Ciencia Latina Revista Científica Multidisciplinar, Ciudad de México, México.
ISSN 2707-2207 / ISSN 2707-2215 (en línea), septiembre-octubre 2024,
Volumen 8, Número 5.

https://doi.org/10.37811/cl_rcm.v8i5

DISEÑO ESTRATÉGICO DE APIS ESCALABLES Y SEGURAS PARA LA INTEGRACIÓN DE SISTEMAS Y APLICACIONES.

STRATEGIC DESIGN OF SCALABLE AND SECURE APIS FOR THE INTEGRATION OF SYSTEMS AND APPLICATIONS

Welington Paul Leones Zambrano
Universidad Estatal Amazónica

Liceth Monserrate Macias Bazurto
Universidad Estatal Amazónica

William Israel Pilla Zuniga
Universidad Estatal Amazónica

Edwin Gustavo Fernández Sánchez
Universidad Estatal Amazónica

DOI: https://doi.org/10.37811/cl_rcm.v8i5.14794

Diseño Estratégico de APIs Escalables y Seguras para la Integración de Sistemas y Aplicaciones.

Welington Paul Leones Zambrano[1]
wp.leonesz@uea.edu.ec
https://orcid.org/0000-0001-9756-6462
Universidad Estatal Amazónica
Provincia de Pastaza- Puyo
Ecuador

Liceth Monserrate Macias Bazurto
lm.maciasb@uea.edu.ec
https://orcid.org/0000-0002-5234-8135
Universidad Estatal Amazónica
Provincia de Pastaza- Puyo
Ecuador

William Israel Pilla Zuniga
wi.pillaz@uea.edu.ec
https://orcid.org/0009-0001-7036-6070
Universidad Estatal Amazónica
Provincia de Pastaza- Puyo
Ecuador

Edwin Gustavo Fernández Sánchez
gfernandez@uea.edu.ec
https://orcid.org/0000-0002-2613-4774
Universidad Estatal Amazónica
Provincia de Pastaza- Puyo
Ecuador

RESUMEN

El diseño estratégico de APIs escalables y seguras es esencial para la integración de sistemas y aplicaciones en el entorno digital actual. Las APIs facilitan la interoperabilidad entre diferentes sistemas, permitiendo a las organizaciones adaptarse a demandas crecientes. Sin embargo, la escalabilidad y la seguridad son dos desafíos clave en su desarrollo. La implementación de microservicios permite una mayor flexibilidad y escalabilidad, pero también introduce complejidades en su gestión, mientras que las arquitecturas monolíticas ofrecen simplicidad en aplicaciones menos dinámicas. La seguridad en las APIs es crucial para proteger los datos de accesos no autorizados y vulnerabilidades como la inyección de código y la autenticación rota. Implementar medidas como OAuth2 y JWT garantiza la protección de la información sensible. Además, el uso de controles de acceso y el cifrado de datos en tránsito y en reposo refuerzan la seguridad de las APIs. Este artículo propone una estructura estratégica que incluye la elección del entorno de ejecución, la gestión de errores, el versionamiento y el monitoreo en tiempo real. También destaca la importancia de la documentación clara para asegurar la sostenibilidad a largo plazo. Estos enfoques permiten el desarrollo de APIs que no solo cumplen con las demandas actuales, sino que también facilitan la adaptación a futuros cambios tecnológicos.

Palabras clave: Microservicios, OAuth2, JWT, APIs escalables, Seguridad en APIs.

[1] Autor principal.
Correspondencia: wp.leonesz@uea.edu.ec

Strategic Design of Scalable and Secure APIs for the Integration of Systems and Applications

ABSTRACT

The strategic design of scalable and secure APIs is essential for the integration of systems and applications in the current digital environment. APIs facilitate interoperability between different systems, allowing organizations to adapt to increasing demands. However, scalability and security are two key challenges in their development. The implementation of microservices allows for greater flexibility and scalability but also introduces complexities in their management, while monolithic architectures offer simplicity in less dynamic applications. Security in APIs is crucial to protect data from unauthorized access and vulnerabilities such as code injection and broken authentication. Implementing measures like OAuth2 and JWT ensures the protection of sensitive information. Additionally, the use of access controls and encryption of data in transit and at rest strengthens API security. This article proposes a strategic structure that includes choosing the execution environment, error management, versioning, and real-time monitoring. It also highlights the importance of clear documentation to ensure long-term sustainability. These approaches allow the development of APIs that not only meet current demands but also facilitate adaptation to future technological changes.

