ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA NO INVASIVO DE MONITOREO DE LA PRESIÓN ARTERIAL Y DE LA FRECUENCIA CARDIACA COMBINADAS

SISTEMA DE ADQUISICIÓN, ENCRIPTAMIENTO EMISIÓN Y PROCESAMIENTO DE LA INFORMACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

KENDY RONNY CAMACHO VELOZ

kendy.camacho@epn.edu.ec

DIRECTOR: EDUARDO FAUSTO ÁVALOS CASCANTE, PhD

eduardo.avalos@epn.edu.ec

DMQ, octubre 2022

CERTIFICACIONES

Yo, KENDY RONNY CAMACHO VELOZ declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Sr. Kendy Ronny Camacho Veloz

Certifico que el presente trabajo de integración curricular fue desarrollado por KENDY RONNY CAMACHO VELOZ, bajo mi supervisión.

PhD. Eduardo Fausto Ávalos Cascante

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Sr. Kendy Ronny Camacho Veloz

PhD. Eduardo Fausto Ávalos Cascante

DEDICATORIA

El siguiente trabajo de titulación lo dedico a mis padres Nelly Veloz, Romulo Camacho y a mis hermanos Joan y Adriana, quienes han sido un pilar fundamental en mi vida, siempre me han apoyado en mis buenos y malos momentos, me han aconsejado cuando he estado por un mal camino y nunca perdieron la esperanza y la fe en mi para poder culminar este largo proceso.

AGRADECIMIENTO

Agradezco a Dios por siempre caminar de mi lado, por siempre cuidarme y por siempre llevarme por buenos senderos y no dejarme tomar malas decisiones en mi vida.

Agradezco a mis padres que han sido un pilar fundamental en mi vida y siempre han estado apoyándome sobre todo en este proceso de formación como un profesional.

Agradezco a mis hermanos por siempre estar ahí cuando los necesitaba, por siempre aguantarme cuando los hacia enojar y por aconsejarme para no desistir en este proceso de mi vida.

Agradezco a mi enamorada-mejor amiga Marisol Medina por formar parte de este proceso en el cual siempre me brindo su cariño, apoyo, amor y comprensión.

Agradezco a mi Tutor de tesis PhD. Eduardo Ávalos por darle seguimiento a mi proyecto, realizar las correcciones, sugerencias con el fin de mejorar y por motivarme a culminar el proyecto.

Agradezco a mis amigos que conocí en la universidad por acompañarme en noches y madrugadas de estudio y por siempre brindarnos apoyo mutuo para no desistir en ninguna materia de la carrera.

Agradezco a mis primos Darwin y Lenin por aconsejarme y apoyarme emocionalmente para poder terminar mi carrera.

ÍNDICE DE CONTENIDO

RESUMEN

El presente Trabajo de Integración Curricular se enfoca en el desarrollo de un SISTEMA de Adquisición, Encriptamiento Emisión Y Procesamiento de la Información proveniente desde un tensiómetro digital, con el objetivo de llevar el seguimiento ambulatorio de la presión arterial y frecuencia cardiaca de un paciente. El presente documento se encuentra constituido por tres apartados que conforman el desarrollo del sistema, los cuales se detallan a continuación.

En el primer capítulo trata acerca del estudio de las herramientas de hardware y software utilizadas para el desarrollo del sistema, así como también la metodología a seguirse y los algoritmos utilizados para el procesamiento de datos provenientes del tensiómetro digital.

El segundo capítulo aborda dos secciones, por un lado, se encuentra la sección de Diseño, en la cual se definen los requisitos del sistema, la arquitectura a emplearse y los componentes que conforman el mismo, por su parte la sección de Implementación detalla el proceso de construcción de la parte de hardware y la codificación de la parte de software, todo esto de acuerdo en los requisitos planteados en la sección de Diseño.

El tercer capítulo se conforma de los resultados de la implementación del sistema, lo que ayuda a verificar que se cumpla con los requisitos establecidos, este capítulo también detalla las conclusiones que se han desprendido del desarrollo del sistema, así como también las recomendaciones a seguir en caso de realizar desarrollos similares.

PALABRAS CLAVE: Raspberry, Python, JavaScript, Socket.io, VueJS

ABSTRACT

This Curriculum Integration Work focuses on the development of a SYSTEM for the Acquisition, Encryption, Emission, and Processing of Information from a digital sphygmomanometer, with the aim of carrying out ambulatory monitoring of blood pressure and heart rate of a patient. This document is made up of three sections that make up the development of the system, which are detailed below.

The first chapter deals with the study of the hardware and software tools used for the development of the system, as well as the methodology to be followed and the algorithms used for the processing of data from the digital tensiometer.

The second chapter deals with two sections, on the one hand, there is the Design section, in which the system requirements, the architecture to be used and the components that make it up are defined, on the other hand the Implementation section details the process construction of the hardware part and the coding of the software part, all in accordance with the requirements set out in the Design section.

The third chapter is made up of the results of the implementation of the system, which helps to verify that the established requirements are met, this chapter also details the conclusions that have emerged from the development of the system, as well as the recommendations to follow in case of similar developments.

KEYWORDS: Raspberry, Python, JavaScript, Socket.io, VueJS

1 INTRODUCCIÓN

Los valores de presión arterial es una medida biológica de utilidad para indicar el estado de salud de una persona, así como el funcionamiento de su sistema cardiovascular, por lo que es de gran utilidad conocerlo para la detección temprana de enfermedades graves [1].

Existen distintos métodos para la obtención de estas medidas, siendo el más común el método de medición oscilométrico [2], el cual consiste tradicionalmente de una válvula, un brazalete inflable y un manómetro de mercurio, sin embargo, debido a la naturaleza analógica de elementos de este sistema, como pueden ser las condiciones variables del mercurio presente en el manómetro, o su aguja que está en constante movimiento existe la probabilidad de que ocurran errores de apreciación, es por ello que surge la necesidad de construir un prototipo digital que implemente elementos digitales y procesamiento de información, el cual dote al sistema de una mayor precisión, lo que ayudará a que los diagnósticos médicos sean más veraces.

Los algoritmos de procesamiento de datos permiten la eliminación del ruido proveniente de los sensores de medición, ya sea este ruido blanco gaussiano, o interferencias electromagnéticas propias de sistemas análogos.

El presente trabajo de integración curricular tiene como finalidad la implementación de la etapa de adquisición, emisión y procesamiento de datos de presión medidos desde un tensiómetro electrónico que se coloca en el paciente de manera no invasiva, permitiendo así determinar el valor de presión arterial y desplegándolo en una interfaz accesible mediante cualquier dispositivo electrónico que cuente con un navegador web, brindando así la posibilidad de registrar los valores de presión arterial correspondientes a un paciente y permitiendo registrar un historial, lo cual será de extrema utilidad para el seguimiento de la calidad de vida de los pacientes.

Con el prototipo digital se podrá realizar un diagnóstico ambulatorio al usuario para que se pueda ir analizando como va cambiando el valor de su presión en un rango de tiempo de 24 horas, con esta información del paciente se podrá visualizar en un navegador web el historial con las medidas del paciente.

1.1 OBJETIVO GENERAL

Desarrollo de algoritmo digital para la adquisición, procesamiento y emisión de la información obtenida con el fin de obtener el valor estimado de presión arterial y frecuencia cardiaca para ser visualizado en una página web.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar una revisión bibliográfica de los diferentes tipos de técnicas que facilitan la estimación de la presión arterial y frecuencia cardiaca.
2. Analizar los lenguajes de programación y herramientas para el desarrollo de algoritmos que permitan procesar la información.
3. Implementar un algoritmo de procesamiento digital de información que cumpla satisfactoriamente las necesidades del proyecto.
4. Realizar un análisis de la validez de los resultados obtenidos mediante una interfaz construida en un servidor web.
5. Seleccionar un sistema embebido que satisfaga las necesidades de procesar la información del proyecto.
6. Escritura de resultados.

1.3 ALCANCE

Se realizará una búsqueda bibliográfica de los requerimientos de hardware, software, programas y herramientas que permitan realizar el procesamiento de la información.

Se realizará una revisión bibliográfica acerca de cómo adquirir información digital entre dos dispositivos que se encuentran conectados en la misma red LAN.

Se realizará un resumen acerca de los tipos de ordenamiento y filtrado de la información, las características del hardware de procesamiento y la compatibilidad del sistema operativo a ejecutarse.

Se realizará un planteamiento de las tareas que debe poseer el sistema de procesamiento de la información (monitoreo, encriptamiento, modelado, sincronización), el alojamiento en un Gateway y la administración de comunicaciones de tipo inalámbrico.

Se desarrollará mediante un algoritmo de procesamiento de información el proceso de obtención del valor de presión arterial y frecuencia cardiaca de un paciente, el cual será solicitado y realizado por un tensiómetro electrónico que estará conectado a una red de área local inalámbrica.

Para el desarrollo del sistema se utilizarán una serie de herramientas de desarrollo de código abierto y que están a libre disposición del público en general, lo cual permitirá que el sistema no esté sujeto al pago de licencias a terceros.

El sistema estará estructurado en dos componentes, los cuales serán: Componente de Procesamiento y Manejo de Datos y Componente de Visualización de Resultados

Componente de Procesamiento y Manejo de Datos:

- Adquisición de los datos emitidos por el tensiómetro digital utilizando y configurando la librería ‘socket.io’.
- En un dispositivo Gateway se programará en lenguaje Python el algoritmo digital que procesará los datos para obtener el valor de la presión arterial y frecuencia cardiaca.
- Almacenar en una base de datos un historial de cada paciente correspondiente a sus mediciones de la presión arterial y frecuencia cardiaca.
- Gestionar el acceso a la información mediante la creación de un sistema de rutas de acceso.

Componente de Visualización de Resultados:

- Proveer una interfaz gráfica accesible a los usuarios por medio de un navegador web.
- Proporcionar menús de opciones para las distintas operaciones realizables en la interfaz gráfica.
- Permitir el registro, gestión y eliminación de pacientes.
- Desplegar en pantalla el valor de presión arterial calculado por el Componente de Procesamiento.
- Mostrar un historial de las mediciones de presión arterial y marcas de tiempo correspondientes a las mismas.

1.4 MARCO TEÓRICO

La siguiente sección tiene como objetivo mostrar la información que se necesita para poder entender la problemática del proyecto. Los temas a tratarse son: Estado del arte de sistemas de procesamiento y análisis de la información de sensores médicos, implementos médicos, herramientas de hardware, herramientas de software, presión arterial con sus componentes y valores normales, método no invasivo oscilométrico de presión sanguínea, métodos para estimar la presión arterial, frecuencia cardiaca.

1.4.1 ESTADO DEL ARTE DE SISTEMAS DE PROCESAMIENTO Y ANÁLISIS DE INFORMACIÓN DE SENSORES MÉDICOS.

Con el objetivo de mejorar la calidad de las mediciones biométricas obtenidas a través de técnicas analógicas y apoyar de mejor manera los diagnósticos a realizarse a los pacientes se han diseñado numerosos proyectos por parte de varios proyectos académicos, a continuación, se presentan una serie de los más relevantes relacionados al presente Trabajo de Integración Curricular.

Medidor Digital De Presión Arterial Y Ritmo Cardiaco: En la década de 1980 se inició el uso de dispositivos electrónicos para la medición de la presión arterial que consistían en sensores reutilizables. Vayas (1981), desarrolló un medidor digital de presión arterial y ritmo cardiaco que tiene como objetivo la implementación de sensores digitales que mejoraban la precisión del elemento de medición [3].

Caracterización Y Modelado De Sensores Capacitivos Para Aplicaciones Médicas: En 2015 Díaz, D. presentó un diseño de un (presión intracraneal, ocular y sanguínea). Si bien la información obtenida por este tipo de sensores es directa, se requiere un sistema de sellado hermético y cubierta biocompatible que permita obtener datos reales del paciente [4].

Diseño De Un Sistema Inalámbrico Para Monitoreo De Pacientes Ambulatorios, Utilizando Sensores De Presión Arterial Y Ritmo Cardiaco E Implementación De Un Prototipo De Prueba: En 2016, Ortiz, S. diseñó un sistema prototipo para el monitoreo inalámbrico de pacientes ambulatorios mediante sensores de presión arterial y ritmo cardiaco mediante el sensor MPX5100 y herramientas online como MySQL, PHP, HTML y JavaScript utilizando la arquitectura MVC [5].

Diseño E Implementación De Un Sistema No Invasivo Para El Monitoreo De Frecuencia Cardiaca Mediante Detección Automática De Una Sección De Piel: El monitoreo no invasivo de frecuencia cardiaca mediante adquisición de datos a través de la piel fue estudiado por Portilla K. 2018, usando para ello un computador de placa reducida

Raspberry Pi 3 Modelo B, cámaras de video y la librería abierta OpenCV, el procesamiento de las imágenes de video fue realizado a partir del algoritmo Kanade-Lucas-Tomasi y el periodo aproximado del procesamiento de datos fue mucho menor a la medición manual y comparable con tensiómetros digitales comerciales [6].

Modelo Sistémico Para La Adquisición De Datos Para Mhealth: En 2018, Hernández A. desarrolló un sistema de monitoreo de pulsos y temperatura con el uso de una tarjeta programable Arduino UNO y un microcontrolador Atmel AVR usando el protocolo de comunicación Bluetooth HC-06 para la posterior visualización de datos en teléfonos inteligentes a través de una aplicación desarrollada con la App Inventor y el almacenamiento de datos local y en la nube (FireBase), en este contexto el autor señaló que el principal inconveniente en el funcionamiento del modelo fue la incompatibilidad del protocolo de comunicación entre el dispositivo móvil y el lector biométrico [7].

Procesamiento Y Análisis De Señales Biomecánicas Adquiridas Por Redes De Sensores: El procesamiento de señales biomédicas fue también estudiado por Concha, D. (2018) mediante el seguimiento del estado motriz de pacientes con la enfermedad de Parkinson, para ello empleó una red de sensores comunicados por módulos XBee 2mW PCB Antenna-ZigBee Mesh, los inconvenientes en la adquisición de datos y la velocidad de transmisión de la información de este sistema tuvieron relación con la disposición de las computadoras con respecto al paciente. Además, usó como sistema alternativo de medición el dispositivo 9DOF Razor IMU (Inertial Measurement Unit), cuyas salidas fueron procesadas por un micro controlador ATmega328 para la conversión de datos, los cuales fueron enviados por Bluetooth y recibidos en un software desarrollado en el lenguaje de programación LabView 2014 que convirtió los datos de Volts a las unidades de cada sensor del dispositivo IMU [8].

Diseño De Prototipo Doctor Pi Para La Medición Y Monitorización De Signos Vitales En Adultos Mayores Utilizando Sensores Biométricos Y Médicos Acoplados A Raspberry Pi: Por su parte, Valencia W., 2018 desarrolló un sistema de telemetría de signos fisiológicos conformado por sensores e-Healt v2, Arduino, Raspberry Pi, sensores y módulos GSM/GPRS [9].

1.4.2 IMPLEMENTOS MÉDICOS

Como base para el desarrollo del sistema se utilizará implementos médicos comerciales con el fin de realizar una comparación y validación de los resultados del procesamiento de datos adquiridos desde el tensiómetro digital, a continuación, se detallan los implementos a usar:

OMRON X3 Comfort: Es un tensiómetro digital que se encuentra clínicamente aprobado, cuyo objetivo es brindar la posibilidad de controlar la hipertensión a usuarios domésticos, el dispositivo es capaz de ser utilizado por dos usuarios y tiene la capacidad de almacenar hasta 60 mediciones de presión arterial y frecuencia cardiaca.

En la Figura 1.1 se puede observar una fotografía del equipo provista por el fabricante.

Figura 1.1. Tensiómetro digital OMRON X3 Comfort.

1.4.3 HERRAMIENTAS DE HARDWARE

El sistema tiene como objetivo servir como Gateway que recibirá los datos de lectura de presión arterial desde un tensiómetro digital, es por ello que surge la necesidad de implementar un dispositivo de hardware que aloje el sistema y permita comunicarse de manera inalámbrica con el tensiómetro digital.

Para el desarrollo del sistema se utilizará un dispositivo de hardware de bajo coste como Gateway para su alojamiento, y para su programación se utilizará una serie de herramientas de programación de código abierto que servirán para el despliegue de un servicio y una interfaz web para la presentación de resultados.

Raspberry Pi Model 3B: La Raspberry Pi Model 3B es una tarjeta de desarrollo que consiste en un sistema de cómputo de bajo coste y tamaño reducido conocido como *Single Board Computer (SBC)* [8], desarrollado y comercializado por la Fundación Raspberry [11] con el objetivo de promover la educación de niños y adultos, particularmente en el campo de las computadoras, ciencias de la computación y temas relacionados [12], cuenta con un procesador basado en la arquitectura ARM [13], y es capaz de ejecutar un sistema operativo que provee de las funciones más básicas de un computador personal, además de contar con interfaces típicas de estos, como son puertos USB, salida de pantalla HDMI, conector para tarjetas MicroSD, puerto Ethernet, entre otros, además interfaces específicas como puertos GPIO y conector para cámara CSI, además de interfaces físicas cuenta con conexiones inalámbricas como son Bluetooth y Wi-Fi.

En la Figura 1.2 se puede apreciar el diagrama de interfaces de la tarjeta de desarrollo Raspberry Pi Model 3B [14].

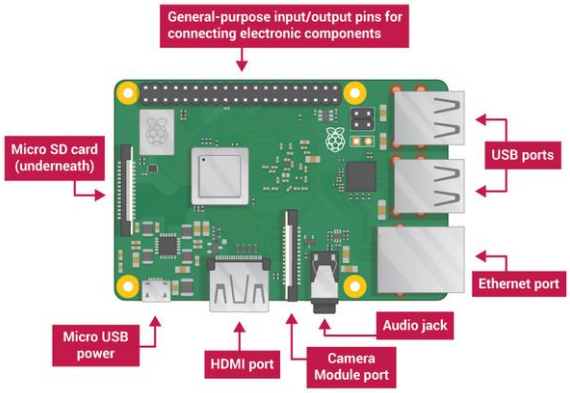


Figura 1.2 Diagrama de Interfaces Raspberry Pi Model 3B.

Enrutador Inalámbrico TP-Link Archer C24: Se trata de un enrutador inalámbrico que permite la creación de una red de área local tanto inalámbrica como cableada, cuenta con el estándar IEEE 802.11n/ac para la comunicación inalámbrica y ofrece una velocidad máxima de 750Mbps, permite la conexión de hasta 32 usuarios y el aprovisionamiento de recursos de red para cada uno de ellos.

En la Figura 1.3 se puede observar una fotografía del enrutador provista por el fabricante.

Figura 1.3. Enrutador inalámbrico TP-Link Archer C24

1.4.4 HERRAMIENTAS DE SOFTWARE

Raspbian OS: Raspbian OS es un sistema operativo de código abierto basado en la distribución *Debian Linux* [15] cuyos componentes de software han sido especialmente adaptados para ser compatibles con el hardware de la Raspberry Pi, provee de los paquetes necesarios para la ejecución de múltiples herramientas de software y lenguajes de programación [16].

MySQL: MySQL es un sistema de gestión de bases de datos relacionales, desarrollada en principio por MySQL AB y mantenida actualmente por Oracle Corporation [17], al ser de código abierto es uno de los gestores de bases de datos más utilizados en la actualidad, lo que le provee de una gran comunidad de desarrolladores y herramientas relacionadas, se basa en una arquitectura cliente-servidor, es decir, provee una interfaz de comunicación que provee servicio a múltiples clientes a la vez, lo que la vuelve asíncrona y perfecta para trabajar con sistemas escalables y que requieran grandes cantidades de datos.

Sequelize: Sequelize es un marco de trabajo *ORM (Object Relational Mapping)* compatible con el lenguaje de programación JavaScript, que provee una capa de abstracción sencilla para facilitar el trabajo con bases de datos, soporta varios gestores como son PostgreSQL, MySQL, MariaDB, SQLite y MSSQL, su objetivo es proveer un método universal para trabajar con programas escritos en los lenguajes de programación TypeScript y JavaScript, cuenta con múltiples características, siendo la más importante el soporte a promesas para el intercambio asíncrono de datos entre aplicaciones cliente y la base de datos [18].

JavaScript: Es un lenguaje de programación del tipo interpretado compatible con navegadores web y entornos de ejecución como *NodeJS o NestJS,* su uso se extiende a la mayoría de páginas web alojadas en internet, ya que es el encargado de proveerlas de interactividad con el usuario, se encuentra actualmente basado en el estándar ECMAScript y es mantenido por *Mozilla Foundation* [19].

NodeJS: Es una tecnología conocida como Entorno de Ejecución compatible con el lenguaje JavaScript, el cual es inicialmente era únicamente ejecutable en navegadores web en el lado del cliente, sin embargo, debido a su popularidad surgió la necesidad de poder ejecutarlo en el lado del servidor, añadiendo así funcionalidades típicas de otros lenguajes de programación del lado del servidor, como acceso al sistema de archivos, símbolo del sistema o acceso a la red, NodeJS ha sido desarrollado utilizando el motor V8 desarrollado por Google y es ejecutable en Linux, Windows o MacOS, por lo que se define como un entorno multiplataforma [20].

REST: *REST (Representational State Transfer)* es una definición de arquitectura para el desarrollo de aplicaciones cliente-servidor, que utiliza el protocolo *HTTP* [21] como medio para el transporte de información, así como también como dialecto de comunicación, utilizando sus distintos métodos de petición conocidos como *HTTP Verbs* [22] para el intercambio y atención de solicitudes [23].

En la Figura 1.4 se presenta un esquema básico de los verbos utilizados por la arquitectura REST.

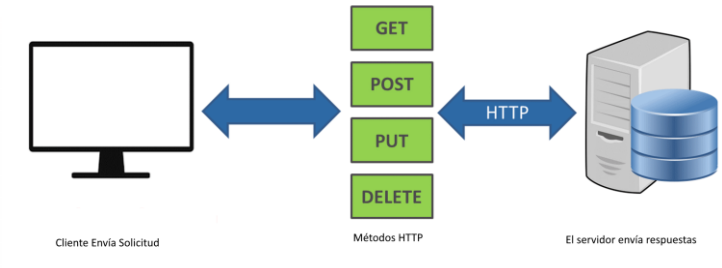


Figura 1.4. Esquema de arquitectura REST

API REST: Una API *(Application Programming Interface)* REST es una aplicación basada en la arquitectura REST que se ejecuta en un servidor de aplicaciones y que permite el intercambio de datos entre un cliente, el cual puede ser una página web, una aplicación de escritorio o una aplicación móvil y el servidor que la aloja, se encarga de proveer de métodos para el acceso a la información y gestiona los datos alojados en el servidor [24].

En la Figura 1.5 se puede observar un esquema del modelo API REST.

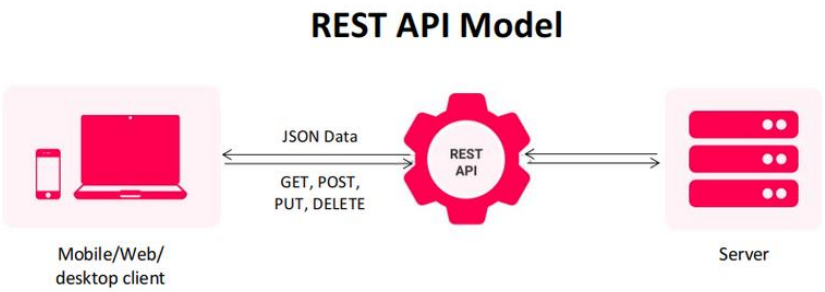


Figura 1.5. Modelo API REST

ExpressJS: ExpressJS es un marco de trabajo compatible con NodeJS, cuyo objetivo es simplificar la creación de aplicaciones web escritas en JavaScript en el lado del servidor, aunque NodeJS por sí solo es capaz de crear una aplicación web Express facilita esta labor, permitiendo ahorrar tiempo de desarrollo y añadiendo funcionalidades de seguridad y gestión conocidas como Middlewares [25].

