Parcial N°3 Sistema de Monitoreo de Sensores

Laboratorio: Sistema de Monitoreo de Sensores con Flask y Firebase

Valentina Hoyos Escobar

Facultad de Ingenierías / Programa de Ingeniería de Software

- Corporación Universitaria Empresarial Alexander Von Humboldt -

Fecha de entrega: *28 de noviembre de 2024*

Resumen

Este proyecto consistió en desarrollar un sistema para monitorear y visualizar en tiempo real los niveles de oxígeno en sangre y el ritmo cardíaco utilizando simulaciones y tecnologías web modernas. Se implementó una API en Flask que interactúa con Firebase para almacenar los datos generados por sensores simulados. Cada sensor mide parámetros biométricos clave asociados a un identificador único, lo que garantiza la trazabilidad de los datos.

La interfaz web, diseñada con HTML, CSS y la biblioteca Chart.js, permite visualizar los datos en gráficos dinámicos y consultar un historial tabulado. Además, se incorporaron funcionalidades como el filtrado de datos por fecha, selección de sensores y la generación automática de gráficos interactivos. Las gráficas en tiempo real destacan tendencias y patrones, facilitando el análisis de los datos biométricos.

La documentación incluyó un manual de usuario, instrucciones técnicas en `README.md` y un archivo `requirements.txt` para la gestión de dependencias del proyecto. La evaluación del sistema demostró su eficacia, cumpliendo con los objetivos iniciales. La integración de simulación, visualización interactiva y almacenamiento centralizado valida la viabilidad del sistema para aplicaciones futuras en el monitoreo automatizado de parámetros de salud.

Palabras clave: Sensores,API,Flask,Firebase.

Abstract

This project involved developing a system to monitor and visualize blood oxygen levels and heart rate in real time using simulations and modern web technologies. An API was implemented in Flask that interacts with Firebase to store the data generated by simulated sensors. Each sensor measures key biometric parameters associated with a unique identifier, ensuring data traceability.

The web interface, designed with HTML, CSS and the Chart.js library, allows data to be visualized in dynamic graphs and a tabulated history to be consulted. In addition, functionalities such as data filtering by date, sensor selection and automatic generation of interactive graphs are incorporated. Real-time graphs highlight trends and patterns, facilitating the analysis of biometric data.

The documentation included a user manual, technical instructions in `README.md` and a `requirements.txt` file for managing project dependencies. The evaluation of the system demonstrated its effectiveness, meeting the initial objectives. The integration of simulation, interactive visualization and centralized storage validates the viability of the system for future applications in automated monitoring of health parameters.

Keywords: Sensors,API,Flask,Firebase.

1. Introducción

En la actualidad, la automatización de procesos en sistemas de monitoreo es un pilar fundamental para mejorar la eficiencia y la precisión en diversas áreas. Un campo clave de aplicación es el monitoreo continuo de parámetros biométricos críticos como los niveles de oxígeno en sangre y el ritmo cardíaco. Este proyecto aborda el diseño e implementación de un sistema de sensores biométricos no invasivos que permiten recolectar y analizar estos datos de manera precisa en tiempo real.

El desarrollo del sistema se centra en la integración de tecnologías modernas de sensores de pulsioxímetro, que ofrecen ventajas como alta sensibilidad, confiabilidad en entornos dinámicos y bajo mantenimiento. Los sensores recolectan datos sobre la saturación de oxígeno (SpO2) y el ritmo cardíaco, los cuales son transmitidos a un sistema central para su monitoreo y análisis. Los datos se procesan y visualizan mediante interfaces gráficas dinámicas que facilitan la interpretación de las tendencias y patrones biométricos.

Con aplicaciones potenciales en sectores como el médico, deportivo y doméstico, este sistema tiene como objetivo optimizar el monitoreo de la salud, permitiendo una gestión más proactiva y personalizada. Además, al ser una solución accesible, eficiente y adaptable a diversas necesidades, establece un precedente en la innovación tecnológica para el cuidado de la salud y la mejora de la calidad de vida.

