Fecha de recepción: 05 de junio de 2025, fecha de publicación en línea: octubre de 2025.

Exploración Teórica de Arquitecturas de Microservicios en Entornos Contenerizados

Samantha Yazmin Elizalde-Valencia [1], José Juan Hernández-Mora [2], María Guadalupe Medina-Barrera [3], y Juan Ramos-Ramos [4]

[1] Tecnológico Nacional de México – Instituto Tecnológico de Apizaco. San Andrés Ahuashuatepec, Municipio de Tzompantepec, Tlaxcala, C.P. 90491, México.
[2] Tecnológico Nacional de México – Instituto Tecnológico de Apizaco. San Andrés Ahuashuatepec, Municipio de Tzompantepec, Tlaxcala, C.P. 90491, México.
[3] Tecnológico Nacional de México – Instituto Tecnológico de Apizaco. San Andrés Ahuashuatepec, Municipio de Tzompantepec, Tlaxcala, C.P. 90491, México.
[4] Tecnológico Nacional de México – Instituto Tecnológico de Apizaco. San Andrés Ahuashuatepec, Municipio de Tzompantepec, Tlaxcala, C.P. 90491, México.

Autor de correspondencia: Autor. Samantha Yazmin Elizalde-Valencia (correo electrónico: m23370041@apizaco.tecnm.mx).

Abstract- This article analyzes container-based microservices architectures as a modern alternative to monolithic systems, highlighting their advantages in scalability, deployment flexibility, and fault isolation. Using a methodological approach that combines technology research engineering and agile practices, it examines how microservices address the key limitations of traditional architectures while introducing new operational complexities.

The findings reveal that microservices offer superior modularity and development speed, particularly for cloud applications. Containerization enables the independent deployment of services and the efficient use of resources, although it requires robust monitoring solutions for distributed environments. The study highlights how agile methodologies effectively manage and promote continuous collaboration with stakeholders.

The comparative analysis shows that microservices outperform monoliths in scenarios that demand high availability and rapid scaling. However, successful adoption requires careful consideration of organizational factors and technical challenges such as data consistency. The research provides insights for organizations transitioning to microservices.

Keywords: System Architecture, Microservices, Containerization, Scalability.

I. INTRODUCCIÓN

En la creación de software, las arquitecturas monolíticas han predominado, estas son arquitecturas que incorporan todos los elementos, tales como interfaz de usuario, lógica empresarial y acceso a datos, en una sola unidad de despliegue [1]. Este modelo brinda beneficios iniciales, tales como sencillez en el desarrollo y despliegues conjuntos, lo que lo convierte en una buena opcion para proyectos de tamaño reducido o equipos pequeños [2]. No obstante, conforme las aplicaciones se vuelven más complejas y los equipos se expanden, los sistemas monolíticos se topan con retos como problemas de mantenimiento, acoplamiento excesivo y un código complicado de administrar [3].

Las arquitecturas basadas en microservicios han surgido como una alternativa que posibilita dividir aplicaciones en servicios autónomos, cada uno con su propia lógica, mientras que la posibilidad de ejecutarse en contenedores asegura portabilidad y escalabilidad [4]. Este enfoque promueve la implementación constante, disminuye los peligros vinculados a modificaciones en el código y potencia la habilidad para ajustarse a nuevas demandas.

Por esta razón, compañías líderes como Netflix, Uber y Amazon, operan bajo estas estructuras. Netflix, vanguardista en la implementación de microservicios, emplea esta estructura para servir a más de 230 millones de usuarios alrededor del mundo. Al separar

características (como motores de sugerencias, transmisión de video y perfiles de usuario), Netflix consigue una elevada tolerancia a errores y un despliegue rápido de funciones. De forma parecida, el sistema de Uber basado en microservicios administra tareas en tiempo real, tales como la asignación de conductores, la geolocalización y el procesamiento de pagos, conservando respuestas con poca latencia. Amazon ilustra aún más este método, en el que servicios autónomos gestionan la administración de inventarios, pasarelas de pago y sugerencias a medida, facilitando el desarrollo simultáneo y la escalabilidad [5].
Este articulo analiza las bases, beneficios y retos de implementar una arquitectura de microservicios en contenedores. Además, se analizan casos de estudio y buenas prácticas vinculadas a la puesta en marcha de microservicios.

