UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

Proyecto: Generador de Reportes de Red con Python

Integrantes:

- Aponte Flores, Santiago Alexander (20221422J)
- Huamán Bernal, Rubi Milagros (20234510J)
- Santillan Crisante, Diogo Gabriel (20221477I)
- Velarde Espinal, José Alejandro (20221062C)

Curso:

- Programación Orientada a Objetos

Docente:

- Yury Oscar Tello Canchapoma

LIMA, PERÚ

2024

Índice

Generador de Reportes de Red con Python

1.- Introducción:

En el mundo actual, especialmente en el mundo digital, las redes de comunicación son los pilares que sustentan las relaciones entre personas, sistemas y organizaciones. Sin embargo, problemas como la alta latencia, el tiempo de inactividad inesperado y el comportamiento errático de la red pueden causar problemas que afecten la experiencia del usuario y la continuidad del negocio.

El proyecto Generador de reportes de Red con Python tiene como objetivo resolver este problema con una interfaz automatizada que permite monitorear y analizar la actividad de la red. Este programa utiliza tecnología de aprendizaje automático para detectar anomalías y generar informes visuales y datos estructurales, proporcionando un enfoque integral para mejorar el análisis y la gestión de la red. Su diseño y enfoque en la visión se puede integrar fácilmente en diferentes entornos, lo que ayuda a encontrar y resolver problemas de red correctamente.

2.- Otros proyectos similares:

El trabajo de fin de grado “RPA: Generador de informes”, del autor Alberto Gómez Rodríguez se basa en la tecnología de Automatización Robótica de Procesos (RPA), el cual busca automatizar la creación y envío de informes que, de otra manera, se realizan manualmente, de tal forma que se ahorre tiempo y se reduzca errores humanos.

Descripción general del proyecto:

Objetivo:

- Crear un robot que, utilizando información de un archivo Excel, navegue por una página web, capture datos y genere informes en formato PowerPoint. Posteriormente, estos informes son enviados por correo electrónico a los destinatarios especificados.

Actividades principales:

- Configuración: El usuario proporciona un archivo Excel con los departamentos, fechas y destinatarios para generar los informes.
- Captura de datos: El robot navega automáticamente por las secciones de un sitio web relacionado con los departamentos seleccionados, captura la información y la guarda en formato imagen.
- Generación del informe: Con la información capturada, el robot genera un informe PowerPoint donde se incluyen las imágenes y textos correspondientes.
- Envío de correos: El robot envía automáticamente los informes generados a los destinatarios especificados en el Excel.
- Manejo de errores: El sistema cuenta con mecanismos para informar de errores en la ejecución, como fallos en la captura de datos o en el envío de correos.

Herramientas y tecnologías investigadas:

- Groovy: Lenguaje de programación utilizado para el desarrollo del robot.
- Jenkins: Servidor de integración continua usado para programar y ejecutar las tareas automáticas.
- Microsoft Office (Excel y PowerPoint): Utilizados para la entrada de datos y la generación de informes.
- Selenium: Herramienta para la automatización de navegadores web.

Para diferenciar el proyecto de **Robot generador de informes RPA** con nuestro proyecto de **Generador de reportes de red con Python**, podemos analizarlos en varios aspectos clave en el siguiente cuadro comparativo:

Aspecto	Generador de Reportes de Red con Python	Proyecto de Referencia (“RPA: Generador de Informes)
Lenguaje de programación	-Python	-Groovy
Principales tecnologías	-Flask (Interfaz web) -Matplotlib (Visualización de datos) -Scikit-learn (Isolation Forest, aprendizaje automático)	- Jenkins (integración continua). - Selenium (automatización de navegadores web).