VueJS: Es un marco de trabajo para el desarrollo de aplicaciones web, basado en el lenguaje JavaScript, tiene como objetivo principal el diseño y desarrollo de interfaces web dinámicas conocidas como Single Page Applications (SPAs), este marco de trabajo dota de funcionalidades adicionales al lenguaje JavaScript, como son la gestión de rutas, manejo de almacenamiento y trabajo con múltiples componentes, su ventaja principal sobre JavaScript puro es su modularidad, siendo su núcleo pequeño y eficiente, pero extensible

si así se requiere, lo que permite dotar a las aplicaciones desarrolladas en el mismo una gran cantidad de funcionalidades en un corto periodo de tiempo [26].

Socket.IO: Es una librería que tiene como objetivo establecer comunicación entre dispositivos que funcionan en una red para el manejo y creación de eventos de notificación en aplicaciones web que trabajan en tiempo real, añade comunicación asíncrona y bidireccional, consta de dos componentes, una biblioteca del lado del cliente, la cual se ejecutará en un navegador web y un servicio web a ejecutarse en el lado del servidor mediante el entorno de ejecución NodeJs [27].

***Python*:** Es un lenguaje de programación multiparadigma de alto nivel de propósito general y multiplataforma de código abierto, cuya filosofía es proveer facilidad y claridad en la escritura del código y una implementación eficiente de software en corto tiempo; características que lo han dotado de gran popularidad en varios campos relacionados a la programación [28].

Flask: Es un marco de trabajo de código abierto y que utiliza la licencia BSD escrito en Python, utilizado para el desarrollo de aplicaciones web, su propósito es ser liviano y proveer un rápido desarrollo y despliegue de aplicaciones con un número mínimo de líneas de código [29].

BCrypt: Es un algoritmo de cifrado creado por Niels Provos y David Maziéres, utilizado en la mayoría de sistemas Linux, este tiene como objetivo la creación de un hash único para cada dato provisto, lo que permite almacenarlos de manera segura en un Gestor de Base de Datos [30].

1.4.5 PRESION ARTERIAL, COMPONENTES Y VALORES NORMALES.

La presión arterial es una fuerza que ejerce la sangre hacia las paredes de las arterias. Cuando el corazón se contrae envía sangre a las arterias por lo que la presión llega a un punto máximo denominándose presión sistólica. En cambio, cuando por las arterias no entra sangre es porque el sistema cardiovascular se encuentra en la etapa de relajación, en esta etapa se puede encontrar la presión mínima en ese momento que es denominada presión diastólica [35].

En la Figura 1.6 se puede observar que las componentes de la presión arterial (sistólica y diastólica) que aparecen durante proceso del desinflado, además, se observa la componente de presión media que generalmente se encuentra en la mitad de las anteriores componentes.

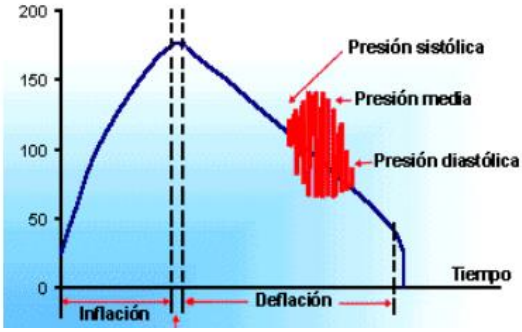


Figura 1.6. Componentes de la presión arterial [36].

En la tabla 1.4 se tiene los valores de presión arterial según la organización mundial de la salud.

Tabla 1.4 Valores de presión arterial [36]

Clasificación	Presión arterial sistólica (mmHg)	Presión arterial diastólica (mmHg)
Adecuada	Menor a 120	Menor a 80
Normal	Entre 120 y 130	Entre 80 y 84
Elevada	Entre 130 y 139	Entre 85 y 89
Alta (hipertensión grado 1)	Entre 140 y 159	Entre 90 y 99
Alta (hipertensión grado 2)	Entre 160 y 179	Entre 100 y 109
Alta (hipertensión grado 3)	Mayor o igual a 180	Mayor o igual a 110
Hipertensión Sistólica aislada	Mayor o igual a 140	Menor o igual que 90

1.4.6 MÉTODO NO INVASIVO OSCILOMÉTRICO DE PRESIÓN SANGUÍNEA

Este método es utilizado a través del inflado y desinflado del brazalete con el fin de obtener oscilaciones de presión en cada latido. Una vez que el brazalete llega a su inflado máximo las paredes de las arterias empiezan a oscilar o vibrar debido a que la sección de la arteria esta ocluida. Con la ayuda de un transductor se puede detectar estas vibraciones de presión. Las oscilaciones llegan a un valor máximo a medida que la presión del brazalete va disminuyendo y se extinguen cuando el flujo de sangre es normal. En la Figura 1.7 se puede apreciar los valores de presión durante el inflado y desinflado, además, se puede observar la forma de onda de las oscilaciones cuando va disminuyendo la presión del brazalete [32].

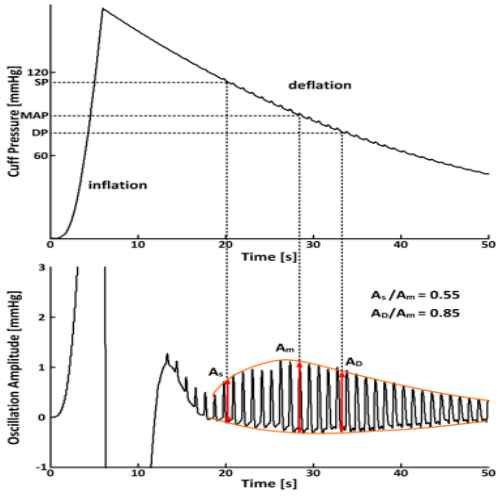


Figura 1.7. Forma de onda de la presión en el mango y sus oscilaciones medidas por un transductor [33].

1.4.7 MÉTODOS PARA DETERMINAR LA PRESIÓN ARTERIAL

Para determinar la presión arterial teniendo como señal de entrada una onda oscilatoria como muestra la Figura 1.7 se la puede estimar en función de la envolvente de la señal para la cual existen dos métodos, uno en función de las alturas y el otro en función de la pendiente. Para cualquiera de los dos métodos primero se necesita obtener la envolvente de la señal en donde se generan las oscilaciones [38] [36].

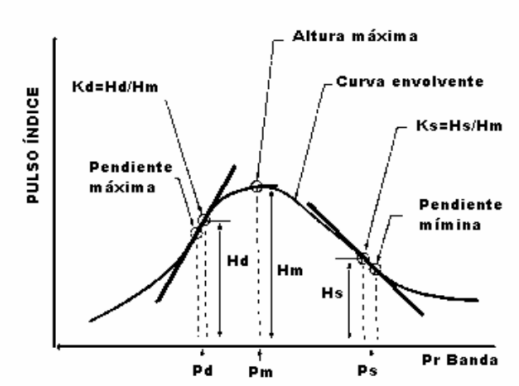


Figura 1.8. Forma de onda índice de la señal oscilométrica [36].

En la Figura 1.8 se tiene los dos métodos para la estimación de la presión arterial. El primer método que es en función de la pendiente tiene como función encontrar la pendiente mínima y máxima en la envolvente de la señal para luego ir a la posición en donde está en valor real de presión con los índices de las pendientes que se estimaron. Para el segundo método en función de las alturas se tiene los parámetros 'Hd-Hs-Hm' que son las alturas de la envolvente 'diastólica-sistólica-máxima'. Para utilizar este método se utilizan las siguientes formulas.

$$K_S = \frac{H_s}{H_m} \quad (1)$$

$$K_D = \frac{H_d}{H_m} \quad (1.1)$$

En donde el valor de ‘Kd-Ks’ toman un valor de ‘0.7 y 0.5’, luego el valor que se debe encontrar es ‘Hs y Hd’ por lo que el ‘Hm’ es el valor máximo de la envolvente generalmente conocido como MPA (índice de presión media máxima) [38].

1.4.8 FRECUENCIA CARDÍACA

La frecuencia cardiaca es el número de veces que el corazón late en un intervalo de tiempo que generalmente es un minuto. Una persona normal adulta tiene entre 60 a 100 latidos por minuto. El ritmo cardiaco es muy variable según como se tome las medidas ya que variarán si la persona se encuentra de pie o sentado, además comienza a variar en función que incrementa la edad de la persona. En las mujeres la frecuencia cardiaca suele ser mayor que en los hombres [37].

2 METODOLOGÍA

En el presente capítulo se describirán los procedimientos realizados para la implementación del sistema.

En el apartado 2.1 se detallarán los requisitos del sistema, mientras que en la etapa 2.2 se detalla la arquitectura base del sistema, mientras que en el apartado 2.3 se describe a detalle la arquitectura del Componente de Procesamiento y Manejo de Información, así mismo en el apartado 2.4 se detalla la arquitectura del Componente de Presentación de Resultados.

2.1 REQUISITOS DEL SISTEMA

En el transcurso de la presente sección se detallan los requisitos a cumplir una vez el sistema sea implementado, los cuales han sido establecidos en base a la investigación de tecnologías de Gateway tanto comerciales como prototipos de investigación analizados en la sección previa, los requisitos del sistema se detallan en la tabla 2.1, mientras que los requisitos del Componente de Procesamiento y Manejo de datos se detallan en la tabla 2.2, en la tabla 2.3 se detallarán los requisitos del Componente de Visualización de Resultados, finalmente se añadirá una tabla adicional 2.4 que detallará los requerimientos adicionales del sistema.

Tabla 2.1. Requisitos del Sistema

Requisito	Detalle	Objetivo en la Implementación
Capacidad de interacción bidireccional entre el sistema y el tensiómetro electrónico.	**Entradas:** Solicitud HTTP POST con datos de medición incluidos en el cuerpo de la petición.	Obtener los valores de presión arterial leídos por el tensiómetro digital para su posterior procesamiento
	Salidas: Solicitud HTTP GET hacia el tensiómetro electrónico para iniciar la lectura de presión arterial.	Accionar el tensiómetro digital por medio de la pulsación de un botón en la interfaz web del sistema.
Comunicación inalámbrica entre el Gateway del sistema y el tensiómetro electrónico.	El sistema debe ser capaz de interactuar de manera inalámbrica con el tensiómetro digital.	Ambos dispositivos deben estar conectados en la misma red de área local inalámbrica. No debe existir impedimentos en la comunicación de ambos dispositivos. Posibilitar la transmisión y recepción de solicitudes HTTP por medio de la red local inalámbrica.
Procesamiento de información de mediciones de presión arterial.	**Entradas:** Solicitud HTTP POST con datos de medición incluidos en el cuerpo de la petición.	Recibir los valores de presión arterial leídos por el tensiómetro digital para su procesamiento.
	Salidas: Dato numérico de presión arterial procesado enviado hacia la base de datos.	Procesar los valores de presión arterial para la obtención de su valor numérico y almacenarlo en la base de datos
Visualización de resultados en interfaz gráfica web	El sistema debe ser capaz de mostrar el resultado del procesamiento de la información en una interfaz web amigable con el usuario.	Mostrar los valores numéricos de presión arterial en la interfaz web como resultado del procesamiento.
Almacenar historial de mediciones de presión arterial	**Entradas:** Valores numéricos de presión arterial resultado del procesamiento de las mediciones leídas por el tensiómetro electrónico.	Realizar un histórico de las mediciones de presión arterial para cada paciente registrado.

	Salidas: Tabla de valores numéricos de presión arterial extraída desde la base de datos.	

Tabla 2.2. Requisitos del Componente de Procesamiento y Manejo de Datos

Requisito	Detalle	Objetivo en la Implementación
Implementación de un sistema gestor de base de datos.	Instalar un sistema gestor de base de datos que registrará los valores de presión arterial, así como también la información de usuarios y pacientes del sistema.	Instalar MySQL en modo servidor y crear una base de datos que contenga las tablas correspondientes a cada tipo de dato usado en el sistema.
Implementación de un sistema ORM.	Implementar un sistema que permita el manejo de la información almacenada en la base de datos.	Implementar SequelizeJS para realizar el mapeo entre las tablas de la base de datos y el resto del componente de procesamiento.
Implementación de API REST	Implementar una API REST para la atención de peticiones de almacenamiento y procesamiento de información.	Desarrollar una API REST en ExpressJS que permita atender las peticiones de procesamiento de datos de las mediciones obtenidas por el tensiómetro digital, así como también permitir las peticiones de almacenamiento de valores numéricos de presión arterial, valores históricos y la información pertinente a usuarios y pacientes.

Tabla 2.3. Requisitos del Componente de Visualización de Resultados

Requisito	Detalle	Objetivo en la Implementación
Creación de una interfaz gráfica para el acceso a las funcionalidades del sistema.	Permitir el acceso a las funcionalidades del sistema por medio de una interfaz gráfica accesible por medio de un navegador web.	Desarrollar una aplicación web con VueJS para la interacción entre el usuario y el sistema.
Creación de menú de acceso a las funcionalidades del sistema.	Permitir el fácil acceso a las funcionalidades del sistema.	Crear un menú de opciones en la interfaz gráfica que permita acceder a cada una de las ventanas correspondientes a las

		distintas funcionalidades del sistema.
Creación de ventana para la interacción con el tensiómetro digital.	Implementar un control para que las mediciones del tensiómetro digital sean especificadas para cada paciente y pueda comenzar el inicio de la lectura de mediciones de presión arterial.	Desarrollar una ventana con un control de tipo botón que realice una solicitud HTTP GET hacia el tensiómetro digital para que este inicie la medición de valores de presión arterial.
Creación de ventana de gestión de usuarios	Permitir el registro, visualización, edición y eliminación de usuarios del sistema.	Desarrollar una ventana con los controles necesarios para poder registrar, visualizar, editar y eliminar usuarios en el sistema.
Creación de ventana de gestión de pacientes	Permitir el registro, visualización, edición y eliminación de pacientes del sistema.	Desarrollar una ventana con los controles necesarios para poder registrar, visualizar, editar y eliminar pacientes en el sistema.
Creación de ventana para la visualización de historial de mediciones de presión arterial.	Permitir visualizar un histórico de mediciones de presión arterial.	Desarrollar una ventana con los controles necesarios para poder visualizar los valores registrados de las lecturas de presión arterial para cada paciente.

Tabla 2.4. Requisitos adicionales del sistema

Requisito	**Detalle**	**Objetivo en la Implementación**
Desplegar el sistema en un Gateway	Instalar el sistema en un dispositivo de hardware independiente conocido como Gateway.	Instalar y desplegar el sistema en la tarjeta de desarrollo Raspberry Pi Model 3B.
Instalar componentes de software necesarios para la ejecución del sistema en el Gateway	Instalar los paquetes de software necesarios para la ejecución automática del sistema en el Gateway.	Instalar el sistema con PM2 para su ejecución automática tras cada reinicio del Gateway.

2.2 ARQUITECTURA BASE DEL SISTEMA

En el presente apartado se detallará la arquitectura base del sistema a implementarse en base a los requisitos detallados en la sección anterior.

El sistema estará estructurado en dos componentes principales, uno externo y una base de datos para el almacenamiento de la información, los componentes principales son: Componente de Procesamiento y Manejo de Datos, y Componente de Visualización de Resultados, el componente externo es el tensiómetro digital, del cual no se detallará la arquitectura, pero si se detallarán las interacciones con el mismo, por último la base de datos del sistema es una base de datos alojada en MySQL que almacenará los datos de usuarios, mediciones y pacientes.

En la Figura 2.1 se puede visualizar el diagrama del sistema y sus componentes correspondientes interacciones.

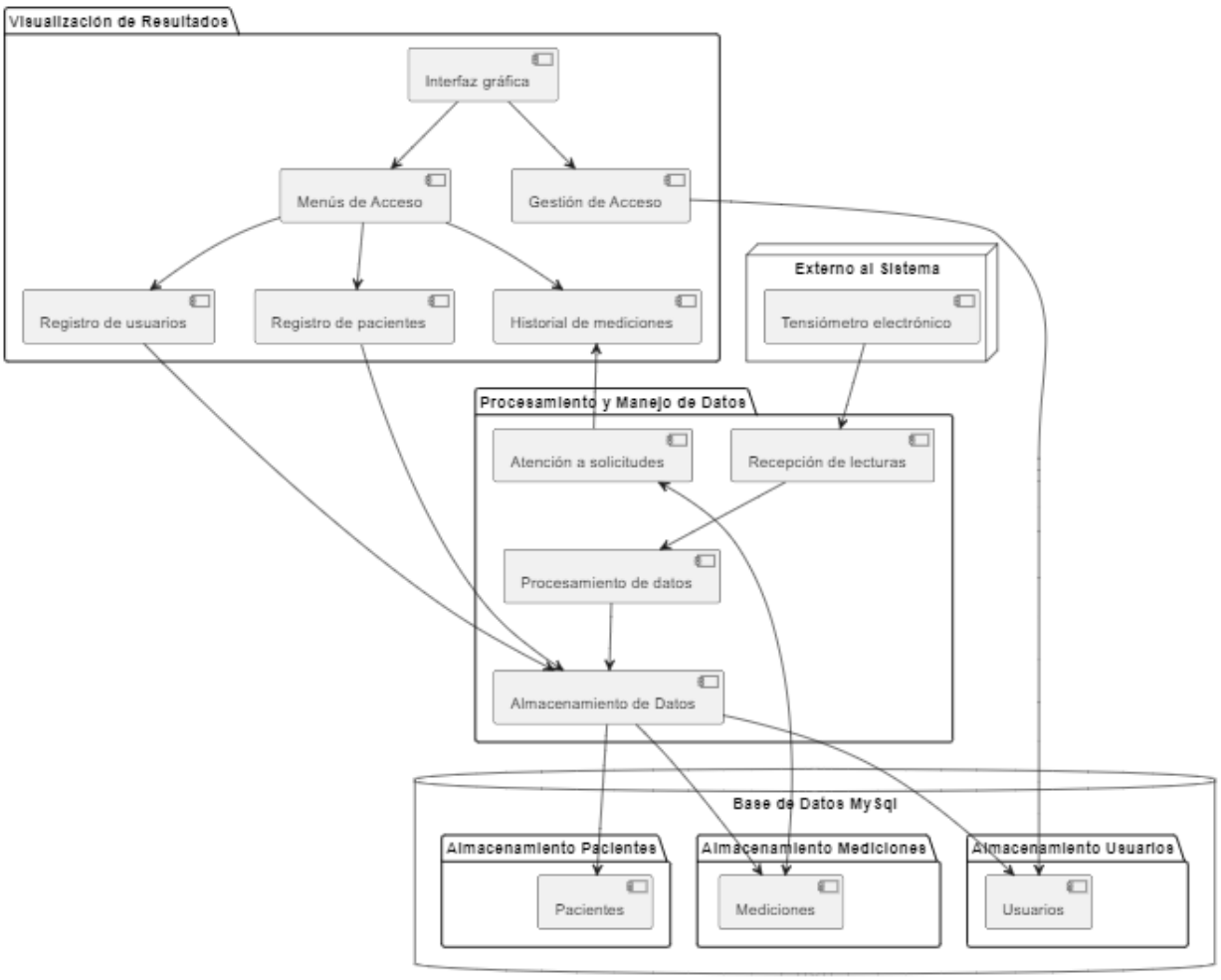


Figura 2.1. Diagrama de componentes del sistema.

2.3 ARQUITECTURA DEL COMPONENTE DE PROCESAMIENTO Y MANEJO DE INFORMACIÓN

El objetivo de este componente es el procesamiento de los valores de medición obtenidos desde el tensiómetro electrónico, su posterior almacenamiento y recuperación, además de permitir almacenar información relevante de usuarios y pacientes del sistema.

El Componente de Procesamiento y Manejo de Información se alojará en el Gateway Raspberry Pi Model 3B, el cual será el encargado de desplegar la pila de software necesaria y de proveer la interfaz de comunicación mediante el estándar IEEE 802.11, el cual define la creación de una red de área local inalámbrica que incorpora las capas: física, de acceso al medio, de transporte y de red necesarias para comunicarse con el tensiómetro electrónico.

La interfaz inalámbrica se compondrá del tensiómetro electrónico, el cual está codificado en el microcontrolador ESP32, así como también un enrutador inalámbrico y el Gateway Raspberry Pi, en la Figura 2.2 se muestra el diagrama de la interfaz inalámbrica.

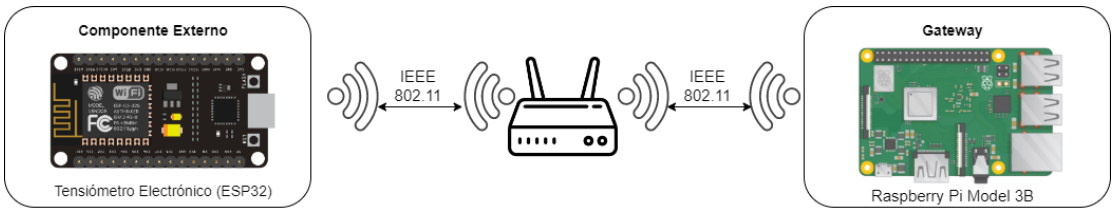


Figura 2.2. Interfaz inalámbrica.

- **Componente Externo:** Se compone del microcontrolador ESP32, mismo que comandará el tensiómetro electrónico y se encargará de enviar los datos de presión arterial hacia el Gateway.
- **Enrutador Inalámbrico:** Se trata de un dispositivo de hardware utilizado para la creación de una red de área local inalámbrica, implementa comunicación bidireccional entre el Gateway y el componente externo mediante el estándar IEEE 802.11.
- **Gateway:** Se trata de la tarjeta de desarrollo Raspberry Pi Model 3, la cual se conecta a la red de área local inalámbrica creada por el enrutador y es el que recibe los datos enviados por el tensiómetro digital.

La pila de software estará compuesta por los frameworks Flask y Express, los cuales se encargarán del procesamiento de la información y su almacenamiento respectivamente,

así como también de una base de datos alojada en el gestor MySQL, en la Figura 2.3 se puede observar un diagrama de la arquitectura del componente.

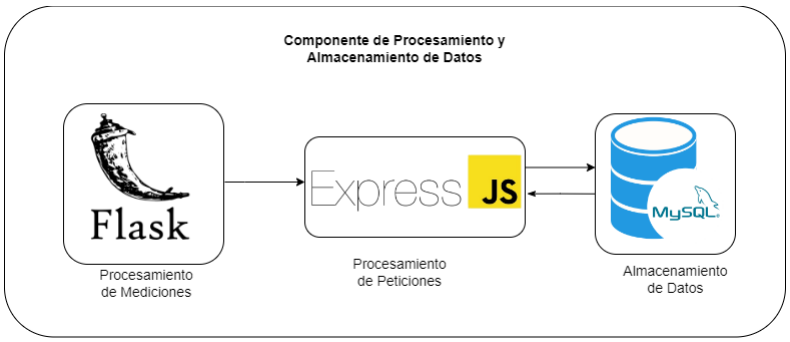


Figura 2.3. Diagrama de pila de software.

- **Flask:** Se encarga de establecer el subcomponente de procesamiento, este recibirá una petición HTTP GET con los valores de presión arterial adquiridos por el tensiómetro inalámbrico.
- **Express.js:** Se encarga de definir una API REST que definirá métodos relacionados al almacenamiento de datos, tanto de presión, usuarios y pacientes.
- **Base de datos MySQL:** Su trabajo es el de definir un esquema estructurado para proveer el almacenamiento permanente de datos del sistema.