2. Objetivos

2.1. Objetivo general

Simular el funcionamiento de un sensor que envía datos a una API desarrollada con Flask.

2.2. Objetivos secundarios

- Crear una API que registre los datos enviados por el sensor en una base de datos Firebase.
- Diseñar una interfaz web que permita consultar y graficar los datos registrados, filtrándolos por sensor y rango de fechas.

3. Metodología

En este proyecto, se realizará la simulación de un sistema de monitoreo para tres pacientes, donde se referenciaran con el id "PULSIOXI- "numero de paciente".

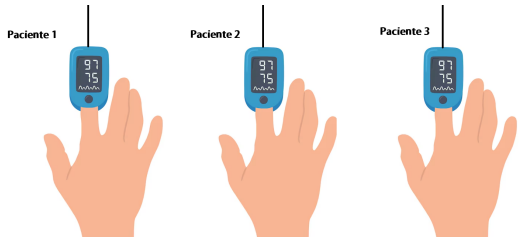

Figura 1 Representación de pacientes

El sensor se encargara de enviar la informacion de ritmo cardiaco y nivel de oxigeno de en la sangre en tiempo real, por lo cual se generara una grafica inspirada en un electrocardiograma para monitorear de manera mas optima cada paciente.

Para la implementación del sistema de monitoreo, se empleará la biblioteca Flask en Python, que facilitará la comunicación entre el servidor en tiempo real alojado en Firebase y los datos generados en campo. Flask actuará como intermediario para enviar peticiones HTTP que contienen información estructurada, incluyendo la fecha, el valor oxigeno medido y el identificador único de cada paciente. Estas peticiones serán procesadas por el servidor en Firebase, almacenándose en un nodo denominado sensor-pulsioximetro.

En Firebase, se configurarán reglas de acceso específicas para garantizar la seguridad de los datos. Estas reglas limitarán la lectura y escritura únicamente a las identificaciones de tanques autorizadas, mitigando posibles vulnerabilidades y asegurando la integridad del sistema.

La visualización de los datos se realizará mediante un gráfico de lineas que representará el ritmo cardiaco de cada paciente y a su vez se imprimira el nivel de oxigeno en sangre. Este gráfico se actualizará automáticamente cada 30 segundos, ofreciendo información en tiempo real al usuario. Adicionalmente, se incluirá un botón para realizar actualizaciones manuales, permitiendo mayor flexibilidad en el monitoreo.

Finalmente, al pie de la interfaz, se dispondrá de un historial universal que recopilará todos los datos ingresados al servidor. Estos datos estarán organizados en una tabla, facilitando su clasificación y consulta según las necesidades del usuario. Esta estructura proporcionará una visión integral del com-

portamiento del sistema a lo largo del tiempo y será clave para el análisis histórico y la trazabilidad.

4. Implementación del Sistema

El proceso de desarrollo del sistema se llevó a cabo siguiendo los pasos descritos a continuación:

1. Se creó una base de datos en tiempo real utilizando la funcionalidad *Real-time Database* de Firebase, configurada para almacenar y procesar los datos generados por los sensores simulados.

2. En la sección de configuración, dentro de *Cuentas de servicio*, se descargó el SDK de Firebase Admin, el cual fue integrado al repositorio del proyecto para garantizar la interacción entre el sistema y el servidor en tiempo real.

3. Se diseñó un archivo `.py` para el entorno de simulación, que genera datos de manera periódica y simula el comportamiento de los sensores.

4. Se desarrolló un archivo `.py` principal (`main`) que recibe los datos generados en el entorno simulado y gestiona las peticiones HTTP al servidor, integrando las funciones del sistema.

5. Se creó un archivo `.py` adicional encargado de ejecutar simultáneamente el entorno de simulación y el archivo principal, facilitando la sincronización entre ambos procesos.

6. Finalmente, se diseñó un archivo HTML que permite la visualización de los datos recopilados. Este archivo incluye una representación gráfica en forma de electrocardiograma. La gráfica se actualiza de forma automática o manual según la preferencia del usuario.