	-Subprocess (Comandos del sistema)	- Microsoft Office (Excel y PowerPoint

Origen de datos	- Datos generados en tiempo real mediante pings a un host de red (latencias).	- Archivo Excel proporcionado por el usuario (con departamentos, fechas y destinatarios). - Información capturada automáticamente de un sitio web.
Procesamiento	- Análisis de datos de red (latencias) para detectar anomalías usando aprendizaje automático (Isolation Forest). - Cálculo de ancho de banda a partir de latencias.	- Manipulación de datos provenientes del archivo Excel y la web para formatearlos en un informe PowerPoint. - No emplea aprendizaje automático.
Resultados generales	- Gráficos de latencias y ancho de banda en formato PNG. - Datos tabulados en CSV (latencias, ancho de banda, anomalías).	- Informes en formato PowerPoint, con imágenes y textos capturados de la web. - Informes enviados automáticamente por correo electrónico.
Automatización	- Automatiza la recolección y análisis de datos de red. - Genera gráficos y CSV automáticamente. - No incluye envío automatizado de resultados.	- Automatiza todo el flujo: captura de datos web, generación de informes PowerPoint y envío por correo electrónico. - Incluye manejo de errores en el flujo.
Alcance del proyecto	- Medición y análisis de rendimiento de red mediante latencias y ancho de banda. - Detección de anomalías en los datos de red.	- Automatización de flujo de trabajo: captura de datos web, generación de informes visuales y distribución automática.
Enfoque principal	- Medición de rendimiento de red. - Análisis técnico con gráficos y detección de anomalías.	- Automatización de procesos para creación y envío de informes administrativos.

Por otro lado, nuestro proyecto tiene diferencias clave respecto al proyecto de **"RPA para la automatización de la gestión administrativa en el área de finanzas de Seidor",** de los autores Balladares, Salinas y Godoy (2020). Mientras que este último se centra en la automatización de tareas repetitivas dentro del departamento financiero de la empresa Seidor, nuestro enfoque está orientado a la automatización de la gestión de redes y la generación de reportes usando Python y RPA.

Diferencias principales:

Ámbito de aplicación: El proyecto de Seidor se enfoca en la gestión administrativa y financiera, optimizando tareas como la creación de contratos, cuentas por cobrar, y programación de pagos. En cambio, nuestro proyecto se centra en la gestión de redes informáticas, automatizando la generación de reportes que permitan un monitoreo y análisis constante del estado de una red.

Herramientas: Mientras que Seidor utiliza UiPath para implementar su automatización, nosotros utilizaremos Python junto con herramientas y bibliotecas como “matplotlib”, “numpy”. “pandas”, “scikit-learn”, entre otras, las cuales están mejor adaptadas para interactuar con datos relacionados a redes y la creación de reportes automatizados empleando IA.

Objetivos específicos: El objetivo de Seidor es optimizar el tiempo y reducir los errores humanos en tareas repetitivas dentro del área financiera. Por nuestra parte, es automatizar la recolección, procesamiento y análisis de datos de red para generar reportes precisos y en tiempo real, mejorando así la capacidad de gestión de redes.

Entorno de aplicación: Nuestro proyecto está más enfocado en el ámbito técnico y tecnológico, donde la automatización permitirá una mejor supervisión del estado de las redes y la resolución de problemas más rápida. En cambio, el proyecto de Seidor busca una mejora operativa en un contexto empresarial, optimizando procesos internos administrativos.

3.- Objetivos:

El objetivo principal de este proyecto es brindar soluciones para gestionar la red.

3.1.- Objetivo general:

- Medir y analizar el rendimiento de la red en tiempo real utilizando métricas clave como la latencia y el ancho de banda.
- Automatizar el proceso de recopilación y análisis de datos de red para reducir la intervención manual y aumentar la eficiencia.