La pila de software propuesta servirá para la creación de una API REST que será la encargada de atender las peticiones de procesamiento y almacenamiento de información.

Se ha utilizado Swagger UI [31] para la definición de la API REST.

En la Figura 2.4 se puede evidenciar los métodos generados en la API REST para el procesamiento de datos.

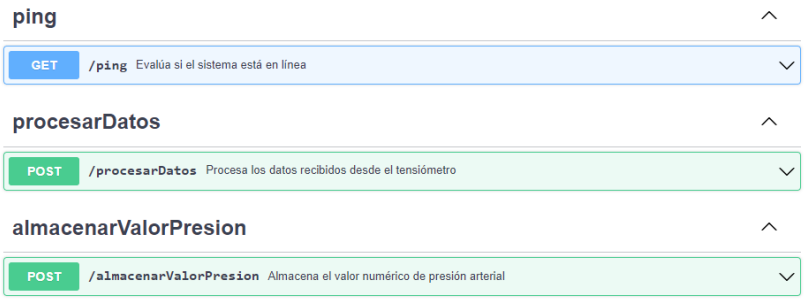


Figura 2.4. Métodos HTTP para el procesamiento de datos.

En la Figura 2.5 se detallan los métodos relacionados a la atención de peticiones para el manejo de usuarios del sistema.

Figura 2.5. Métodos HTTP para el manejo de usuarios.

La Figura 2.6 define los métodos para el intercambio de solicitudes HTTP relacionados con los datos de pacientes.

Figura 2.6. Métodos HTTP para el manejo de pacientes.

2.4 ARQUITECTURA DEL COMPONENTE DE VISUALIZACIÓN DE RESULTADOS

El objetivo del presente Componente es mostrar los resultados de las mediciones realizadas por el tensiómetro electrónico, además de proveer un historial para el diagnóstico ambulatorio de un paciente, el cual permitirá analizar las variaciones de presión arterial en un rango de tiempo determinado.

La arquitectura del componente se basa en el framework VueJS, el cual será utilizado para la creación de una interfaz gráfica que permita interactuar con el Componente de Procesamiento y Manejo de Información.

La interfaz gráfica está compuesta por varios módulos que se conocen como ventanas, estas ventanas definen controles de usuario para la manipulación del sistema de una forma fácil e intuitiva.

Al igual que el Componente de Manejo y Procesamiento de Información, el Componente de Visualización de Resultados se alojará en el Gateway Raspberry Pi Model 3B y se compondrá de una pila de software que será la encargada de crear y desplegar la interfaz gráfica para presentación al usuario final.

En la Figura 2.7 se aprecia el diagrama de la arquitectura del Componente de Visualización de Resultados, así como también sus interacciones con el usuario final y los demás componentes del sistema.

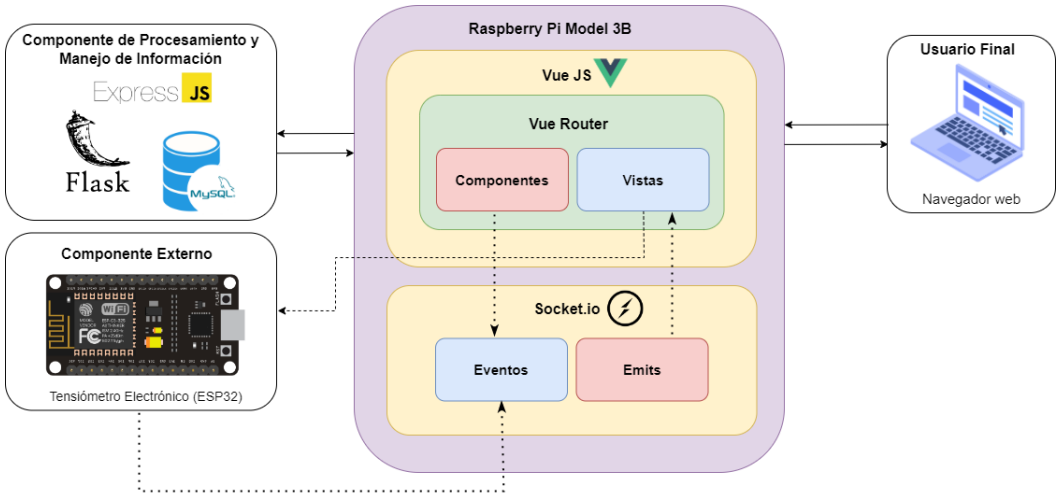


Figura 2.7. Diagrama de Arquitectura del Componente de Visualización de Resultados.

- **Vue JS:** Se encarga de definir la interfaz gráfica que utilizará el usuario final, definiendo los módulos correspondientes a lecturas, registro y seguimiento ambulatorio de pacientes.
- **Socket.io:** Se encarga de proveer una interfaz asíncrona de comunicación entre la interfaz de usuario y los Componentes de Procesamiento y Manejo de Información y Componente Externo, para ello define métodos conocidos como Eventos y Emits, que son los mecanismos de intercambio de Información.
- **Componente Externo:** Se encarga de enviar un mensaje a través de un Emit por medio de Socket.io para indicar a la interfaz del usuario que se ha realizado una medición de presión arterial.

- **Componente de Procesamiento y Manejo de Información:** Se encarga de proveer los mecanismos para la ejecución de las acciones definidas en las interfaces de usuario.

2.5 DISEÑO DE HARDWARE

2.5.1. ALIMENTACIÓN

El elemento de Hardware principal del sistema será la tarjeta de desarrollo Raspberry Pi Model 3B, y esta será la que actúe como Gateway, para lo cual este debe contar con un sistema de alimentación de energía que cumpla con los parámetros establecidos por el fabricante, siendo este un conector de tipo micro USB conectado a una fuente de alimentación de 5 voltios con una corriente de 2.5 amperios, con una vida media de 50000 horas y protección anti-cortocircuitos [32].

2.5.2. RED DE ÁREA LOCAL INALÁMBRICA

Mediante el enrutador inalámbrico Archer C24 se creará una red de área local inalámbrica que permita conectar al Gateway con el tensiómetro digital, así como también a los dispositivos clientes que visualizarán los resultados del procesamiento, esta red se configurará mediante la interfaz web del enrutador, estableciéndose una red con el SSID: TP-Link_A216, y cuya contraseña será: 57723709, en la Figura 2.8. se muestra la interfaz del enrutador inalámbrico y los parámetros a configurar de la misma.

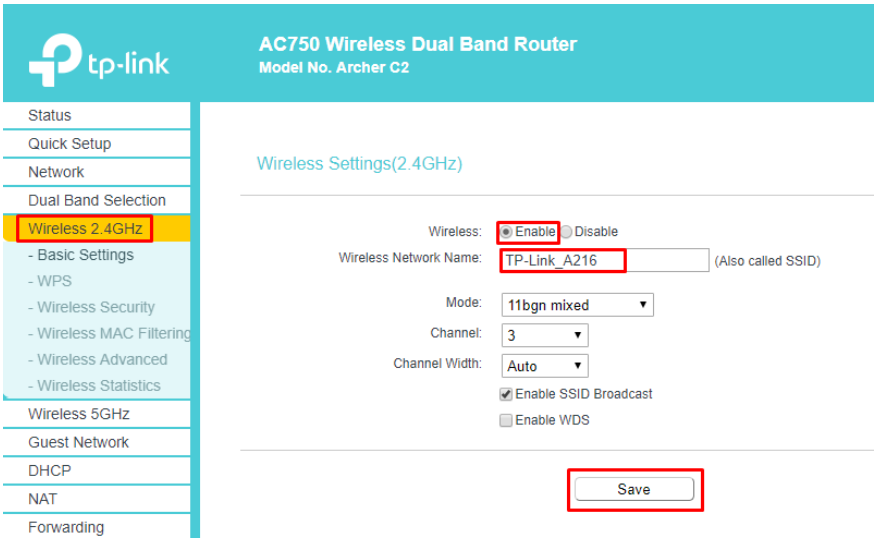


Figura 2.8. Configuración de red inalámbrica.

2.5.3 DISEÑO DE CASE EN 3D

El Gateway del sistema, al estar basado en una Raspberry Pi Model 3B tiene una naturaleza de una tarjeta de desarrollo, es por ello que no incluye con una carcasa o protector por defecto, lo cual puede derivar en posibles problemas de hardware, debido a

que sus componentes se encuentran desprotegidos, haciéndolos susceptibles a posibles corto circuitos o estrés mecánico, es por esta razón que se ha diseñado un case que protegerá al Gateway. El case ha sido desarrollado en la herramienta de modelado Autodesk Fusión 360, adoptando un diseño modular dividido en 3 partes, lo cual permitirá su fácil manipulación y armado. Las partes que constituyen el case son, tapa superior y tapa inferior las cuales serán presentadas a continuación.

La tapa inferior se conforma de una base para la tarjeta de desarrollo Raspberry Pi Model 3B, y cuenta con orificios de ventilación en la parte inferior, además de los orificios correspondientes a los puertos de entrada y salida del dispositivo. La Figura 2.9 muestra la vista superior de la tapa inferior del modelo 3D.

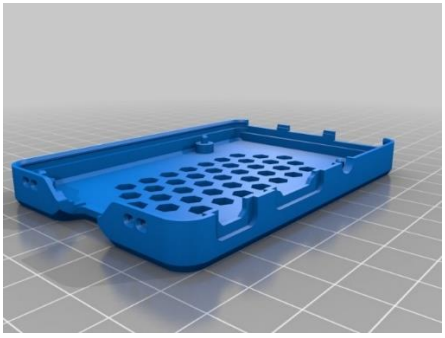


Figura 2.9. Vista superior de la tapa inferior del case 3D.

Por su parte la tapa superior cuenta con orificios encaminados a dar soporte a los pines GPIO de la Rasberry Pi Model 3B, además de complementar los orificios correspondientes a los puertos del dispositivo, en la Figura 2.10 se puede evidenciar una captura del modelo 3D de la tapa superior.

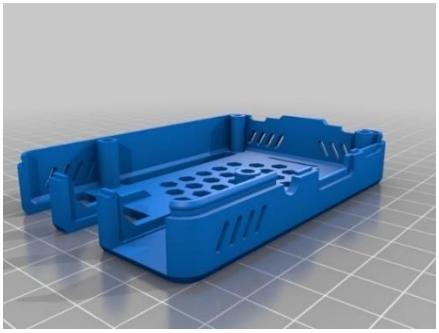


Figura 2.10. Tapa superior del case 3D.

2.6 DISEÑO DE SOFTWARE

2.6.1 CODIFICACIÓN DEL COMPONENTE DE PROCESAMIENTO Y MANEJO DE DATOS

El Componente de Procesamiento y Manejo de Datos será el encargado, tanto de procesar los datos que provienen del tensiómetro digital, para la codificación de este Componente ha sido realizada utilizando como base el framework ExpressJS y el Middleware Sequelize, el cual permite la creación de un proyecto que conecte una API REST desarrollada en ExpressJS con una Base de Datos.

Este componente consta de tres subsistemas, conocidos como Base de Datos, Servicio de Manejo de Datos, y Servicio de Procesamiento de Datos, los cuales se detallan a continuación:

2.6.2 CODIFICACIÓN DE LA BASE DE DATOS

El almacenamiento de información del sistema será responsabilidad de una Base de Datos Relacional alojada el Gestor de Base de Datos MySQL, el cual permite almacenar información estructurada en tablas que guardará la información de cada una de las entidades que intervienen en el sistema, para ello se han codificado una serie de archivos conocidos como modelos mediante el Middleware Sequelize, el cual permite realizar un mapeo de la información recibida desde el tensiómetro digital hacia la Base de Datos sin necesidad de agregar capas intermedias de conexión que dificulten el desarrollo del subsistema, para ello como primer paso se creó un proyecto con el gestor de paquetes NPM que creará los archivos iniciales de configuración del mismo.

El Código 2.1 ilustra la creación del proyecto con NPM.

```
1. npm install -g sequelize
2. npm install sequelize-cli --global
3. npx sequelize-cli init
```

Código 2.1. Creación del proyecto con NPM.

Como resultado de la creación del proyecto de Sequelize se generará un directorio con la estructura base del proyecto que será detallada en la Figura 2.11.

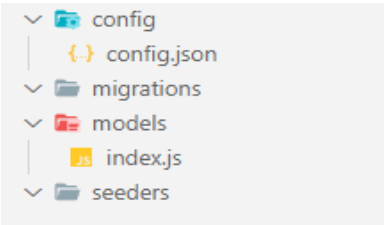


Figura 2.11. Estructura base del proyecto Sequelize.

El proyecto está conformado por las carpetas config, migrations, models y seeders, que son los encargados de guardar los códigos correspondientes a cada una de las funcionalidades a desarrollarse, siendo el principal config.json, ubicado en la carpeta config, su contenido es el encargado de especificar la base de datos a utilizar, así como también sus credenciales, en el Código 2.2 del anexo D se detalla el contenido del mismo.

Una vez configurada la comunicación con la Base de Datos se procede a crear las entidades que conforman al sistema, en el caso de Sequelize estas pasarán a llamarse modelos, para ello se utilizará los comandos detallados en el Código 2.3.

```
1. npx sequelize-cli db:drop
2. npx sequelize-cli db:create
3. npx sequelize-cli model:generate --name Usuario
4. npx sequelize-cli model:generate --name Paciente
5. npx sequelize-cli model:generate --name Lectura
6. npx sequelize-cli model:generate --name Medicamento
7. npx sequelize-cli model:generate --name Configuracion
```

Código 2.3. Comandos necesarios para la creación de los modelos del sistema

Como resultado de los comandos, NPX creará una serie de archivos relacionados a las entidades del sistema, estos se almacenarán en las carpetas migrations y models, los cuales se detallan en la Figura 2.12.

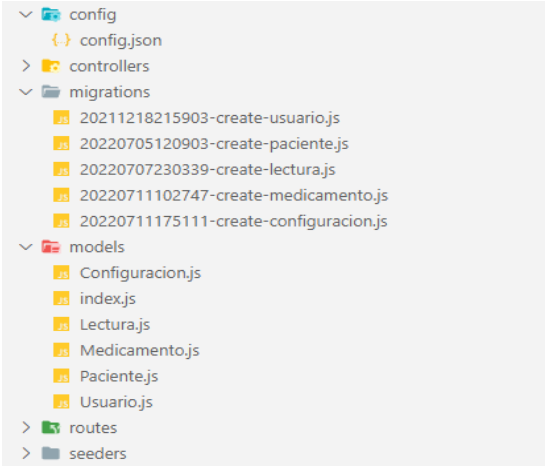


Figura 2.12. Resultado de la creación de modelos del sistema.

Como se puede observar en la Figura 2.12, a la creación de cada modelo le precede la creación de dos archivos, estos almacenados en las carpetas migrations y models siguiendo su nombre respectivo, en estos archivos se detallará la estructura de cada uno

de los modelos, a continuación, se mostrarán los códigos correspondientes al modelo Lectura.

En el Código 2.4. del anexo D se detalla el código correspondiente al archivo 20220707230339-create-lectura.js de la carpeta migrations, el cual es el encargado de realizar la creación de la tabla Lecturas en la Base de Datos MySQL.

En la Figura 2.13 se muestra el resultado de la ejecución del Código 2.4, mismo que tiene como resultado la creación de la tabla Lecturas en la Base de Datos.

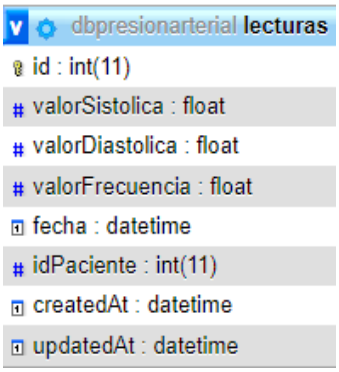


Figura 2.13. Tabla Lecturas creada en la Base de Datos dbPresionArterial.

En el Código 2.5 del anexo D se detalla el contenido del archivo Lectura.js de la carpeta models, el cual se encarga de interactuar con la tabla Lecturas creada previamente.

Al igual que los archivos para el modelo Lectura el resto de los archivos del proyecto se encargarán de la generación de los demás modelos del sistema, los cuales se adjuntarán en el Anexo D.

2.6.3 CODIFICACIÓN DE SERVICIO DE MANEJO DE DATOS

Para la implementación del Servicio de Manejo de Datos se utilizó el framework Express JS estudiado en el Capítulo 1, Sección 1.4.4. para el desarrollo de una API REST que se encargará de atender las solicitudes del Componente de Visualización de Resultados por medio de mecanismos conocidos como rutas de acceso, mismos que utilizan el protocolo HTTP para establecer una interfaz común que permita el acceso a los datos del sistema.

El principal componente del Servicio de Manejo de Datos es el framework Express que es el que provee la interacción con el resto de las herramientas de software y las peticiones de almacenamiento de datos, estas interacciones pueden visualizarse en la Figura 2.14.

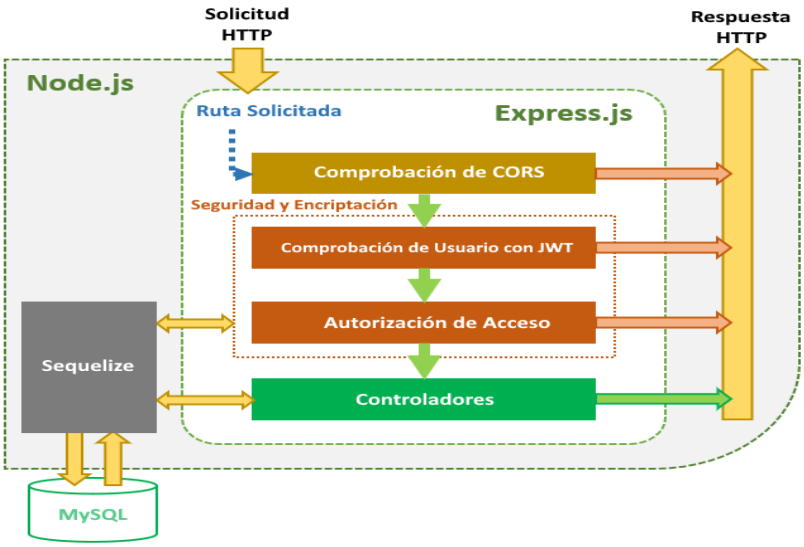


Figura 2.14. Esquema del Servicio de Manejo de Datos.

Este subsistema se codificará en conjunto con el de Base de Datos, por lo que se añadirá una serie de directorios y archivos al mismo, dando como resultado la estructura que se puede observar en la Figura 2.15.

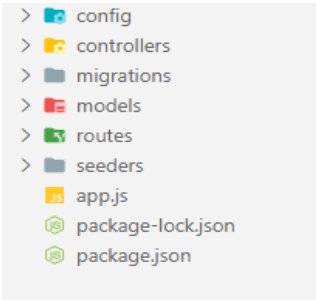


Figura 2.15. Estructura de directorios del Servicio de Manejo de Datos

Como se aprecia en la Figura 2.15 se ha añadido el archivo app.js, el cual será el archivo principal, en el código 2.6 del anexo D se puede observar el encabezado de este, el cual establece el punto de entrada al Servicio de API REST.

En la Figura 2.16 se puede apreciar de manera detallada el contenido de las carpetas routers y controllers, las cuales son las encargadas de almacenar los archivos que definen la interacción entre los modelos y la API REST, en el caso de los archivos de la carpeta routers será donde se establecerán los mecanismos de acceso a los datos por medio de verbos HTTP llamados rutas; por otro lado los archivos de la carpeta controllers son los encargados de dictar la interacción entre las rutas de acceso y el servicio de Base de Datos.

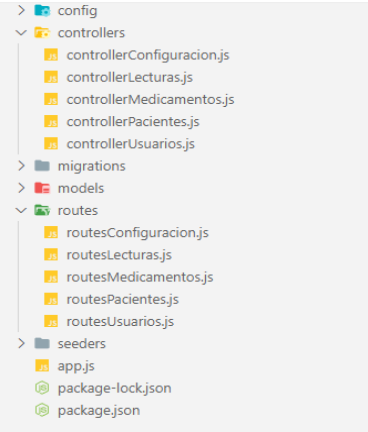


Figura 2.16. Estructura de directorios Routers y Controllers

A continuación, se detallan los contenidos de los archivos controllerLecturas.js y routesLecturas.js, mientras que el resto de los archivos de las carpetas Routers y Controllers se adjuntarán en el Anexo A.

En el Código 2.7 del anexo D se puede encontrar el contenido del archivo routesLecturas.js, el cual detalla las rutas correspondientes al modelo Lectura.

El archivo controllerLecturas.js se puede evidenciar en el Código 2.8 del anexo D, en este se definen las acciones a seguir con los datos consultados por medio de cada una de las rutas establecidas en el archivo routesLecturas.js.

Por su parte, el código 2.9 del anexo D muestra el método utilizado para la encriptación de los datos por parte del sistema, esto se ha logrado a través de la utilización de la librería bcryptjs, la cual se encargará de tomar los datos sensibles, como contraseñas y almacenarlos en forma de hash en lugar de texto plano.

El resultado de la utilización de bcryptjs se muestra en la Figura 2.17, donde se puede evidenciar el contenido de la tabla usuarios, en la cual, a pesar de conocerse los datos de nombre de usuario y correo, no se puede saber al completo la información del usuario, al encontrarse el campo “contrasenia” encriptado y con su función hash calculada.

id	nombre	correo	tipoUsuario	contrasenia
1	Administrador	admin@epn.edu.ec	1	$2a$10$0hBCNJvyQllEhLHlLu1.qeAVLLrw1CXx6GLpqm8uH6W...
2	Usuario1	usuario1@epn.edu.ec	2	$2a$10$Li0ZnHedkxOsYLpqhGr.geJEeVNMd5f/IcUsKB8Wt2k...
3	Usuario2	usuario2@epn.edu.ec	2	$2a$10$G/kPH3pFlqLkfW2mDgWq8uIJ2QaD/61Id/SVhjz6XII...
5	Usuario4	usuario4@epn.edu.ec	2	$2a$10$uMH2J2XmYQ85UOC.9WsaquJ76pG1p.LBxB0twshokYO...

Figura 2.17. Contenido de la tabla usuarios.

2.6.4 CODIFICACIÓN DEL SERVICIO DE PROCESAMIENTO DE DATOS

Para el procesamiento de datos se ha creado un proyecto en el lenguaje de programación Python, con ayuda del Framework Flask, el cual establece una interfaz de procesamiento para los datos, mismo que expone una URL que será la cual se entregará al tensiómetro digital, posteriormente este devolverá los datos de presión arterial y estos pasarán a través de una etapa de filtrado que devolverá un valor de presión arterial que será presentado en el Componente de Visualización de Resultados.

En la Figura 2.18 se puede visualizar la estructura del Servicio de Procesamiento de datos, la forma de recibir los datos de entrada por medio de solicitudes HTTP y la respuesta enviada hacia el Componente de Visualización de Resultados.