II. PROBLEMATICA

Las arquitecturas monolíticas y los sistemas *legacy* (Sistemas obsoletos, críticos para operaciones empresariales, pero con alta dependencia de tecnologías desactualizadas y difícil mantenimiento) [6], presentan retos en entornos tecnológicos modernos, donde la agilidad y la escalabilidad son requisitos esenciales. A pesar de ser inicialmente eficaces, estos sistemas producen elevados gastos operativos debido a la falta de especialistas en tecnologías anticuadas y la complejidad de su mantenimiento, además de generar riesgos de seguridad al no poder poner en marcha actualizaciones esenciales [7]. Esta dificultad se intensifica cuando se compara con las necesidades actuales de integración con paradigmas como *cloud computing* e *IoT* (Internet of Things o Internet de las cosas en español). En este contexto, la arquitectura de microservicios, implementada por líderes tecnológicos como Netflix y Amazon, brinda beneficios clave al desacoplar funcionalidades en servicios ligeros, reutilizables y fácilmente escalables [8]. Precisamente por estas limitaciones de los sistemas tradicionales y las posibilidades que brindan los nuevos enfoques, se expone a continuación un estudio de las arquitecturas de microservicios, valorando su habilidad para abordar estos retos.

III. ESTADO DEL ARTE

A pesar de que todavía existen sistemas que funcionan bajo esquemas anticuados y arquitecturas rígidas, cada vez más organizaciones están adoptando soluciones basadas en microservicios y contenedores, evidenciando su efectividad en ambientes de producción. A continuación, se muestran ejemplos específicos que evidencian estas implementaciones exitosas.
En los casos analizados, los sistemas creados con estos métodos han demostrado avances notables en su capacidad de respuesta, mantenibilidad y eficiencia operativa, particularmente cuando se combinan con metodologías ágiles.

La investigación de Saransig [9] expone el desarrollo histórico de la arquitectura de software, centrándose en las arquitecturas monolíticas y de microservicios. La hipótesis principal plantea que, al implementarse en contenedores, la arquitectura de microservicios incrementa el rendimiento en un 15% en comparación con la arquitectura monolítica.
El estudio resalta que el análisis comparativo de rendimiento corrobora la hipótesis propuesta, evidenciando que las aplicaciones con arquitectura de microservicios en contenedores logran optimizar el uso de los recursos en comparación a las arquitecturas monolíticas. Estos hallazgos no solo satisfacen las metas concretas de la investigación, sino que también cuentan con el apoyo de investigaciones anteriores, incrementando la fiabilidad de los descubrimientos y posibilitando futuros estudios.
Aunque la obra se centra principalmente en la comparación de arquitecturas, también aborda las herramientas utilizadas para cada una de ellas y las estructuras en las que se están implementando. Este estudio ofrece una visión clara de cómo operan estas arquitecturas desde ese punto de vista, lo que aporta de manera significativa a la investigación en curso.

Por otro lado, Indrasiri [10], analiza la aplicación de microservicios en el desarrollo de aplicaciones, resaltando su habilidad para incrementar la escalabilidad y la rapidez frente a las arquitecturas monolíticas. Se utiliza una metodología basada en la generación e implementación de microservicios presentados como imágenes de contenedores, lo que simplifica su administración mediante herramientas como *Kubernets (*Permite orquestar varios contenedores en distintos servidores como si fueran uno solo) [11]. Los hallazgos indican que esta arquitectura facilita una implementación ágil y una gestión más eficiente de la comunicación entre servicios, aspecto vital para aplicaciones complejas.

Además, la investigación de Auer [12] sugiere un enfoque de evaluación fundamentado en pruebas para la transición de sistemas monolíticos a microservicios, con el objetivo de reconocer indicadores esenciales como el rendimiento, la escalabilidad y los costos de

infraestructura. Los resultados indican que los microservicios en contenedores exhiben un desempeño superior (hasta un 15% superior en peticiones/segundo) en comparación con los monolíticos, aunque demandan un uso más intensivo de CPU debido a la sobrecarga de la red. La capacidad para escalar servicios de manera individual disminuye los gastos operativos, comprobando su efectividad en casos de prueba.