3.2.- Objetivos específicos:

- Utilizar el comando ping para realizar una sesión de prueba para probar la latencia del host.
- Utilizar técnicas de aprendizaje automático (Isolation Forest) para analizar datos de latencia para evaluar el rendimiento de la red.
- Ver los resultados en gráficos claros y concisos que representan el tiempo y el ancho de banda a lo largo del tiempo y son fáciles de entender.
- Generar los informes de registro en un formato CSV que contengan métricas, longitud, ancho de banda e y detección de las anomalías
- Proporcionar un buen sitio web financiero donde los usuarios pueden completar y realizar un seguimiento de las pruebas.
- Utilizar el historial y las imágenes registradas para resolver problemas, detectar incidentes y tomar medidas. Los problemas de red se pueden identificar fácilmente con información clara del sistema.

4.- Cronograma:

En el cronograma inicial presentamos las actividades que iríamos desarrollando semana tras semana con el fin de avanzar progresivamente con nuestro proyecto, pero a la par que fueron pasando las semanas, nuestro cronograma también fue cambiando, quedando de la siguiente forma:

Semana 6: Revisión de diagrama de clases del proyecto

Al principio del proyecto creamos un diagrama UML para darnos una idea de cómo sería nuestro proyecto, en la semana 6 nos dimos cuenta de que nuestro UML no estaba orientado a

la automatización por lo que hicimos una revisión y agregamos la clase “BotRPA” siendo al final la parte central de nuestro UML final.

Semana 7: Desarrollo del módulo de recolección de datos

En la semana 7 nos propusimos a desarrollar un módulo de recolección el cual se encargaría de juntar los datos necesarios para tener las mediciones de Latencia y Tiempo de Espera que eran las dos mediciones que elegimos inicialmente que tendría que mostrar nuestro programa.

Semana 9: Desarrollo del módulo de procesamiento de datos

Hasta la semana 7 ya teníamos los datos recolectados, pero no había forma de que el usuario las visualice, por lo que en la semana 9 desarrollamos el módulo de procesamiento de los datos recolectados, el cual al final mostraba una gráfica para la Latencia y una gráfica para el tiempo de espera y además nos mostraba un archivo CSV donde se lograba visualizar de forma más precisa las mediciones realizadas.

Semana 10: Despliegue del programa a sitio web

Para seguir avanzando nuestro proyecto nos vimos en la necesidad de usar Flask para crear una página web por lo que recurrimos a Python Anywhere que nos daba esa facilidad de una página web y también desarrollamos lo que sería la interfaz que visualizaría el usuario al entrar.

Semana 11: Implementación de la IA para detectar anomalías

En esta semana implementamos la IA para la detección de anomalías de nuestras mediciones con la ayuda del Isolation Forest y también una explicación del porque se daban estas anomalías y algunas recomendaciones.

Semana 12: Optimización del código y mejora del Frontend

Hasta la semana 11 nuestro proyecto iba por buen camino, pero encontramos un problema muy grande el cual era que nuestra página demoraba 10 minutos aproximadamente en tomar las mediciones por lo que en la semana 12 optimizamos el código eliminando hilos innecesarios y haciendo que las mediciones se hagan de manera simultánea.

Esta optimización nos dio un resultado muy grato, ya que nuestro código paso de hacer las mediciones en 10 minutos a menos de 30 segundos y como ultimo avance de la semana mejoramos el aspecto de nuestra interfaz para que se vea más llamativa para el usuario

Semana 13: Pruebas Automatizadas

En la semana 13 decidimos cambiar la medición de Tiempo de Espera por la de Ancho de banda. Y nos pasamos de usar Python Anywhere a usar VS Code.

