Sistema de gestión de bioseñales en el sector de telemedicina con uso de IoMT

Bio-signal management system in the telemedicine sector using IoMT

Leonardo Juan Ramírez López [1] y Luz Dary Ávila Chacón

[1] Universidad Militar Nueva Granada, Bogotá, Colombia.
ltigumeonardo.ramirez@unimilitar.edu.co

Resumen. Las telecomunicaciones dentro de un marco sustancial e innovador representan métodos y soluciones pertinentes que permiten ejecutar sistemas capaces de mejorar procedimientos en distintos sectores claves de la sociedad. Uno de estos sectores es la medicina la cual requiere de desarrollos amplios y competentes que puedan agilizar procesos debido a la urgencia que sostienen las practicas clínicas en virtud de la salud de las personas. En este trabajo se presenta el desarrollo de una solución tecnológica para el monitoreo remoto de bioseñales, diseñada para mejorar el acceso a la atención médica, donde se aprovecha la tecnología para el seguimiento de pacientes a distancia. La plataforma seleccionada fue e-health platform V2.0, y este proyecto incorpora seis sensores que miden variables críticas como por ejemplo la temperatura corporal. Finalmente, con el fin de establecer una arquitectura y comunicación eficiente se tuvieron en cuenta varios criterios, como el protocolo de red, frameworks y dockerización, garantizando la gestión e integridad de los datos y la cohesión entre las diferentes capas del trabajo.

***Palabras claves**: Telemedicina, bioseñales, sockets, salud digital, código fuente, monitoreo remoto.*

Abstract. Telecommunications within a substantial and innovative framework represent relevant methods and solutions that allow the execution of systems capable of improving procedures in different key sectors of society. One of these sectors is medicine, which requires comprehensive and competent developments that can streamline processes due to the urgency that support clinical practices under the health of people. This paper presents the development of a technological solution for remote monitoring of biosignals, designed to improve access to medical care, where technology is used to track patients remotely. The selected platform was e-health platform V2.0, and this project incorporates six sensors that measure critical variables such as body temperature. Finally, in order to establish an efficient architecture and communication, several criteria were taken into account, such as network protocol, frameworks and dockerization, ensuring data management and integrity and cohesion between the different layers of the work.

***Keywords**: Telemedicine, bio signal, sockets, e-health, open-source, remote monitoring.*

1. Introducción

La tecnología según [1] debe ser entendida por fuera de su concepto instrumental y teniendo en cuenta la experiencia asociada a su desarrollo, esto para comprender que ópera dentro de un escenario determinado. De tal forma, la tecnología ha pasado de ser una herramienta simple y novedosa a ser un mecanismo evolutivo que aborda características perspectivas y de conocimiento del mundo [1], lo que la permite influir en diversos ámbitos de la población. Desarrollos técnicos, impulsos por aplicar nuevas formas de entendimiento, acontecimientos radicales, sucesos de fuerza mayor y en definitiva la ocurrencia de sistemas obsoletos han hecho prácticamente obligatorio que la sociedad tenga que relacionarse y usar la tecnología como sustento en su accionar cotidiano.

De tal forma adquiere importancia el término "remoto", el cual ha sido un adjetivo capaz de trasladar ciertos métodos de trabajo estructurado a trabajo a distancia con el fin de agilizar procedimientos, tomando en consideración que recurre en varios desafíos y dilemas dependiendo de las organizaciones que adopten la practica [2]. Específicamente en medicina, las fuerzas que han llevado a utilizar esta práctica han sido eventos que han provocado la saturación del sistema y por lo tanto han solicitado la búsqueda de sistemas capaces que beneficien a los ecosistemas clínicos evitando de cierto modo el sobreuso del sistema en todo sentido, optimizando condiciones laborales y mejorando la ejecución de las actividades médicas.

Con ello, simultáneamente ha surgido el fortalecimiento de la telemedicina, donde según [3] anualmente la tasa de crecimiento de esta disciplina está en alrededor de un 20% con un aumento exponencial de inversión económica debido a la satisfacción de los pacientes y la aceptación de los médicos expertos. Como la telemedicina se ha vuelto transcendente ha ocasionado la demanda e incremento progresivo del IoMT (Internet de las Cosas Medicas) y su integración con entornos de nube que favorezcan el acceso y la disponibilidad a los recursos y datos, resultando como dice [4] en un alivio a causa de colapsos operativos por el flujo constante que posee el sector medico; de tal forma con la telemedicina se provee establecer un sistema de salud inteligente que permita la gestión de conjuntos de datos grandes y permita combinar técnicas a fines tales como la robótica, telecomunicaciones, entre otras [4].