Cada etapa del proceso fue documentada cuidadosamente para asegurar la replicabilidad del sistema y su correcta operación en diferentes entornos.

4.0.1. Sensores

Usaremos el Sensor Pulsioxímetro GMD Pulsax 500E, debido a lo económico y simples que son.

Algunas caracteristicas de este es que mide ritmo cardiaco a su vez de nivel de oxigeno en una pantalla led.

El sistema utiliza una simulación de sensores de pulsioxímetro para monitorear en tiempo real dos

Figura 2 Sensor Pulsioxímetro GMD Pulsax 500E

Figura 3 Sensor Pulsioxímetro GMD Pulsax 500E caracteristicas

parámetros biométricos fundamentales: la saturación de oxígeno en sangre (SpO2) y el ritmo cardíaco. Se han configurado tres sensores virtuales, cada uno identificado de forma única mediante un código alfanumérico que incluye un identificador de usuario y un número de sensor, asegurando una trazabilidad clara en la base de datos.

Cada sensor mide y registra los siguientes valores:

- **Saturación de oxígeno (SpO2):** Representa el porcentaje de oxígeno transportado por la hemoglobina en la sangre. Este valor se encuentra en el rango del 95 % al 100 % en condiciones normales.

- **Ritmo cardíaco (lpm):** Indica la cantidad de latidos por minuto, oscilando en un rango típico de 60 a 100 lpm en reposo.

Estos datos se generan de forma continua y se almacenan en una base de datos central, donde se asocian con una marca temporal para facilitar el análisis posterior. Además, los valores recolectados se pueden visualizar mediante gráficos en tiempo real, lo que permite interpretar tendencias y realizar un monitoreo eficaz.

Los sensores envían los datos en formato JSON a una API desarrollada con Flask. Cada paquete de datos contiene:

- `idsensor`: Identificador único del sensor.
- `ritmo cardiaco`: Valor entero del ritmo cardiaco.
- `nivel oxigeno`: Valor entero del nivel de oxigeno en sangre.

El envío de datos se realiza a intervalos regulares, definidos en dos segundos, para garantizar una actualización constante y precisa de los parámetros biométricos monitoreados. En caso de errores durante la transmisión, el sistema registra los fallos para facilitar su diagnóstico y asegurar la continuidad del monitoreo. Además, los datos enviados por cada sensor son almacenados en estructuras locales, lo que permite su posterior visualización en gráficos interactivos que destacan tendencias en los niveles de oxígeno y ritmo cardíaco.

La implementación en Python permite que cada sensor funcione de manera concurrente mediante hilos de ejecución (*threads*), optimizando el rendimiento y asegurando que los datos de todos los sensores se envíen y procesen en tiempo real. Este diseño modular facilita la escalabilidad del sistema y su integración con bases de datos externas, como Firebase, para almacenamiento centralizado y análisis avanzado.

4.0.2. Firebase

Para la implementación del sistema, se configuró un proyecto en Firebase, seleccionando como servidor la región de Estados Unidos para optimizar la conectividad y el rendimiento del sistema. En la Figura 4, se presenta la configuración inicial de la base de datos.

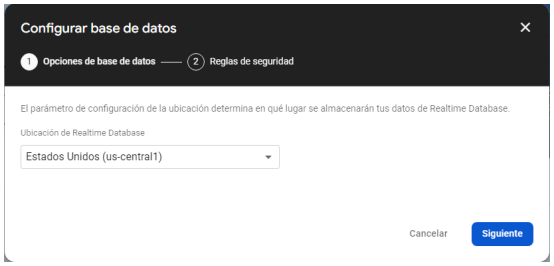

Figura 4 Configuración inicial del proyecto en Firebase.

Durante el proceso, se seleccionó el *modo prueba* para facilitar las pruebas iniciales, como se ilustra en la Figura 5.