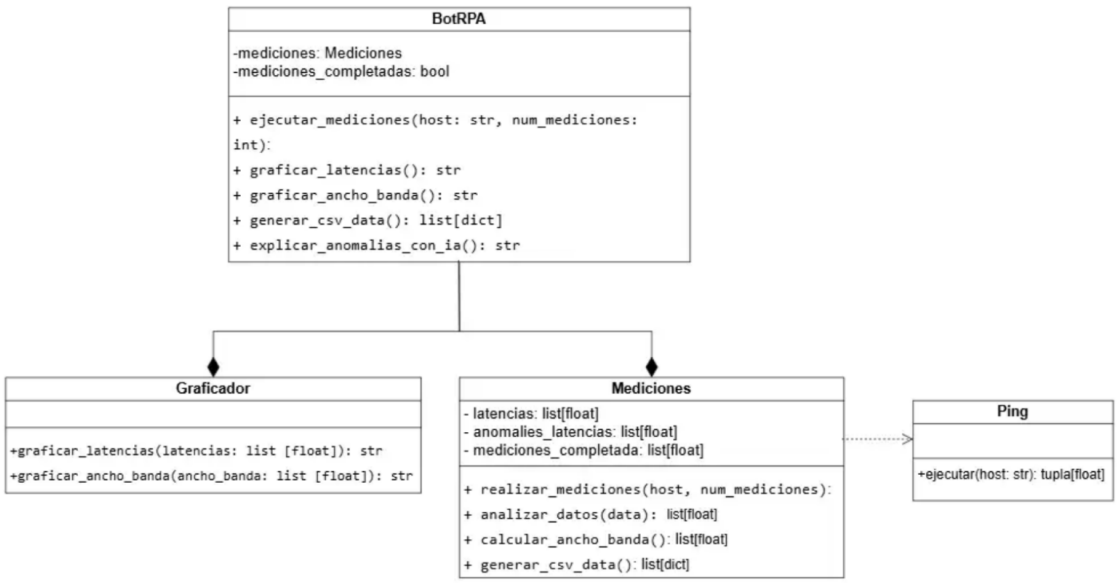
También nos encargamos de realizar las pruebas automatizadas con la ayuda del UNITTEST el cual nos ayudó a corroborar que nuestro proyecto estaba funcionando como debía.

Semana 14: Documentación y entrega

En la última semana nos encargamos de la documentación en el cual mostraríamos la fase final de nuestro proyecto, con el diagrama UML, las diferentes librerías que utilizamos y por último a las conclusiones que llegamos.

5.- Diseño del proyecto:

5.1.- Diagrama UML:



5.2- Explicación del Diagrama

Un diagrama UML es una herramienta gráfica que muestra la organización de los principales componentes de un sistema. Muestra las partes más importantes en cuanto a sus características, procesos y conexiones entre ellas.

5.2.1. Clase Ping

Rol: Ejecuta un comando ping para obtener la latencia desde un host.

- **Métodos:**
  - **+ ejecutar(host: str) -> tuple[float | None, None]**
    Realiza un ping al host especificado y devuelve un tuple con la latencia obtenida (en milisegundos) o None en caso de error.

5.2.2 Clase Mediciones

Rol: Gestiona la obtención de latencias y analiza los datos resultantes.

- **Atributos:**
  - **latencias: list[float]**
    Lista que almacena las latencias medidas.
  - **anomalias_modelo: list[float]**
    Lista que almacena anomalías detectadas mediante un modelo de aprendizaje automático.
  - **anomalias_umbral: list[float]**
    Lista de anomalías identificadas mediante un umbral predefinido.
- **Métodos:**
  - **realizar_mediciones(host: str, num_mediciones: int) -> None**
    Realiza múltiples pings al host especificado y almacena las latencias.
  - **analizar_datos() -> tuple[list[float], list[float]]**
    Detecta anomalías en los datos de latencia utilizando modelos de ML (como Isolation Forest) y umbrales.
  - **calcular_ancho_banda() -> list[float]**
    Calcula el ancho de banda a partir de las latencias obtenidas.
  - **generar_csv_data() -> list[dict]**
    Genera los datos en formato estructurado para exportar como CSV.

5.2.3. Clase Graficador

Rol: Genera gráficos de latencias y anchos de banda.

- **Métodos:**
    - **graficar_latencias(latencias: list[float]) -> str | None**
      Genera un gráfico de las latencias y lo devuelve en formato Base64.
    - **graficar_ancho_banda(ancho_banda: list[float]) -> str | None**
      Genera un gráfico del ancho de banda y lo devuelve en formato Base64.