Así pues, las tecnologías de salud electrónica representan un amplio potencial para mejorar la calidad de la atención sanitaria [5], y que, además, no simbolizan una transición o reemplazo de los actores humanos puesto que el uso de la salud electrónica consiste en interacción social y material [6] a partir del sistema tradicional que no se fundamenta por el mundo digital [7]. Por ello, se requieren de habilidades especificas con el objetivo de emplear la informática en la medicina [6] y con relación a este trabajo, algunas de esas soluciones son los monitoreos e-health los cuales permiten capturar y transmitir señales biomédicas.

Estos monitoreos se pueden llevar a cabo con una arquitectura de software y hardware libre capaz de satisfacer el sistema y por ende según [8], el uso de placas Arduino en entornos de IoT permite desarrollar propios artefactos para cubrir necesidades generales con el fin de que los objetos con conexión a la red puedan intercambiar información vital.

Adicionalmente, el uso de placas como Raspberry Pi optimizan el monitoreo en línea y continuo en busca de tratamientos médicos inmediatos y una excelente comunicación paciente-medico [9].

Para completar este tipo de solución, se debe garantizar la transmisión eficiente de datos en tiempo real que implique una comunicación completa de las dos partes, por este motivo se consideran tecnologías como WebSockets que posibilitan una conexión bidireccional entre dispositivos y servidores, reduciendo la latencia y optimizando el flujo de datos [10]. Esto es esencial en entornos críticos como la telemedicina, donde la respuesta rápida asegura primordialmente un diagnóstico constante en relación con el bienestar del paciente.

2. Algunos conceptos preliminares

La base de este trabajo se centra en diversas teorías y conceptos que fundamentan cada paso secuencialmente realizado y por ello justifican su correlación con el desarrollo de la solución elaborada. Inicialmente, la teoría de la información, también conocida como teoría matemática de la comunicación, es una propuesta teórica presentada por Claude E. Shannon y Warren Weaver a finales de la década de 1940. Esta teoría está relacionada con las leyes matemáticas que rigen la transmisión y el procesamiento de la información y se ocupa de la medición de la información y de la representación de esta, así como también de la capacidad de los sistemas de comunicación para transmitir y procesar información [11]. En términos médicos, los bioseñales son señales eléctricas, ópticas, mecánicas o térmicas generadas por el cuerpo humano o cualquier organismo vivo. Estas señales proporcionan información sobre los procesos fisiológicos internos [12]. Los bioseñales se pueden clasificar en varias categorías según su origen y las características típicas, pero los más comunes son: ECG, electroencefalograma (EEG), presión arterial, sonidos cardíacos y medición de la temperatura corporal [12].

Uno de los procesos más revisados en este trabajo fue el desarrollo de software, este se refiere a un conjunto de actividades informáticas dedicadas al proceso de creación, diseño, implementación y soporte de software, de tal manera el desarrollo de software implica una serie de pasos e instrucciones que los desarrolladores siguen para crear software funcional y de alta calidad [13]. Estos pasos incluyen requisitos, diseño, implementación, pruebas y mantenimiento [14].

2.1 **Open Source**:

El término Open Source o Código Abierto se refiere a software que se distribuye con una licencia que otorga al usuario acceso al código fuente, permitiéndole estudiarlo y modificarlo libremente. Además, esta licencia permite la redistribución del software, siempre y cuando se respeten las condiciones estipuladas por la licencia original.

Es fundamental destacar que Open Source no implica necesariamente que el software sea gratuito; más bien, hace referencia al acceso libre al código fuente y a la posibilidad de utilizarlo, modificarlo y redistribuirlo bajo los términos especificados en su licencia [18]. Aunque el software de código abierto puede ser vendido a cualquier precio que el

distribuidor decida, lo que lo diferencia del software comercial tradicional es la flexibilidad que otorga al usuario para inspeccionar y personalizar el código según sus necesidades. [19]

2.2 **Modelo cliente servidor**:

El modelo cliente-servidor es una arquitectura que permite la interacción entre dos componentes: un cliente y un servidor, facilitando el intercambio de información entre ambos. Este sistema está diseñado para que múltiples usuarios puedan acceder simultáneamente a una misma base de datos, lo que permite el uso compartido de recursos e información en tiempo real [20]. El servidor actúa como el encargado de almacenar grandes volúmenes de datos, mientras que los clientes realizan solicitudes al servidor para obtener o enviar información.

Este tipo de arquitectura es común en aplicaciones en red, ya que permite una administración centralizada de los datos y un acceso eficiente por parte de múltiples usuarios desde diferentes ubicaciones. Además, asegura que los datos estén organizados y protegidos, facilitando la gestión y actualización de la información de manera controlada.

2.3 **Arquitectura orientada a servicios**:

La arquitectura orientada a servicios (SOA) es un enfoque de TI diseñado para alinear la infraestructura tecnológica con las necesidades y objetivos del negocio, facilitando la integración de sus procesos en forma de tareas o servicios empresariales reutilizables [21]. Este enfoque permite que las diversas funciones empresariales se desglosen en componentes modulares y repetibles, que pueden ser fácilmente combinados y reutilizados para diferentes aplicaciones y escenarios [21].