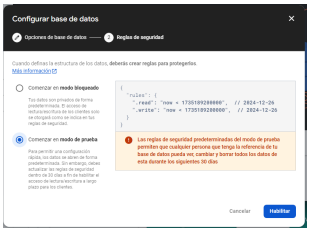

Figura 5 Selección del modo prueba.

Posteriormente, se creó una base de datos en tiempo real (*Real-time Database*), adecuada para manejar la comunicación continua entre los sensores simulados y el servidor, como se muestra en la Figura 6.

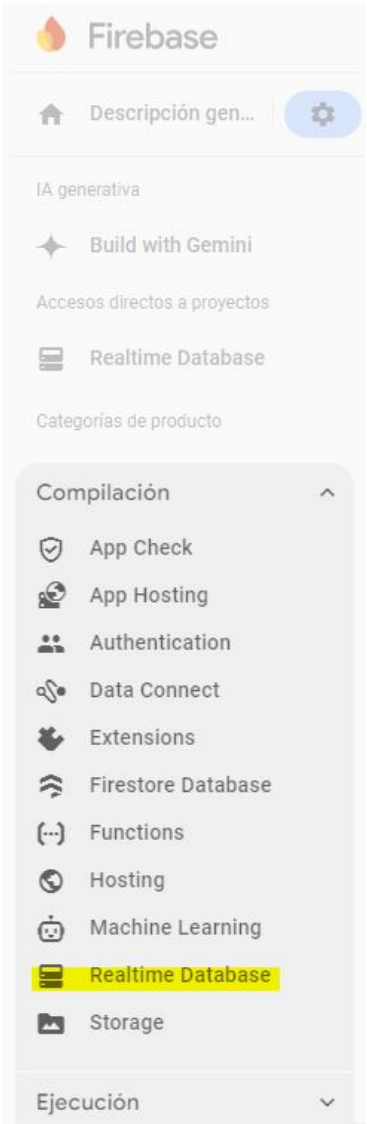

Figura 6 Configuración de la base de datos en tiempo real.

En la sección de configuración, dentro de *Cuentas de servicio*, se descargó el SDK de Firebase Admin. Este archivo se integró en el repositorio del proyecto para garantizar la interacción adecuada entre el sistema local y el servidor en tiempo real. El proceso de generación de la clave privada necesaria se detalla en la Figura 7.

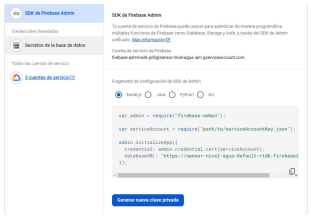

Figura 7 Generación de clave privada para Firebase Admin SDK.

Inicialmente, las reglas de acceso a la base de datos fueron configuradas para permitir cualquier conexión IP. Aunque Firebase advierte sobre los riesgos de esta configuración, al tratarse de un proyecto a pequeña escala, se decidió utilizar estas reglas provisionales, como se observa en la Figura 8, para facilitar las pruebas.

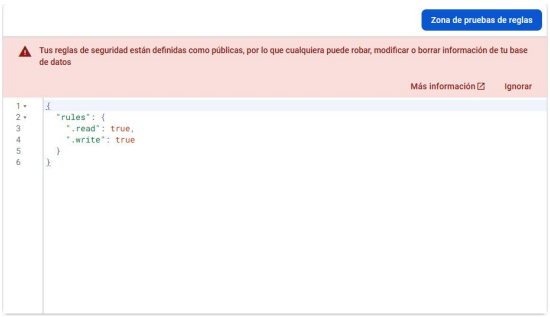

Figura 8 Reglas provisionales de acceso a la base de datos.

Para la fase final, se definieron reglas más estrictas que permiten únicamente la lectura y escritura de datos correspondientes a las identificaciones de los sensores de proximidad. Estas reglas definitivas garantizan la seguridad del sistema, como se muestra en la Figura 9.

Finalmente, se identificó el nodo dentro de la base de datos en tiempo real donde se almacenarán los datos enviados por los sensores, como se indica en la Figura 10.

4.0.3. API REST

Para implementar la API en Python que conecta los datos de los sensores con Firebase, se siguieron los

Figura 9 Reglas definitivas de acceso seguro.