***Keywords*:** Microservices, OAuth2, JWT, Scalable APIs, API Security

Artículo recibido 30 octubre 2024

Aceptado para publicación: 20 noviembre 2024

INTRODUCCIÓN

La rápida evolución de la tecnología y la transformación digital han impulsado la necesidad de integrar sistemas y aplicaciones de manera eficiente, tanto en organizaciones como en instituciones. Un componente esencial en esta transformación es el uso de Interfaces de Programación de Aplicaciones (APIs), que permiten la interoperabilidad entre diferentes sistemas o capas de una misma aplicación, facilitando la comunicación entre aplicaciones sin importar el lenguaje de programación o la plataforma utilizada. La creciente demanda de APIs ha dado lugar a la proliferación de servicios y aplicaciones interconectadas, creando ecosistemas complejos que son fundamentales en la era digital.

A pesar de la adopción masiva de APIs, las organizaciones e instituciones enfrentan importantes desafíos en el diseño de APIs que sean simultáneamente escalables y seguras. La falta de estrategias de diseño unificadas y guías claras provoca implementaciones inconsistentes, en las que las APIs pueden fallar al gestionar aumentos de carga de trabajo o quedar expuestas a vulnerabilidades de seguridad. Este problema se agrava debido a la complejidad inherente en la gestión de versiones, la integración de servicios dispares y la selección de arquitecturas y tecnologías, como REST, GraphQL, arquitecturas monolíticas y microservicios.

Abordar este problema es crucial, ya que la eficiencia operativa y la seguridad de los sistemas dependen en gran medida de la capacidad de las APIs para gestionar mayores cargas y proteger los datos. Una API mal diseñada no solo compromete la interoperabilidad y la experiencia del usuario, sino que también pone en riesgo la integridad y confidencialidad de la información, generando importantes riesgos para las operaciones de las organizaciones e instituciones. Por lo tanto, establecer estrategias claras para el diseño de APIs escalables y seguras es esencial para garantizar la sostenibilidad y competitividad en el entorno digital actual.

El marco de este estudio se sustenta en teorías y prácticas reconocidas en el campo del desarrollo de software y la seguridad informática. En términos de diseño de APIs, se destacan principios que promueven la creación de interfaces intuitivas y eficientes para facilitar la comunicación entre sistemas. Pacheco et al. (2021) señalan que los avances tecnológicos y la creciente complejidad de los sistemas heterogéneos han incrementado el interés por diseñar APIs más genéricas, robustas y adaptables. La

escalabilidad de las APIs implica implementar estrategias para garantizar que una API pueda manejar incrementos en la carga de trabajo sin perder rendimiento.

Respecto a la seguridad de las APIs, es fundamental protegerlas contra amenazas y accesos no autorizados. Según Ruiz Barea (2023), es indispensable implementar medidas que sigan las directrices de organizaciones como OWASP para evitar riesgos como accesos no autorizados, ataques de denegación de servicio y ataques de inyección. Estos riesgos pueden tener graves consecuencias, como pérdidas financieras, daños a la reputación y responsabilidades legales. La seguridad es especialmente crucial en instituciones que gestionan datos sensibles, como el sector financiero, de salud y gubernamental.

La *OWASP* (2017) proporciona una lista detallada de vulnerabilidades comunes en APIs y cómo mitigarlas, destacando la necesidad de incorporar seguridad desde el inicio del diseño (**ver Tabla 1**). Estas vulnerabilidades incluyen problemas como la autenticación rota, exposición excesiva de datos y gestión inadecuada de recursos. Aplicar controles adecuados es clave para proteger las APIs frente a amenazas comunes, especialmente en aplicaciones críticas.

Tabla 1 | Directrices principales de OWASP 2017 para proteger APIs contra amenazas y accesos no autorizados.