SOA se basa en el concepto de organizar y gestionar capacidades distribuidas, lo que implica que estas funciones o servicios pueden estar controlados por distintas áreas o propietarios dentro de una organización. Estos servicios se comunican entre sí a través de interfaces estándar, lo que permite que, independientemente de dónde se encuentren o quién los controle, puedan interoperar de manera eficiente y flexible [22].

Este paradigma facilita la adaptabilidad y escalabilidad del negocio, ya que permite reutilizar los servicios en múltiples contextos, lo que reduce la redundancia y optimiza el uso de recursos tecnológicos. Además, SOA fomenta la agilidad organizativa, ya que facilita la rápida implementación de cambios y mejoras en los procesos empresariales a medida que evolucionan las necesidades del negocio [22].

2.4 **Node.js**:

Node.js es un entorno de servidor de código abierto que permite la ejecución de código JavaScript en el lado del servidor. Este entorno es completamente gratuito y es compatible con diversas plataformas, como Windows, Linux, Unix y Mac OS X. Una de las principales características de Node.js es su capacidad para utilizar JavaScript en el servidor, lo que

permite a los desarrolladores unificar el lenguaje tanto en el front-end como en el back-end de sus aplicaciones [23].

Node.js está construido sobre el motor JavaScript V8 de Chrome, lo que le otorga un rendimiento optimizado. Su modelo de programación se basa en eventos y en una entrada/salida no bloqueante, lo que lo convierte en una opción muy eficiente y ligera en cuanto al uso de recursos. Esta arquitectura lo diferencia de otros entornos, ya que evita la espera por la finalización de operaciones antes de continuar con nuevas solicitudes, facilitando la ejecución de múltiples tareas de manera eficiente [23].

2.5 **Websockets**:

Es un protocolo de comunicación bidireccional sobre un socket TCP, que se caracteriza por mantener una conexión abierta entre el cliente y el servidor [24].

2.6 **Express Framework**:

Express es un framework back-end de aplicaciones web para Node.js que ofrece un conjunto amplio de funcionalidades para el desarrollo tanto de aplicaciones web como móviles. Es utilizado para crear aplicaciones de una sola página (single-page), de múltiples páginas (multipage) y aplicaciones web híbridas. Este framework actúa como una capa adicional sobre Node.js, facilitando la gestión de servidores y rutas de manera eficiente [25].

2.7 **React Framework**:

React.js, a diferencia de Node.js, es una biblioteca de JavaScript orientada al cliente. Su propósito es el desarrollo de interfaces de usuario, especialmente para aplicaciones de una sola página que requieren un modelo de interacción dinámico. Con React, se puede gestionar la capa de vista de las aplicaciones web, permitiendo a los desarrolladores definir sus interfaces en función de un estado que varía con el tiempo [26]. React Diseña vistas simples para cada estado de la aplicación y, cuando los datos cambian, React actualiza y renderiza de manera eficiente solo los componentes necesarios [26].

3. Método

La metodología utilizada se basó en los principios de las metodologías ágiles, que permiten un desarrollo iterativo y adaptativo, mejorando la capacidad de respuesta ante cambios y optimizando la entrega de valor al cliente de manera continua. En particular, se implementó la metodología SCRUM debido a su particular estructura que apoya el proceso de desarrollo [15].

Se utilizó el Product Backlog, una lista priorizada de todas las características, mejoras y correcciones necesarias para el proyecto. Esta lista sirve como fuente principal de trabajo para el equipo y permite la adaptación continua de los requerimientos según las necesidades

que permiten obtener un trabajo final satisfactorio [16]. A partir del Product Backlog, se derivan los Sprint Backlogs, que son los subconjuntos de tareas priorizadas a abordar durante cada sprint, un ciclo de trabajo que puede durar entre 1 y 4 semanas [17].

Adicionalmente, se realizaron revisiones semanales al final de cada sprint, donde el equipo presentaba el progreso alcanzado y se recogía feedback de las partes interesadas.

4. Resultados

La integración inicial del sistema se centró en conectar la tarjeta de desarrollo e-health v2.0, la cual es compatible con diversas librerías proporcionadas por Cooking Hacks, con la tarjeta Arduino. Esta integración permitió la adquisición de señales biomédicas, ya que la tarjeta Arduino actúa como el núcleo que procesa las mediciones de los sensores conectados. Los sensores utilizados incluyen aquellos para medir temperatura, electrocardiografía (ECG), flujo de aire, pulso y saturación de oxígeno en la sangre. Las librerías específicas de e-health v2.0 facilitaron el acceso a los datos brutos generados por estos sensores, habilitando la recolección eficiente y precisa de los valores biomédicos necesarios para el monitoreo remoto.