Respecto a la tesis de García [13], el propósito principal es desarrollar una aplicación que simplifique la adopción de animales de compañía, además de sensibilizar a la población acerca del mantenimiento y posesión de mascotas. Este problema necesita una respuesta que pueda expandirse en cuanto a usuarios y funciones. Para ello, se decide emplear un enfoque ágil que posibilita segmentar el proyecto en sprints o etapas breves. De igual manera, se propone como meta principal el diseño de API REST y contenedores virtuales.
La arquitectura elegida fusiona los fundamentos de la arquitectura hexagonal con la arquitectura de cebolla. Por lo tanto, se puede apreciar una base firme para construir un sistema que no solo es eficaz en cuanto a recursos, sino que también se puede ajustar con facilidad a modificaciones futuras.

En la tesis de Abad [14], examina el caso del Banco Central de Ecuador, que afrontaba desafíos importantes debido a su sistema monolítico convencional, tales como flexibilidad limitada y escalabilidad, fallos extendidos del sistema, problemas para incorporar nuevas tecnologías y cumplir con normativas, además de un desempeño restringido y un mantenimiento complicado. Para resolver estos problemas, la organización decidió cambiar a una arquitectura fundamentada en microservicios y microfrontends, que facilita un desarrollo modular y desacoplado, la reutilización de componentes en varios procesos y una escalabilidad autónoma por servicio. Los hallazgos de esta migración evidenciaron que los ambientes de desarrollo aislados son sumamente escalables y tienen la habilidad de ajustarse a demandas variables, garantizando así un desempeño óptimo y una escalabilidad eficaz. Además, los test de carga y estrés proporcionaron resultados confiables que corroboran la capacidad de la solución sugerida, corroborando que estas arquitecturas posibilitan a las organizaciones construir y administrar sistemas de software con mayor eficiencia, velocidad y capacidad de adaptación a las exigencias tecnológicas contemporáneas.

IV. ARQUITECTURA MONOLÍTICA

Para seguir con este estudio, es crucial mantener los conceptos básicos, iniciando con la arquitectura monolítica, que es un modelo de diseño de software en el que todos los elementos de una aplicación (interfaz de usuario, lógica de negocio y capa de acceso a datos) se integran en una única unidad de despliegue y comparten el mismo espacio de memoria y recursos [15]. Este enfoque se distingue por su fuerte interconexión y dependencia directa entre sus módulos, lo que complica la modificación individual de componentes, como se muestra en la Figura 1 [16]. La aplicación se establece como un único artefacto ejecutable, lo que facilita la distribución inicial pero restringe la actualización de funciones específicas.

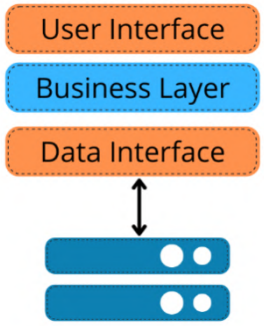

Figura 1. Arquitectura Monolítica.

A pesar de que su sencillez permite su desarrollo en fases iniciales, los sistemas monolíticos se topan con retos críticos en términos de escalabilidad (al necesitar duplicar toda la aplicación para incrementar su capacidad) y mantenimiento (dado que las modificaciones impactan en el sistema en su totalidad).

V. ARQUITECTURA DE MICROSERVICIOS

Los microservicios son una forma de arquitectura derivada de la Arquitectura Orientada a Servicios (SOA en inglés), que se distingue por la fragmentación de aplicaciones en servicios pequeños, autónomos y de alta especialización. Cada microservicio se centra en una única funcionalidad corporativa, se desarrolla de forma autónoma y se relaciona con otros servicios a través de interfaces claramente establecidas, usualmente a través de protocolos como REST (Figura 2) [16].

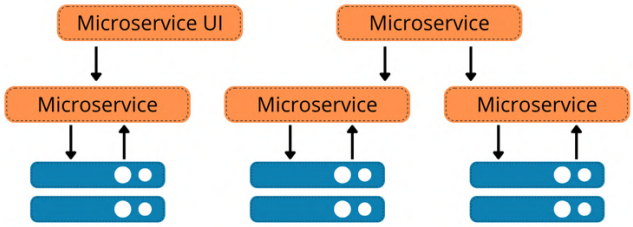

Figura 2. Arquitectura de Microservicios.

