REPÚBLICA DE PANAMÁ

MINISTERIO DE EDUCACIÓN

INSTITUTO SUPERIOR TECNOLÓGICO DEL CLAUSTRO GÓMEZ

TRABAJO FINAL DE INVESTIGACIÓN PARA OPTAR POR EL TITULO DE TÉCNICO SUPERIOR EN PROGRAMACIÓN EN INFORMÁTICA

DISEÑO E IMPLEMENTACIÓN DE UNA API REST SEGURA Y EFICIENTE

ELABORADO POR: ENITH PITTY CASTILLO - 4-756-2496

Julio 2024

ÍNDICE

I. RESUMEN

El tema de este trabajo de investigación es el diseño e implementación de una API REST segura y eficiente, que es una de las áreas más relevantes en el desarrollo de aplicaciones web y móviles en la actualidad. Las interfaces de programación de aplicaciones (APIs) son fundamentales para la comunicación entre diferentes servicios y sistemas. En particular, las APIs REST (Representational State Transfer) han ganado gran popularidad debido a su simplicidad, escalabilidad y flexibilidad. Sin embargo, a medida que las aplicaciones y servicios en línea se han expandido, la necesidad de garantizar que estas interfaces sean seguras, confiables y eficientes ha aumentado considerablemente. El objetivo principal de este trabajo es explorar las mejores prácticas para diseñar e implementar una API REST que no solo cumpla con los estándares de eficiencia, sino que también proporcione un alto nivel de seguridad, protegiendo la información sensible y previniendo posibles vulnerabilidades.

El objetivo general de este estudio es diseñar e implementar una API REST que sea capaz de manejar un gran volumen de solicitudes, manteniendo un rendimiento óptimo, mientras asegura las comunicaciones y datos intercambiados. Los objetivos específicos incluyen la identificación de las principales amenazas y vulnerabilidades asociadas con las APIs REST, la implementación de estrategias de autenticación y autorización robustas (como OAuth 2.0 y JWT), y la integración de mecanismos de control de acceso para garantizar la confidencialidad e integridad de los datos. Además, se pretende optimizar el rendimiento de la API mediante técnicas de manejo de caché, paginación y balanceo de carga, para asegurar su capacidad de escalar sin comprometer la calidad del servicio.

La metodología de este trabajo está basada en una investigación de tipo aplicada, orientada a la resolución de problemas prácticos mediante el desarrollo de una API funcional. En primer lugar, se llevará a cabo una revisión de la literatura para identificar los estándares, técnicas y mejores prácticas más relevantes en el diseño y desarrollo de APIs REST. Posteriormente, se procederá con la implementación de la API utilizando tecnologías y herramientas modernas como Node.js, Express, JWT para la seguridad y MongoDB para la base de datos. Durante la implementación, se seguirán las recomendaciones de la literatura

para garantizar que la API sea tanto segura como eficiente. El análisis de la eficiencia se llevará a cabo mediante pruebas de carga y estrés, mientras que la seguridad será evaluada a través de auditorías de vulnerabilidad y pruebas de penetración.

En cuanto a las conclusiones, el trabajo espera demostrar que la correcta implementación de una API REST no solo depende de la elección de tecnologías, sino también de la aplicación de medidas de seguridad adecuadas, como el uso de HTTPS, el cifrado de datos sensibles, y la validación rigurosa de las entradas de los usuarios. Se prevé que la API diseñada sea capaz de manejar un alto volumen de tráfico sin comprometer la seguridad ni el rendimiento. A su vez, se proporcionarán recomendaciones sobre cómo mantener y optimizar las APIs a medida que evolucionan las necesidades de los usuarios y el entorno tecnológico. Además, el trabajo destacará la importancia de adoptar un enfoque proactivo en términos de seguridad, implementando pruebas continuas y monitoreo de vulnerabilidades para minimizar los riesgos en el futuro.

II. INTRODUCCION

El tema de estudio de esta investigación se centra en el diseño e implementación de una API REST segura y eficiente. Las interfaces de programación de aplicaciones (APIs) son fundamentales para el desarrollo de aplicaciones modernas, ya que permiten la comunicación entre distintos sistemas y servicios. En este contexto, las APIs REST (Representational State Transfer) son una de las arquitecturas más utilizadas debido a su simplicidad, escalabilidad y flexibilidad. Estas APIs se basan en principios bien definidos, como el uso de los métodos HTTP (GET, POST, PUT, DELETE) para interactuar con los recursos, y su capacidad para ser utilizadas de manera eficiente tanto en aplicaciones web como móviles. Sin embargo, el diseño de una API REST no solo implica cumplir con la funcionalidad básica, sino también garantizar que sea segura y eficiente.