Posteriormente, para transmitir las señales adquiridas, se implementó un ordenador Raspberry Pi modelo B+. Esta pequeña computadora se conectó al sistema Arduino, aprovechando sus capacidades de procesamiento y conectividad. La Raspberry Pi actúa como intermediaria entre el hardware de adquisición de señales y el servidor central, permitiendo la transmisión fluida de datos biomédicos hacia una plataforma de monitoreo remoto.

En cuanto al software de conectividad, se evaluaron diversas opciones para la transmisión de los datos en tiempo real. Finalmente, se optó por utilizar el protocolo Websockets en lugar de HTTP, debido a sus características de comunicación bidireccional y continua. Este protocolo permite una conexión constante entre el servidor y el cliente, facilitando el intercambio de datos sin la necesidad de múltiples peticiones HTTP, lo que se traduce en una mayor eficiencia en la transmisión de señales biomédicas en tiempo real.

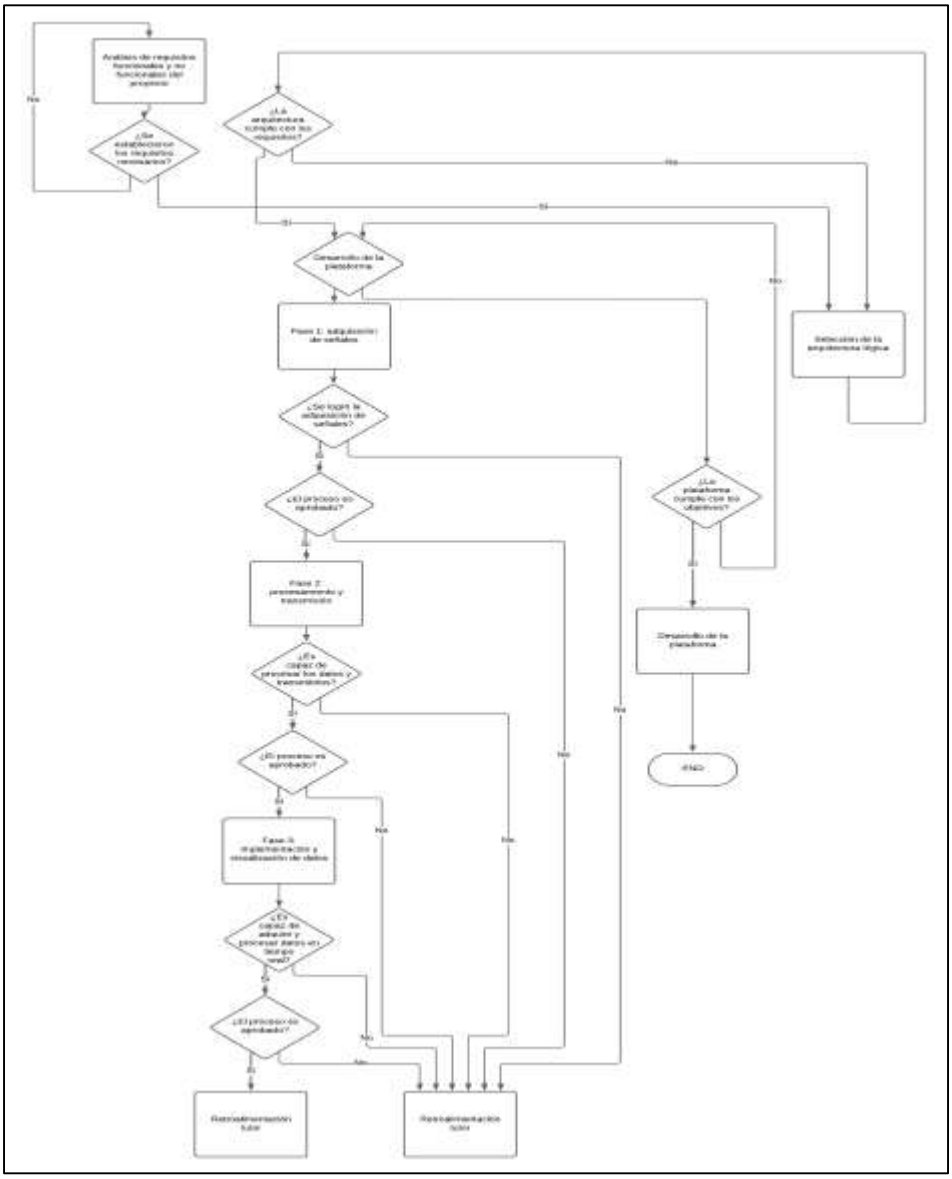
El bajo consumo de ancho de banda es otro de los beneficios claves de Websockets en este proyecto, ya que las señales biomédicas requieren un flujo constante de datos que no debe saturar la red ni generar latencias significativas. Aunque se identificaron algunos desafíos menores, como ciertas vulnerabilidades en términos de seguridad, Websockets cumplió con los requerimientos esenciales para el monitoreo remoto de señales en tiempo real, brindando una solución sólida y eficiente para la transmisión de datos biomédicos.