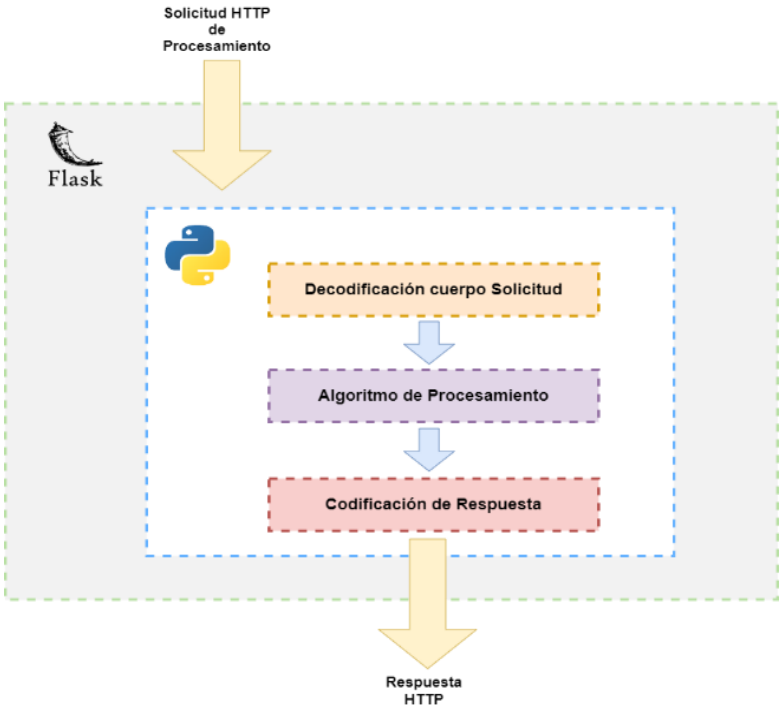


Figura 2.18. Diagrama Servicio de Procesamiento de Datos.

En el Código 2.10 del anexo D se detalla el código de Python que hace uso del framework Flask, desarrollado para realizar el procesamiento de información y su posterior devolución hacia el Componente de Visualización de Resultados.

2.6.5 CODIFICACIÓN DE ALGORITMO DE PROCESAMIENTO

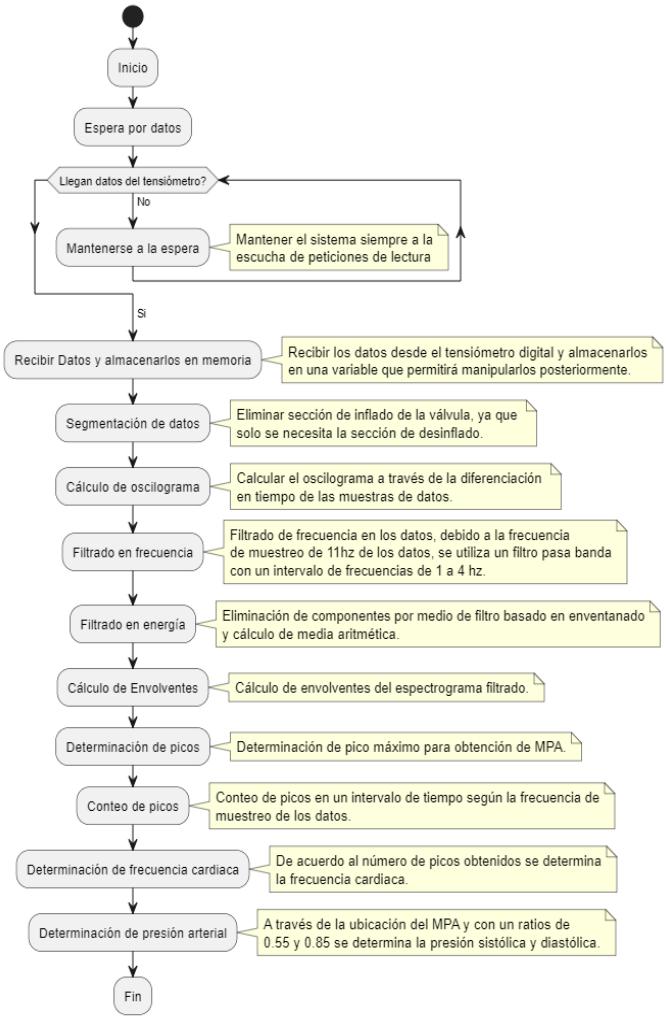


Figura 2.19. Algoritmo de procesamiento de datos desde el tensiómetro digital.

Las lecturas provenientes del tensiómetro digital son tomadas por un transductor que determina la presión en el brazalete según la etapa de inflado y desinflado, sin embargo, estas carecen de un procesamiento adecuado que determine un valor nominal de presión arterial y frecuencia cardiaca, es por ello que surge la necesidad de implementar un algoritmo que se encargue de este cometido.

Este algoritmo se compone de distintas etapas las cuales son segmentación de datos, cálculo de espectrograma, filtrado en frecuencia, filtrado en energía, cálculo de envolventes, determinación de picos, conteo de picos, determinación de frecuencia cardiaca y determinación de presión arterial. A continuación la Figura 2.19 detalla el algoritmo utilizado y sus etapas.

A continuación, se presentará la codificación correspondiente a cada una de las etapas del algoritmo de procesamiento.

El Código 2.10 del anexo D indica la sección correspondiente a la creación de un método de escucha del sistema para recibir los datos de lectura, esto se logra a través de un método HTTP que estará disponible a cualquier petición del tensiómetro digital y realizará la recepción y almacenamiento de datos para su posterior utilización.

Cuando el tensiómetro digital envíe sus datos de lectura de presión, el sistema ejecutará este método y almacenará los datos en una variable, la Figura 2.20 muestra un ejemplo de la recepción de datos de presión y su posterior almacenamiento en la variable dataTens.

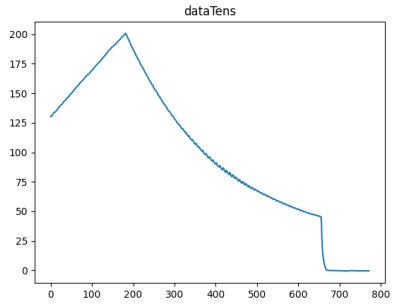


Figura 2.20. Datos provenientes del tensiómetro digital.

Una vez recibidos los datos se procederá con el resto de las etapas del algoritmo, a continuación, el código 2.12 nos presenta la sintaxis necesaria para la realización de la etapa de segmentación de datos.

```
1. posMax=np.argmax(dataTens)
2. dataNoInf=dataTens[math.floor(posMax*1.2):len(dataTens)]
3. finalData=dataNoInf[0:math.floor(len(dataNoInf)*0.75)]
```

Código 2.12. Código necesario para la segmentación de datos de presión.

El resultado del Código 2.12 se muestra en la Figura 2.21, allí se puede evidenciar que se ha segmentado los datos para obtener únicamente la sección de desinflado del brazalete, misma que servirá posteriormente para la determinación del oscilograma.

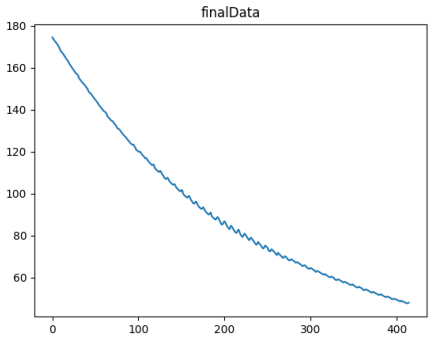


Figura 2.21. Resultado de segmentación de datos, sección de desinflado de brazalete.

Como paso posterior a la segmentación se encuentra el cálculo del oscilograma , esto será detallado en el Código 2.13 del anexo D, el oscilograma se lo calcula restando dos muestras, el valor final con el inicial, esto se realiza para la cantidad de muestras que se tengan. [34]

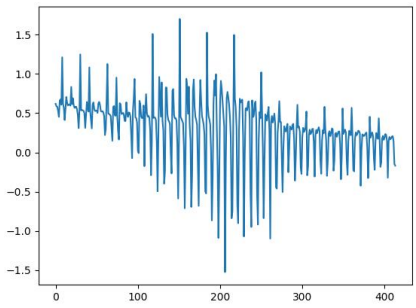


Figura 2.22. Oscilograma de los datos de presión arterial.

Una vez que se obtiene el oscilograma se procede a realizar un filtrado de la señal con el fin de eliminar datos que sean producidos por el ruido o sean producto de un movimiento brusco cuando se realizó la medida. El filtro utilizado es un filtro Butterworth pasa banda de octavo orden con frecuencia de corte de 1 y 4 Hz. Estas frecuencias fueron seleccionadas ya que la frecuencia cardiaca se encuentra en ese rango, además, con el filtrado se puede ver como mejora el oscilograma que es necesario para la obtención de la presión arterial, esto se puede evidenciar en la Figura 2.23 y la sintaxis se aclara en el Código 2.13 del anexo D. [34]

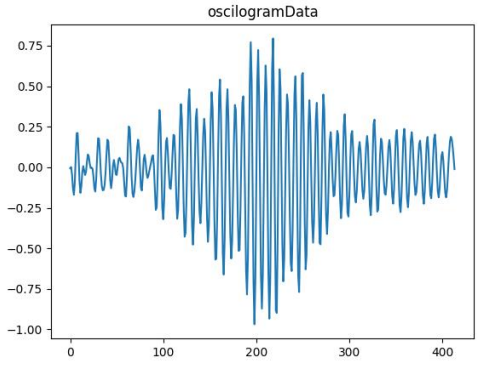


Figura 2.23. Oscilograma de presión arterial.

Una vez obtenido el oscilograma de presión arterial se realizará la obtención del MPA (presión media máxima) para lo cual primero se necesita obtener la envolvente de la señal. La envolvente se la calcula mediante un código el cual está formado por un bucle iterativo. El resultado se muestra en la Figura 2.24. Luego se aplica un filtro de media el cual es para reducir los picos bruscos. Finalmente, para calcular el MPA se mide el valor del segmento más grande, esto se detalla en el código 2.14 del anexo D. [37]

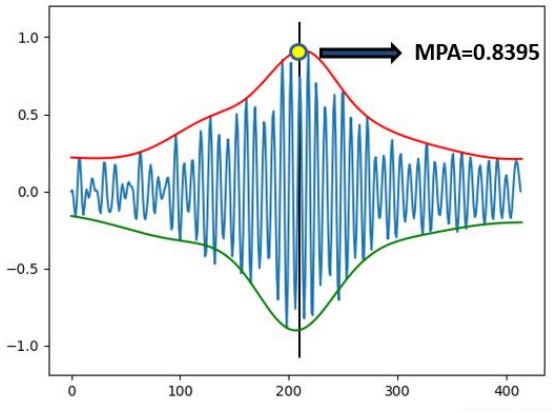


Figura 2.24. Oscilograma de presión arterial aplicando filtros y obtención del MPA.

Una vez que se ha fijado el valor del MPA inmediatamente se aplica el criterio de las alturas que se mencionó en la sección 1.4.7 para obtener el valor de la presión sistólica y diastólica. Con este criterio se aplica la ecuación (1) y (1.2) para hallar los valores de 'Hd' y 'Hs' para luego proceder a ubicar esos puntos en el eje de las abscisas de las muestras de la Figura 2.20, luego ya ubicado el valor se procede a ir al eje de las ordenadas como se observa en la Figura 2.25, resultando el valor real de la presión sistólica y diastólica, en el código 2.16 del anexo D se muestra este procedimiento para la obtención de dichos valores.

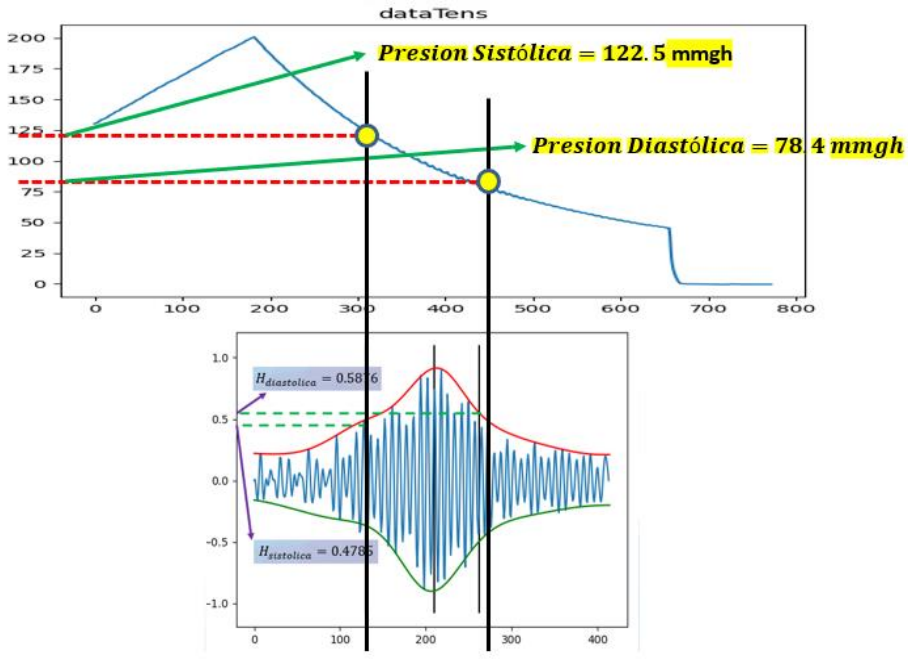


Figura 2.25. Obtención de presión sistólica y diastólica por método de las alturas.

Una vez realizada la estimación de la presión arterial se procede a realizar la estimación de la frecuencia cardiaca para la cual se hace una aproximación en función de conteo de picos, esta acción ayudará a determinar la frecuencia cardiaca con una relación de tiempo,

misma que será almacenada para su posterior transmisión, esto será reflejado en el Código 2.15. En la Figura 2.26, misma que indica los picos encontrados en el oscilograma, sin embargo, se puede evidenciar que no todos los picos son tomados en cuenta, ya que esto se debe a la naturaleza de la medición, la cual establece que solo los picos distanciados a una distancia de como mínimo 6 muestras sean establecidos como picos válidos [37].

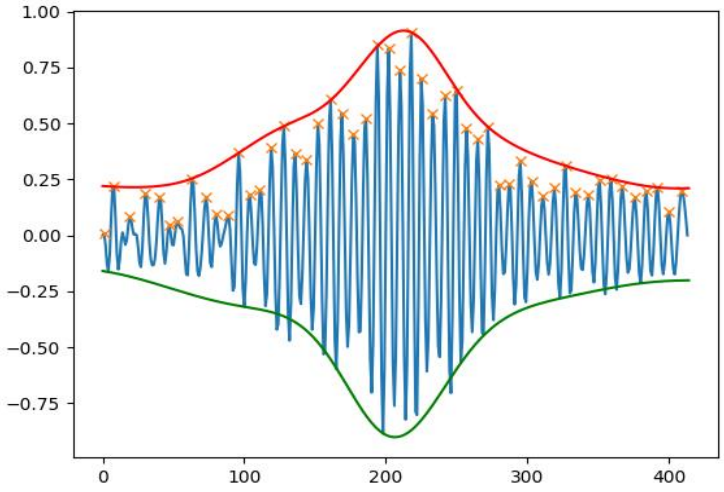


Figura 2.26. Determinación de picos del oscilograma.

Como etapa final del procesamiento, el Código 2.17 del anexo D muestra la sintaxis necesaria para la transmisión de los valores de presión arterial y frecuencia cardiaca, mismos que serán retornados al Componente de Visualización de Resultados.

2.6.6 CODIFICACIÓN DEL COMPONENTE DE VISUALIZACIÓN DE RESULTADOS

Los datos enviados por el tensiómetro digital son procesados por el Componente de Procesamiento y Manejo de Información, sin embargo, este no tiene la capacidad de comunicarse con los usuarios finales, por lo tanto no es posible una visualización de los resultados de medición de presión arterial, es por ello que surge la necesidad de crear una interfaz gráfica intuitiva que permita a los usuarios una visualización clara de los resultados, con el fin de realizar el seguimiento ambulatorio de presión arterial, esta interfaz tomará el nombre del Componente de Visualización de Resultados.

El presente componente ha sido desarrollado utilizando el framework VueJS, el cual establece define una serie de reglas a seguir para el desarrollo de una interfaz gráfica, esto por medio de un proyecto de tipo Vue, el cual será creado utilizando el gestor de paquetes NPM, el código 2.18 ilustra la creación de un proyecto Vue.

```
1. vue create dashboard
2. npm install
3. npx run serve
```

Código 2.18. Creación del proyecto Vue con NPM.

El resultado de la creación de un proyecto Vue por medio del gestor de paquetes NPM es una estructura de directorios que cuenta con archivos de código fuente y configuración necesarios para la ejecución del proyecto, en la Figura 2. 27 se muestra la estructura base del proyecto.

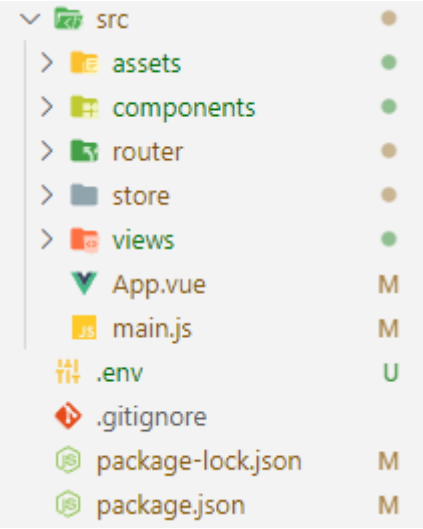


Figura 2.27. Estructura base del proyecto Vue.

El proyecto Vue cuenta con una serie de carpetas que son las encargadas de almacenar los archivos de código correspondientes a cada una de las funcionalidades del sistema, a continuación, se detallara sus funciones.

- **Assets:** Se encarga de almacenar los archivos correspondientes a recursos estáticos que serán utilizados por el resto de los módulos del proyecto.
- **Components:** Almacena los archivos de componentes, mismos que son fragmentos de código permiten realizar funciones específicas pero que pueden ser invocados desde cualquier otro archivo del proyecto, por lo que cumplen la función de componentes reutilizables.
- **Router:** Se encarga de almacenar la información de las rutas del sistema, mismas que permitirán definir los métodos de acceso a los diferentes módulos que componen la interfaz.

- **Store:** Se encarga de definir métodos de control de almacenamiento de entidades de la interfaz gráfica.
- **Views:** Se define como la carpeta principal donde se especifican archivos para cada una de las ventanas que componen el proyecto.

Además de las carpetas que almacenan código fuente se encuentran archivos de configuración como son App.vue, main.js, .env, .gitnignore y package.json, los cuales se encargan de establecer la configuración interna del proyecto.

2.6.7 CODIFICACIÓN DE URLS DEL COMPONENTE

El Componente de Visualización de resultados será accedido a través de un navegador web, por lo cual se necesita definir una serie de URLs que darán acceso a cada una de las ventanas de la interfaz, este cometido se llevará a cabo mediante el archivo index.js, ubicado en la carpeta Router, en el código 2.19 se detalla parte del contenido del archivo index.js, esta sección corresponde a la configuración del elemento routers, la cual establece un objeto encargado de las URLs del proyecto.

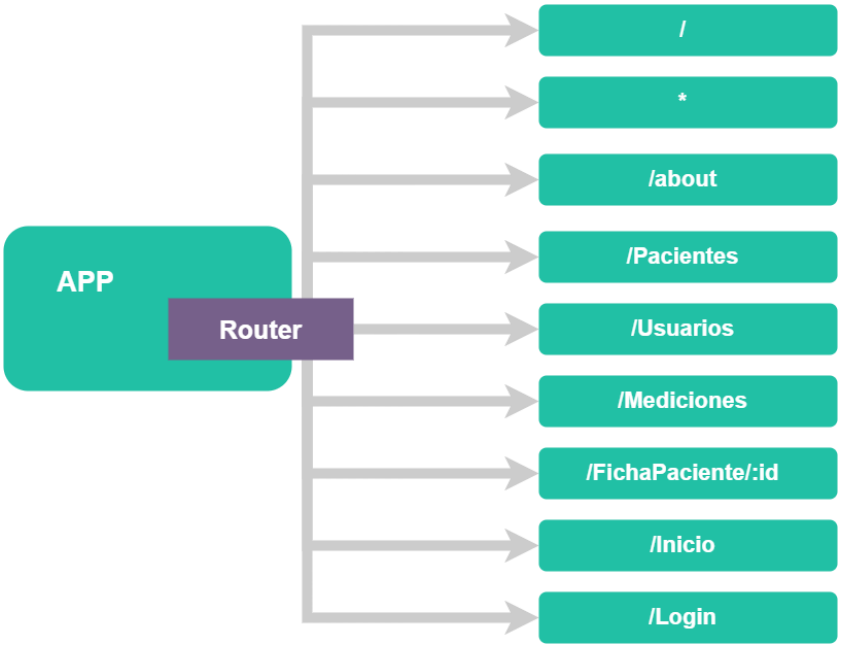


Figura 2.28. Árbol de Rutas del componente router.

La Figura 2.28 da como resultado lo que se conoce como árbol de rutas del componente router del proyecto Vue.

```
1. import Vue from 'vue'
2. import VueRouter from 'vue-router'
3. import Inicio from '../views/Inicio.vue'
4. Vue.use(VueRouter)
```

Código 2.19. Inicialización del componente router.

El componente router inicializado en el Código 2.19 será el encargado de recibir las peticiones de acceso por parte de los usuarios y creará un array de rutas, las cuales especificarán cada una de las URLs correspondientes a las ventanas del proyecto, en los códigos 2.20 y 2.21 del anexo D se detallan cada una de las rutas, las cuales serán descritas a continuación.

Las rutas establecidas en los Códigos 2.20 y 2.21 corresponden a los métodos de acceso correspondientes a las ventanas del componente, las cuales se detallan a continuación:

- **Ruta /:** Se conoce como la ruta raíz del proyecto, y es la encargada de establecer el punto de entrada a la ventana de inicio por parte del navegador.
- **Ruta *:** Se establece como la ruta de redirección del proyecto, esta se encarga de redirigir las peticiones de URLs no existentes hacia la ruta /, es decir si no existe la URL consultada el sistema dirigirá al usuario al inicio del proyecto.
- **Ruta /about:** Define el método de acceso a la ventana de información del proyecto, en esta se visualizará la información acerca del desarrollador e información adicional del funcionamiento del Componente de Visualización de Resultados.
- **Ruta /Pacientes:** Detalla el método de acceso por el cual se podrá visualizar la ventana que despliega la información de los pacientes registrados en el Componente de Visualización de Resultados.
- **Ruta /Usuarios:** Realiza la importación de la vista de la ventana Usuarios, la cual es la encargada del registro y gestión de usuarios autorizados para el uso del Componente de Visualización de Resultados, en esta ventana se detallará si el usuario está a modificar información de los pacientes o si solo puede realizar mediciones de presión arterial, esto se logra mediante la asignación de roles de usuario por parte del campo TipoUsuario.
- **Ruta /Mediciones:** Establece el mecanismo de acceso a la ventana de mediciones, esta será la cual interactúe con el tensiómetro digital, recibiendo las mediciones de

presión arterial y enviándoselas al Componente de Manejo y Procesamiento de Datos.

- **Ruta /FichaPaciente/id:** Crea la URL de acceso a la ventana FichaPaciente, tomando como parámetro el id del paciente del cual se desea conocer la información.
- **Ruta /Inicio:** Establece la URL que se enlazará a la ventana de inicio del Componente de Visualización de Resultados, en esta se desplegará información importante acerca del funcionamiento actual del componente.
- **Ruta /Login:** Establece la URL correspondiente el formulario de inicio de sesión del componente, mismo que sirve para proveer de seguridad al sistema y que solo usuarios registrados en el mismo sean capaces de visualizar la información almacenada en la base de datos.

Como pie del archivo index.js del router del Componente de Visualización de Resultados, en el Código 2.22 se establece el mecanismo de historiador del proyecto Vue, mismo que se encarga de definir si se provee al usuario la capacidad de volver a las ventanas anteriormente consultadas.

```
1. const router = new VueRouter({
2.   mode: 'history',
3.   base: process.env.BASE_URL,
4.   routes
5. })
6.
7. export default router
```

Código 2.22. Configuración del router del proyecto en modo historiador.

2.6.8 CODIFICACIÓN DE VISTAS DEL COMPONENTE

Una vez realizado el establecimiento de los métodos de acceso a las ventanas del Componente de Visualización de resultados, estas deben codificarse para su funcionamiento, a continuación, se detallará el proceso de codificación de las mismas.