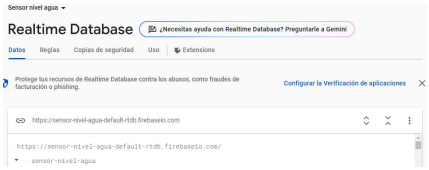

Figura 10 Identificación del nodo de almacenamiento de datos.

siguientes pasos:

1. **Inicialización de Flask y Firebase:** Se utilizó Flask como marco de trabajo para la API. Además, se configuró Firebase mediante la integración del archivo `.json` que contiene la clave generada desde el servicio de cuentas de Firebase Admin.

2. **Validación del formato de la ID del sensor:** Se definió un patrón mediante expresiones regulares (`regex`) para validar las identificaciones de los sensores (`idsensor`). Este formato asegura que las ID sigan la estructura definida para los sensores activos (por ejemplo, $\texttt{PULSIOXI}_1a$

2. 2. `idsensor: Identificador único del sensor.`

3. `ritmo cardiaco: Valor entero del ritmo cardiaco.`

4. `nivel oxigeno: Valor entero del nivel de oxigeno en sangre.`

La función valida los datos, obtiene la fecha y hora actuales, y construye un registro en forma de diccionario. Este registro se almacena en Firebase mediante el comando `push`, que genera una clave única para cada entrada.

Confirmación de operación: La función imprime un mensaje en la consola para confirmar si los datos se guardaron exitosamente en Firebase o si ocurrió algún error durante el proceso. Esto facilita la depuración y el monitoreo del funcionamiento de la API.

Visualización de datos almacenados: Se definió otra ruta (`/ver_datos`) que permite recuperar y visualizar los datos almacenados en Firebase. La información se convierte a un formato de lista para su posterior uso o visualización.

Integración con una interfaz web: Se creó una ruta principal (`/`) que carga una página HTML. Esta página servirá como interfaz gráfica para visualizar y gestionar los datos enviados por los sensores.

Finalmente, el servidor se ejecuta en modo `debug` para facilitar las pruebas y ajustes durante el desarrollo. Esta implementación asegura una comunicación robusta y en tiempo real entre los sensores simulados y la base de datos en Firebase.

Este código HTML, con estilos CSS y scripts JavaScript, implementa una interfaz gráfica interactiva para visualizar y gestionar los datos enviados por sensores de pulsioxímetros. A continuación, se detalla la estructura y funcionalidad de cada sección:

1. **Encabezado HTML (`<head>`)**:
   - Define la codificación de caracteres (`UTF-8`) y asegura la adaptabilidad a dispositivos móviles mediante la etiqueta `viewport`.
   - Incluye la biblioteca `Chart.js`, utilizada para generar gráficos interactivos de los valores de oxígeno y ritmo cardíaco.
   - Define estilos personalizados en CSS para la estructura y estética del contenido, destacando el uso de una paleta minimalista y elementos centrados.
2. **Cuerpo de la página (`<body>`)**: Contiene una estructura organizada dentro de un contenedor principal (<div class=çontainer») que incluye:
   - **Título principal:** Un encabezado centrado (`<h1>`) con el texto "Datos de Oxígeno y Ritmo Cardiaco".
   - **Sección de control:**
     - Un menú desplegable para seleccionar el sensor (`<select>`) con opciones generadas dinámicamente a partir de los datos recuperados del servidor.
     - Un campo de entrada de fecha (<input type="date») que permite filtrar datos por fecha.
     - Un botón para actualizar manualmente los datos.
   - **Gráficos interactivos:**
     - Un gráfico de líneas para los valores de oxígeno (<canvas id="grafica-oxigeno»).
     - Un gráfico de líneas para los valores de ritmo cardíaco (<canvas id="grafica-ritmo»).
   - **Tabla de datos:** Una tabla interactiva (`<table>`) que muestra los valores tabulados, con columnas para ID del sensor, oxígeno, ritmo, fecha y hora. La tabla permite ordenar y explorar los datos recuperados.
3. **JavaScript para interactividad**:
   - **Funciones principales:**
     - `cargarSensores`: Recupera los sensores disponibles del servidor y actualiza el menú desplegable.
     - `cargarDatos`: Filtra los datos en función del sensor y la fecha seleccionados, actualizando la tabla y los gráficos correspondientes.
     - `actualizarGrafica`: Genera o actualiza gráficos de línea dinámicamente utilizando `Chart.js`, mostrando valores de oxígeno o ritmo cardíaco a lo largo del tiempo.
   - **Interacción inicial:** La función `cargarSensores` se ejecuta al cargar la página, asegurando que las opciones de selección estén disponibles de inmediato.