4.1 Arquitectura de datos

La solución fue desplegada en la infraestructura del grupo de investigación TIGUM de la Universidad Militar Nueva Granada (UMNG), lo que aporta al proyecto una gran

La Figura 1 presenta el flujograma de las actividades realizadas.

Figura 1. Flujograma de trabajo



confiabilidad en términos de seguridad y estabilidad. Esta infraestructura está específicamente diseñada para alojar aplicaciones de investigación, lo que brinda un

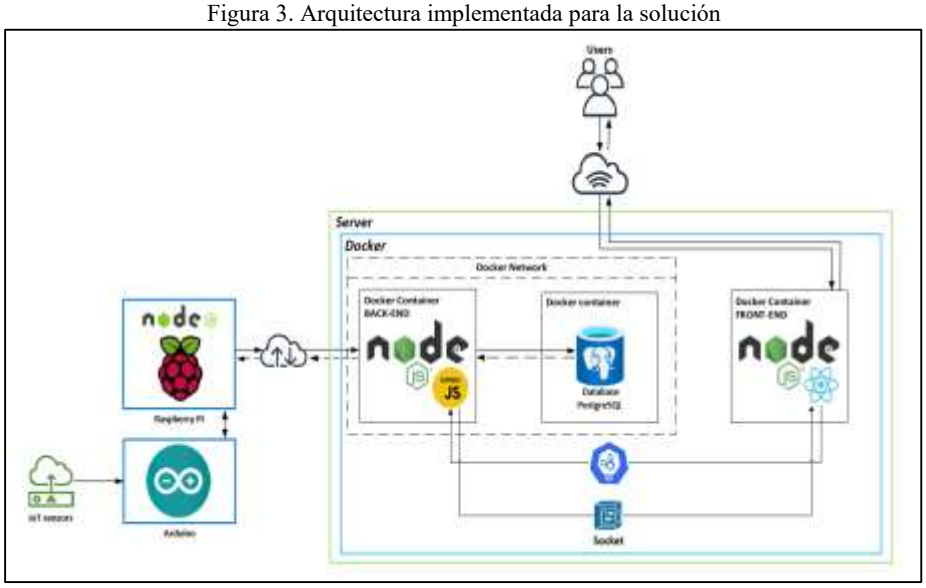
entorno robusto para la ejecución de la plataforma. Además, el servidor corre bajo la distribución de Linux Redhat, un sistema reconocido por su estabilidad y control sobre recursos de red, garantizando así la disponibilidad ininterrumpida del servicio.

Dentro del servidor se configuró una base de datos, donde se almacenan todas las señales biomédicas adquiridas por los sensores, junto con los datos relacionados con los usuarios. Esto permite que toda la información sea registrada de manera segura y organizada, para que pueda ser consultada por el personal médico o los usuarios finales, asegurando la continuidad y disponibilidad de los datos en cualquier momento que se necesiten.

Para la interfaz de usuario, se empleó el framework React, el cual permitió el desarrollo de un front-end dinámico y fácil de usar. Gracias a React, los usuarios pueden visualizar las señales biomédicas en tiempo real de manera clara e interactiva, lo que mejora considerablemente la experiencia de uso al ofrecer una plataforma intuitiva que no requiere conocimientos técnicos avanzados.