La ventana principal, ubicada en la ruta / se detalla en la Figura 2.29, en esta se mostrará información relevante al estado actual del sistema, teniendo en cuenta el número de usuarios y pacientes registrados, así como también el número de mediciones recibidas desde el tensiómetro digital.

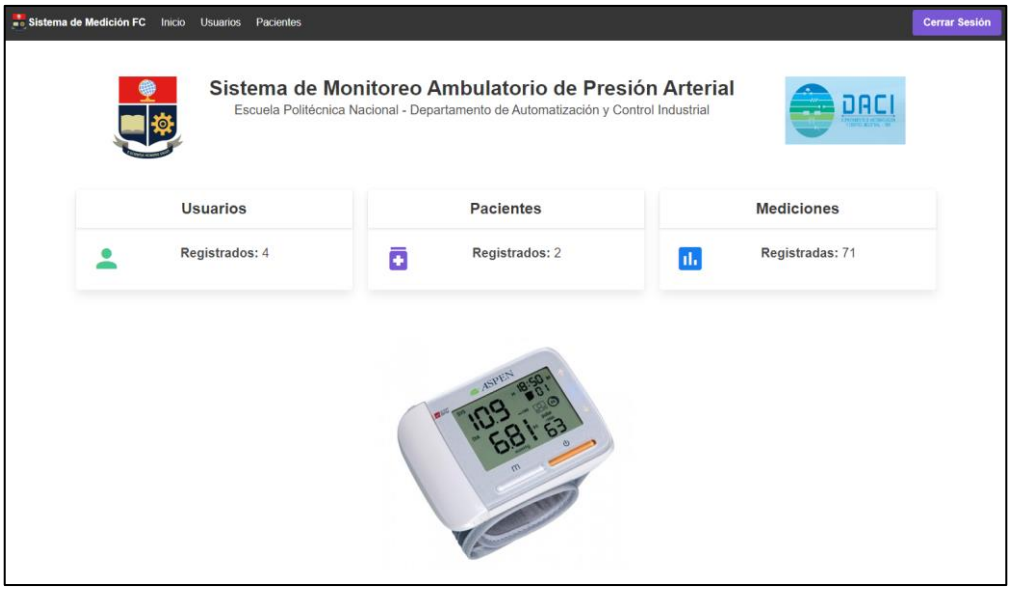


Figura 2.29. Ventana principal del Componente de Visualización de Resultados

La ventana principal, al igual que las demás que componen el sistema, se codificó con ayuda del framework VueJS, el código fuente correspondiente a estas se encuentra estructurado en archivos de tipo vue, los cuales cuentan con una estructura como la mostrada en la Figura 2.30.

```
<template>
....
</template>

<script>
....
</script>

<style>
....
</style>
```

Figura 2.30. Estructura de código fuente de un archivo *.vue.

El objetivo de dividir el código fuente de un archivo de tipo vue es abstraer en secciones específicas el comportamiento de una ventana para obtener un código fuente más modular y ordenado, a continuación, se detalla la función que cumple cada una de estas secciones:

- La sección template del código fuente se escribe con ayuda del lenguaje de etiquetado HTML, y es la encargada de definir la estructura visual de la ventana, lo cual incluye etiquetas de texto, imágenes, y controles como botones o áreas de texto.

- La sección script se codifica por medio del lenguaje de programación JavaScript, y es la encargada de manejar las interacciones que el usuario realiza con los elementos visuales de la ventana, así como también de realizar el procesamiento de fondo correspondiente a desplegar información al usuario en la sección template.
- La sección style es codificada en el lenguaje CSS, y se encarga de definir los colores y efectos gráficos a aplicarse a los elementos gráficos de la interfaz definida en la sección template.

A continuación, se muestra a grandes rasgos el código fuente correspondiente al archivo Inicio.vue, correspondiente a la ventana principal, el resto de las ventanas contará con una estructura de código similar y serán incluidos en el Anexo B.

En el Código 2.23 del anexo D se muestra el contenido de la sección template del archivo Inicio.vue, en este se incluyen etiquetas HTML correspondientes a definir la estructura visual de la ventana, en estas se puede evidenciar un contenedor principal el cual incluye el título e imágenes vistas en la Figura 2.29.

En la sección script del archivo Inicio.vue, se encuentran contenidas las funciones encargadas de interactuar con el Componente de Visualización de Resultados y cargar desde la base de datos el número de usuarios, pacientes y mediciones registradas y mostrarlas en los contenedores definidos en la sección template, esta sección se subdivide a su vez en cuatro sub-secciones, siendo estas Data, Computed, Components y Methods, además de añadirse sub-secciones especiales conocidas como lifecycle hooks, siendo mounted() la más común de todas, estas sub-secciones se presentarán a continuación, así como las funciones que cumplen dentro del script.

El Código 2.24 detalla la sub-sección components, la cual es la encargada de establecer los componentes a utilizarse por parte de la ventana, siendo el SecurityComponent el más importante, ya que es el encargado de gestionar la seguridad del sistema, este determinará si el usuario está autorizado a visualizar la información provista por el Componente de Manejo y Procesamiento de Datos.

```
import LineChart from '@/components/Line.vue'
import SecurityComponent from "@/components/SecurityComponent.vue";
export default
{
  components:{
    LineChart,
    SecurityComponent
  },
}
```

Código 2.24. Sub-sección components del archivo Inicio.vue

El Código 2.25 detalla la sección data, la cual se encarga de inicializar las variables utilizadas para realizar la carga del número de usuarios, pacientes y mediciones.

```
data(){
    return{
      numUsuarios:0,
      numPacientes:0,
      numMediciones:0
    }
  }
```

Código 2.25. Sub-sección data del archivo Inicio.vue

La sintaxis correspondiente al Código 2.26 describe el código necesario para la sub-sección Computed, misma que se encarga de establecer variables globales a ser utilizadas por el sistema, en este caso se cuenta únicamente con la variable isNavBarVisible, la cual se encarga de determinar si la barra de navegación es visible o no.

```
computed: {
    isNavBarVisible() {
      return this.$store.state.isNavBarVisible;
    },
  }
```

Código 2.26. Subsección computed del archivo Inicio.vue

La sub-sección Methods es la más importante de la sección script, ya que en esta se encuentran detallados los métodos para la carga de información desde el Componente de Manejo y Procesamiento de datos, esta se ve reflejada en el Código 2.27 del anexo D.

La sub-sección methods detalla los métodos fetchUsuarios, fetchPacientes y fetchLecturas, los cuales son los encargados de consultar al Componente de Manejo y

Procesamiento de Datos el número de registros existentes y desplegarlos en los campos de texto correspondientes detallados en la sección template.

Por último, el Código 2.28 detalla los life cycle hooks de la ventana, los cuales se encargan de ejecutar los métodos fetchUsuarios, fetchPacientes y fetchLecturas una vez se haya cumplido un ciclo de vida de la ventana, siendo mounted el cual desencadenará esta ejecución.

```
1. mounted(){
2.     this.fetchUsuarios()
3.     this.fetchPacientes()
4.     this.fetchLecturas()
5.   }
```

Código 2.28. Life Cycle mounted del archivo Inicio,vue.

2.6.9 CODIFICACIÓN DE ALERTAS DEL COMPONENTE

El Componente de Visualización de resultados trabaja en conjunto con el tensiómetro digital, sin embargo, las ventanas desarrolladas en VueJS no cuentan con un sistema de alertas que permitan notificar la llegada de lecturas de presión arterial en tiempo real, es por ello que se ha desarrollado una serie de métodos adicionales a estas con la ayuda de la librería Socket.io.

Los códigos detallados a continuación se agregarán al proyecto vue y servirán para agregar un sistema de notificaciones al Componente de Visualización de Resultados.

El Código 2.29 indica la sintaxis necesaria para la instalación de la librería dentro del proyecto vue por medio del gesto de paquetes npm.

```
1. npm install socket-io
2. npm install socket-io-client
3. npm install vue-socketio
4. npm install vue-socketio-client
```

Código 2.29. Instalación de socket.io.

La librería socket.io debe incluirse tanto en el Componente de Manejo y Procesamiento de Datos, como en el Componente de visualización de resultados, a continuación, el Código 2.29 indica la inclusión de la librería Socket.io al Componente de Manejo y Procesamiento de Datos, mismo que recibirá un dato de lectura desde el tensiómetro digital, realizará su

procesamiento y lo enviará como un evento de tipo emit hacia el Componente de Visualización de Resultados.

Una vez incluida la librería Socket.io, el Componente de Manejo y Procesamiento de Datos será capaz de detectar en tiempo real la conexión y desconexión de clientes, disponiendo así de la capacidad de enviar notificaciones al Componente de Visualización de Resultado cuando se haya almacenado una lectura de presión arterial por parte del tensiómetro digital.

En la Figura 2.31 se puede evidenciar la conexión y desconexión de un cliente al Componente de Manejo y Procesamiento de Datos.

```
> nodemon app.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Server UP running at: localhost:3000
a user connected
user disconnected
```

Figura 2.31. Conexión y desconexión de cliente al Componente de Manejo y Procesamiento de Datos.

Para la inclusión de la librería Socket.io en el Componente de Visualización de Resultados se debe primero importarla en el archivo JavaScript principal del proyecto vue, esto se debe realizar en el encabezado del archivo main.js, ubicado en la raíz del proyecto. El Código 2.31 del anexo D demuestra la importación de Socket.io.

Una vez incluida la librería al proyecto, es posible realizar la conexión del Componente de Visualización de Resultados hacia el Componente de Manejo y Procesamiento de datos, esto se logrará a través de la inclusión de una sub-sección llamada *sockets* al archivo de la ventana principal, el Código 2.32 del anexo D muestra la sintaxis necesaria para la creación de la sub-sección sockets en el archivo Inicio.vue.

Como se puede ver en el Código 2.32, se han creado tres métodos dentro de sockets, siendo estos, connection, disconnect y lecturaOk, y serán los encargados de manejar la conexión en tiempo real, a continuación, se describe el funcionamiento de cada uno de estos métodos.

- El Método connection es el encargado de establecer la conexión en tiempo real con el Componente de Manejo y Procesamiento de Datos, y será este quien determine si ha sido exitosa, o en su defecto la reintentará hasta que lo sea.
- El método disconnect es el encargado de cerrar la conexión en tiempo real, y su cometido es liberar los recursos de memoria y CPU destinados a mantener la conexión, lo que permite optimizarla.
- El método lecturaOk es el encargado de recibir una notificación de lectura correcta desde el tensiómetro digital, así como también de notificar al usuario los resultados del procesamiento realizado en el Componente de Manejo y Procesamiento de Datos.

Una vez codificada la conexión en tiempo real entre el Componente de Visualización de Resultados y el Componente de Manejo y Procesamiento de Datos, al abrirse la ventana principal, en las opciones de depuración se puede evidenciar la conexión por medio de Socket.io. En la Figura 2.32 se evidencia la ejecución del método connection, el cual tiene como resultado el establecimiento de esta conexión.

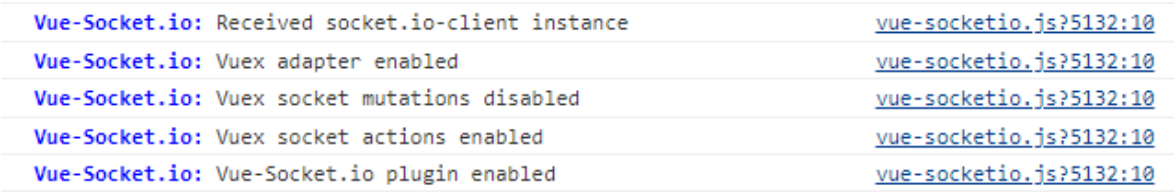


Figura 2.32. Conexión en tiempo real por medio de Socket.io.

Si la ventana del navegador se cierra o el Componente de Manejo y Procesamiento de Datos decide finalizar la conexión en tiempo real se ejecutará el método disconnect, cuyo resultado se muestra en la Figura 2.33.

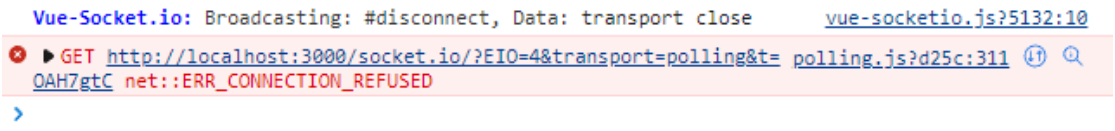


Figura 2.33. Cierre de la conexión por medio del método disconnect.

La ejecución de los métodos connection y disconnect son inherentes a la apertura o cierre de la ventana y no están en control del usuario, por lo que se ejecutarán una sola vez cada que se abra una ventana del Componente de Visualización de Resultados, sin embargo, el método lecturaOk tiene como objetivo ser desencadenado cada vez que el tensiómetro digital realice una lectura de presión arterial exitosa, este recibirá como parámetros el

resultado de procesamiento de los datos enviados desde el tensiómetro y procesados por parte del Componente de Manejo y Procesamiento de Datos.

En la Figura 2.34. se puede evidenciar el resultado de la ejecución del método lecturaOk, así como los resultados que arroja a la ventana de depuración del sistema.

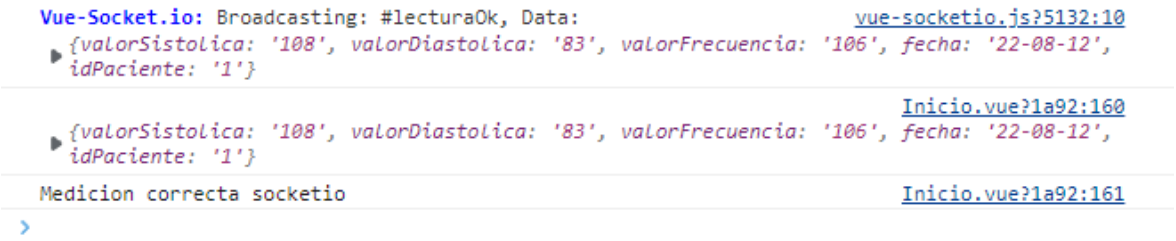


Figura 2.34. Ejecución del método lecturaOk.

Una vez ejecutado el método lecturaOk, este se encargará de la creación de una notificación en las ventanas del Componente de Visualización de Resultados, a continuación, la Figura 2.34 muestra un evento de notificación desencadenado por la lectura de presión arterial desde el tensiómetro Digital.

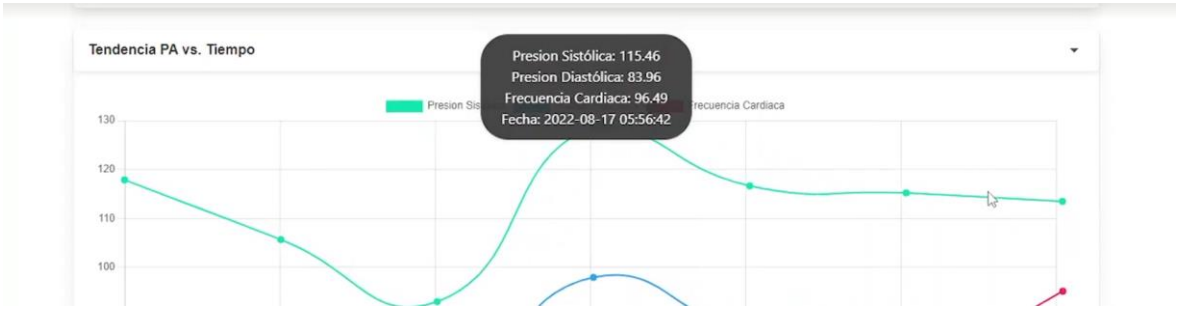


Figura 2.34. Notificación de lectura de presión arterial.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

3.1.1 PROTOTIPO DEL GATEWAY DEL SISTEMA

En esta sección se procede a mostrar el resultado de la implementación del dispositivo que servirá como Gateway del Sistema, las partes que lo constituyen, así como el diseño final de su case impreso en 3d.

La Figura 3.1 muestra en conjunto el diseño de hardware del Gateway del sistema, el cual está constituido por la tarjeta de desarrollo Raspberry Pi Model 3B, su fuente de

alimentación y el enrutador inalámbrico encargado de crear la red inalámbrica que comunicará al Gateway con el tensiómetro digital, además se puede apreciar que el Gateway ya se encuentra encapsulado en su case impreso en 3D.

Figura 3.1. Resultado de diseño de hardware del sistema.

3.1.2. COMPONENTE DE MANEJO Y PROCESAMIENTO DE DATOS

En esta sección se presentará los resultados de la implementación del Componente de Manejo y Procesamiento de datos, estos resultados serán devueltos en formato json, por lo cual para facilitar su visualización se ha utilizado la herramienta Imsomnia REST Client.

En la Figura 3.2 se puede visualizar una captura de pantalla de la herramienta a utilizarse, la cual establece una serie de controles que permiten la ejecución de peticiones hacia el Componente de Manejo y Procesamiento de Datos.

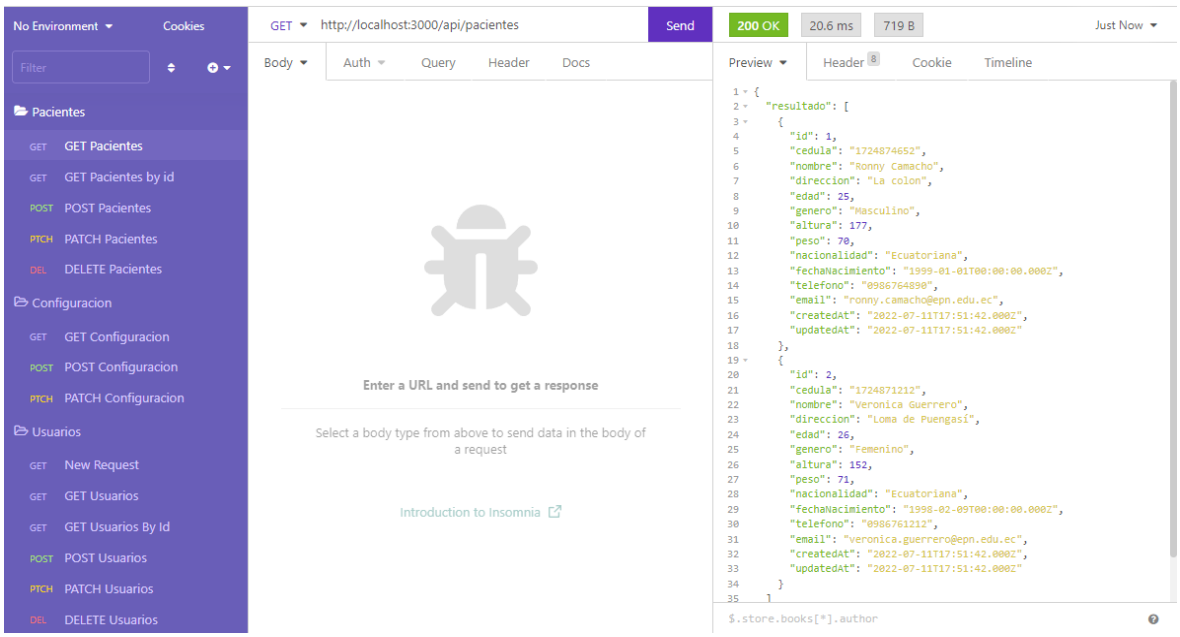


Figura 3.2. Herramienta Insomnia REST Client.

En la Figura 3.2 se puede evidenciar la interfaz gráfica de Insomnia, la cual cuenta con diversos controles que nos permiten la realización de consultas, visualización de resultados y almacenamiento de consultas previas.

A continuación, se procede a mostrar los resultados de las consultas relacionadas a la entidad Paciente, las cuales se componen por los métodos HTTP GET, POST, PATCH y DELETE.

La Figura 3.3 muestra el resultado de la consulta “GET Pacientes”, esta consulta es la encargada de recuperar la información de los pacientes del sistema almacenada en la base de datos MySQL.

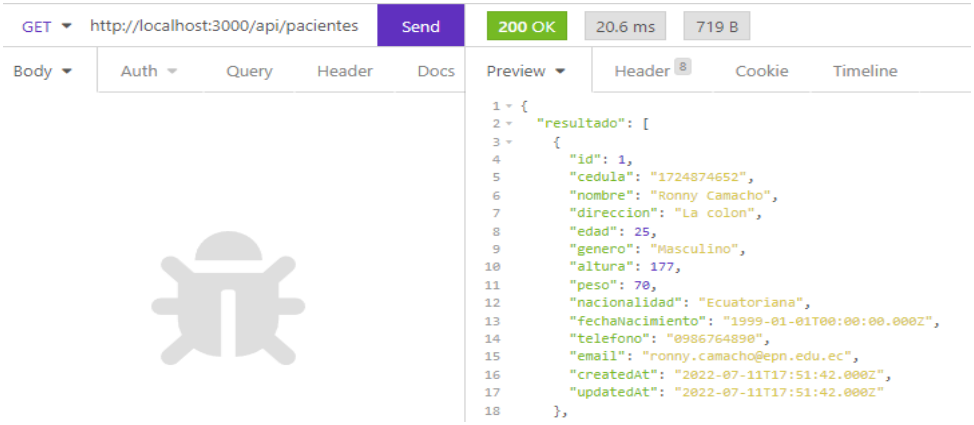


Figura 3.3. Consulta “GET Pacientes”.

La Figura 3.4 nos muestra el resultado de la consulta “GET Paciente by id”, esta se diferencia de la anterior que obtiene la información de todos los pacientes registrados en el sistema, mientras que la presente consulta obtiene la información de un solo paciente teniendo como parámetro su id, esto es útil en casos de despliegue de información de un solo paciente, por lo cual utilizar este método obtiene solo la información necesaria a desplegarse.

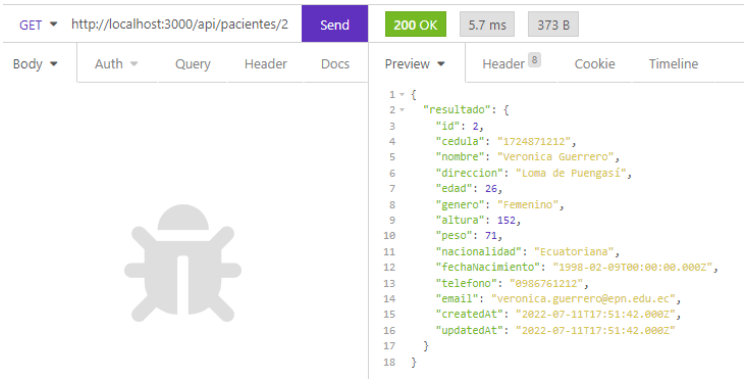


Figura 3.4. Consulta “Get Paciente by id”.

Por su parte, la Figura 3.5 muestra el resultado de la consulta “POST Paciente”, la cual tiene como objetivo el registro de un nuevo paciente en la base de datos, los parámetros se indicarán en la sección Form de la interfaz de Insomnia.

