
  max-width: 867px;
  margin: 25px 10px;
  height: calc(100% - 50px);
  border: var(--border);
  border-radius: 5px;
  background: var(--msger-bg);
  box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}

.msger-header {
  /* display: flex; */
  font-size: medium;
  justify-content: space-between;
  padding: 10px;
  text-align: center;
  border-bottom: var(--border);
  background: #eee;
  color: #666;
}

.msger-chat {
  flex: 1;
```

```
    overflow-y: auto;
    padding: 10px;
  }
  .msger-chat::-webkit-scrollbar {
    width: 6px;
  }
  .msger-chat::-webkit-scrollbar-track {
    background: #ddd;
  }
  .msger-chat::-webkit-scrollbar-thumb {
    background: #bdbdbd;
  }
  .msg {
    display: flex;
    align-items: flex-end;
    margin-bottom: 10px;
  }

  .msg-img {
    width: 50px;
    height: 50px;
    margin-right: 10px;
    background: #ddd;
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
    border-radius: 50%;
  }
  .msg-bubble {
    max-width: 450px;
    padding: 15px;
    border-radius: 15px;
    background: var(--left-msg-bg);
  }
  .msg-info {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 10px;
  }
  .msg-info-name {
    margin-right: 10px;
    font-weight: bold;
  }
  .msg-info-time {
    font-size: 0.85em;
  }

  .left-msg .msg-bubble {
    border-bottom-left-radius: 0;
  }

  .right-msg {
    flex-direction: row-reverse;
  }
  .right-msg .msg-bubble {
```

```
    background: var(--right-msg-bg);
    color: #fff;
    border-bottom-right-radius: 0;
  }
  .right-msg .msg-img {
    margin: 0 0 0 10px;
  }

  .msger-inputarea {
    display: flex;
    padding: 10px;
    border-top: var(--border);
    background: #eee;
  }
  .msger-inputarea * {
    padding: 10px;
    border: none;
    border-radius: 3px;
    font-size: 1em;
  }
  .msger-input {
    flex: 1;
    background: #ddd;
  }
  .msger-send-btn {
    margin-left: 10px;
    background: rgb(0, 196, 65);
    color: #fff;
    font-weight: bold;
    cursor: pointer;
    transition: background 0.23s;
  }
  .msger-send-btn:hover {
    background: rgb(0, 180, 50);
  }