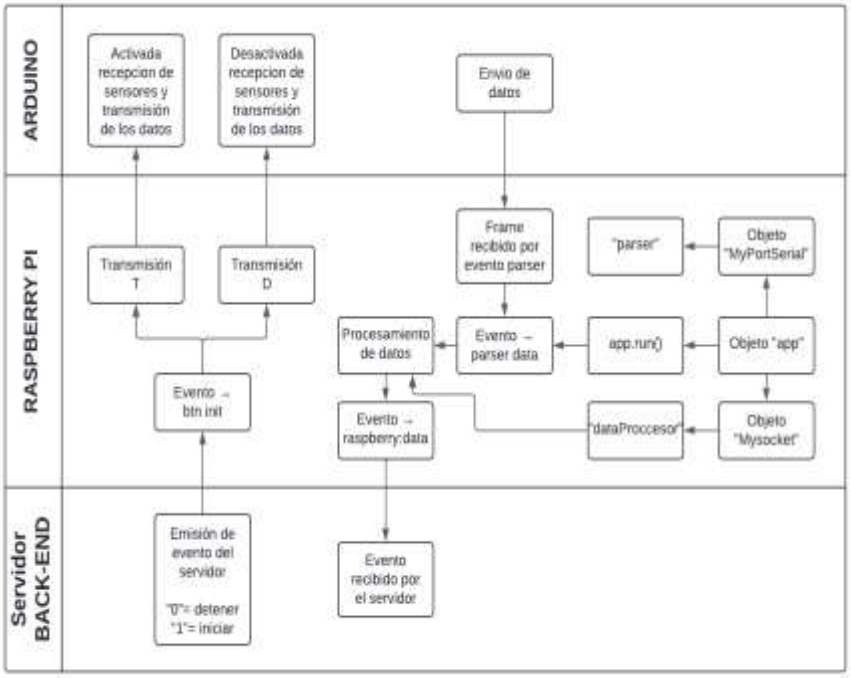
Por otra parte, para garantizar una adecuada gestión del sistema, se optó por una arquitectura de microservicios soportada por contenedores. En esta arquitectura, cada componente del sistema (ya sea la capa de back-end, la base de datos o la interfaz gráfica) se ejecuta de manera independiente en contenedores Docker, permitiendo que cada uno se comunique sin fricciones dentro de una red interna. Esto no solo facilita la interacción entre los distintos elementos del sistema, sino que también optimiza el rendimiento general, al garantizar que las tareas se distribuyan de forma eficiente entre los diferentes servicios.

Figura 3. Arquitectura implementada para la solución



El proceso de adquisición de los datos es llevado bajo tres bloques estructurados teniendo en cuenta el diagrama anterior, por ende, para adquirir los datos se tuvo en cuenta el servidor backend, la Raspberry Pi y el Arduino.

Figura 4. Proceso de adquisición de los datos.



En términos del procesamiento de las señales biomédicas capturadas, estas son procesadas de manera diferente dependiendo del tipo de muestreo. En el caso de las señales con un muestreo largo, como la temperatura o la saturación de oxígeno, se calcula el promedio entre el valor más reciente y el valor anterior. Este enfoque permite suavizar las fluctuaciones en los datos y proporciona un valor más estable para el monitoreo continuo.

Por otro lado, las señales con un muestreo corto, como el electrocardiograma (ECG) o el flujo de aire, se analizan de manera distinta. Aquí, el sistema evalúa el número de picos registrados en un minuto para determinar la frecuencia de ocurrencia de estos eventos, lo que es esencial para identificar posibles irregularidades en las señales vitales, como arritmias cardíacas o patrones respiratorios anómalos.

El procesamiento de estas señales permite identificar patrones específicos en los datos, los cuales pueden revelar cambios significativos en el estado de salud del usuario. Estos valores en tiempo real son comparados automáticamente con límites de referencia predefinidos por expertos en salud. Estos límites actúan como parámetros de referencia para clasificar el estado de cada variable biomédica.

Para facilitar la interpretación de los resultados, se implementa un sistema de clasificación por colores que utiliza "banderas" que indican el nivel de gravedad del estado

de salud del usuario. La clasificación contiene los siguientes colores y su indicación respectiva:

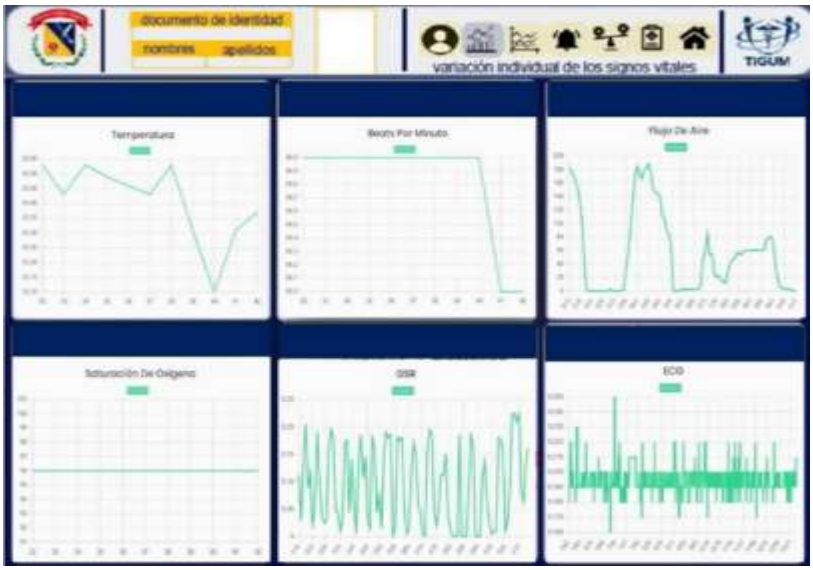
- ✓ La bandera verde indica que las señales están dentro de los rangos normales
- ✓ La bandera amarilla señala una leve anomalía que debe ser monitoreada
- ✓ La bandera naranja indica una situación que requiere atención médica pronta
- ✓ La bandera roja advierte sobre un estado crítico que necesita intervención inmediata.
    - o En total son 6 señales biomédicas:
        - ▪ Temperatura BPM Saturación de O2 Flujo de aire
        - ▪ GSR ECG

Por tal motivo se establecieron cuatro zonas de alerta, que indican de forma secuencial el estado de gravedad del paciente por cada señal biomédica. El límite superior de cada zona de alerta coincide prácticamente con el límite inferior de la zona siguiente, de manera que la transición entre los diferentes niveles de gravedad es continua y progresiva.

