Revista de investigación de sistemas e informática
17(2): 169-184 (2024)
DOI: https://doi.org/10.15381/risi.v17i2.29666

ISSN: 1815-0268 (Impreso) / ISSN: 1816-3823 (Electrónico)
Facultad de Ingeniería de Sistemas e Informática - UNMSM

Una revisión sistemática de la literatura: análisis de riesgos en la migración de arquitectura monolitica a microservicio

A systematic literature review: risk analysis in migration from monolithic to microservice architecture

Oscar Aldo Quispe Santa María
https://orcid.org/0009-0003-0049-379X
oscar.quispe2@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos,
Lima, Perú

Recibido: 07/12/2024 - Aceptado: 15/12/2024 - Publicado: 31/12/2024

RESUMEN

Desde 2011, el término "microservicios" comenzó a ser articulado como una evolución de la arquitectura orientada a servicios, destacándose por su enfoque en servicios pequeños, independientes y alineados con objetivos de negocio. Con el auge de tecnologías como los contenedores y herramientas de orquestación, los microservicios se establecieron como un estándar para arquitecturas escalables y modernas. Además, la aparición de frameworks, patrones y herramientas especializadas simplifico su adopción, consolidándolos como una práctica esencial en el desarrollo de software. Migrar microservicios conlleva diversos riesgos que pueden afectar la estabilidad, los costos y operación del sistema. La mayor complejidad técnica se manifiesta al diseño, implementación y pruebas debido a la naturaleza distribuida de los microservicios, sumado a problemas de comunicación entre servicios como latencia y errores de red. Riesgo de fallos en cascada por la interdependencia entre servicios puede propagar errores, impactando la disponibilidad del sistema, especialmente si no se implementan mecanismos de tolerancia a fallos. Este estudio identificará los riesgos en la migración a microservicios utilizando el método de Revisión sistemática de la literatura. Las bases de datos utilizadas en la selección de artículos que coinciden con los criterios incluyen: IEEE Xplore, ScienceDirect, Scopus, Google Scholar y entre 2015 y 2024. Los resultados de este estudio, fueron 33 artículos seleccionados de acuerdo con los criterios y revisados. El resultado de este estudio permitió identificar 21 riesgos los cuales fueron agrupados; microservicios, migración, arquitectura y otros, cuyas cantidades representan el 19%, 29%, 14% y 38% respectivamente e incluyendo las acciones de mitigación.

Palabra clave: Microservicio, Arquitectura Microservicio: Riesgo, Deuda Técnica, Antipatrones, Límites.

ABSTRACT

Since 2011, the term "microservices" began to be articulated as an evolution of service-oriented architecture, standing out for its focus on small, independent services aligned with business objectives. With the rise of technologies such as containers and orchestration tools, microservices were established as a standard for scalable and modern architectures. In addition, the emergence of specialized frameworks, patterns and tools simplified their adoption, consolidating them as an essential practice in software development. Migrating microservices entails various risks that can affect the stability, costs and operation of the system. The greatest technical complexity is manifested in the design, implementation and testing due to the distributed nature of microservices, coupled with communication problems between services such as latency and network errors. Risk of cascading failures due to the interdependence between services can propagate errors, impacting system availability, especially if fault tolerance mechanisms are not implemented. This study will identify the risks in migrating to microservices using the Systematic Literature Review method. The databases used in the selection of articles that match the criteria include: IEEE Xplore, ScienceDirect, Scopus, Google Scholar and between 2015 and 2024. The results of this study were 33 articles selected according to the criteria and reviewed. The result of this study allowed to identify 21 risks which were grouped; microservices, migration, architecture and others, whose quantities represent 19%, 29%, 14% and 38% respectively and including mitigation actions.

Keywords: Microservice, Microservice Architecture: Risk, Technical Debt, Antipatterns, Boundaries.

I. INTRODUCCIÓN

La revisión sistemática de la literatura (SLR) es un método de investigación que permite analizar, evaluar y sintetizar de manera exhaustiva los hallazgos de estudios previos. Según Al-Ruithe, este enfoque se emplea en artículos de revisión para seleccionar y examinar publicaciones siguiendo criterios previamente definidos y validados (Al-Ruithe, 2019).