Directriz OWASP	Descripción	Recomendaciones
Broken Object Level Authorization (BOLA)	La falta de controles adecuados sobre el acceso a objetos permite el acceso a datos de otros usuarios.	Implementar control de acceso basado en el usuario autenticado. Validar siempre la propiedad del objeto.
Broken Authentication	Métodos de autenticación inseguros o mal configurados que permiten acceso no autorizado.	Usar autenticación sólida como OAuth2, OpenID Connect. Evitar credenciales hardcodeadas.
Excessive Data Exposure	Las API exponen más datos de los necesarios, lo que puede ser explotado por un atacante.	Limitar los datos expuestos a lo estrictamente necesario. Filtrar y sanitizar las respuestas antes de enviarlas.
Lack of Resources & Rate Limiting	No limitar adecuadamente el acceso a los recursos de la API permite ataques de denegación de servicio.	Implementar límites de uso y mecanismos de rate-limiting. Monitorizar y controlar el uso de recursos.

Mass Assignment	Los parámetros enviados por el cliente pueden actualizar propiedades de objetos no previstos.	Validar las propiedades de los objetos que pueden ser modificadas. Utilizar listas blancas en las asignaciones.
Security Misconfiguration	Configuraciones incorrectas o inseguras de la API pueden dejar vulnerabilidades abiertas.	Aplicar principios de seguridad desde el diseño. Utilizar configuraciones seguras por defecto y revisarlas periódicamente.
Injection	Las inyecciones de código malicioso, como SQLi, en parámetros de la API pueden comprometer el sistema.	Usar ORM y consultas parametrizadas. Evitar la concatenación directa de parámetros en las consultas.
Improper Assets Management	La exposición de endpoints no documentados o versiones antiguas de la API puede ser explotada.	Mantener un inventario actualizado de las APIs y sus versiones. Desactivar versiones obsoletas o no seguras.
Insufficient Logging & Monitoring	La falta de registros y monitoreo adecuado impide la detección temprana de ataques.	Implementar monitoreo continuo y alertas de seguridad. Registrar eventos críticos con detalles suficientes.
Insecure Deserialization	Deserialización insegura de objetos puede permitir la ejecución remota de código en el servidor.	Validar y filtrar datos deserializados. Utilizar formatos como JSON y evitar la deserialización de objetos complejos.

La arquitectura de microservicios y la arquitectura monolítica representan dos enfoques contrastantes en el desarrollo de aplicaciones. Mientras que la primera se caracteriza por su flexibilidad y escalabilidad al descomponer una aplicación en pequeños servicios autónomos, la segunda estructura todo el sistema como una unidad interconectada que sigue un modelo más tradicional con tres capas: acceso a datos, interfaz de usuario y controlador. Según Mamani Rodríguez et al. (2020), la arquitectura de microservicios ha ganado popularidad por su capacidad de adaptación a entornos complejos, permitiendo que las aplicaciones evolucionen y se ajusten a demandas crecientes. De la misma forma, Hernández et al. (2021) resaltan que la arquitectura monolítica, aunque más rígida, tiene el beneficio de una estructura consolidada que facilita la comunicación interna entre sus componentes.

Ambos enfoques ofrecen ventajas según el contexto en el que se apliquen. Mientras los microservicios son ideales para proyectos que requieren escalabilidad y flexibilidad, los sistemas monolíticos pueden ser más adecuados para aplicaciones que no anticipan cambios frecuentes o que necesitan un diseño más simple y directo. La elección entre ambos dependerá de las necesidades y los objetivos del proyecto.

Marculescu et al. (2022) señalan que los servicios web basados en RESTful son comúnmente empleados en el desarrollo de diversas aplicaciones empresariales. Con el incremento de la cantidad y la variedad de aplicaciones que aprovechan las API RESTful, se ha visto un aumento en los recursos destinados al desarrollo y la prueba de estos sistemas. La automatización en la creación de datos de prueba surge como una solución efectiva para generar datos de manera rápida y eficiente. No obstante, dicha automatización también conlleva la generación de grandes conjuntos de pruebas que pueden resultar difíciles de analizar y evaluar manualmente. Por su parte, Quiña-Mera et al. (2023) destacan que GraphQL, como lenguaje de consulta y motor de ejecución para APIs web, ha sido propuesto como una alternativa para mejorar problemas relacionados con el acceso a datos y la gestión de versiones en las API basadas en REST.

La evolución de las API, especialmente RESTful, ha impulsado el desarrollo empresarial, pero también ha generado desafíos en la gestión y pruebas automatizadas de grandes volúmenes de datos. Aunque la automatización es prometedora, su complejidad demanda nuevas soluciones. Alternativas como GraphQL abordan problemas de acceso y versiones, ofreciendo mayor flexibilidad y eficiencia. Es fundamental elegir la tecnología adecuada según las necesidades de cada proyecto.