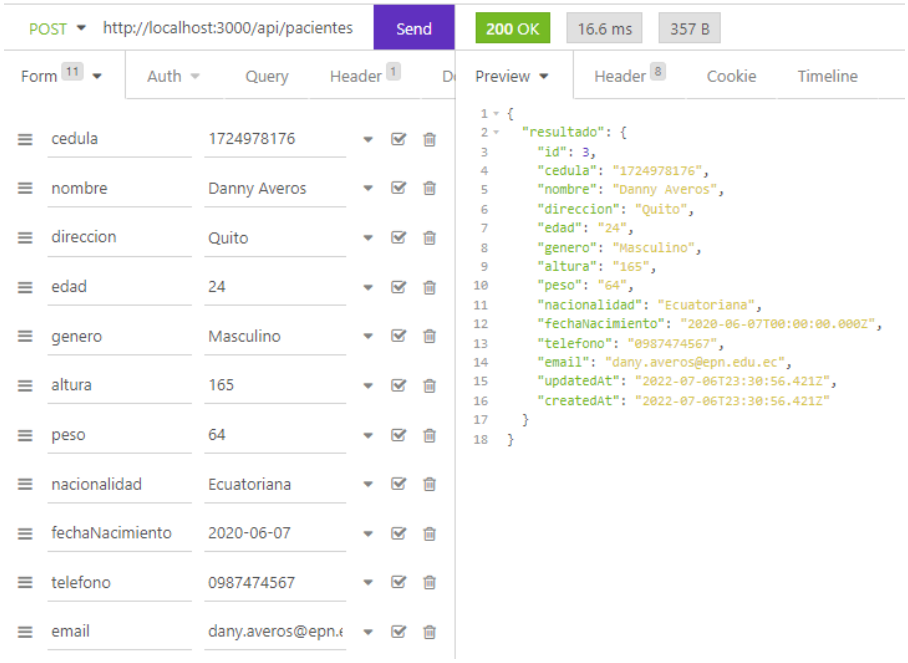


Figura 3.5. Consulta “POST Paciente”.

En la Figura 3.6 podemos evidenciar la ejecución de la consulta “PATCH Pacientes”, misma que será la encargada de editar la información registrada de un paciente existente en la base de datos, tomando como parámetro la id del mismo, y detallando en la sección Form la información que se desea editar.

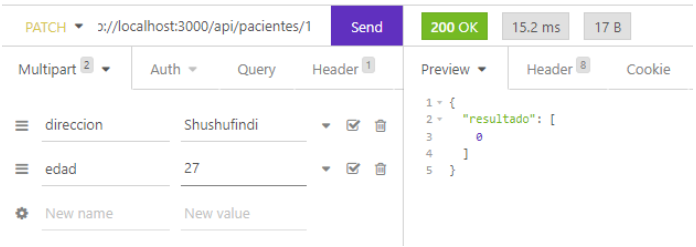


Figura 3.6. Consulta “PATCH Paciente”.

Como última consulta pertinente a la entidad Paciente tenemos “DELETE Paciente”, la cual es la encargada de eliminar toda la información registrada del paciente cuyo id se pasa como parámetro, la Figura 3.7 muestra el resultado de su ejecución.

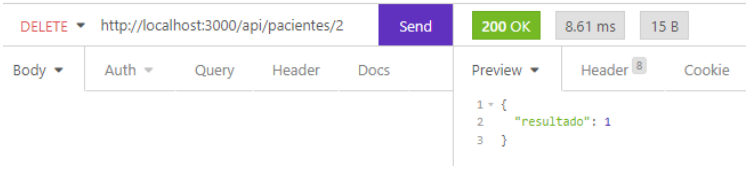


Figura 3.7. Consulta “DELETE Paciente”.

Al igual que la entidad Pacientes, el resto del Componente de Manejo y Procesamiento de datos cuenta con las consultas GET, POST, PATCH y DELETE correspondientes al resto de entidades que intervienen en este.

Además de las peticiones a las entidades se encuentra la consulta de procesamiento de datos enviados desde el tensiómetro digital, misma que tiene como resultado lo evidenciado en la Figura 3.8.

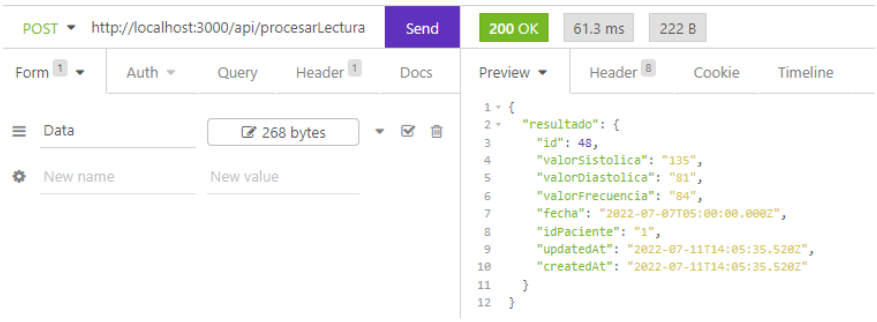


Figura 3.8. Resultado de procesamiento de lectura del tensiómetro digital.

3.1.3. COMPONENTE DE VISUALIZACIÓN DE RESULTADOS

La presente sección tiene como objetivo presentar los resultados de la implementación del Componente de Visualización de Resultados a través de capturas de pantalla de las ventanas codificadas en la implementación del presente componente.

La Figura 3.9 presenta el resultado de la implementación de la ventana de información pertinente a los Pacientes del sistema de forma tabulada, esta tabla cuenta en sus columnas la descripción de los atributos del paciente, además de una columna especial llamada Acciones, la cual define una serie de botones a utilizarse para acceder a la ventana de seguimiento ambulatorio de presión arterial y realizar mediciones de frecuencia cardiaca, así como también editar su información personal o eliminarlo.

Figura 3.9. Ventana de información de Pacientes.

En la ventana Pacientes también se incluye el botón “Crear”, mismo que será el encargado de desplegar un formulario que permite el registro de un nuevo paciente, este se muestra en la Figura 3.10.

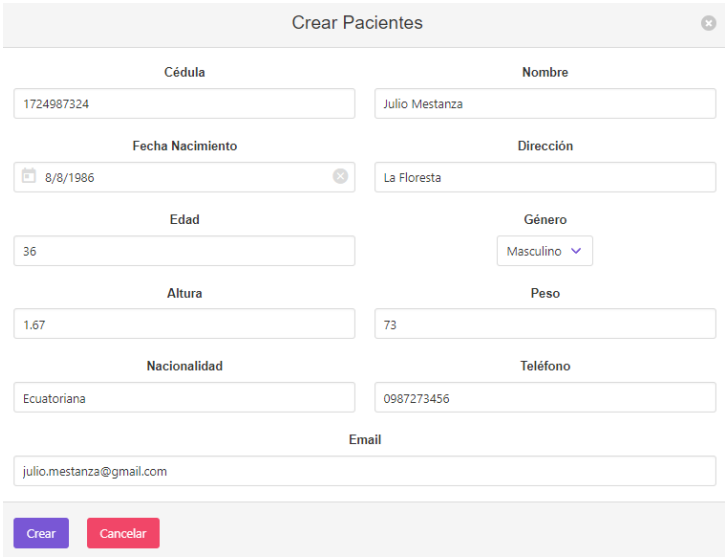


Figura 3.10. Formulario de Registro de nuevo Paciente.

El resultado de la ejecución del formulario de registro de nuevo paciente se puede evidenciar en la Figura 3.11, donde en la tabla pacientes de la ventana Pacientes se puede evidenciar un nuevo registro correspondiente al Paciente registrado.

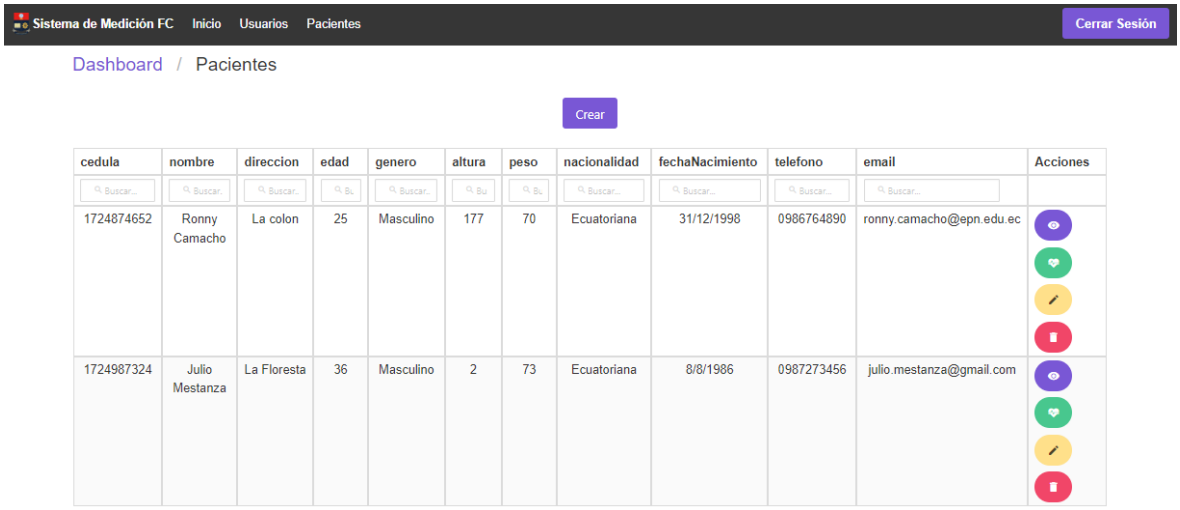
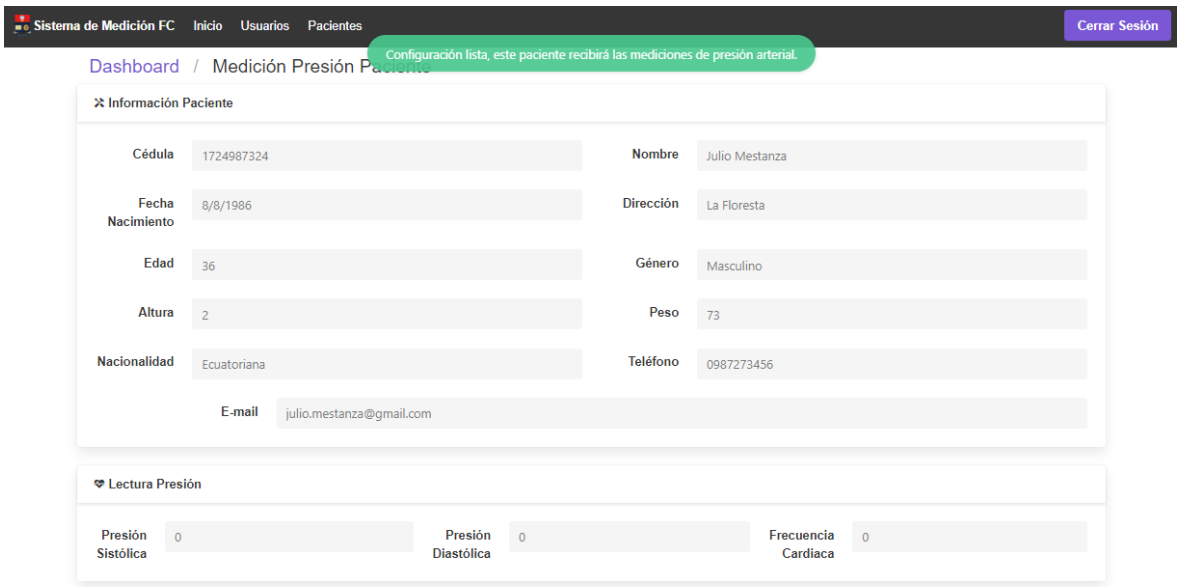


Figura 3.11. Resultado de registro de nuevo Paciente.

Una vez existan pacientes registrados, se puede proceder con la medición de presión arterial con ayuda del tensiómetro digital, en la Figura 3.12 se detalla la ventana de mediciones correspondiente a un paciente registrado.



Además de la información del paciente, se desplegará una notificación que indica que el presente usuario es el seleccionado para registrar las mediciones de presión arterial que se tomen con el tensiómetro digital a partir del ingreso a su ventana de mediciones.

Una vez se haya seleccionado el paciente que recibirá la medición de presión arterial, se procede a realizarla con ayuda del tensiómetro digital, esta medición será en tiempo real y el Componente de Visualización de Resultados mostrará una notificación que se

encontrará por encima de la ventana de medición, la Figura 3.13 detalla el funcionamiento de esta notificación.

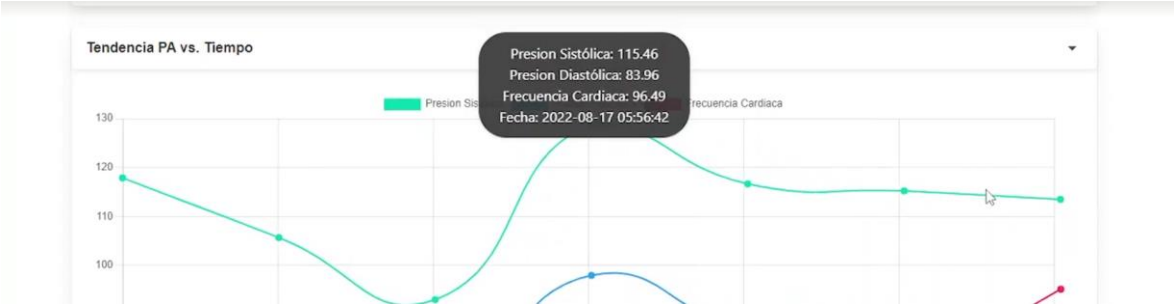


Figura 3.13. Alerta de medición de presión arterial.

Una vez registradas mediciones de presión arterial, es posible la visualización del informe de seguimiento de presión ambulatorio para el paciente al cual se registraron estas mediciones.

El informe de seguimiento de presión ambulatorio se despliega en la pestaña Ficha Paciente, la cual se divide en distintas secciones que ayudarán a mostrar el citado informe.

La Figura 3.14 detalla la sección Información del Paciente de la ventana Ficha Paciente, la cual se encarga de desplegar la información personal del paciente para identificar de manera correcta a quien pertenece el informe de presión ambulatorio.

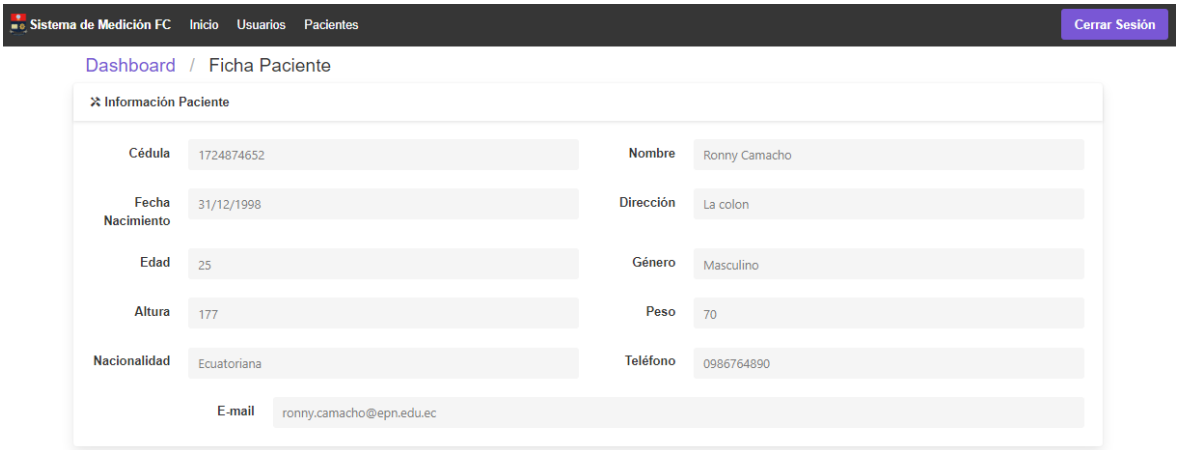


Figura 3.14. Sección de información personal del paciente.

Posteriormente se encuentra la sección de resultados conocida como MAPA, en la cual se visualizan valores estadísticos encargados de dar mayor información del seguimiento de presión ambulatorio, esta sección se detalla en la Figura 3.15.

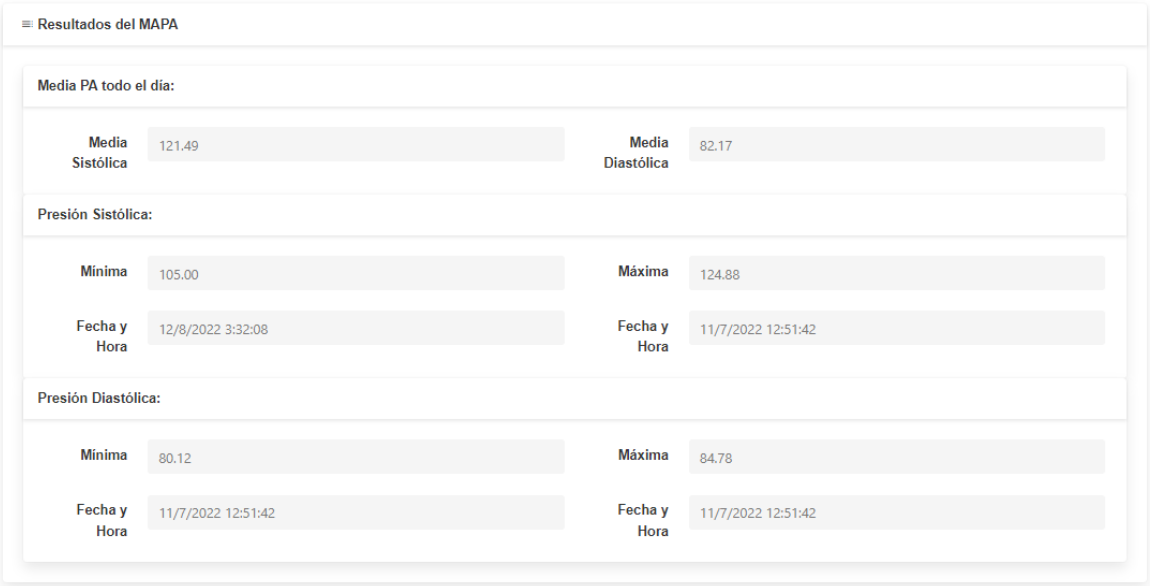


Figura 3.15. Resultados estadísticos de seguimiento de presión ambulatorio.

Posteriormente se encuentra una serie de secciones correspondientes a la visualización gráfica de los resultados de las mediciones de presión arterial.

También se ha incluido una sección correspondiente a mostrar un historial de mediciones de presión arterial por medio de una tabla que desplegará esta información en una ventana.

La Figura 3.16 muestra la sección de Tendencia de Presión Arterial vs. Tiempo, esto se logra a través de un gráfico comparativo de presión sistólica, diastólica y frecuencia cardiaca.

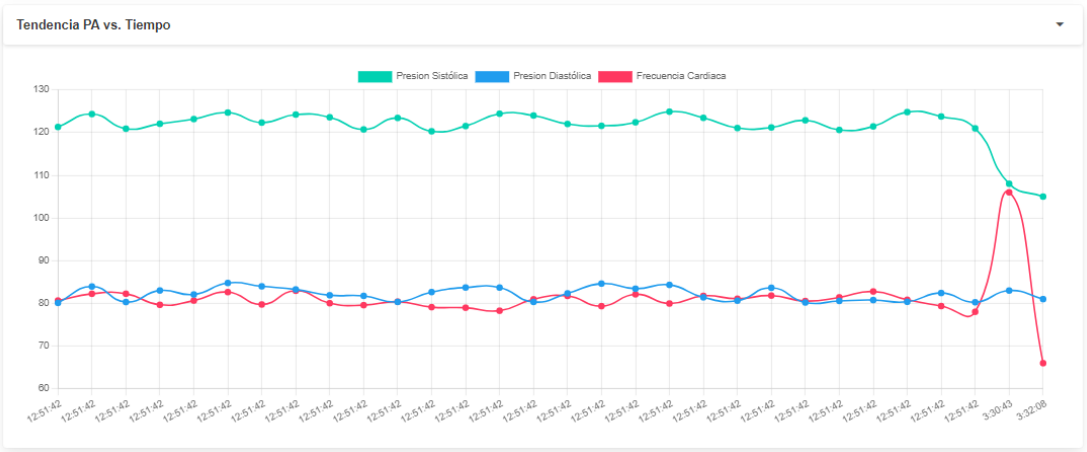


Figura 3.16. Gráfico de Tendencia PA vs. Tiempo.

Además del gráfico de tendencia mostrado en la Figura 3.16, se han incluido otra serie de gráficos de utilidad para monitorizar las mediciones de presión arterial de un paciente y mejorar el seguimiento ambulatorio por medio de los histogramas se clasificará las

mediciones en intervalos de 10mmHg para conocer cuáles son los porcentajes representativos de cada intervalo, por su parte los gráficos de pastel tienen 3 intervalos que indican el porcentaje de mediciones en intervalos de presión y frecuencia baja, media y alta. Estos gráficos serán presentados en la Figura 3.17.

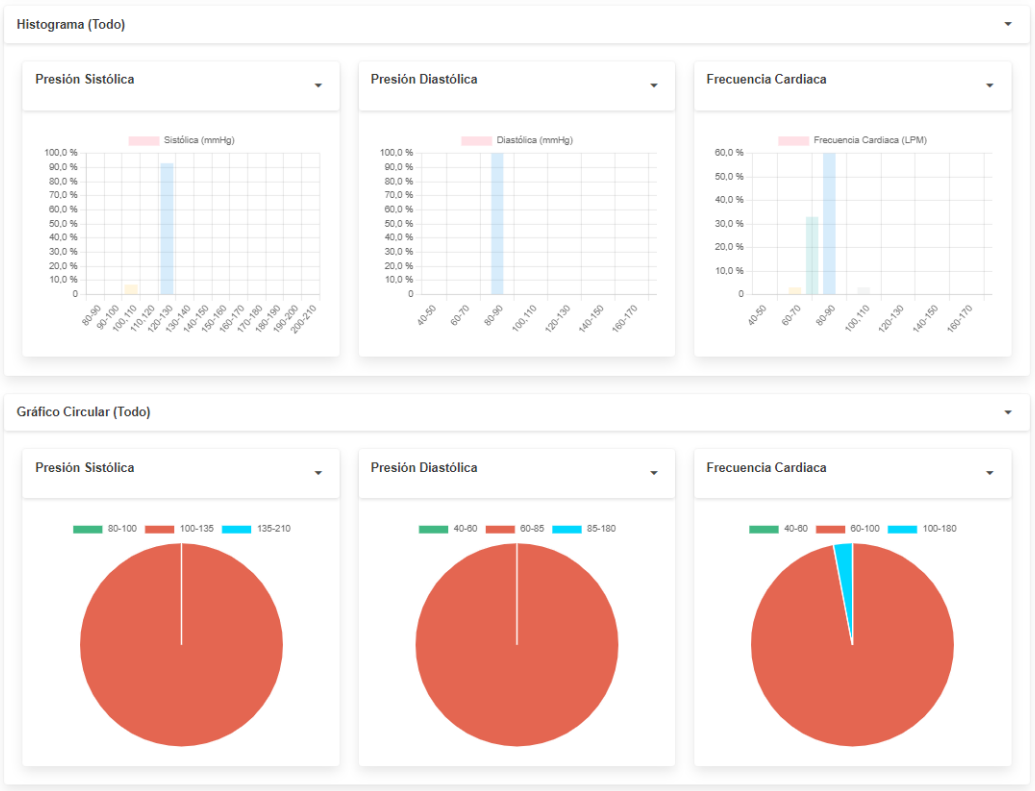


Figura 3.17. Gráficos auxiliares: Histograma y Circular.

Por último, se encuentra la sección de Historial de mediciones, mismo que se encarga de ordenar de primera a última medición, con el objetivo de conocer cuáles han sido las mediciones previas y conocer sus valores y la fecha en la que han sido realizadas y mostrarlas en forma de tabla para facilitar su visualización por parte del usuario. La Figura 3.18 será la encargada de detallar el cómo se ve la tabla de historial de mediciones.