Una de las características fundamentales de los microservicios es que cada servicio puede ser creado, desplegado y escalado de manera autónoma, empleando distintas tecnologías y lenguajes de programación según se requiera. Esta estructura se diferencia de los sistemas monolíticos, en los que todas las funcionalidades se encuentran unidas en una única unidad de despliegue. Los microservicios brindan beneficios tales como una mayor resistencia, escalabilidad y sencillez para adoptar nuevas tecnologías. No obstante, también plantean retos, tales como la complejidad en la administración de la comunicación entre servicios, y la demanda de diversas habilidades técnicas en los equipos de desarrollo [17].

VI. API REST

Una API RESTful (Representational State Transfer) es un enfoque arquitectónico para la creación de interfaces de programación de aplicaciones (APIs), como el protocolo HTTP, con el objetivo de facilitar la comunicación entre sistemas.

De acuerdo con Amazon Web Services [18], las APIs RESTful se basan en recursos identificables a través de URIs (Uniform Resource Identifiers), donde cada recurso puede ser gestionado a través de las operaciones estándar de consulta (GET), creación (POST), actualización (PUT) y eliminación (DELETE), en la que cada petición del cliente al servidor debe incluir toda la información requerida para su procesamiento, optimizando así la escalabilidad y el desempeño del sistema. Entre sus mayores beneficios sobresalen su simplicidad, escalabilidad y compatibilidad con diversos formatos de datos como JSON o XML, lo que las hace una elección predominante para la incorporación de servicios en aplicaciones web y móviles.

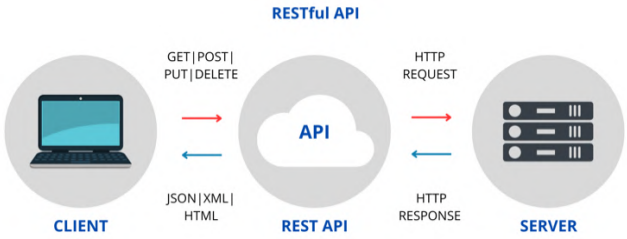

Figura 3. Funcionamiento API RESTful.

La Figura 3 [19], muestra una parte del funcionamiento esencial que realizan las APIs RESTful, resaltando los procedimientos fundamentales (GET, POST, PUT y DELET), además de los formatos de intercambio convencionales como JSON, XML y HTML. La imagen describe el proceso de comunicación entre cliente y servidor, un cliente inicia la comunicación enviando una solicitud HTTP a la API REST, que procesa la petición y se comunica con el servidor; en consecuencia, el servidor responde con un HTTP.

VII. CONTENEDORES

Los contenedores han revolucionado la forma en que se desarrollan, implementan y ejecutan las aplicaciones en la actualidad. De acuerdo con Google Cloud [20], los contenedores son paquetes de software que contienen todo lo requerido para que una aplicación funcione de forma autónoma y eficiente: código, bibliotecas, herramientas y configuraciones, como se puede observar en la figura 4 [21]. Una de sus mayores fortalezas es la portabilidad, dado que pueden funcionar sin alteraciones en diferentes contextos (desarrollo local, producción o la nube), lo que permite una implementación consistente y disminuye los errores.

Además, su ligereza los convierte en perfectos para ambientes *cloud*, posibilitando la ejecución de múltiples aplicaciones en un solo servidor sin interrupciones, optimizando recursos y disminuyendo los gastos operativos.

Otro beneficio principal es su escalabilidad ágil y fácil. Los contenedores son una tecnología adaptable que revoluciona el desarrollo y la implementación de aplicaciones.