5.2.4. Clase BotRPA

Rol: Orquesta la interacción entre las clases Mediciones y Graficador para ejecutar mediciones, generar gráficos y exportar datos.

- **Atributos:**
    - **mediciones: Mediciones**
      Instancia de la clase Mediciones que gestiona el flujo de datos.
    - **mediciones_completadas: bool**
      Indicador del estado de las mediciones.
- **Métodos:**
    - **ejecutar_mediciones(host: str, num_mediciones: int) -> None**
      Llama al método realizar_mediciones de la clase Mediciones.
    - **graficar_latencias() -> str | None**
      Genera el gráfico de latencias utilizando Graficador.
    - **graficar_ancho_banda() -> str | None**
      Genera el gráfico del ancho de banda utilizando Graficador.
    - **generar_csv_data() -> list[dict]**
      Obtiene los datos en formato estructurado desde Mediciones.
    - **explicar_anomalias_con_ia() -> str**
      Explica las anomalías detectadas utilizando IA.

5.2.5 Relaciones Generales

En este caso, la relación sería la siguiente:

1. **BotRPA y Graficador (Composición)**:
    - La relación entre **BotRPA** y **Graficador** es de **composición**, como indica el rombo negro. Esto significa que **BotRPA** tiene a **Graficador** como parte de su estructura y controla su ciclo de vida. Si **BotRPA** es destruido, **Graficador** también lo será.
2. **Mediciones y Graficador (Composición)**:
    - Similarmente, la relación entre **Mediciones** y **Graficador** también es de **composición**. Esto implica que **Mediciones** contiene a **Graficador** y también controla su ciclo de vida. Si **Mediciones** se destruye, **Graficador** también se destruirá.
3. **Mediciones y Ping (Asociación)**:
    - La relación entre **Mediciones** y **Ping** es una **asociación**, indicada por la flecha punteada. **Mediciones** depende de **Ping** para ejecutar la acción de medición (latencia), pero no tiene una relación fuerte con **Ping** en términos de ciclo de vida, es solo una dependencia en el uso de sus métodos.

5.3 Integración con Flask

La integración con Flask se basa en un sistema que permite a los usuarios interactuar con el proceso a través de solicitudes HTTP. Esto muestra que Flask funciona como un gancho, permitiendo que el proyecto se convierta en una única API. Por lo tanto, el método se puede utilizar fácilmente mediante un navegador o una solicitud remota.

5.3.1 Rutas en Flask (@app.route)

Las rutas definidas en Flask permiten la interacción con el sistema. Las siguientes rutas están implementadas:

1. **Ruta /**: Esta ruta carga la página principal del sistema donde el usuario puede visualizar y realizar mediciones de latencia.
2. **Ruta /iniciar_mediciones**: Inicia las mediciones de latencia y ancho de banda al hacer ping a un host determinado (por ejemplo, 8.8.8.8) durante un número específico de veces (15 en el ejemplo).

3. **Ruta /obtener_resultados**: Esta ruta devuelve los resultados de las mediciones, incluyendo las imágenes de los gráficos de latencia y ancho de banda, así como los datos en formato CSV.
4. **Ruta /explicar_anomalias**: Solicita a la API de OpenAI una explicación sobre las anomalías detectadas en las mediciones y su impacto en el rendimiento de la red.

5.3.2 Flujo General del Código

1. **Inicio de mediciones**:
    - Cuando el usuario accede a la ruta /iniciar_mediciones, la clase **BotRPA** comienza a realizar mediciones de latencia hacia el host indicado.
    - La clase almacena las latencias y calcula las anomalías.

2. **Obtención de resultados**:
    - A través de la ruta /obtener_resultados, los resultados de las mediciones se envían al cliente. Esto incluye los gráficos generados y los datos tabulados.
3. **Explicación de anomalías**:
    - Al acceder a la ruta /explicar_anomalias, el sistema se comunica con la API de OpenAI para generar una explicación detallada sobre el impacto de las anomalías en la red, lo cual se presenta al usuario.