Por último, el sistema de monitoreo está diseñado para ofrecer una visualización en tiempo real de cada una de las señales biomédicas a través de un dashboard interactivo. Cada señal capturada es representada en su propia gráfica, que refleja el estado de la variable en un formato visual claro y comprensible.

Dependiendo del color asignado a cada señal, que corresponde a los límites de referencia previamente establecidos, la gráfica cambiará automáticamente de color en tiempo real, adaptándose de forma continua al estado de la señal. Esto permite que el personal médico o los usuarios finales puedan identificar rápidamente cualquier cambio en las condiciones de salud del paciente, facilitando una evaluación y diagnóstico dinámico y ágil. La capacidad de mostrar estos cambios instantáneamente en el dashboard asegura que las alertas visuales sean claras y accesibles, mejorando la capacidad de respuesta ante cualquier situación crítica.

Figura 5. Dashboard para staff médico



5. Conclusiones

Este trabajo ha demostrado que la implementación de soluciones tecnológicas basadas en plataformas de código abierto impulsa el desarrollo de la telemedicina, optimizando los procesos de monitoreo remoto y mejorando el acceso a servicios médicos a distancia.

La integración de tecnologías como la tarjeta e-health v2.0 y la Raspberry Pi modelo B+ ha permitido la adquisición precisa de bioseñales a través de dispositivos adecuados, cumpliendo con los objetivos de capturar y gestionar datos biomédicos en tiempo real.

El uso del protocolo Websockets, permitió una conexión ideal que logró que el proyecto se pudiera ejecutar sin problemas, este fue clave para establecer una comunicación bidireccional continua entre los dispositivos y el servidor.

Otro aspecto fundamental fue la contenerización del sistema mediante Docker. Esta estrategia facilitó la gestión y comunicación entre los diferentes componentes del sistema (back-end, front-end y base de datos).

Agradecimientos

Los autores agradecen a la Vicerrectoría de Investigación de la Universidad Militar Nueva Granada por el financiamiento del proyecto código INV-ING-3948. Los autores quieren reconocer el valioso aporte de Juan Alejandro Forero Gómez y Julián Andrés Duarte Suarez estudiantes de Ingeniería en Telecomunicaciones.

Referencias

[1] L. M. Brown, "What Is Technology?," *Advances in educational technologies and instructional design book series*, pp. 16–40, Jan. 2019, doi: https://doi.org/10.4018/978-1-5225-5519-3.ch002.

[2] R. Zakaria and A. Bliven, "Past, Present, and Future of Remote Work: The Swing of a Pendulum?," *Sagepub.com*, Jan. 2021, doi: https://doi.org/10.4135/9781529767292.

[3] M. Waller and C. Stotler, "Telemedicine: a Primer," *Current Allergy and Asthma Reports*, vol. 18, no. 10, pp. 1–9, Oct. 2018, doi: https://doi.org/10.1007/s11882-018-0808-4.

[4] "Potential of Internet of Medical Things (IoMT) applications in building a smart healthcare system: A systematic review," *Journal of Oral Biology and Craniofacial Research*, vol. 12, no. 2, pp. 302–318, Mar. 2022, doi: https://doi.org/10.1016/j.jobcr.2021.11.010.

[5] T. H. Tebeje and J. Klein, “Applications of e-Health to Support Person-Centered Health Care at the Time of COVID-19 Pandemic,” *Telemedicine Journal and e-Health*, vol. 27, no. 2, pp. 150–158, Feb. 2021, doi: https://doi.org/10.1089/tmj.2020.0201.

[6] D. Hallberg and N. Salimi, “Qualitative and Quantitative Analysis of Definitions of e-Health and m-Health,” *Healthcare Informatics Research*, vol. 26, no. 2, pp. 119–128, Apr. 2020, doi: https://doi.org/10.4258/hir.2020.26.2.119.

[7] Bach Xuan Tran *et al.*, “Feasibility of e-Health Interventions on Smoking Cessation among Vietnamese Active Internet Users,” *International Journal of Environmental Research and Public Health*, vol. 15, no. 1, pp. 165–165, Jan. 2018, doi: https://doi.org/10.3390/ijerph15010165.