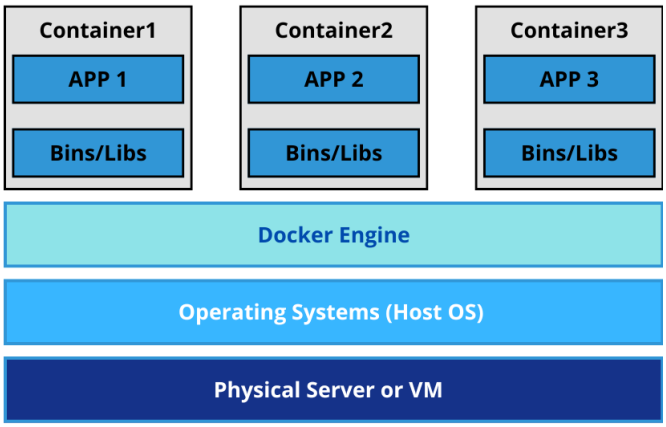

Figura 4. Estructura de contenedores de software.

VIII. METODOLOGÍAS

Para llevar a cabo el estudio de las arquitecturas de microservicios en contenedores, se empleará un enfoque de investigación tecnológica en ingeniería. Este método es apropiado para proyectos que no buscan formular nuevas teorías, sino reconstruir procesos a través de la adaptación y optimización de soluciones ya existentes. La investigación tecnológica facilitará el análisis de soluciones anteriores, la elección de sus componentes más destacados e incorporarlos a las demandas del proyecto [22].

Figura 5. Metodología de investigación propuesta.

La figura 5 muestra un proceso iterativo que inicia con la identificación de necesidades especiales y la definición exacta del problema, para luego pasar a una etapa de documentación de soluciones ya existentes. Después, a través de un análisis crítico, se elabora un proyecto de intervención (como prototipos o arquitecturas), que se evalúa en cuanto a factibilidad antes de su ejecución regulada. Luego, la observación de los resultados promueve modificaciones constantes, reiniciando el ciclo para mejorar la solución.

Mientras que en la etapa de intervención, se utilizarán técnicas ágiles, que mejoran la administración de proyectos de software a través de flexibilidad, constante cooperación con el cliente y entregas progresivas de productos funcionales. De acuerdo con Shafir [23], estas técnicas potencian la comunicación interna, incrementan la capacidad de adaptación a cambios y disminuyen los riesgos de fracaso. Elementos como el respaldo de la organización, la formación del equipo y la implicación directa del cliente son esenciales para su éxito. Asimismo, su método iterativo facilita la identificación de fallos en fases iniciales, optimizando recursos y gastos.

Figura 6. Metodología propuesta para desarrollo del proyecto de intervención.

Se inicia con la definición del problema, los objetivos del proyecto y la creación de un backlog con tareas priorizadas. Posteriormente, en la etapa de planificación se realizan iteraciones cortas (sprints) en las que el equipo selecciona las tareas más relevantes para desarrollarlas dentro de un flujo visual representado en un tablero Kanban, con columnas como “por hacer”, “en proceso” y “terminado”. A lo largo del proceso se llevan a cabo reuniones diarias breves para dar seguimiento, limitar el trabajo en curso y fomentar la colaboración. Al finalizar cada sprint se realiza una revisión de resultados y una retrospectiva para identificar áreas de mejora. Esta metodología permite adaptarse a los cambios, monitorear el avance en tiempo real y generar entregables funcionales durante todo el ciclo del proyecto

Para fundamentar las bases de estas metodologías y la implementación de una arquitectura de microservicios, se trabajará en conjunto con el Instituto del Deporte de Tlaxcala (IDET). Las condiciones de su proyecto lo convierten en apropiado para probar este método, dado que necesita un sistema de alta durabilidad, facilidad para incorporar funciones y capacidad de escalado. La solución se organizará en microservicios autónomos, tales como el frontend, backend (que actuara como middleware), servicios de reportes y verificaciones de datos, entre otros. Este modelo facilitará la incorporación de nuevas funcionalidades a través de microservicios adicionales, sin alterar las operaciones actuales, asegurando de esta manera escalabilidad y modularidad.

IX. TRABAJOS FUTUROS

Este artículo se centra en el estudio teórico de las arquitecturas de microservicios, tratando aspectos esenciales como las arquitecturas monolíticas y las arquitecturas fundamentadas en microservicios. Esta revisión proporciona un panorama que sirve como base para comprender las diferencias, ventajas y desafíos de cada enfoque, promoviendo de esta manera futuras investigaciones y avances en el área.