Además, se examinan las diferencias entre REST y GraphQL en términos de diseño y rendimiento de APIs. Capote Pérez (2023) destaca que una de las principales diferencias entre ambos enfoques es la mayor flexibilidad de GraphQL, que permite al cliente especificar los datos exactos que necesita, optimizando la eficiencia. En contraste, las APIs REST siguen una estructura predeterminada, lo que puede requerir múltiples endpoints para acceder a diferentes recursos.

Pernil Bronchalo (2024) destaca varias ventajas de GraphQL frente a REST, explicando lo que ofrece esta tecnología. GraphQL se concibe como una herramienta para crear APIs declarativas, donde el cliente especifica exactamente qué datos necesita, evitando la creación de múltiples endpoints con diferentes estructuras de datos. Con GraphQL, se utiliza un único endpoint que responde de manera precisa a las solicitudes del cliente. Además, introduce el concepto de suscripciones, lo cual permite

implementar un modelo de comunicación "servidor-cliente" en el que los datos se envían automáticamente a los clientes que están a la escucha. Esto resulta clave para notificaciones en tiempo real, como aquellas que informan al usuario cuando su turno en una cola está disponible. Otra ventaja importante es la reducción en el costo de servidores y accesos a la base de datos, al eliminar la necesidad del polling con peticiones HTTP, ya que solo se actualiza la información requerida por el usuario mientras se mantiene una conexión activa con WebSockets. Esto optimiza el uso de recursos y permite refrescar los datos en tiempo real, lo que favorece una arquitectura escalable. Además, GraphQL facilita la integración estandarizada de WebSockets, un protocolo ligero sobre TCP, lo que simplifica la implementación y reduce la necesidad de herramientas adicionales.

Por otro lado, Culcay Oñate (2022) subraya que las API REST ofrecen ventajas significativas en el desarrollo, como la capacidad de conectar aplicaciones y dispositivos de manera eficiente, tanto individualmente como en conjunto. REST promueve la independencia entre el cliente y el servidor, mejorando la portabilidad de las interfaces y la escalabilidad de los proyectos. Además, facilita la adición de nuevas características y la exposición del sistema a diferentes plataformas, lo que permite que los usuarios accedan a través de aplicaciones, dispositivos IoT, navegadores, y programas. No es de extrañar, entonces, que Liu et al. (2022) señalen que las API RESTful son los puntos finales más populares para acceder a servicios web.

Sin embargo, a pesar de las fortalezas de REST, GraphQL ha ganado relevancia por su capacidad para optimizar recursos y ofrecer actualizaciones en tiempo real. A diferencia de REST, GraphQL permite que el cliente defina exactamente qué datos necesita, lo que, junto con su integración con WebSockets, lo convierte en una opción atractiva para arquitecturas escalables. Aunque REST es robusto, puede ser menos eficiente en situaciones donde es crucial minimizar la carga en servidores y bases de datos. En este sentido, Amodeo (2013) sugiere que la elección entre GraphQL y REST depende del caso de uso de la API. Si se prevé la necesidad de gestionar entidades hijas mediante operaciones complejas, es recomendable utilizar una colección intermedia. En cambio, si dichas funcionalidades no son necesarias, optar por enlaces directos simplifica el diseño y mejora la eficiencia.

La seguridad y la autenticación también son aspectos clave en el diseño de APIs. Amengual Bauza (2019) subraya la importancia de implementar protocolos robustos de autenticación, como OAuth2 y

JWT, para proteger la información sensible y garantizar que cada solicitud sea procesada de manera segura. El uso de estándares probados es esencial para minimizar vulnerabilidades y mejorar la seguridad general de la API.

Por otro lado, Marin Diaz et al. (2020) destacan que las mejores prácticas en el desarrollo de APIs incluyen la implementación de sistemas, herramientas y técnicas que han demostrado ser eficaces en organizaciones de renombre mundial. La adopción de estas prácticas, como el control de versiones y la documentación clara, es esencial para garantizar la sostenibilidad y eficiencia de las APIs a largo plazo.