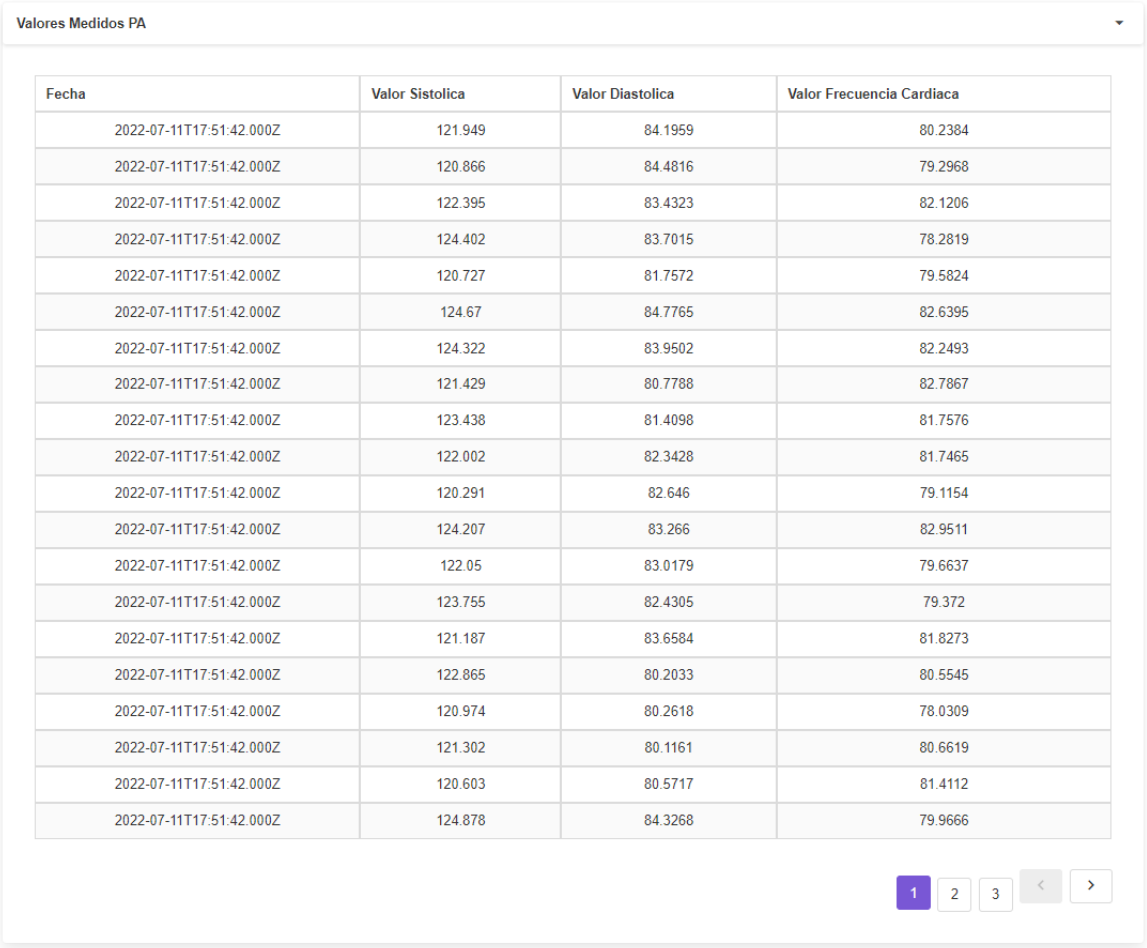


Figura 3.18. Tabla de historial de mediciones de PA.

3.2. CONCLUSIONES

El presente Trabajo de Integración Curricular se ha desarrollado por medio de una serie de etapas fundamentales, como son: Marco Teórico, Fase de Diseño, Implementación y Obtención de resultados, derivando así en un sistema que cumple con los objetivos planteados al inicio del desarrollo de este, y del cual se desprendes las conclusiones que se detallan a continuación.

- Mediante el desarrollo de un algoritmo digital para la adquisición, procesamiento y emisión de información obtenida desde un tensiómetro digital, con el fin de estimar valores de presión arterial y frecuencia cardiaca con capacidad de visualizar sus resultados en una página web y una base de datos que almacena automáticamente estos valores, se ha logrado mejorar el proceso de seguimiento ambulatorio de presión arterial y simplificarlo de cara a los usuarios finales.
- Para el desarrollo del sistema se analizó y adquirió conocimientos referentes a distintas tecnologías acerca del desarrollo de hardware, software, redes

inalámbricas y sistemas operativos. Además de estos conocimientos generales se obtuvieron habilidades específicas para la implementación de tecnologías y marcos de desarrollo como ExpressJS, VueJS, Socket.io y Python.

- Habiéndose tomado en cuenta los requisitos del sistema, se ha implementado un algoritmo de procesamiento digital de información que cumple con estos y satisface las necesidades detalladas con el fin de ofrecer un sistema robusto que aporte precisión a las mediciones tomadas por el tensiómetro digital.
- Los resultados obtenidos han pasado por un proceso de revisión exhaustivo, mediante pruebas realizadas con herramientas externas al sistema en un ambiente controlado que simula un escenario real, debidamente estandarizado y correctamente documentado, lo que nos permite asegurar la validez de cada uno de sus componentes y por lo tanto del sistema en su totalidad.
- El sistema se encuentra alojado en el Gateway escogido en la etapa de diseño de hardware, y funciona de manera fluida con una alta disponibilidad, sin caídas del servicio, ni interrupciones por retardos de procesamiento, lo que nos permite concluir que el sistema embebido cumple con las necesidades de procesamiento del sistema en su totalidad.

3.3. RECOMENDACIONES

Durante el transcurso del desarrollo del presente Trabajo de Integración Curricular existieron ciertas incidencias que afectaron al mismo, sin embargo, estas fueron resueltas a medida que iban apareciendo, lo cual nos permite detallar las siguientes recomendaciones.

- Al trabajar con una API REST, se recomienda utilizar sistemas de documentación estandarizados para la especificación de los métodos de acceso a los recursos del sistema, como ha sido el caso del presente Trabajo de Integración Curricular, el cual utiliza la herramienta de documentación Swagger.
- Al trabajar con un proyecto de desarrollo de software, es muy recomendable el uso de un sistema de gestión de versiones, como puede ser el caso de GIT, lo que permitirá llevar un control exhaustivo de las distintas versiones del proyecto, adicionalmente se recomienda el uso de un sistema de repositorios de código no locales que hagan uso de GIT, como pueden ser Github o Gitlab.
- Al desarrollar un proyecto que consista en un servicio o aplicación web, se recomienda la adición de métodos auxiliares que permitan conocer si este se

encuentra en línea a través de consultas sencillas y que no expongan los datos almacenados en el sistema.

- Para el trabajo con bases de datos se recomienda la adición de una capa de seguridad adicional al servidor o Gateway donde se encuentre alojado su sistema de gestión, esta capa de seguridad adicional puede ser provista por sistemas de respaldo en caso de eventos fortuitos y sistemas firewall para evitar la intrusión de usuarios no autorizados.

- Al trabajar con sistemas que son expuestos a una red, se debe asegurar que el Gateway del sistema cuente con las mayores capacidades de hardware posibles, con el objetivo de no provocar cuellos de botella que afecten al rendimiento del sistema, asegurando así una alta disponibilidad del mismo.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] O. Mejía-Rodríguez, R. Paniagua-Sierra, M. del R. Valencia-Ortiz, J. Ruiz-García, B. Figueroa-Núñez, y V. Roa-Sánchez, «Factores relacionados con el descontrol de la presión arterial», *Salud Pública de México*, vol. 51, n.º 4, pp. 291-297, ago. 2009.

[2] M. Rosa Oltra *et al.*, «Factores que influyen en la diferencia de medida de presión arterial entre el método auscultatorio y el oscilométrico», *Medicina Clínica*, vol. 127, n.º 18, pp. 688-691, nov. 2006, doi: 10.1157/13095096.

[3] C. A. Vayas Valdivieso, «Medidor digital de presión arterial y ritmo cardiaco», *Escuela Politécnica Nacional*, dic. 1981.

[4] D. Díaz, «Caracterización y modelado de sensores capacitivos para aplicaciones médicas», Accedido: 5 de mayo de 2022. [En línea]. Disponible en: https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/28/1/DiazAlD.pdf

[5] F. T. O. Lima y S. L. S. Maita, «DISEÑO DE UN SISTEMA INALÁMBRICO PARA MONITOREO DE PACIENTES AMBULATORIOS, UTILIZANDO SENSORES DE PRESIÓN ARTERIAL Y RITMO CARDÍACO E IMPLEMENTACIÓN DE UN PROTOTIPO DE PRUEBA», p. 212.

[6] K. P. P. Pedraza, «DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA NO INVASIVO PARA EL MONITOREO DE FRECUENCIA CARDIACA MEDIANTE DETECCIÓN AUTOMÁTICA DE UNA SECCIÓN DE PIEL», p. 122.

[7] A. Hernández Sánchez, «Modelo sistémico para la adquisición de datos para mhealth», Modelo sistemico usando cloud computing para mhealth, nov. 2019, Accedido: 15 de julio de 2022. [En línea]. Disponible en: http://tesis.ipn.mx:8080/xmlui/handle/123456789/27606

[8] P. D. Concha Gómez, «Procesamiento y análisis de señales biomecánicas adquiridas por redes de sensores», oct. 2018, Accedido: 15 de julio de 2022. [En línea]. Disponible en: http://tesis.ipn.mx:8080/xmlui/handle/123456789/26077

[9] W. A. Valencia Zambrano, «Diseño de prototipo doctor Pi para la medición y monitorización de signos vitales en adultos mayores utilizando sensores biométricos y médicos acoplados a raspberry Pi», may 2018, Accedido: 15 de julio de 2022. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/15570

[10] G. Mitchell, «The Raspberry Pi single-board computer will revolutionise computer science teaching [For & Against]», *Engineering & Technology*, vol. 7, n.º 3, pp. 26-26, abr. 2012, doi: 10.1049/et.2012.0300.

[11] C. Severance, «Eben Upton: Raspberry Pi», *Computer*, vol. 46, n.º 10, pp. 14-16, oct. 2013, doi: 10.1109/MC.2013.349.

[12] M. L. Salcedo-Tovar, «Minicomputador educacional de bajo costo Raspberry Pi: Primera parte», vol. 7, n.º 1, p. 18, 2015.

[13] D. Jaggar, «Arm Architecture And Systems», *IEEE Micro*, vol. 17, n.º 4, pp. 9-11, jul. 1997, doi: 10.1109/MM.1997.612174.

[14] V. B. Jadhav, T. S. Nagwanshi, Y. P. Patil, y D. R. Patil, «Digital Notice Board Using Raspberry PI», vol. 03, n.º 05, p. 5.

[15] M. F. Krafft, *The Debian System: Concepts and Techniques*. No Starch Press, 2005.

[16] B. Balon y M. Simic, «Using Raspberry Pi Computers in Education», en *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, may 2019, pp. 671-676. doi: 10.23919/MIPRO.2019.8756967.

[17] E. C. Foster y S. Godbole, «Overview of MySQL», en *Database Systems*, Berkeley, CA: Apress, 2016, pp. 451-460. doi: 10.1007/978-1-4842-1191-5_24.

[18] N. Barsoti y D. Gibertoni, «IMPACTO QUE O SEQUELIZE TRAZ PARA O DESENVOLVIMENTO DE UMA API CONSTRUÍDA EM NODE.JS COM EXPRESS.JS», *Revista Interface Tecnológica*, vol. 17, n.º 2, Art. n.º 2, dic. 2020, doi: 10.31510/infa.v17i2.964.

[19] S. H. Jensen, A. Møller, y P. Thiemann, «Type Analysis for JavaScript», en *Static Analysis*, Berlin, Heidelberg, 2009, pp. 238-255. doi: 10.1007/978-3-642-03237-0_17.

[20] «Introduction to Node.js», *Introduction to Node.js*. https://nodejs.dev/learn/introduction-to-nodejs (accedido 7 de junio de 2022).

[21] «Generalidades del protocolo HTTP - HTTP | MDN». https://developer.mozilla.org/es/docs/Web/HTTP/Overview (accedido 7 de junio de 2022).

[22] «Métodos de petición HTTP - HTTP | MDN». https://developer.mozilla.org/es/docs/Web/HTTP/Methods (accedido 7 de junio de 2022).

[23] «REST - MDN Web Docs Glossary: Definitions of Web-related terms». https://udn.realityripple.com/docs/Glossary/REST (accedido 7 de junio de 2022).

[24] «Introduction to web APIs - Learn web development | MDN». https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (accedido 7 de junio de 2022).

[25] A. Mardan, Express.js Guide: The Comprehensive Book on Express.js. Azat Mardan, 2014.

[26] N. Patrylo y M. Miłosz, «Comparison of AngularJS and VueJS frameworks efficiency», Journal of Computer Sciences Institute, vol. 5, pp. 204-207, dic. 2017, doi: 10.35784/jcsi.622.

[27] «Socket.IO». https://socket.io/ (accedido 15 de julio de 2022).

[28] G. van Rossum (Guido), «Python reference manual», *Department of Computer Science [CS]*, n.º R 9525. CWI, 1 de enero de 1995. Accedido: 2 de marzo de 2022. [En línea]. Disponible en: https://ir.cwi.nl/pub/5008

[29] «Welcome to Flask — Flask Documentation (2.1.x)». https://flask.palletsprojects.com/en/2.1.x/ (accedido 16 de junio de 2022).

[30] «Bcrypt Algorithm». https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node5.html (accedido 24 de agosto de 2022).

[31] «OpenAPI Specification - Version 3.0.3 | Swagger». https://swagger.io/specification/ (accedido 3 de mayo de 2022).

[32] R. P. Ltd, «Buy a Raspberry Pi 1, 2 and 3 Power Supply», *Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-universal-power-supply/ (accedido 15 de julio de 2022).

[33] Karel Iser Brito, René A. Llanes Machado. «Algoritmos para la detección de la presión sistólica y diastólica en señales oscilométricas y auscultatoria »,UNIVERSIDAD CENTRAL“MARTA ABREU” DE LA VILLAS, Villa Clara 2005. Disponible en: http://dspace.uclv.edu.cu:8089/handle/123456789/5629

[34] J. Liu, J. Hahn and R. Mukkamala, "An initial step towards improving the accuracy of the oscillometric blood pressure measurement," 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2013, pp. 4082-4085, doi: 10.1109/EMBC.2013.6610442.

[35] Vayas Valdivieso, C., 1981. Medidor digital de presión arterial y ritmo cardíaco. [online] Bibdigital.epn.edu.ec. Available at: <http://bibdigital.epn.edu.ec/handle/15000/10664> [Accessed 27 July 2022].

[36] Vinueza Cahuasquí, M., 2013. Diseño e implementación de un dispositivo de monitorización de la presión arterial. [online] Repositorio.espe.edu.ec. Available at: <http://repositorio.espe.edu.ec/handle/21000/7159> [Accessed 27 July 2022].

[37] J. Quintanar-Gómez, D. Robles-Camarillo, F. R. Trejo-Macotela and I. Campero-Jurado, "Telemonitoring Device of Blood Pressure and Heart Rate Through Multilayer Perceptrons and Pulse Rate Variability," in IEEE Latin America Transactions, vol. 19, no. 7, pp. 1233-1241, July 2021, doi: 10.1109/TLA.2021.9461853.

[38] R. B. Rodríguez Salazar, "Monitor portátil de signos vitales con un PDA", septiembre de 2007. [En línea]. Disponible: http://bibdigital.epn.edu.ec/handle/15000/9921

[39] J. Quintanar-Gómez, D. Robles-Camarillo, F. R. Trejo-Macotela and I. Campero-Jurado, "Telemonitoring Device of Blood Pressure and Heart Rate Through Multilayer

Perceptrons and Pulse Rate Variability," in IEEE Latin America Transactions, vol. 19, no. 7, pp. 1233-1241, July 2021, doi: 10.1109/TLA.2021.9461853.

5 ANEXOS

Anexo A: Digital del código fuente del Componente de Manejo y Procesamiento de Datos (link).

Anexo B: Digital del código fuente del Componente de Visualización de resultados (link)

Anexo C: Manual de Usuario

Anexo D: Códigos

ANEXO A

Digital del código fuente del Componente de Manejo y Procesamiento de Datos (link).

https://epnecuador-my.sharepoint.com/:f:/g/personal/kendy_camacho_epn_edu_ec/EtPavA0MkK1Hhuts2woBU0wBnW54owGUTw25oI-51McFYQ?e=wY30XC

ANEXO B

Digital del código fuente del Componente de Visualización de resultados (link)

https://epnecuador-my.sharepoint.com/:f:/g/personal/kendy_camacho_epn_edu_ec/EpJVvZiqjJ9Nr-gCWhwyV7wB1iLE7oGnYi0dl7H_7Mn4bA?e=MJvIPf

ANEXO C: MANUAL DE USUARIO

SISTEMA DE ADQUISICIÓN, ENCRIPTAMIENTO EMISIÓN Y PROCESAMIENTO DE LA INFORMACIÓN

1. ALIMENTACIÓN DEL DISPOSITIVO UTILIZADO PARA EL PROCESAMIENTO DE LA INFORMACIÓN.

La Raspberry pi necesita una alimentación de 5 voltios DC con una corriente de 2.5 amperios.

2. RED DE CONEXIÓN

La Raspberry pi y el tensiómetro digital deben estar conectados a una misma red inalámbrica WIFI debido a que la comunicación entre estos dos dispositivos se la realizara por dicho medio. En este caso se cuenta con un router inalámbrico que simula una conexión de red local entre estos dos dispositivos. El nombre de la red es 'TP-Link_B538_5G' y la contraseña es: '57723709'.

Figura 2.1. Nombre de la red

3. REVISIÓN DE ESTADO DE CONEXIÓN DE LA RASPBERRY

La Raspberry pi es una minicomputadora que cuenta con puertos USB y HDMI que sirven para poder visualizar en un monitor el sistema operativo. Para ver que este dispositivo que funciona como Gateway funcione correctamente se debe hacer lo siguiente.

- Abrir el CMD del sistema operativo de la Raspberry.

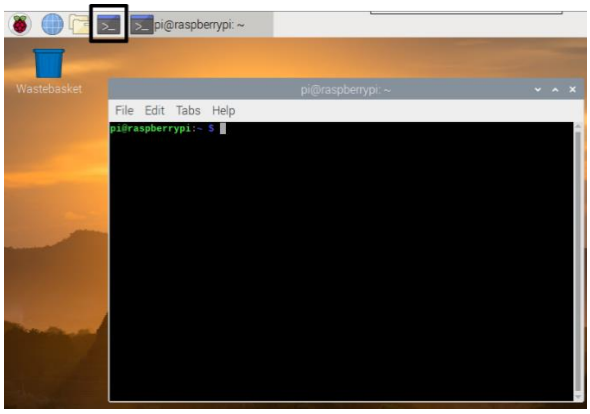


Figura 3.1. CMD de sistema operativo de la Raspberry

- Ejecutar el comando ‘pm2 status’

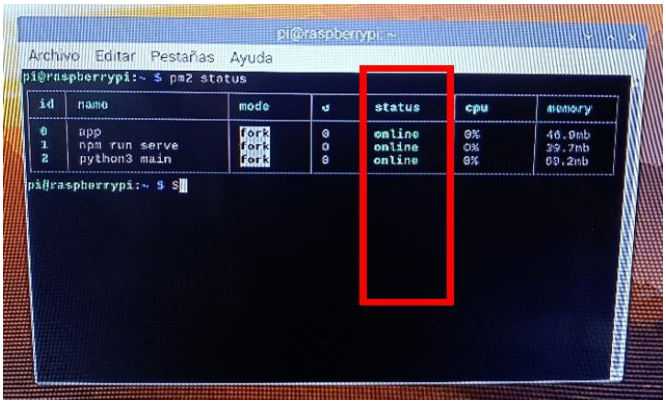


Figura 3.2. Estado de conexión de la raspberry en el CMD

Como se puede observar en la Figura 3.2 se tiene 3 opciones enumeradas del 0 al 2. Estas opciones debes estar en ‘online’ para poder desplegar y ejecutar sin ningún inconveniente la página web, la base de datos y el procesamiento del algoritmo digital.

4. VISUALIZACIÓN DE RESULTADOS

Para poder observar los resultados se debe utilizar un computador o celular cualquiera y acceder a la dirección 192.168.0.107:8080 desde cualquier navegador web. Una vez se haya accedido a esa dirección les aparecerá la siguiente página como en la F bigura 4.1.

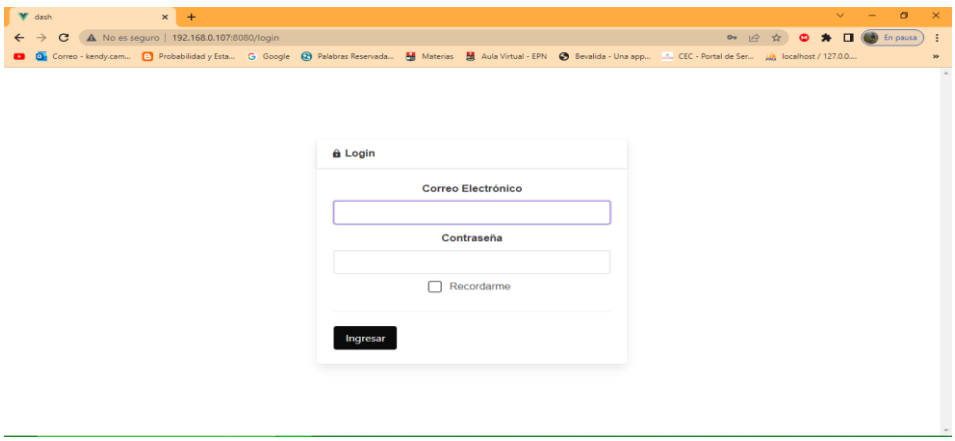


Figura 4.1. Página web con la dirección 192.168.0.107:8080

En donde debe ser colocado el correo y la clave del administrador.

- Usuario: admin@epn.edu.ec
- Contraseña: admiepn

Una vez que se ingrese como administrador aparecerá la siguiente pestaña que se visualiza en la Figura 4.2 en la cual se tiene las opciones de inicio, usuarios y pacientes.

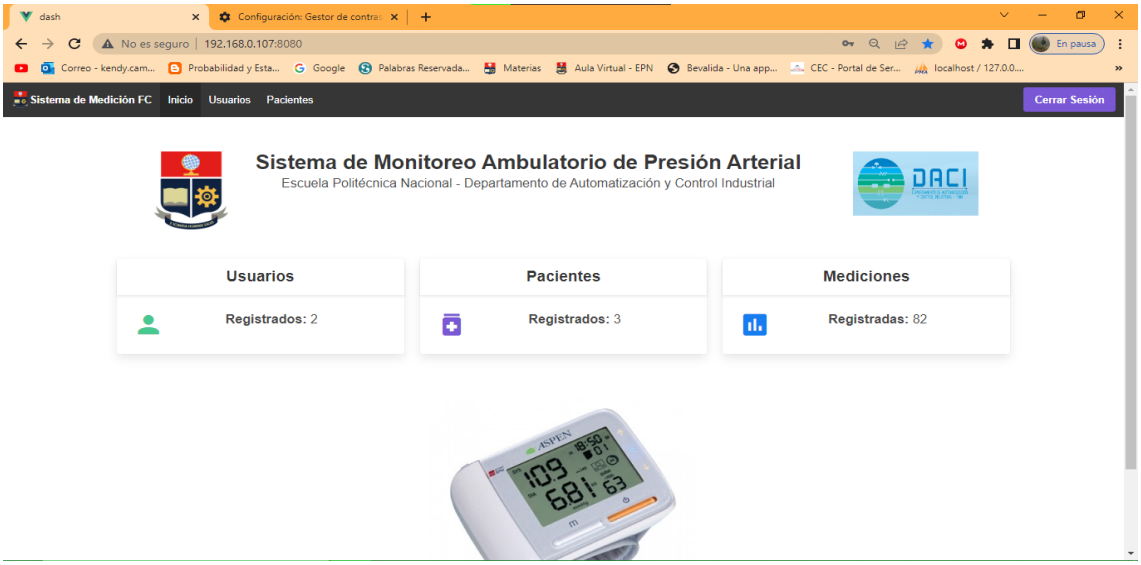


Figura 4.2. Página de inicio

4.1. OPCIÓN PARA CREAR USUARIOS.

Primero se debe seleccionar la opción de usuarios luego seleccionar la opción de crear y de esa manera se crea un usuario el cual puede realizar las mismas acciones que el usuario administrador con la diferencia de que no puede crear, editar ni eliminar usuarios, esa opción solo la puede realizar el usuario administrador.