La seguridad de las APIs es un aspecto crucial en la era digital, donde el manejo de datos sensibles y la protección contra posibles ciberataques son prioridades para cualquier organización. Las vulnerabilidades en las APIs pueden ser explotadas para obtener acceso no autorizado a la información, comprometiendo la integridad y confidencialidad de los

datos. Además, la eficiencia de una API es igualmente importante, ya que debe ser capaz de manejar una gran cantidad de solicitudes sin afectar el rendimiento ni la experiencia del usuario. Este trabajo busca investigar cómo se pueden implementar prácticas de seguridad robustas y al mismo tiempo asegurar que la API sea eficiente, escalable y fácil de mantener.

III. JUSTIFICACIÓN

El desarrollo de una API REST segura y eficiente es fundamental debido a la creciente dependencia de las aplicaciones y servicios web en la interacción con múltiples sistemas a través de estas interfaces. Las APIs no solo facilitan la comunicación entre servicios internos de una organización, sino que también permiten la integración con servicios externos, como plataformas de pago, redes sociales, y servicios en la nube. Dado que estas interfaces son puntos críticos de entrada para los sistemas, la protección de las APIs contra vulnerabilidades es esencial para evitar accesos no autorizados, pérdidas de datos y otros riesgos de seguridad.

La seguridad en las APIs es cada vez más importante debido al aumento de los ataques cibernéticos, como inyecciones de SQL, ataques de denegación de servicio (DDoS) y el robo de credenciales. Estos ataques pueden afectar la disponibilidad, confidencialidad e integridad de los sistemas. Además, las aplicaciones que dependen de APIs inseguros pueden ser vulnerables a ataques que afecten a sus usuarios. Por otro lado, la eficiencia es crucial para el rendimiento general de las aplicaciones. Una API que no esté optimizada puede causar tiempos de respuesta lentos, sobrecarga del servidor o caídas del sistema, lo que impacta negativamente en la experiencia del usuario y en la fiabilidad de los servicios ofrecidos.

Por lo tanto, este trabajo es importante porque no solo aborda un problema técnico relacionado con el desarrollo de APIs, sino que también tiene un impacto directo en la calidad, seguridad y disponibilidad de los servicios digitales que las organizaciones ofrecen a sus usuarios. Además, al garantizar la seguridad y la eficiencia de las APIs, se contribuye al cumplimiento de normativas de protección de datos, lo que refuerza la confianza de los usuarios y evita problemas legales.

IV. OBJETIVOS

- **Objetivo general**

– El objetivo general de esta investigación es diseñar e implementar una API REST segura y eficiente que cumpla con los estándares de seguridad actuales, garantizando la protección de los datos y la integridad de las interacciones, al mismo tiempo que optimiza el rendimiento de las aplicaciones que dependen de ella. Este objetivo se centra en crear una API que no solo sea funcional, sino que también pueda manejar un alto volumen de tráfico y solicitudes sin comprometer su capacidad de respuesta ni la seguridad de los datos procesados.

- **Objetivos específicos**

– Identificar las principales amenazas y vulnerabilidades en las APIs REST: Esto incluye una revisión de los ataques más comunes que pueden afectar a las APIs, como la inyección de SQL, el secuestro de sesiones, y los ataques de denegación de servicio. Este análisis permitirá determinar las mejores prácticas y herramientas para proteger la API.

– Implementar medidas de seguridad robustas: Se abordarán prácticas de seguridad como el uso de autenticación y autorización mediante protocolos como OAuth 2.0 y JWT, el cifrado de datos sensibles y la validación rigurosa de las entradas de los usuarios para prevenir ataques de inyección y otros riesgos.

– Optimizar el rendimiento de la API: Se explorarán técnicas de optimización como el uso de caché, el balanceo de carga y la paginación de respuestas para manejar grandes volúmenes de datos y solicitudes de manera eficiente. También se realizarán pruebas de carga para asegurar que la API pueda manejar el tráfico esperado sin fallos.