La integración efectiva de sistemas también juega un papel importante en este proceso. Davas Rodríguez (2024) sugiere que el uso de estándares abiertos facilita la conexión de sistemas heterogéneos, mejorando la interoperabilidad entre plataformas.

Los microservicios mejoran la flexibilidad y escalabilidad de las aplicaciones, facilitando una rápida adaptación a cambios en los requisitos del negocio. Reynés Fernández (2023) señala que, aunque los microservicios presentan beneficios, también suponen desafíos en la gestión y seguridad de múltiples componentes.

Por esta razón, Arbieto Batallanos (2021) indica que los sistemas o tecnologías de información basados en tecnologías WEB han cambiado los paradigmas de operar de las organizaciones o empresas actuales, dado que se logran considerables mejoras a comparación de los sistemas de escritorio, pues optimizan los procesos informáticos que operan las empresas, y a su vez proporcionan información de apoyo, para el proceso de toma de decisiones en el preciso momento, motivando el alcance de obtener ventajas competitivas.

Finalmente, Perales Chavez (2024) señala que uno de los principales desafíos que enfrentan las empresas es la incapacidad de lograr una adecuada interoperabilidad entre sus sistemas y procesos. A medida que estas organizaciones crecen, es común que experimenten dificultades en la comunicación y en la eficiencia de sus sistemas, especialmente cuando integran software de planificación de recursos empresariales (ERP) que, en lugar de ofrecer una solución integral, fragmentan los procesos y generan sistemas heterogéneos. Esta falta de cohesión no solo genera pérdidas significativas en términos de tiempo y dinero, sino que también puede comprometer la estabilidad financiera de la empresa.

En línea con estos desafíos de integración, Patajalo (2023) advierte que las vulnerabilidades en APIs representan un riesgo significativo para la seguridad de las aplicaciones web, por lo que es esencial contar con mecanismos sólidos de autenticación y autorización. El autor enfatiza la importancia de adoptar buenas prácticas de desarrollo, como la implementación de pruebas automatizadas, para mejorar la calidad y seguridad de las APIs a lo largo del tiempo.

Asimismo, Suescún Monsalve et al. (2024) destacan que las empresas digitales emergentes están impulsando una transformación significativa que afecta a grandes organizaciones. Estas sinergias, facilitadas en gran parte por el uso eficiente de las APIs, permiten que las compañías mantengan su competitividad en mercados dinámicos. Las APIs, al integrar sistemas y promover modelos de negocio innovadores, se consolidan como herramientas fundamentales para asegurar la continuidad de este proceso de cambio.

Aunque estos estudios ofrecen información valiosa, existe una brecha en la literatura sobre la falta de una guía integral que combine estos aspectos y ofrezca directrices claras para el diseño estratégico de APIs que aborden tanto la escalabilidad como la seguridad. Este trabajo busca llenar ese vacío, consolidando estos elementos y proporcionando recomendaciones prácticas basadas en una revisión exhaustiva y en experiencias aplicadas.

La investigación se desarrolla en el contexto de la transformación digital que enfrentan organizaciones e instituciones, donde la integración de sistemas y aplicaciones es cada vez más necesaria. En sectores como salud, educación, finanzas y gobierno, la interoperabilidad eficiente no solo optimiza los procesos internos, sino que también mejora la satisfacción del usuario y la capacidad de adaptarse rápidamente a cambios en el entorno tecnológico.

El propósito de este estudio es establecer estrategias de diseño que permitan la creación de APIs escalables y seguras, facilitando la integración eficiente y protegida de sistemas en entornos tecnológicos diversos. Se busca identificar las principales prácticas y tecnologías que contribuyen a la escalabilidad y seguridad de las APIs, proporcionando directrices claras que mejoren la consistencia y calidad de las mismas.

METODOLOGÍA

Para llevar a cabo la investigación, se propone una estructura para el diseño de APIs escalables y seguras que faciliten la integración de sistemas y aplicaciones. Con el fin de alcanzar este objetivo, se realizó una búsqueda exhaustiva y sistemática de literatura en bases de datos académicas y técnicas, complementada con el análisis de proyectos de desarrollo de APIs implementados en diversos sectores. Se consultaron fuentes académicas y estudios técnicos que abordan temas clave como el diseño de APIs, escalabilidad, seguridad y tecnologías como REST, GraphQL, OAuth2 y JWT, aplicadas en entornos organizacionales e institucionales. Además, se incorporó la experiencia práctica obtenida de implementaciones reales, lo que permitió obtener una visión más profunda y aplicada sobre los retos y soluciones en la creación de APIs.