Este código proporciona una experiencia interactiva y visual para el monitoreo de parámetros biométricos clave. Combina gráficos dinámicos, tablas de datos y opciones de filtrado para ofrecer una interfaz eficiente y atractiva para el análisis de los datos enviados por los sensores de pulsioxímetros.

4.0.4. Documentacion

La documentación del proyecto se elaboró cuidadosamente para garantizar que los usuarios y desarrolladores puedan comprender y utilizar el sistema de manera efectiva. Se generaron los siguientes archivos clave:

1. **`manual_usuario.md`:** Este archivo contiene una guía detallada para el usuario final. Incluye instrucciones paso a paso sobre cómo instalar,

configurar y operar el sistema. También describe las funcionalidades principales, como la visualización de datos y filtrado por fecha, además de soluciones a problemas comunes.

2. **`README.md`:** Este archivo actúa como una introducción al proyecto. Incluye una descripción general del propósito del sistema, las tecnologías utilizadas (como Flask, Firebase, y Chart.js) y un resumen de las características implementadas. También proporciona instrucciones rápidas para la instalación, ejecución y pruebas del sistema.

3. **`requirements.txt`:** Este archivo lista las dependencias necesarias para ejecutar el proyecto, incluyendo las versiones específicas de las bibliotecas requeridas. Permite a los usuarios instalar todas las dependencias de manera eficiente utilizando `pip` con el comando:

```
pip install -r requirements.txt
```

Estos archivos aseguran que tanto los usuarios como los desarrolladores cuenten con la información necesaria para instalar, configurar, operar y mantener el sistema, facilitando su adopción y comprensión.

5. Resultados y evaluación

El desarrollo del sistema concluyó con una página web funcional que permite la visualización de datos biométricos en tiempo real y el acceso al historial de registros de los sensores. A continuación, se presentan los resultados obtenidos:

1. **Interfaz Principal:** La página web se diseñó con un enfoque en la usabilidad y claridad. La interfaz principal, como se muestra en la Figura 11, incluye herramientas para seleccionar sensores y fechas específicas, actualizar manualmente los datos, y visualizar gráficos interactivos de oxígeno y ritmo cardíaco. Además, se incorporaron campos de filtrado para realizar consultas más precisas.

2. **Gráficas Interactivas:** La página incluye gráficos dinámicos generados con `Chart.js` para representar los niveles de oxígeno y el ritmo cardíaco de los sensores seleccionados, como se observa en la Figura 12. Estas gráficas permiten visualizar tendencias en tiempo real, destacando variaciones en los datos a lo largo del tiempo.

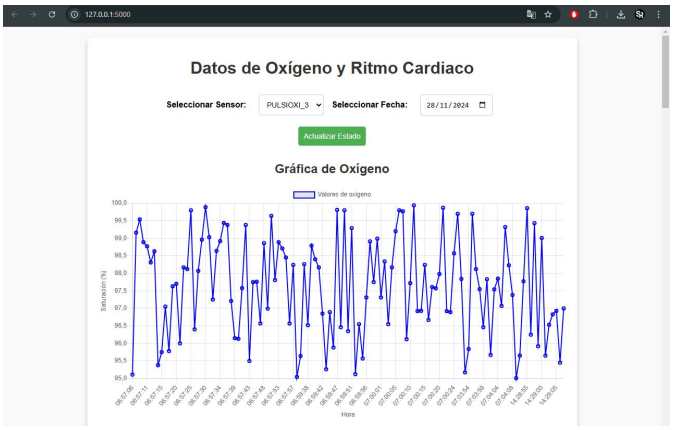

Figura 11 Página web - Interfaz principal.