En trabajos posteriores se pretende profundizar en las metodologías utilizadas durante el proceso de investigación y en la elaboración del proyecto de intervención (Sistema de gestión medico deportiva del IDET), incluyendo la creación de una arquitectura que represente la interacción entre las diferentes capas del sistema. Además, se presentará una vista física que permita observar el funcionamiento real de una arquitectura de microservicios aplicada a un caso de prueba. También se detallarán aspectos técnicos como la configuración de APIs, contenedores y microservicios, lo cual permitirá comprender cómo se integran y comunican estos elementos dentro del sistema.

X. CONCLUSIONES

El estudio realizado demuestra que las arquitecturas de microservicios constituyen un enfoque eficaz y escalable para el desarrollo de sistemas informáticos. En contraste a los sistemas monolíticos, que tienen limitaciones en términos de escalabilidad, mantenimiento y adaptabilidad, los microservicios brindan beneficios significativos, tales como modularidad, flexibilidad en la implementación y resistencia frente a errores, tal como lo respaldan varios estudios analizados.

Los microservicios no solo superan a los monolíticos en contextos que buscan alta disponibilidad y escalabilidad, tales como ambientes cloud o sistemas distribuidos, sino que también se adecuan a las demandas presentes de ágilidad y eficacia en el desarrollo de software.

REFERENCIAS

[1] M. Fowler and J. Lewis, "Microservices: a definition of this new architectural term," *martinfowler.com*. Disponible: https://martinfowler.com/microservices/.

[2] M. Fowler, "How to Break a Monolith into Microservices," *martinfowler.com*, 2019. Disponible: https://martinfowler.com/articles/break-monolith-into-microservices.html.

[3] A. Guimarey, "Beneficios y riesgos de migrar una arquitectura monolítica a microservicios," *Universidad de Palermo*, 2020. Disponible: https://www.researchgate.net/publication/348309479.

[4] AWS, "Monolítico frente a microservicios: diferencia entre arquitecturas de desarrollo de software," *Amazon Web Services*, s.f. Disponible: https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/.

[5] M. Sarzoza, "Arquitecturas de software que utilizan Netflix, Uber y más," *LinkedIn*, 2023. [En línea]. Disponible: https://www.linkedin.com/pulse/arquitecturas-de-software-que-utilizan-netflix-uber-y-marcos-sarzoza-e6ytf/

[6] M. E. Barzola, "Migración semiautomática de sistemas Legacy hacia arquitecturas orientadas a servicios," Tesis de licenciatura, Lic. en Sistemas, Univ. Nac. de La Plata, 2018. Disponible: https://sedici.unlp.edu.ar/handle/10915/82154

[7] D. Hossain y T. Sultana, "The role of Microservice Approach in Edge Computing: opportunities, challenges, and research directions," *ScienceDirect*, 2023. Disponible: https://www.sciencedirect.com/topics/computer-science/monolithic-system

[8] LinkedIn, "Sistema Legacy: Qué es, características y migración con Whitecloud," dic. 26, 2024. Disponible: https://www.linkedin.com/pulse/sistema-legacy-qu%C3%A9-es-caracter%C3%ADsticas-y-migraci%C3%B3n-con-whitecloud-s4tgf/

[9] A. F. Saransig Chiza, "Análisis de rendimiento entre una arquitectura monolítica y una arquitectura de microservicios - tecnología basada en contenedores," Tesis de Maestría, Escuela Politécnica Nacional, Ecuador, 2018. Disponible: https://core.ac.uk/download/pdf/200323828.pdf

[10] K. Indrasiri, "Microservices in practice: Key architectural concepts of an MSA," WSO2, Tech. Rep., 2019. Disponible: https://resources.wso2.com/whitepapers/microservices-in-practice-key-architectural-concepts-of-an-msa

[11] DZone, *Kubernetes in the Enterprise: A Guide to Orchestrating Containers at Scale*, DZone Refcardz, 2019. Disponible: https://dzone.com/storage/attachments/14131598-dzone-kubernetesbundle.pdf

[12] F. Auer, "From monolithic systems to Microservices: An assessment framework," *Inf. Softw. Technol,* 2021. Disponible: https://www.sciencedirect.com/science/article/pii/S0950584921000793