6.- Justificación de las Decisiones de Diseño

- **Uso del paradigma POO (Programación Orientada a Objetos):** El código está estructurado de manera modular mediante clases y métodos independientes. Esto facilita la reutilización y extensión del código de manera eficiente. Al incorporar el modelo IsolationForest y la integración de OpenAI, el sistema se mantiene flexible, permitiendo agregar nuevas funcionalidades, como nuevas métricas o algoritmos de análisis sin alterar el flujo principal del programa. Esta modularidad es clave para escalar el sistema o adaptarlo a diferentes tipos de monitoreo.

Uso de Bibliotecas:

1. **Flask==3.1.0:**

    - Flask sigue siendo el microframework elegido para la construcción de la aplicación web. Su flexibilidad y simplicidad permiten crear aplicaciones rápidas y ligeras que permiten interactuar con el sistema de monitoreo en tiempo real y con la integración de OpenAI para explicar las anomalías detectadas. Flask es la opción ideal para aplicaciones pequeñas a medianas que requieren desarrollo rápido y bajo consumo de recursos (Pallets Projects, 2024).

2. **Scikit-learn==1.5.2:**
    - La integración del modelo IsolationForest para la detección de anomalías es una herramienta robusta en el ámbito del aprendizaje automático. Esta biblioteca es esencial para el análisis predictivo y clasificación, permitiendo identificar patrones inusuales en las mediciones de latencia y tiempos de espera de la red. Su facilidad de integración con otras bibliotecas de Python y su eficiencia hacen de scikit-learn una herramienta clave para el análisis de datos en el sistema (Scikit-learn Developers, 2023).
3. **OpenAI==0.28.0:**
    - La biblioteca Python de OpenAI proporciona un acceso cómodo a la API de OpenAI desde aplicaciones escritas en el lenguaje Python. Incluye un conjunto predefinido de clases para recursos de API que se inicializan de forma dinámica a partir de respuestas de API, lo que la hace compatible con una amplia gama de versiones de la API de OpenAI.OpenAI. (2023)
4. **Pandas==2.2.3:**
    - Pandas sigue siendo la biblioteca fundamental para la manipulación de datos, especialmente para manejar los resultados de las mediciones en formato de DataFrame. Esta herramienta facilita las operaciones sobre los datos de red, como la limpieza, filtrado y agregación de mediciones, así como la exportación de resultados en formato CSV (The pandas development team, 2023).
5. **NumPy==2.0.2:**
    - Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, incluidas operaciones matemáticas, lógicas, manipulación de formas,

ordenamiento, selección, E/S, transformadas de Fourier discretas, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más. (Harris et al., 2023).

6. **Matplotlib==3.9.2:**
    - Matplotlib sigue siendo la herramienta principal para la visualización de datos. La generación de gráficos en tiempo real permite al usuario observar el comportamiento de la red a medida que se realizan las mediciones, lo que es crucial para la toma de decisiones y la resolución de problemas (Matplotlib Developers, 2024).

Formato JSON y Base64:

- **JSON (JavaScript Object Notation):** JSON continúa siendo utilizado como el formato para intercambiar datos entre el servidor y el cliente. Su uso facilita la transmisión eficiente de datos en la aplicación web y es ampliamente utilizado en las tecnologías actuales para garantizar compatibilidad entre sistemas (Bray, 2017).
- **Base64:** El formato Base64 sigue siendo crucial para la codificación de archivos, como las gráficas generadas, para que puedan ser embebidos directamente en el HTML sin necesidad de hacer múltiples peticiones al servidor. Esto mejora la eficiencia y facilita la integración de los datos visuales en la interfaz web sin recargar la página constantemente (RFC 4648, 2006).