RESULTADOS Y DISCUSIÓN

Resultados

Proponer una API va más allá de desarrollar una simple interfaz de comunicación. El éxito de una API radica en su capacidad para ser escalable, segura, flexible y fácilmente integrable. Para ello, se sugiere la siguiente estructura estratégica para el diseño de APIs escalables y seguras:

Tabla 2 | Propuesta de estructura estratégica para el diseño de APIs escalables y seguras.

Aspecto	Descripción
Elección del entorno de ejecución	Definir un entorno que optimice el desempeño, como un contenedor Docker para escalabilidad y facilidad de despliegue, o una máquina virtual para proyectos que requieran mayor personalización del sistema operativo.
Arquitectura: monolito vs. microservicios	Evaluar la complejidad del proyecto para decidir entre una arquitectura monolítica (para simplicidad y rapidez en proyectos pequeños) o una de microservicios (para escalabilidad y modularidad en aplicaciones grandes).
Selección de la tecnología	Escoger entre API REST (común y fácil de implementar) o GraphQL (más control sobre las consultas), dependiendo del nivel de flexibilidad y personalización de las necesidades del cliente y de los datos.
Uso de ORM para bases de datos	Aislar la gestión de datos en un ORM que facilite la interacción con bases de datos, permitiendo escalabilidad y eficiencia en las consultas. El ORM debe mantenerse lo más separado posible de la lógica de negocio y solo interactuar cuando se requiere la persistencia de datos.
Buenas prácticas de desarrollo de software	Seguir prácticas como control de versiones (Git), revisión de código, pruebas unitarias e integración continua para asegurar que todo el ciclo de vida del desarrollo siga principios sólidos de escalabilidad, mantenibilidad y seguridad. Esto debe abarcar toda la estructura de la API.

Protección frente al acceso no autorizado	Implementar autenticación robusta (OAuth2, JWT) y cifrado TLS para proteger tanto el acceso a la API como las transacciones de datos sensibles. Las APIs deben incluir separación entre endpoints públicos y privados, con controles de acceso en cada capa.
Pruebas y manejo de errores	Asegurar que las pruebas unitarias, pruebas de integración y pruebas automatizadas sean parte del ciclo continuo de desarrollo. Implementar un manejador de errores global que centralice la gestión de errores y devuelva mensajes adecuados.
Versionamiento	Mantener un sistema de versionado claro, en línea con las mejores prácticas de semver, para asegurar que las nuevas versiones no rompan la compatibilidad con clientes que usen versiones anteriores.
Cifrado de datos	Asegurar que los datos en tránsito y en reposo estén cifrados de acuerdo con las normativas de seguridad más estrictas. Implementar políticas de rotación de claves y manejo de incidentes de seguridad.
Regulación de llamadas (Rate Limiting)	Implementar un sistema de control de la cantidad de solicitudes, devolviendo códigos de error 429 cuando se excedan los límites. Este sistema de rate limiting es clave para evitar ataques de denegación de servicio y sobrecarga del servidor.
Documentación clara y consistente	Usar herramientas como Swagger o OpenAPI para generar documentación precisa y actualizada. Asegurarse de que los términos sean consistentes y claros, incluyendo ejemplos de uso y respuestas de error. La documentación es parte integral de la implementación y mantenimiento de una API bien diseñada.
Monitoreo y análisis en tiempo real	Implementar monitoreo en tiempo real con herramientas como StatusPage o UptimeRobot para detectar caídas o degradación de rendimiento antes de que afecten al cliente. Realizar análisis estadísticos periódicos para detectar áreas de mejora.