Según Lelovic y otros opinan que una arquitectura de microservicios, cada servicio actúa como una unidad que se puede implementar de forma independiente (Lelovic et al., 2024). Por lo tanto; los microservicios (MS) ofrecen beneficios significativos en términos de escalabilidad, flexibilidad y facilidad de mantenimiento. Sin embargo, la descomposición de un sistema de arquitectura monolítica en MS es un proceso complejo que requiere asegurar que cada servicio sea lo más independiente posible, manteniendo una fuerte cohesión interna y minimizando la dependencia unos de otros. Esto puede ocurrir a medida que el sistema crece y muchos servicios se intercomunican entre sí a través de API u otros tipos de dependencia (Lelovic et al., 2024). También, cada microservicio puede aumentar la superficie de ataque, ya que expone múltiples puntos de entrada a través de APIs, lo que convierte la seguridad y autentificación en un reto debido a la naturaleza distribuida del sistema. En lugar de tener una superficie monolítica expuesta, se tienen varios servicios pequeños expuestos (Matias, Ferreira, Mateus-Coelho, & Ferreira, 2024). Asimismo, es crucial considerar los costos operativos y de infraestructura que demandan los microservicios, estos pueden aumentar significativamente si no se administran de manera eficiente. Las aplicaciones monolíticas tienden a ser complejas y de bajo nivel, estrechamente ligadas a plataformas y tecnologías específicas, lo que dificulta su portabilidad y limita su accesibilidad para usuarios con menos experiencia. (Esparza-Peidro, Muñoz-Escoí, & Bernabéu-Aubán, 2024).

Según Auer y otros consideran que todos los estudios coinciden en que no está claro cuándo las empresas deben migrar a los MS y qué características deben tener las empresas o el software para beneficiarse de las ventajas de los MS (Florian Auer, Lenarduzzi, Felderer, & Taibi, 2021). Ante ello, es importante realizar un estudio con el propósito de recopilar y analizar la información disponible en la literatura para identificar los riesgos a los que se enfrentan durante la migración de sistemas monolíticos hacia una arquitectura basada en microservicios. Una migración bien planificada tiene el potencial de ser exitosa, pero exige una preparación técnica y organizativa rigurosa; pensar en MS implica descomponer las funcionalidades en servicios independientes, lo cual introduce desafíos adicionales en términos de comunicación, sincronización de datos, seguridad y la gestión en la complejidad del sistema.

El objetivo de esta revisión es identificar, analizar, clasificar y definir los elementos clave que representan riesgos en la migración hacia microservicios, como su complejidad operativa, la seguridad, la gestión de datos y el impacto potencial en la operación y el rendimiento del sistema, asociados con la adopción de una arquitectura de microservicios (MSA) considerando que los sistemas más antiguos basados en MS no tienen más de 15 años y su arquitectura es muy diferente a la que se diseñó originalmente debido a varios cambios aplicados a los sistemas debido a la implementación de nuevas características y la corrección de errores (Cerny, Abdelfattah, Bushong, Al Maruf, & Taibi,

2022). Con la identificación de los riesgos al migrar a MS se busca garantizar una transición eficiente y segura, minimizando impactos negativos y maximizando beneficios. Esto incluye mitigar problemas técnicos, planificar estrategias de contingencia, reducir costos, asegurar la continuidad operativa, maximizar la escalabilidad e innovación, y mejorar la colaboración entre equipos, permitiendo una migración estructurada y orientada a resultados. La migración de un sistema monolítico a MS también implica un esfuerzo considerable en refactorización para garantizar la funcionalidad independiente de cada servicio. El uso de patrones de consistencia y el uso de técnicas pueden ayudar a mitigar riesgos, facilitando la coordinación de transacciones y preservando la coherencia entre los servicios.