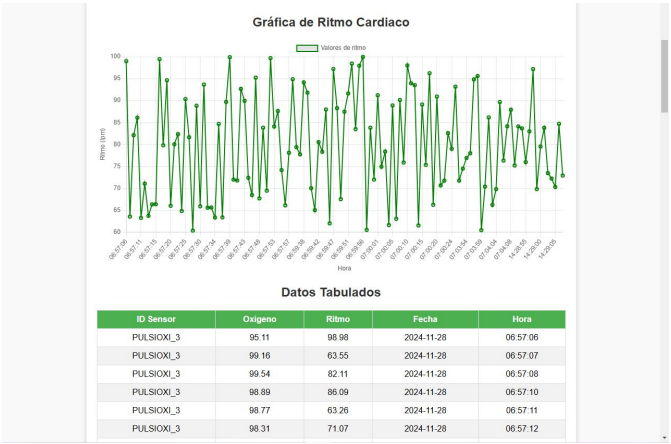

Figura 12 Página web - Gráficas de oxígeno y ritmo cardíaco.

3. **Tabla de Historial de Datos:** En la parte inferior de la página web, se incluyó una tabla interactiva que permite visualizar el historial completo de datos enviados por los sensores. Esta tabla incluye columnas para el identificador del sensor, valores de oxígeno, ritmo cardíaco, fecha y hora, como se muestra en la Figura 13. Los datos se pueden ordenar y filtrar para facilitar el análisis detallado. Ademas de graficar en matplotlib para ver los datos enviados al servidor.

4. **Consistencia de Datos:** El sistema fue probado con datos simulados enviados por los sensores. Las gráficas y la tabla permitieron verificar la consistencia y precisión de los valores generados, asegurando que los datos correspondan a parámetros biométricos realistas, como se ilustra en las gráficas dinámicas.

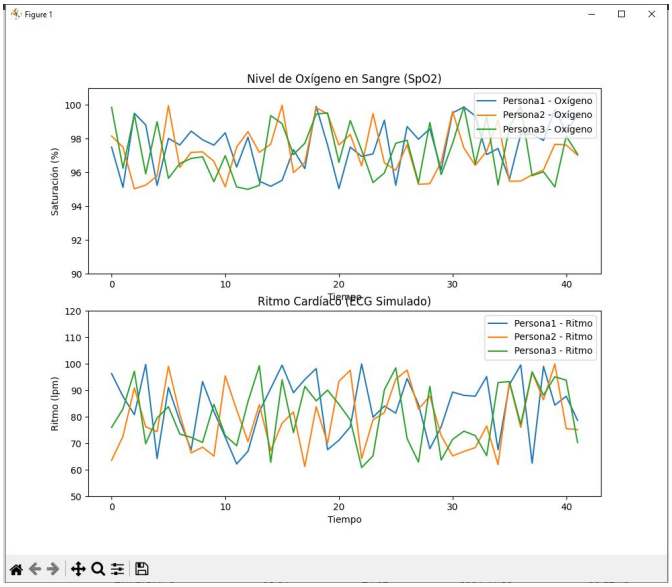

Figura 13 Página web - Tabla interactiva de datos históricos.

5. **Escalabilidad y Extensibilidad:** El sistema fue diseñado de manera modular, permitiendo la fácil incorporación de nuevos sensores o características, como la integración con bases de datos externas o análisis avanzado de datos históricos.

Estos resultados demuestran el funcionamiento correcto del sistema, desde la simulación y transmisión de datos hasta su visualización gráfica y tabular. El sistema cumple con los objetivos planteados al inicio del proyecto, proporcionando una solución eficiente y accesible para el monitoreo de parámetros biométricos clave.