[13] G. García Castañeda, "Diseño y desarrollo de una aplicación software, en el contexto del enfoque ágil, como soporte a la reducción del número de animales de compañía sin hogar en el

distrito metropolitano de Quito," Tesis de grado, Escuela Politécnica Nacional, Ecuador, 2024. Disponible: https://bibdigital.epn.edu.ec/handle/15000/25528

[14] F. L. Abad León and S. M. Guamán Cabrera, "Propuesta de una arquitectura basada en micro servicios y micro frontend con integración de una plataforma de mensajería y procesamiento de eventos masivos. Caso de estudio: Aplicación en sistemas de transferencia de fondos," *Universidad Politécnica Salesiana*, 2024. Disponible: https://dspace.ups.edu.ec/bitstream/123456789/29407/1/UPS-CT011876.pdf

[15] IBM, "Monolithic Architecture: Definition and Characteristics,". 2023. Disponible: https://www.ibm.com/think/topics/monolithic-architecture.

[16] Medium, Monolithic vs. Microservices Architecture: Understanding the Key Differences, 2024. Disponible: https://medium.com/@jain.yash1909/monolithic-vs-microservices-architecture-understanding-the-key-differences-7ddf328565d0

[17] H. M. Ayas, R. Hebig, and P. Leitner, "An empirical investigation on the competences and roles of practitioners in Microservices-based Architectures," *The Journal of Systems & Software*, 2024. Disponible: https://www.sciencedirect.com/science/article/pii/S0164121224001006?ref=pdf_download&fr=RR-2&rr=947fe8970b0c49df

[18] Amazon Web Services, "¿Qué es una API RESTful?," 2023. Disponible: https://aws.amazon.com/es/what-is/restful-api/

[19] onnuri.log. RESTful API. 2022. Disponible: https://velog.io/@onnuri/RESTful-API

[20] Google Cloud, "Contenedores en Compute Engine," 2024. Disponible: https://cloud.google.com/compute/docs/containers?hl=es-419

[21] Evolution. 2025. ¿Qué son los contenedores de software?. Disponible: https://evolutioncode.us/que-son-los-contenedores-de-software

[22] C. De la Cruz Casaño, *Metodología de la investigación tecnológica en ingeniería*. Universidad Continental, 2016. Disponible: https://www.academia.edu/94930372/Metodología_de_la_investigación_tecnológica_en_ingeniería

[23] M. Shafir et al., "The Success Factors of Agile Methodologies in Software Development based on Developing Countries' Software Firms," ScienceDirect, 2025. Disponible: https://www.sciencedirect.com/science/article/pii/S1877050925007069

SAMANTHA YAZMIN ELIZALDE VALENCIA Ingeniera en Tecnologías de la Información y las Comunicaciones. Estudiante de Maestría en Sistemas Computacionales en el Instituto Tecnológico Nacional de México (TecNM) - Campus Apizaco.

JOSÉ JUAN HERNÁNDEZ MORA Ingeniero en Computación por la Universidad Autónoma de Tlaxcala. Tiene el grado de Maestro en Ciencias de la Computación en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), de Cuernavaca, Morelos y Doctor en Excelencia Docente por la Universidad de los Ángeles. Es Profesor con Perfil Deseable por parte del PRODEP, es líder del cuerpo académico "Sistemas de Información" y nivel de candidato del SNII del Conahcyt. Sus líneas de investigación incluyen: Ingeniería del Software, Desarrollo de Aplicaciones de Tecnologías de la Información, Procesamiento Digital de Imágenes (PDI), Redes Neuronales Artificiales (RNA), Heutogogía y Cibergogía.

MARÍA GUADALUPE MEDINA BARRERA Tiene el grado de Maestra en Ciencias de la Computación en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), de Cuernavaca, Morelos y Doctora en Decanato de Ingenierías por la Universidad Popular Autónoma del Estado de Puebla. Es Docente en la División de Estudios de Posgrado e Investigación en el Instituto Tecnológico Nacional de México (TecNM) - Campus Apizaco.

JUAN RAMOS RAMOS Profesor de Tiempo completo y jefe de Proyecto de Investigación (Sistemas y Computación) del Instituto Tecnológico Nacional de México (TecNM) - Campus Apizaco.