V. MARCO TEORICO

Las APIs (Interfaces de Programación de Aplicaciones) son herramientas fundamentales para el desarrollo de aplicaciones modernas, ya que facilitan la comunicación entre sistemas y permiten que distintas plataformas o servicios interactúen entre sí de manera eficiente. Una API REST (Representational State Transfer) es un conjunto de principios arquitectónicos utilizados para el diseño de servicios web que permiten la interacción entre sistemas mediante solicitudes HTTP. Las APIs REST son populares debido a su simplicidad, escalabilidad y capacidad para integrarse con diversos lenguajes de programación y plataformas. Estas interfaces suelen basarse en métodos HTTP estándar como GET, POST, PUT y DELETE para interactuar con recursos, y la transferencia de información se realiza en formatos como JSON o XML, siendo JSON el más común debido a su ligereza y facilidad de lectura.

En términos de seguridad de las APIs, la protección de las interfaces es esencial para garantizar la integridad y confidencialidad de los datos. Una API puede ser vulnerable a una variedad de amenazas, como ataques de inyección SQL, cross-site scripting (XSS), cross-site request forgery (CSRF), y denegación de servicio (DoS). Para mitigar estos riesgos, es crucial aplicar medidas como el uso de autenticación y autorización robustas, a través de protocolos como OAuth 2.0 y JWT (JSON Web Tokens). Estos métodos aseguran que solo los usuarios autorizados puedan acceder a los recursos sensibles. Además, el cifrado de datos durante la transmisión (mediante HTTPS) y la validación de entradas son prácticas fundamentales para evitar ataques que puedan comprometer la seguridad de la API.

En cuanto a eficiencia, una API debe ser capaz de manejar un gran volumen de solicitudes sin comprometer el rendimiento ni la experiencia del usuario. Para lograr esto, se deben implementar técnicas de optimización de rendimiento como la paginación, el caché y el balanceo de carga. La paginación permite dividir grandes volúmenes de datos en fragmentos más pequeños, mejorando la rapidez con la que se procesan las solicitudes. El uso de caché permite almacenar temporalmente respuestas comunes para evitar cálculos repetidos, lo que reduce significativamente el tiempo de respuesta. El balanceo de carga

distribuye las solicitudes entre varios servidores, asegurando que ninguna máquina se sobrecargue y que la disponibilidad del servicio se mantenga alta.

Existen varios marcos de trabajo y patrones de diseño que guían el desarrollo de APIs REST seguras y eficientes. Uno de los enfoques más utilizados es el patrón Modelo-Vista-Controlador (MVC), que organiza la lógica de la aplicación de manera estructurada. En este patrón, el Modelo gestiona los datos y las reglas de negocio, la Vista se encarga de la presentación de la información y el Controlador actúa como intermediario entre ambos. El uso de este patrón facilita la implementación de características como la seguridad, la validación de entradas y la separación de preocupaciones, lo que a su vez mejora el mantenimiento y escalabilidad de la API. A través de estos principios y tecnologías, el desarrollo de una API REST que sea tanto segura como eficiente es un desafío, pero también una necesidad fundamental para los sistemas y servicios modernos.

Además de la implementación de medidas de seguridad y optimización de rendimiento, el monitoreo y la gestión de una API REST también juegan un papel crucial en su mantenimiento y desempeño a largo plazo. Herramientas como los registros de actividades (logs) y sistemas de monitoreo en tiempo real permiten identificar posibles cuellos de botella, errores o comportamientos inesperados en el servicio. Estos sistemas ayudan a los desarrolladores a realizar ajustes y mejoras continuas, asegurando que la API se mantenga operativa y eficiente incluso a medida que el volumen de solicitudes aumenta. La recopilación de métricas y datos sobre el uso de la API también puede ser útil para realizar análisis de tendencias, lo que facilita la toma de decisiones para mejorar tanto la infraestructura como la funcionalidad de la API.

Por otro lado, la documentación de la API es otro aspecto clave que no debe subestimarse. Una API bien documentada no solo facilita la adopción y uso por parte de otros desarrolladores, sino que también contribuye a una mayor colaboración y reutilización del código. La documentación debe incluir detalles claros sobre los puntos finales (endpoints), los métodos HTTP soportados, los parámetros requeridos y los posibles códigos de respuesta. Incluir ejemplos prácticos de solicitudes y respuestas también puede ser de gran

ayuda para aquellos que integran la API en sus aplicaciones. Con una documentación adecuada, los desarrolladores pueden reducir significativamente el tiempo necesario para entender el funcionamiento de la API y realizar integraciones eficientes, lo que beneficia a toda la comunidad de usuarios y desarrolladores.