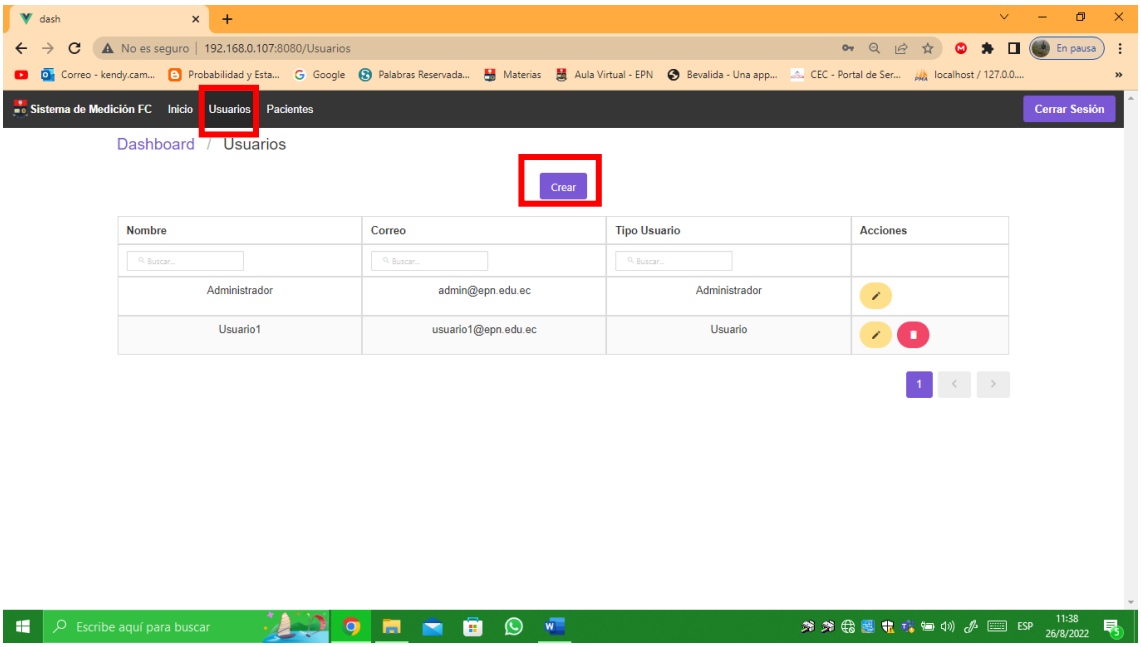


Figura 4.3. Opción de crear usuarios

4.2. OPCIÓN PARA CREAR HISTORIAL DEL PACIENTE.

Para acceder a la opción de los historiales se debe hacer clic en la opción de pacientes y aparecerá la pestaña que se muestra en la Figura 4.4. En esta opción se puede crear un historial con la información de cada paciente. En la parte derecha se tiene para cada paciente 4 acciones las cuales son:

Accion1: Sirve para ver toda la información ambulatoria y todas las medidas que se le han efectuado al paciente. (En color morado y con un ojo)

Acción 2: Sirve para seleccionar que las mediciones efectuadas por el tensiómetro digital se almacenen en el historial de dicho paciente. (En color verde y con un corazón)

Acción 3: Sirve para editar el historial del paciente. (En color amarillo y con un lápiz)

Acción 4: Sirve para eliminar el historial del paciente. (En color rojo y con un basurero)

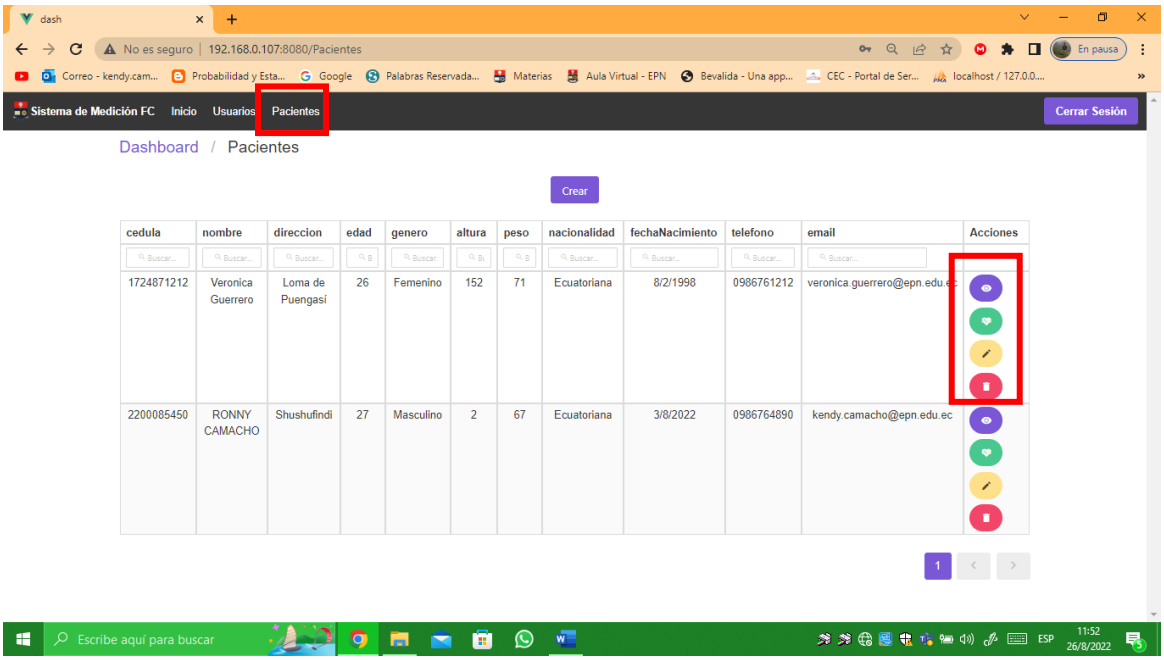


Figura 4.4. Opción de pacientes

ANEXO C

Código principal utilizado en el proyecto

```
{
"development": {
"username": "userPA",
"password": "passPA",
"database": "dbPresionArterial",
"host": "127.0.0.1",
"dialect": "mysql"
},
"test": {
"username": "userPA",
"password": "passPA",
"database": "dbPresionArterial",
"host": "127.0.0.1",
"dialect": "mysql"
},
"production": {
"username": "userPA",
"password": "passPA",
"database": "dbPresionArterial",
"host": "127.0.0.1",
"dialect": "mysql"
}
}
```

Código 2.2. Contenido del archivo config.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('Lecturas', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      valorSistolica: {
        type: Sequelize.FLOAT
      },
      valorDiastolica: {
        type: Sequelize.FLOAT
      },
      valorFrecuencia: {
        type: Sequelize.FLOAT
      },
      fecha: {
        type: Sequelize.DATE
      },
      idPaciente: {
        type: Sequelize.INTEGER,
        references:{
          model: 'Pacientes',
          key: 'id'
        },
        onDelete: 'CASCADE',
        onUpdate: 'CASCADE'
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('Lecturas');
  }
};
```

Código 2.4. Contenido del archivo 20220707230339-create-lectura.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Lectura extends Model {
    static associate(models) {
      Lectura.belongsTo(models.Paciente,{
        foreignKey: 'idPaciente',
        target_Key: 'id'
      })
    }
  };
  Lectura.init({
    valorSistolica: DataTypes.FLOAT,
    valorDiastolica: DataTypes.FLOAT,
    valorFrecuencia: DataTypes.FLOAT,
    fecha: DataTypes.DATE,
    idPaciente: DataTypes.INTEGER
  }, {
    sequelize,
    modelName: 'Lectura',
  });
  return Lectura;
};
```

Código 2.5. Contenido del archivo Lectura.js

```
const express=require('express')
const app=express()
const http = require('http').createServer(app);

const io = require('socket.io')(http, {
    cors: {
      origins: ['http://localhost:8080']
    }
  });
const cors = require('cors')
const cookieParser = require('cookie-parser')
const modeloLectura = require('./models').Lectura;
app.use(cors({
    credentials: true,
    origin:
['http://localhost:3000','http://localhost:8080','http://localhost:8081'
,'http://localhost:8082']
}))
app.use(cookieParser())
app.use(express.json())
app.use(express.urlencoded({ extended: true}))
app.io=io;
```

Código 2.6. Punto de entrada a la API REST

```
1. const express = require("express");
2. const router = express.Router();
3. const controllerLecturas =
   require('../controllers/controllerLecturas')
4.
5. router.get("/",controllerLecturas.getLecturas);
6. router.get("/getchart/:id",controllerLecturas.getLecturaCh
   art);
7. router.get("/getLast",controllerLecturas.getLastLectura);
8. router.get("/:id",controllerLecturas.getLecturaById);
9. router.post("/",controllerLecturas.createLectura);
10.     router.patch("/:id",controllerLecturas.editLectura);
11.     router.delete("/:id",controllerLecturas.deleteLectura
   );
12.
13.     module.exports = router;
```

Código 2.7. Contenido de routesLecturas.js

```
1.  const modeloLectura = require('../models').Lectura;
2.  const modeloPaciente = require('../models').Paciente;
3.  module.exports = class controllerLecturas {
4.      static async getLecturas(req,res){
5.          modeloLectura.findAll({})
6.          .then((data)=>{res.json({resultado:data})})
7.          .catch((error)=>{
8.              res.json({error:error})})}
9.      static async getLecturaById(req,res){
10.         modeloLectura.findOne({ where:{ id : req.params.id } })
11.         .then((data)=>{res.json({resultado:data})})
12.         .catch((error)=>{res.json({error:error})})
13.     }
14. }
```

Código 2.8 Contenido de controllerLecturas.js

```
1. const bcrypt = require("bcryptjs");
2. const jwt = require("jsonwebtoken");
3.
4. const salt = await bcrypt.genSalt(10);
5.         const hashedPassword = await
   bcrypt.hash(usuario['contrasenia'],salt);
6.
7.         usuario['contrasenia']=hashedPassword;
8.
9.         modeloUsuario.create(usuario)
10.             .then((data)=>{
11.                 res.json({resultado:data})
12.             })
13.             .catch((error)=>{
14.                 res.json({error:error})
15.             })
```

Código 2.9. Código para la encriptación de contraseñas.

```
1. @app.route('/procesarSensor', methods=['GET', 'POST'])
2. def procesarSensor():
3.     if request.method == 'POST':
4.         valorStr  = request.form.get('valor')
5.         listaValores=valorStr.split('\r\n')
6.         listaValores.pop()
7.         time.sleep(2)
8.         url = 'http://localhost:3000/api/lecturas'
9.         today = date.today()
10.             pd = randint(105, 110)
11.             sp = randint(77, 84)
12.             freqC= randint(55,120)
13.
   urlConfig='http://localhost:3000/api/configuracion/1'
14.             configResponse=requests.get(urlConfig)
15.             configData=configResponse.json()
16.             print(configData)
17.
   idPaciente=int(configData['resultado']['lastPaciente'])
18.             print(idPaciente)
19.             myobj = {'valorSistolica':
   str(pd),"valorDiastolica":str(sp),'valorFrecuencia':str(fr
   eqC),'fecha':today.strftime("%y-%m-
   %d"),'idPaciente':idPaciente}
20.             url2 ='http://localhost:3000/lecturaPOk'
21.             x = requests.post(url2, data = myobj)
22.             respuestaJson={"respuestaFinal":"respuesta
   ok"}
23.             response =
   make_response(jsonify(respuestaJson),200,        )
24.             response.headers["Content-Type"] =
   "application/x-www-form-urlencoded"
25.             return response
```

Código 2.10. Servicio de Procesamiento de Dato

```
@app.route('/procesarSensor', methods=['GET', 'POST'])
def procesarSensor():
    if request.method == 'POST':
        valorStr  = request.form.get('valor')
        listaValores=valorStr.split('\r\n')
        listaValores.pop()
        dataTens = np.array(listaValores).astype(float)
```

Código 2.11. Método de escucha de datos.

```
def filterData(f1,f2,fs,data):
    fn1=f1/fs
    fn2=f2/fs

    b, a = signal.butter(8, [fn1,fn2], 'bandpass')
    filteredData = signal.filtfilt(b, a, data)
    return filteredData

def filtroMedia(porcentaje,data):
    nMuestras=math.floor(len(data)*(porcentaje/100))
    startIntervalo=0
    endIntervalo=nMuestras

    for i in range(0,math.floor(100/porcentaje)):
        if i==math.floor(100/porcentaje)-1:
            endIntervalo=len(data)
            muestraDatos=data[startIntervalo:endIntervalo]
            media=np.absolute(np.mean(muestraDatos))

data[startIntervalo:endIntervalo]=data[startIntervalo:endIntervalo]-
media
            startIntervalo=startIntervalo+nMuestras
            endIntervalo=endIntervalo+nMuestras
        else:
            muestraDatos=data[startIntervalo:endIntervalo]
            media=np.absolute(np.mean(muestraDatos))

data[startIntervalo:endIntervalo]=data[startIntervalo:endIntervalo]-
media
            startIntervalo=startIntervalo+nMuestras
            endIntervalo=endIntervalo+nMuestras
    return data

oscilogramData=finalData[0:len(finalData)-1]-finalData[1:len(finalData)]
fs=11
f1=1
f2=4
oscilogramData=filterData(f1,f2,fs,oscilogramData)
oscilogramData=filtroMedia(5,oscilogramData)
```

Código 2.13. Cálculo de oscilograma y filtrado en frecuencia y energía.

```
def hl_envelopes_idx(s, dmin=1, dmax=1, split=False):
    lmin = (np.diff(np.sign(np.diff(s))) > 0).nonzero()[0] + 1
    lmax = (np.diff(np.sign(np.diff(s))) < 0).nonzero()[0] + 1
    if split:
        s_mid = np.mean(s)
        lmin = lmin[s[lmin]<s_mid]
        lmax = lmax[s[lmax]>s_mid]
    lmin = lmin[[i+np.argmin(s[lmin[i:i+dmin]]) for i in range(0,len(lmin),dmin)]]
    lmax = lmax[[i+np.argmax(s[lmax[i:i+dmax]]) for i in range(0,len(lmax),dmax)]]
    return lmin,lmax
low_idx,high_idx = hl_envelopes_idx(oscilogramData,dmin=8,dmax=5)
t = np.linspace(0,len(oscilogramData),len(oscilogramData))
yupper=oscilogramData[high_idx]
ylower=oscilogramData[low_idx]
fEnvUpper = Rbf(t[high_idx], yupper,function="cubic")
fEnvLower = Rbf(t[low_idx], ylower,function="cubic",fill_value="extrapolate")
envelopeUpper=fEnvUpper(t)
envelopeLower=fEnvLower(t)
```

Código 2.14. Cálculo de envolventes.

```
peaks, _ = find_peaks(oscilogramData, height=0,distance=6)
secs=len(oscilogramData)/11
FrecuenciaCard=len(peaks)*60/secs
```

Código 2.15. Determinación y conteo de picos para la determinación de frecuencia cardiaca

```
diffEnvelopes=envelopeUpper-envelopeLower

posMaxDiff=np.argmax(diffEnvelopes)

plt.vlines(x = t[posMaxDiff], ymin = np.amin(envelopeLower)*1.2, ymax =
np.amax(envelopeUpper)*1.2,
        colors = 'black',
        label = 'vline_multiple - full height')
maxDiff=diffEnvelopes[posMaxDiff]
MPA=finalData[posMaxDiff]
print("MPA: "+str(MPA))

posSP=posMaxDiff-1
for i in range(posMaxDiff-1,0,-1):
    posSP=i
    porcentajeMPA=diffEnvelopes[i]/maxDiff
    if porcentajeMPA<=0.55:
        break

posDP=posMaxDiff+1
for i in range(posMaxDiff+1,len(diffEnvelopes)):
    posDP=i
    porcentajeMPA=diffEnvelopes[i]/maxDiff
    if porcentajeMPA<=0.95:
        break

plt.vlines(x = t[posDP], ymin = np.amin(envelopeLower)*1.2, ymax =
np.amax(envelopeUpper)*1.2,
        colors = 'black',
        label = 'vline_multiple - full height')

plt.vlines(x = t[posSP], ymin = np.amin(envelopeLower)*1.2, ymax =
np.amax(envelopeUpper)*1.2,
        colors = 'black',
        label = 'vline_multiple - full height')

valSP=finalData[posSP]
print("SP: "+str(valSP))

valDP=finalData[posDP]
print("DP: "+str(valDP)
```

Código 2.16. Determinación de presión arterial.

```
1. url = 'http://localhost:3000/api/lecturas'
2. today = date.today()
3. configResponse=requests.get(urlConfig)
4. configData=configResponse.json()
5. print(configData)
6. idPaciente=int(configData['resultado']['lastPaciente'])
7. myobj = {'valorSistolica':
   str(valDP),"valorDiastolica":str(valSP),'valorFrecuencia':str(Frecuencia
   Card),'fecha':today.strftime("%y-%m-%d"),'idPaciente':idPaciente}
```

Código 2.17. Resultado del algoritmo de procesamiento a enviarse al Componente de Visualización de Resultados

```
1.  const routes = [
2.    {
3.      path: '/',
4.      name: 'Inicio',
5.      component: Inicio
6.    },
7.    {
8.      path: '*',
9.      redirect: '/'
10.   },
11.   {
12.     path: '/about',
13.     name: 'About',
14.     component: function () {
15.       return import(/* webpackChunkName: "about" */
    '../views/About.vue')
16.     }
17.   },
18.   {
19.     path: '/Pacientes',
20.     name: 'Pacientes',
21.     component: function () {
22.       return import(/* webpackChunkName: "about" */
    '../views/Pacientes.vue')
23.     }
24.   },
25.   }
26. ]
```

Código 2.20. Rutas del proyecto Vue parte 1.

```
1.  <template>
2.      <div>
3.        <SecurityComponent/>
4.      <div class="container">
5.        <div class="hero-body">
6.          <div class="columns">
7.            <div class="column is-1">
8.              <figure class="image is-128x128">
9.              <img width="100" height="100"
    :src="require('../assets/logoEPN.png')">
10.             </figure>
11.           </div>
12.           <div class="column is-9">
13.             <p class="title">
14.           Sistema de Monitoreo Ambulatorio de Presión Arterial
15.         </p>
16.             <p class="subtitle">
17.           Escuela Politécnica Nacional - Departamento de Automatización
    y Control Industrial
18.         </p>
19.           </div>
20.           <div class="column is-2">
21.             <figure class="image is-16by9">
22.             <img :src="require('../assets/logoDaci.png')">
23.             </figure>
24.           </div>
25.         </div>
26.       </div>
27. </template>
```

Código 2.23. Sección template del archivo Inicio.vue

```
methods:{
    fetchUsuarios() {
      try {
        fetch(
          process.env.VUE_APP_TITLE + ":3000/api/usuarios",
          {
            method: "GET",
            headers: { "Content-Type": "application/json" },
            credentials: "include",
          }
        )
    },
    fetchPacientes() {
      try {
        fetch(
          process.env.VUE_APP_TITLE +":3000/api/pacientes",
          {
            method: "GET",
            headers: { "Content-Type": "application/json" },
            credentials: "include",
          }
        )
   },
    fetchLecturas() {
      try {
        fetch(
          process.env.VUE_APP_TITLE +":3000/api/lecturas",
          {
            method: "GET",
            headers: { "Content-Type": "application/json" },
            credentials: "include",
          }
        )
    },
  }
```

Código 2.27. Sub-sección methods del archivo Inico.vue.

```
const http = require('http').createServer(app);

const io = require('socket.io')(http, {
    cors: {
      origins: ['http://localhost:8080','http://localhost:8081']
    }
  });

app.io=io;

app.io.on('connection', (socket) => {
    console.log('a user connected');
    socket.on('disconnect', () => {
      console.log('user disconnected');
    });
  });
  app.post('/lecturaPOk',(req,res)=>{
    console.log("Testio")
    console.log(req.body)
    modeloLectura.create(req.body)
    .then((data)=>{
        app.io.emit('lecturaOk',req.body)
        res.json({resultado:data})
    })
    .catch((error)=>{
        res.json({error:error})
    })
})
```

Código 2.30. Código de Socket.io en el Componente de Manejo y Procesamiento de Datos

```
1.  import Vue from 'vue'
2.  import App from './App.vue'
3.  import router from './router'
4.  import store from './store'
5.
6.  import VueSocketIO from 'vue-socket.io'
7.  import SocketIO from 'socket.io-client'
8.
9.  import Buefy from 'buefy'
10. import 'buefy/dist/buefy.css'
11. import '@mdi/font/css/materialdesignicons.css'
12. Vue.config.productionTip = false
13. Vue.use(Buefy)
14. const options = {};
15. Vue.use(new VueSocketIO({
16.     debug: true,
17.     connection: SocketIO(process.env.VUE_APP_TITLE+':3000', options),
18.   })
19. );
20. new Vue({
21.   router,
22.   store,
23.   render: function (h) { return h(App) }
24. }).$mount('#app')
```

Código 2.31. Importación de la librería Socket.io al Componente de Visualización de Resultados.

```
sockets: {
    connection() {
      this.isConnected = true;
      console.log("Medicion Connected")
    },

    disconnect() {
      this.isConnected = false;
    },
          messageChannel(data) {
            this.socketMessage = data
          },

          lecturaOk(data){
            console.log(data)
            console.log("Medicion correcta socketio")
            this.$buefy.toast.open({message:"<p>Presion
  Sistólica: "+data.valorSistolica+"</p> <p>Presion Diastólica:
  "+data.valorDiastolica+"</p> <p>Frecuencia Cardiaca:
  "+data.valorFrecuencia+"</p> <p>Fecha: "+data.fecha+"</p>",
                                    duration:5000,
                                    type: 'is-success'})

            this.fetchUsuarios()
            this.fetchPacientes()
            this.fetchLecturas()
          }
        },
```

Código 2.32. Creación de la sub-sección sockets dentro del archivo Inicio.vue.

```
{
    path: '/Usuarios',
    name: 'Usuarios',
    component: function () {
      return import(/* webpackChunkName: "about" */ '../views/Usuarios.vue')
    }
  },
  {
    path: '/Mediciones',
    name: 'Mediciones',
    component: function () {
      return import(/* webpackChunkName: "about" */ '../views/Medicion.vue')
    }
  },
  {
    path: '/FichaPaciente/:id',
    name: 'Ficha Paciente',
    component: function () {
      return import(/* webpackChunkName: "about" */ '../views/FichaPaciente.vue')
    }
  },
  {
    path: '/MedicionPaciente/:id',
    name: 'Medición Paciente',
    component: function () {
      return import(/* webpackChunkName: "about" */ '../views/MedicionPaciente.vue')
    }
  },
  {
    path: '/Inicio',
    name: 'redInicio',
    component: function () {
      return import(/* webpackChunkName: "about" */ '../views/Inicio.vue')
    }
  },
  {
    path: '/Login',
    name: 'login',
    component: function () {
      return import(/* webpackChunkName: "about" */ '../views/Login.vue')
    }
  }
```

Código 2.21. Rutas del proyecto Vue parte 2.