[8] “Arduino y el Internet de las cosas,” *Google Books*, 2018. https://books.google.com.co/books?hl=es&lr=&id=FIlyDwAAQBAJ&oi=fnd&pg=PA14&dq=%22arduino%22&ots=xa3cyRzeSk&sig=krVlqGQddn6koTyk9m3_AU2n4nQ&redir_esc=y#v=onepage&q=%22arduino%22&f=false (accessed Sep. 16, 2024).

[9] Sunilkumar Laxmanbhai Rohit and B. V. Tank, “Iot Based Health Monitoring System Using Raspberry PI - Review,” Apr. 2018, doi: https://doi.org/10.1109/icicct.2018.8472957.

[10] K.E. Ogundeyi and C Yinka-Banjo, “WebSocket in real time application,” *Nigerian Journal of Technology*, vol. 38, no. 4, pp. 1010–1010, Dec. 2019, doi: https://doi.org/10.4314/njt.v38i4.26.

[11] C. de, “rama de las matemáticas que estudia la cuantificación, almacenamiento y comunicación de la información digital,” *Wikipedia.org*, Feb. 08, 2002. https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_informaci%C3%B3n (accessed Sep. 16, 2024).

[12] “Bioseñales: Procesamiento & Técnicas | StudySmarter,” *StudySmarter ES*, 2019. https://www.studysmarter.es/resumenes/ingenieria/ingenieria-biomedica/biosenales/ (accessed Sep. 16, 2024).

[13] “¿Qué es el desarrollo de software? | IBM,” *Ibm.com*, May 10, 2024. https://www.ibm.com/es-es/topics/software-development (accessed Sep. 16, 2024).

[14] adminstark, “¿Qué es el Desarrollo de Software? | Blog | StarkCloud,” *Starkcloud.com*, Oct. 12, 2023. https://www.starkcloud.com/starkcloud-blog/cloud/que-es-el-desarrollo-de-software (accessed Sep. 16, 2024).

[15] M. Morandini, Thiago Adriano Coleti, E. Oliveira, and P. Luiz, “Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams,” *Computer Science Review*, vol. 39, pp. 100314–100314, Feb. 2021, doi: https://doi.org/10.1016/j.cosrev.2020.100314.

[16] T. Sedano, P. Ralph, and C. Peraire, “The Product Backlog,” May 2019, doi: https://doi.org/10.1109/icse.2019.00036.

[17] Alhejab Alhazmi and S. Huang, “A Decision Support System for Sprint Planning in Scrum Practice,” Apr. 2018, doi: https://doi.org/10.1109/secon.2018.8479063.

[18] A. Stefania, “Que es Open Source,” *Academia.edu*, Feb. 27, 2016. https://www.academia.edu/22533644/Que_es_Open_Source (accessed Sep. 19, 2024).

[19] “OPEN SOURCE.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.openbiz.com.ar/Open%20Source.pdf

[20] Shakirat Haroon- Sulyman, “Client-Server Model,” *ResearchGate*, 2014. https://www.researchgate.net/publication/271295146_Client-Server_Model (accessed Sep. 19, 2024).

[21] “CSCI 5828 -Spring 2010 Foundations of Software Engineering -Arpit Sud.” Accessed: Sep. 19, 2024. [Online]. Available: https://home.cs.colorado.edu/~kena/classes/5828/s10/presentations/soa.pdf

[22] J. Erickson and K. L. Siau, “Service Oriented Architecture: A Research Review from the Software and Applications Perspective,” *ResearchGate*, 2009. https://www.researchgate.net/publication/288204978_Service_Oriented_Architecture_A_Research_Review_from_the_Software_and_Applications_Perspective (accessed Sep. 19, 2024).

[23] “What is Node.js?” Accessed: Sep. 19, 2024. [Online]. Available: https://aec.edu.in/aec/Instruction_Material/mst_unit3.pdf

[24] Liu Qigang and X. Sun, “Research of Web Real-Time Communication Based on Web Socket,” *ResearchGate*, 2012. https://www.researchgate.net/publication/276491172_Research_of_Web_Real-Time_Communication_Based_on_Web_Socket (accessed Sep. 19, 2024).

[25] A. Sharma, “Express JS Tutorial,” *Simplilearn.com*, Mar. 18, 2022. https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-

js#:~:text=StackExplore%20Program-
,What%20Is%20Express%20JS%3F,helps%20manage%20servers%20and%20routes.
(accessed Sep. 19, 2024).

[26] UXPin, “Node JS vs React JS – A Quick Comparison of Two JavaScript Frameworks,” *Studio by UXPin*, Aug. 20, 2024. https://www.uxpin.com/studio/blog/nodejs-vs-reactjs/ (accessed Sep. 19, 2024).