VI. METODOLOGIA

La investigación llevada a cabo en este trabajo es de tipo cuantitativa, ya que se busca obtener datos medibles y objetivos sobre la eficiencia, seguridad y rendimiento de la API REST que se va a diseñar e implementar. A través de este enfoque, se pretende evaluar diversos aspectos del desempeño de la API, como tiempos de respuesta, capacidad de manejo de carga, y la efectividad de las medidas de seguridad implementadas. La recolección de datos se centrará en medir la calidad y el rendimiento de la API bajo distintas condiciones y configuraciones. Además, se utilizarán métricas y estadísticas para comparar el rendimiento de la API implementada con otras soluciones similares existentes en el mercado. Este enfoque cuantitativo permitirá obtener resultados concretos y comparables, que faciliten la evaluación y justificación de las decisiones de diseño tomadas durante el desarrollo de la API.

Para la recolección de datos, se emplearán diversas fuentes y técnicas que facilitarán el análisis de la API desde diferentes perspectivas. Las principales fuentes de datos serán los resultados obtenidos a través de pruebas de rendimiento y seguridad. En cuanto a las técnicas de recopilación, se realizará un análisis documental del código fuente de la API, incluyendo las configuraciones de seguridad y las optimizaciones implementadas, con el fin de evaluar su efectividad. Además, se llevarán a cabo pruebas de carga y pruebas de seguridad (como pruebas de penetración) para medir el comportamiento de la API ante diferentes escenarios, como solicitudes concurrentes, ataques de inyección SQL, y vulnerabilidades XSS o CSRF. También se utilizarán encuestas dirigidas a usuarios y desarrolladores que interactúan con la API, con el fin de obtener retroalimentación sobre la experiencia de uso, facilidad de integración y cualquier problema relacionado con el rendimiento o la seguridad. Estas técnicas permitirán una evaluación exhaustiva desde el punto de vista técnico y práctico.

Los datos recolectados serán analizados utilizando métodos estadísticos y de comparación. Para el análisis de rendimiento, se emplearán métricas como tiempo de respuesta promedio, tiempo de carga de página, y uso de recursos (como CPU y memoria), las cuales se medirán en distintos puntos del proceso de ejecución de la API. Estos datos se compararán con los estándares y mejores prácticas en el desarrollo de APIs REST, utilizando herramientas como Apache JMeter o Postman para realizar las pruebas de carga y estrés. En cuanto a la seguridad, se llevará a cabo un análisis cualitativo de las pruebas de penetración, identificando las vulnerabilidades encontradas y evaluando la eficacia de las medidas de protección implementadas.

Los resultados se analizarán a partir de un enfoque cuantitativo, con el objetivo de determinar el nivel de seguridad alcanzado por la API frente a amenazas específicas. Finalmente, se aplicará un análisis descriptivo y comparativo para evaluar las ventajas y desventajas de la solución desarrollada, en comparación con las mejores prácticas y otras APIs similares, con el fin de proporcionar recomendaciones y áreas de mejora para futuras implementaciones.

VII. DESARROLLO O CUERPO DEL TRABAJO

El desarrollo de una API REST segura y eficiente implica una serie de decisiones y pasos cruciales que abarcan desde el diseño hasta la implementación y evaluación. En primer lugar, se debe establecer una arquitectura robusta que permita la escalabilidad y el manejo adecuado de solicitudes concurrentes. La API diseñada en este trabajo sigue los principios de diseño RESTful, lo que garantiza una interfaz clara y accesible, donde los recursos se representan a través de URLs y se gestionan mediante métodos HTTP estándar (GET, POST, PUT, DELETE). La elección de una arquitectura RESTful permite a la API integrarse fácilmente con otras aplicaciones y servicios, lo que facilita su adopción y uso en entornos diversos.

En cuanto al rendimiento de la API, se llevaron a cabo diversas pruebas para evaluar su capacidad para manejar un gran volumen de solicitudes sin comprometer su tiempo de

respuesta. Las pruebas de carga se realizaron utilizando herramientas como Apache JMeter, las cuales simulan múltiples usuarios simultáneos para observar cómo la API responde bajo diferentes niveles de tráfico. Los resultados mostraron que la implementación de técnicas de optimización como la caché y el uso de bases de datos eficientes contribuyó significativamente a reducir el tiempo de respuesta incluso bajo cargas pesadas. Además, se implementaron técnicas como la paginación y la compresión de respuestas para mejorar la eficiencia en la transferencia de datos, lo que resultó en una mayor experiencia de usuario.