En la **Figura 1** se observa la estructura de la propuesta estratégica para el diseño de APIs escalables y seguras:

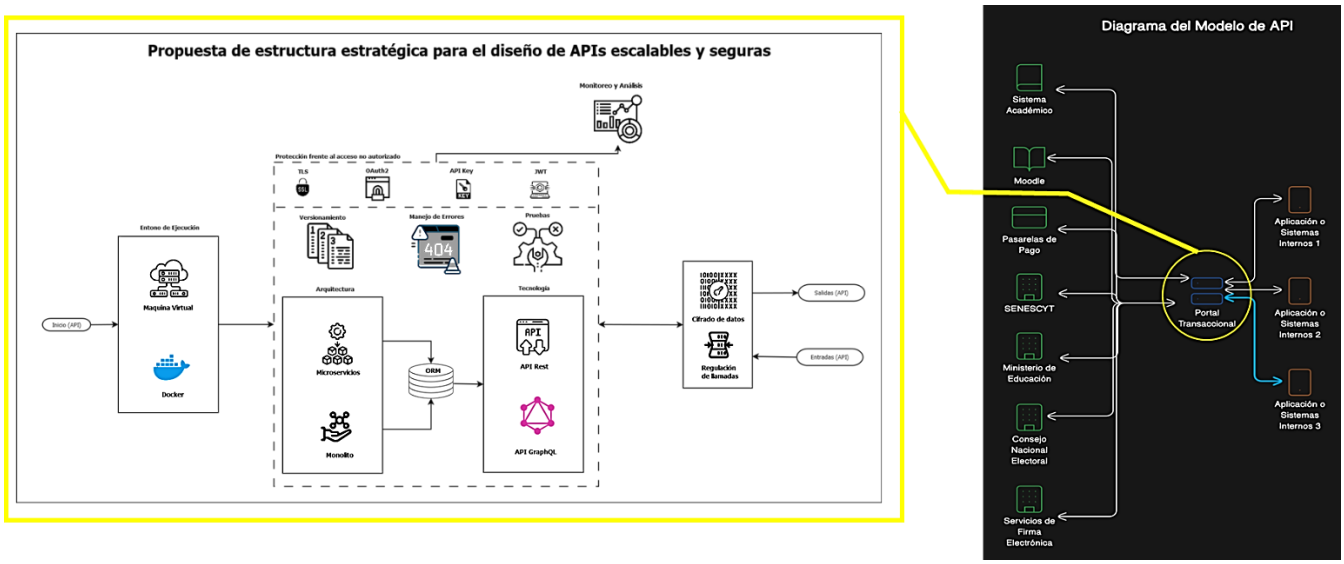

Figura 1 | Propuesta de estructura estratégica para el diseño de APIs escalables y seguras (API Rest).

Discusión

Esta estructura proporciona un marco estratégico para el diseño de APIs escalables y seguras, alineándose con las mejores prácticas y tecnologías actuales. Aunque cualquier programador puede desarrollar una API básica, seguir esta estructura asegura que la API sea eficiente, segura y capaz de

manejar las demandas crecientes de los sistemas y aplicaciones modernas. La adopción de estas prácticas ayuda a evitar problemas comunes como la falta de escalabilidad, vulnerabilidades de seguridad y dificultades en la integración por parte de terceros.

Además, la implementación de estos principios permite que las APIs sean más sostenibles y flexibles a largo plazo, lo que es fundamental para organizaciones que buscan adaptarse rápidamente a la evolución tecnológica. La documentación clara y el monitoreo en tiempo real también garantizan que los equipos de desarrollo y operaciones puedan gestionar la API de manera eficiente y responder a cualquier problema de manera proactiva.

CONCLUSIONES

- La integración de medidas de seguridad como OAuth2 y JWT desde el inicio del diseño de APIs es crucial para proteger los datos y evitar vulnerabilidades, garantizando una API segura y confiable.
- La adopción de microservicios permite una mayor escalabilidad y flexibilidad, aunque implica desafíos adicionales en su gestión, requiriendo herramientas y capacidades adecuadas para su correcta implementación.
- La documentación clara y el versionamiento adecuado son esenciales para asegurar la sostenibilidad y facilidad de uso de las APIs, evitando problemas de compatibilidad y facilitando su integración en diferentes entornos.
- Persisten interrogantes sobre la gestión de microservicios en entornos complejos, lo que abre la puerta a futuros estudios enfocados en simplificar su manejo y mejorar su seguridad en contextos de alta demanda.

REFERENCIAS BIBLIOGRÁFICAS

Amengual Bauza, M. (2019). *Security in API and API managers*. https://openaccess.uoc.edu/handle/10609/95286

Amodeo, E. (2013). *Principios de diseño de APIs REST*.