Estas decisiones de diseño garantizan un sistema escalable, eficiente y fácilmente extensible, integrando la automatización del monitoreo en tiempo real con herramientas avanzadas de inteligencia artificial, aprendizaje automático y visualización de datos.

7.- Conclusiones

1. Este proyecto nos permite monitorear y mejorar el desarrollo a través de inteligencia artificial, no solo para monitorear el control y el tiempo de espera de manera inmediata, sino también conectar la inteligencia -inteligencia humana a través de OpenAI para brindar información detallada sobre anomalías encontradas en las mediciones. Esto agrega un nivel de comprensión que puede ser útil para mejorar la toma de decisiones y el análisis de redes, destacando la importancia de utilizar IA para interpretar resultados de manera más accesible.

2. Mejora en la detección de anomalías: A través del modelo IsolationForest y la implementación de un umbral adicional para latencias superiores a 0.2 segundos, el código detecta de manera más precisa las anomalías en las mediciones. Lo cual nos permite identificar problemas en tiempo real con mayor confiabilidad y flexibilidad, lo que es importante para el monitoreo continuo de redes y sistemas.

3. Desarrollo de visualización de datos: crear y exportar gráficos en formato base64 para integrarlos en aplicaciones de red es un aspecto importante que desarrolla la visualización de resultados para mostrar gráficos de latencia y ancho de banda en un espacio que sea fácil de entender la estructura y el comportamiento de la red, lo cual es importante para velocidad.

4. Integración de Flask y capacidades de colaboración avanzadas: Flask admite la visualización y medición de datos y le permite integrar inteligencia artificial para obtener información única. La combinación de Flask y Python le permite crear aplicaciones web confiables y dinámicas, ideales para entornos de control en tiempo real.

5. Generación y exportación de datos mejorada: Este código optimiza la exportación de los resultados al CSV, que ahora incluye tanto las anomalías detectadas por el modelo como las que superan el umbral de latencia. Esto ofrece un análisis más completo de las mediciones, y el uso del formato CSV permite realizar un análisis detallado o almacenamiento a largo plazo de los datos, lo que es útil para futuras revisiones o investigaciones.

8.- Referencias

- Balladares C. (2020). *RPA para la automatización de la gestión administrativa en el área de finanzas de Seidor* [Trabajo de investigación, Universidad Científica del Sur]. Repositorio Institucional Científica. Disponible en: https://repositorio.cientifica.edu.pe/bitstream/handle/20.500.12805/1710/TB-Balladares%20C-et%20al-Ext.pdf?sequence=2&isAllowed=y
- Gómez Rodríguez, A. (2020). *RPA: Generador de informes* [Trabajo de fin de grado, Universidad de Cantabria]. Repositorio Institucional de la Universidad de Cantabria. Disponible en: https://repositorio.unican.es/xmlui/bitstream/handle/10902/20929/Gomez%20Rodriguez%20Alberto.pdf?sequence=1&isAllowed=y
- Bray, T. (2017). *The JavaScript Object Notation (JSON) data interchange format*. https://www.ietf.org/rfc/rfc8259.txt
- RFC 4648. (2006). *Base 64 encoding and decoding*. https://tools.ietf.org/html/rfc4648
- Matplotlib Developers. (2024). *matplotlib 3.9.2 documentation*. https://matplotlib.org/stable/contents.html
- NumPy Developers (2024). *numpy 2.0.2 documentation*. https://numpy.org/doc/2.0/index.html
- Pallets Projects. (2024). *Flask 3.1.0 documentation*. https://flask.palletsprojects.com/en/2.3.x/
- Pandas development team. (2023). *pandas 2.2.3 documentation*. https://pandas.pydata.org/pandas-docs/stable/
- Scikit-learn Developers. (2023). *scikit-learn 1.5.2 documentation*. https://scikit-learn.org/stable/
- OpenAI. (2023). *openai (Versión 0.28.0)* [Software]. https://pypi.org/project/openai/0.28.0/