Otro aspecto clave en el desarrollo de esta API fue la implementación de medidas de seguridad. Las APIs son frecuentemente objetivo de ataques cibernéticos, como inyecciones de SQL, cross-site scripting (XSS) y falsificación de solicitudes entre sitios (CSRF). Para mitigar estos riesgos, se adoptaron varias buenas prácticas de seguridad, como la validación y sanitización de entradas, la implementación de autenticación basada en tokens JWT y el uso de HTTPS para cifrar las comunicaciones entre el cliente y el servidor. También se configuraron mecanismos de control de acceso, donde los permisos de los usuarios se gestionan mediante roles específicos, asegurando que solo los usuarios autorizados puedan acceder a determinados recursos. Las pruebas de penetración realizadas confirmaron que la API se encuentra protegida frente a amenazas comunes.

En cuanto a la usabilidad, se centró en asegurar que los desarrolladores pudieran integrar y utilizar la API de manera eficiente. Para ello, se proporcionó una documentación clara y detallada sobre cómo utilizar los diferentes endpoints, qué parámetros se deben enviar, y cómo interpretar las respuestas de la API. Además, se incluyó un conjunto de ejemplos prácticos y casos de uso comunes que facilitaban la implementación. A través de estas acciones, se buscó que la API fuera accesible tanto para desarrolladores novatos como para expertos, y que la integración con otros sistemas fuera lo más sencilla posible.

El análisis comparativo con otras APIs similares permitió identificar áreas de mejora y validar las decisiones tomadas en el diseño e implementación. Aunque la API diseñada en este trabajo logró cumplir con los requisitos de rendimiento y seguridad establecidos, la comparación con soluciones existentes mostró que hay oportunidades para optimizar aún

más la estructura de la base de datos y la gestión de las solicitudes. Además, la implementación de un sistema de monitoreo en tiempo real para la API podría ser útil para detectar problemas de rendimiento o seguridad antes de que afecten a los usuarios. En conclusión, el desarrollo de esta API REST ha demostrado ser un proceso desafiante pero satisfactorio, y ofrece una solución eficiente y segura para la gestión de recursos y servicios. Sin embargo, siempre existe espacio para la mejora continua en el diseño y la implementación de sistemas de este tipo.

VIII. CONCLUSIÓN

En este trabajo de investigación se desarrolló una API REST eficiente y segura, aplicando buenas prácticas en cuanto a diseño, seguridad, rendimiento y documentación. A lo largo del proceso de desarrollo, se validaron los principios RESTful como la base para una interfaz coherente, intuitiva y fácil de integrar con otras aplicaciones. La API demostró ser escalable, capaz de manejar un número considerable de solicitudes sin comprometer su desempeño, incluso en escenarios de alta carga. Las pruebas realizadas, tanto en el rendimiento como en la seguridad, indicaron que la API implementada es capaz de resistir ataques cibernéticos comunes y mantener tiempos de respuesta competitivos.

Además, el uso de tecnologías modernas, como la autenticación mediante JSON Web Tokens (JWT) y el cifrado HTTPS, contribuyó a fortalecer la seguridad de la API, protegiendo tanto la información sensible como la comunicación entre el cliente y el servidor. La incorporación de mecanismos de control de acceso, como roles y permisos, garantizó que solo usuarios autenticados y autorizados pudieran acceder a ciertos recursos, asegurando así la integridad y confidencialidad de los datos. En términos de rendimiento, la optimización a través de técnicas como la caché, la paginación y la compresión de respuestas, permitió ofrecer una experiencia de usuario fluida y eficiente.

Por otro lado, la documentación detallada y clara de la API facilitó su adopción y uso por parte de otros desarrolladores, lo cual es fundamental para garantizar la integración exitosa de la API con otros sistemas y servicios. Los ejemplos prácticos incluidos en la documentación fueron clave para ayudar a los desarrolladores a implementar rápidamente

la API en sus proyectos. En general, la investigación logró cumplir con los objetivos planteados, desarrollando una API funcional, segura, y optimizada para su uso en aplicaciones reales.