Arbieto Batallanos, C. E. (2021). *Desarrollo de un sistema de gestión de certificados SOAT, aplicando metodología ágil Scrum*. https://repositorio.ucsm.edu.pe/handle/20.500.12920/10533

Capote Pérez, Á. A. (2023). *FreeMyPark: Aplicación para la gestión de aparcamiento*. https://riull.ull.es/xmlui/handle/915/33531

Culcay Oñate, G. (2022). *API REST para la transmisión de información y control de redes de sensores IOT* [bachelorThesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Carrera de Ingeniería en Electrónica y Comunicaciones]. https://repositorio.uta.edu.ec:8443/jspui/handle/123456789/35022

Davas Rodríguez, R. M. (2024). *Integración de las Apis Rest de Figshare y Github mediante una aplicación orientada a servicios para publicar contenido Open Science* [bachelorThesis]. https://repositorio.utn.edu.ec/handle/123456789/15630

Hernández, L. M. A., Romero, V. A. P., González, S. A. S., & Rodríguez, J. A. V. (2021). Arquitectura REST para el desarrollo de aplicaciones web empresariales. *Revista Electrónica sobre Tecnología, Educación y Sociedad*, *8*(15), Article 15. https://www.ctes.org.mx/index.php/ctes/article/view/748

Liu, Y., Li, Y., Deng, G., Liu, Y., Wan, R., Wu, R., Ji, D., Xu, S., & Bao, M. (2022). Morest: Model-based RESTful API testing with execution feedback. *Proceedings of the 44th International Conference on Software Engineering*, 1406-1417. https://doi.org/10.1145/3510003.3510133

Mamani Rodríguez, Z. E., Del Pino Rodríguez, L., & Gonzales Suarez, J. C. (2020). Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua. *Industrial Data*, *23*(2), 141-149. https://doi.org/10.15381/idata.v23i2.17278

Marculescu, B., Zhang, M., & Arcuri, A. (2022). On the Faults Found in REST APIs by Automated Test Generation. *ACM Trans. Softw. Eng. Methodol.*, *31*(3), 41:1-41:43. https://doi.org/10.1145/3491038

Marin Diaz, A., Trujillo Casañola, Y., Buedo Hidalgo, D., Marin Diaz, A., Trujillo Casañola, Y., & Buedo Hidalgo, D. (2020). Estrategia de pruebas para organizaciones desarrolladoras de software. *Revista Cubana de Ciencias Informáticas*, *14*(3), 83-104.

OWASP Top Ten | OWASP Foundation. (2017, enero 1). https://owasp.org/www-project-top-ten/

Pacheco, A., Escobar Mendoza, J., & Trujillo, E. (2021). *Uso de patrones de diseño y metaprogramación para construir APIs de IoT usando C++*.

Patajalo, G. A. R. (2023). Seguridad en desarrollo web: Mejores prácticas para proteger aplicaciones y datos. *Dominio de las Ciencias*, *9*(3), Article 3. https://doi.org/10.23857/dc.v9i3.3552

Perales Chavez, J. A. (2024). Implementación de un modelo de arquitectura de industria 4.0 para mejorar la interoperabilidad entre sistemas de una empresa peruana. *Repositorio Institucional - USS*. http://repositorio.uss.edu.pe//handle/20.500.12802/12740

Pernil Bronchalo, R. (2024). *Aplicación técnica de arquitectura basada en microservicios con GraphQL y tecnologías serverless en cloud*. https://riuma.uma.es/xmlui/handle/10630/30628

Quiña-Mera, A., Fernandez, P., García, J. M., & Ruiz-Cortés, A. (2023). GraphQL: A Systematic Mapping Study. *ACM Comput. Surv.*, *55*(10), 202:1-202:35. https://doi.org/10.1145/3561818

Reynés Fernández, D. (2023, mayo). *Creación y gestión de microservicios a través de un Api Gateway* [Info:eu-repo/semantics/bachelorThesis]. E.T.S. de Ingenieros Informáticos (UPM). https://oa.upm.es/75036/

Ruiz Barea, R. (2023). *Protección de APIs REST*. https://openaccess.uoc.edu/handle/10609/148140

Suescún Monsalve, E., Tabares Betancur, M., González Palacio, L., & Vásquez Escobar, M. (2024). *Hacia un modelo de gobierno de APIs, mapeo sistemático de la literatura.* http://dspace.espoch.edu.ec/handle/123456789/21212