  .msger-chat {
    background-color: #fcfcfe;
    background-image: url("https://i.pinimg.com/564x/57/e1/df/57e1df2212e52fa90990e653d5ffd8f4.jpg");
    height: 500px; /* You must set a specified height */
    background-position: center; /* Center the image */
    background-repeat: no-repeat; /* Do not repeat the image */
    background-size: cover; /* Resize the background image to cover the entire container */
  }
```

Anexo 15: Programación HTML de la interfaz del historial médico

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Historia Clínica</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
  <script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>
</head>
<body>

<div class="container mt-3">
  <h2 class="text-center">Historia Clínica</h2>
  <p>A continuación se muestra de forma detallada tus consultas con su respectivo diagnóstico:</p>

  <div class="table-responsive">
    <table class="table table-bordered">
      <thead class="table-dark">
        <tr>
          {% for header in headings %}
          <th class="text-center">{{ header }}</th>
          {% endfor %}
        </tr>
      </thead>
      <tbody>
        {% for row in data %}
        <tr>
          {% for cell in row %}
          <td class="text-center">{{ cell }}</td>
          {% endfor %}
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
</div>

<form action="/login">
  <div class="d-grid gap-2">
    <a class="btn btn-dark btn-lg" href="{{ url_for('logout') }}">Salir</a>
    <p class="text-center">Gracias por visitarnos!</p>
  </div>
</form>

<!-- Footer -->
<footer class="text-center text-lg-start bg-black text-muted">
  <!-- Section: Social media -->
  <section class="d-flex justify-content-center justify-content-lg-between p-4 border-bottom">
    <!-- Left -->
    <div class="me-5 d-none d-lg-block">
      <span>Conéctate con nosotros en las redes sociales:</span>
    </div>
    <!-- Left -->

    <!-- Right -->
    <div>
      <a href="https://www.facebook.com/aso.fisei.1" class="me-4 link-secondary">
        <i class="fab fa-facebook-f"></i>
      </a>
      <a href="https://twitter.com/UTecnicaAmbato" class="me-4 link-secondary">
        <i class="fab fa-twitter"></i>
      </a>
      <a href="https://goo.gl/maps/vWASVcNHPtr3cUWg9" class="me-4 link-secondary">
        <i class="fab fa-google"></i>
      </a>
      <a href="https://www.instagram.com/utecnicaambato/" class="me-4 link-secondary">
        <i class="fab fa-instagram"></i>
      </a>
      <a href="https://www.linkedin.com/school/universidad-t%C3%A9cnica-de-ambato/"
      class="me-4 link-secondary">
        <i class="fab fa-linkedin"></i>
      </a>
      <a href="https://github.com/luhuisicnu/The-Flask-Mega-Tutorial-zh" class="me-4 link-secondary">
        <i class="fab fa-github"></i>
      </a>
    </div>
```

```
78      <!-- Right -->
79    </section>
80    <!-- Section: Social media -->
81
82    <!-- Section: Links  -->
83    <section class="">
84      <div class="container text-center text-md-start mt-5">
85        <!-- Grid row -->
86        <div class="row mt-3">
87          <!-- Grid column -->
88          <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">
89            <!-- Content -->
90            <h6 class="text-uppercase fw-bold mb-4">
91              <i class="fas fa-gem me-3 text-secondary"></i>UTA-FISEI
92            </h6>
93            <p>
94              Formar profesionales líderes competentes con visión humanista y pensamiento
95              crítico a través de la Docencia, la Investigación y la Vinculación, que apliquen,
96              promuevan y difundan el conocimiento respondiendo a las necesidades del país.
97            </p>
98          </div>
99          <!-- Grid column -->
100
101         <!-- Grid column -->
102         <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">
103           <!-- Links -->
104           <h6 class="text-uppercase fw-bold mb-4">
105             Centros Médicos
106           </h6>
107           <p>
108             <a href="https://goo.gl/maps/msShdXyryEF72DPC6" class="text-reset">Hospital Regional Ambato</a>
109           </p>
110           <p>
111             <a href="https://goo.gl/maps/baus983us6kzyUpM6" class="text-reset">Hospital del IESS Ambato</a>
112           </p>
113           <p>
114             <a href="https://goo.gl/maps/MbQaYywpB7UKqcCM7" class="text-reset">Hospital Durán de Ambato</a>
115           </p>
116           <p>
117             <a href="https://goo.gl/maps/k6vWdZnXj2CYFW7X7" class="text-reset">Hospital Santa Inés Ambato</a>
118           </p>
119         </div>
120         <!-- Grid column -->
121
122         <!-- Grid column -->
123         <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">
124           <!-- Links -->
125           <h6 class="text-uppercase fw-bold mb-4">
126             Enlaces útiles
127           </h6>
128           <p>
129             <a href="https://www.who.int/es/health-topics/coronavirus#tab=tab_1" class="text-reset">COVID-19</a
130           </p>
131           <p>
```

```
132             <a href="https://www.uis.edu.co/intranet/calidad/documentos/bienestar_estudiantil/guias/GBE.15.pdf"
133           </p>
134           <p>
135             <a href="https://www.quironsalud.es/es/comunicacion/notas-prensa/oms-estima-40-poblacion-urbana-pre
136           </p>
137           <p>
138             <a href="https://apps.who.int/iris/bitstream/handle/10665/331860/WHO-2019-nCoV-SARI_treatment_cente
139           </p>
140         </div>
141         <!-- Grid column -->
142
143         <!-- Grid column -->
144         <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">
145           <!-- Links -->
146           <h6 class="text-uppercase fw-bold mb-4">Contacto-Soporte Técnico</h6>
147           <p><i class="fas fa-home me-3 text-secondary"></i> Parroquia Picahiua, Ambato-Ecuador</p>
148           <p><i class="fas fa-envelope me-3 text-secondary"></i>naythanvillafuerte1n@gmail.com</p>
149           <p><i class="fas fa-phone me-3 text-secondary"></i> +593 0969128272</p>
150           <p><i class="fas fa-phone me-3 text-secondary"></i> +593 0986366267</p>
151         </div>
```

```
152             <!-- Grid column -->
153           </div>
154           <!-- Grid row -->
155         </div>
156       </section>
157       <!-- Section: Links  -->
158
159       <!-- Copyright -->
160       <div class="text-center p-4" style="background-color: rgba(0, 0, 0, 0.025);">
161         © 2022 Copyright:
162         <a class="text-reset fw-bold" href="https://fisei.uta.edu.ec/v4.0/index.php/facultad/mision-vision/">
163           FISEI.com</a>
164       </div>
165       <!-- Copyright -->
166     </footer>
167     <!-- Footer -->
168
169     </body>
170     </html>
```