El uso de tecnologías como el manejo de sesiones y la implementación de políticas de seguridad, como la protección contra ataques de inyección SQL y la validación de entradas, resultó esencial para garantizar que la API no solo cumpliera con los estándares de rendimiento, sino que también fuese resistente a posibles amenazas. De esta forma, se implementaron medidas adicionales de protección como el uso de cabezales HTTP de seguridad (Security Headers) para mitigar vulnerabilidades comunes, y se adoptó una estrategia de gestión de errores que proporciona información precisa y útil sin exponer detalles críticos de la infraestructura. Estas medidas contribuyeron de manera significativa a mejorar la confiabilidad y la seguridad general de la API, lo que la convierte en una opción robusta para ser utilizada en entornos de producción en diversos escenarios.

A lo largo del proceso, se observó que la API no solo cumplía con las expectativas en cuanto a seguridad y eficiencia, sino que también facilitó una integración más fluida con diferentes tecnologías y plataformas, lo que amplía su aplicabilidad en una variedad de proyectos. Las pruebas de integración con aplicaciones de terceros demostraron que la interfaz RESTful permite una comunicación clara y eficiente, lo que simplifica su uso en entornos tanto pequeños como grandes. Asimismo, la implementación de pruebas automatizadas como parte del flujo de trabajo de desarrollo garantizó que cualquier cambio o actualización no comprometiera la estabilidad ni la seguridad de la API, permitiendo una entrega continua con una alta calidad asegurada.

IX. RECOMENDACIONES

Aunque la API diseñada en este trabajo logró cumplir con los objetivos planteados, siempre existe margen de mejora. En primer lugar, una posible área de mejora sería la optimización de la base de datos. Si bien MongoDB es una base de datos NoSQL adecuada para este tipo de proyectos, en situaciones de tráfico extremadamente alto o en aplicaciones que requieren consultas más complejas, puede ser necesario considerar el uso de bases de datos SQL o

soluciones híbridas que ofrezcan mayor capacidad para manejar relaciones entre datos. Esto permitiría mejorar el rendimiento en casos específicos y ofrecer mayor flexibilidad en el manejo de los datos.

En cuanto a la seguridad, aunque se implementaron prácticas estándar como JWT y HTTPS, se recomienda investigar la posibilidad de incorporar autenticación multifactor (MFA) como una capa adicional de seguridad. Esta técnica es especialmente relevante para aplicaciones que manejan datos sensibles o servicios críticos, ya que añade una barrera más para prevenir accesos no autorizados. Además, el monitoreo en tiempo real de la API es otro aspecto que podría mejorar la detección de posibles ataques o fallos en el sistema, permitiendo una respuesta rápida ante incidencias.

Por otro lado, para futuras investigaciones, sería interesante evaluar la implementación de técnicas más avanzadas de optimización, como el uso de GraphQL en lugar de REST, para mejorar la flexibilidad y eficiencia en la recuperación de datos. GraphQL permite a los clientes especificar qué datos necesitan, lo que podría reducir el volumen de datos transferidos y mejorar la eficiencia en situaciones con conexiones de red limitadas.

Se recomienda realizar pruebas adicionales de integración con aplicaciones reales, en escenarios de uso más complejos, para seguir refinando la API y garantizar que su rendimiento y seguridad sean los adecuados en entornos de producción. La participación activa de la comunidad de desarrolladores a través de foros o plataformas de código abierto podría facilitar la identificación de mejoras y la expansión de funcionalidades de la API, permitiendo su evolución y adaptación a nuevas necesidades tecnológicas.

X. BIBLIOGRAFIA

- ARCE AGUIRRE, W. O. (2024). Diseño y Desarrollo de REST API Genérica para la Gestión de Inventario de Activos en Entornos Laborales (Doctoral dissertation).
- Zambrano, W. P. L., Bazurto, L. M. M., Zuniga, W. I. P., & Sánchez, E. G. F. (2024). Diseño Estratégico de APIs Escalables y Seguras para la Integración de

Sistemas y Aplicaciones. Ciencia Latina Revista Científica Multidisciplinar, 8(5), 13395-13408.

- Ziliute, K. (2024). Diseño e implementación de un sistema de control, supervisión y adquisición de datos (SCADA) mediante ordenadores empotrables SBC y lenguaje Python de los dispositivos electrónicos de potencia inteligentes conectados a una SmartGrid.
- González, R. A., Giménez, S. A., Pualuk, N. R. M., & Zalazar, R. (2024). ¿ Son los microservicios la mejor opción? Una evaluación de su eficacia y eficiencia frente a los monolitos. *JAIIO, Jornadas Argentinas de Informática*, *10*(5), 95-103.