II. MÉTODO

Para responder a las preguntas de investigación e identificar términos clave para la búsqueda adoptamos una metodología de estudio basado en pautas como PRISMA 2020, acrónimo de Preferred Reporting Items of Systematic Reviews and Meta-Analyses, el cual refleja avances en los métodos para identificar, seleccionar, evaluar y sintetizar los resultados de estudios previos de manera exhaustiva (Yepes-Nuñez, Urrútia, Romero-García, & Alonso-Fernández, 2021). Esta metodología es reconocida por asegurar un enfoque sistemático y transparente en la revisión de la literatura.

Búsqueda en bases de datos	La investigación se llevó a cabo en varias bases de datos académicas relevantes.
Criterios de inclusión y exclusión	Los artículos fueron seleccionados según su pertinencia en el tema de la migración de sistemas monolíticos a microservicios.
Proceso de selección	Se realizó una eliminación automática de entradas duplicadas y no relevantes, seguida de una selección manual.
Selección final	Se aplicaron estrictos criterios de exclusión en un examen detallado de los artículos restantes.

2.1. Criterios de elegibilidad

En la identificación del problema, nuestro enfoque para llevar a cabo esta investigación es incluir aquellos artículos que involucren aspectos referentes a una MSA en el rango de los últimos 10 años. Se excluirán las entrevistas u opiniones de expertos ya que determinan opiniones sesgadas. Sin embargo, si se considerará estudios secundarios y/o terciarios relacionado a MS o ingeniería de software como una práctica establecida para realizar e informar sobre estos estudios. También, se considerarán todos los estudios relacionados a los procesos de migración de sistemas monolíticos y riesgos asociados en la implementación de microservicios, incluyendo aquellos relacionados con malas prácticas de codificación en la refactorización (antipatrones), problemas de seguridad (exposición), acoplamiento, dependencias entre MS y otros mecanismos identificados como posibles riesgos. Hasta donde saben los autores, no existe una base de datos de aplicaciones basadas en MS listas para usar con las que comparar el lenguaje de modelado (Esparza-Peidro et al., 2024). Ante ello, se excluirán los estudios que aborden la migración a MS en el contexto de la migración a la nube, ya que esta última presenta riesgos específicos que requieren un tratamiento aparte. Se contrasta la arquitectura monolítica con la arquitectura objetivo (microservicios), permitiendo identificar riesgos potenciales, la identificación de deuda técnica, la falta de patrones en la refactorización, latencia, sincronización de datos, el uso de orquestadores y/o coreografía en el empleo de microservicios. Sin embargo, migrar de una arquitectura monolítica a una MSA es un proceso complejo que exige estrategias específicas para abordar los numerosos desafíos que surgen durante la transición (Martínez Saucedo, Rodríguez, Gomes Rocha, & Santos, 2025).

Es fundamental identificar y evaluar los marcos de migración, las estrategias utilizadas en la evolución de las APIs. Lercher hace un estudio de la evolución de las API, para él es el acoplamiento flexible mejora la capacidad de mantenimiento, la escalabilidad y la tolerancia a fallas , plantea nuevos desafíos para el proceso de evolución (Lercher, Glock, Macho, & Pinzger, 2024), la importancia de los orquestadores de MS, patrones de resiliencia que, aunque mitigan ciertos inconvenientes, incrementan la complejidad del desarrollo y el tiempo de ejecución. Además, es crucial identificar brechas entre las características más conocidas de una MSA y las prácticas reales en su implementación evaluando el desempeño del nuevo servicio.

El objetivo de nuestro estudio de mapeo sistemático es responder las siguientes preguntas de investigación (RQ):

RQ1: ¿Cuáles son los tipos de riesgos técnicos asociados con la migración a microservicios?

El propósito es identificar y clasificar los riesgos técnicos específicos que pueden surgir al separar las funcionalidades de un sistema monolítico en MS independientes.

RQ2: ¿Cuáles son los diferentes componentes de riesgos en la migración a microservicios?

Clasificamos varias formas de “componentes de riesgos” que se encuentran en la literatura respecto a la migración a microservicios, que generalmente se refiere a estos componentes utilizando varias palabras clave: amenaza, impacto, factores de riesgos, vulnerabilidad y daños.

RQ3: ¿Cómo afecta la aplicación de malas prácticas de desarrollo y diseño en la migración a microservicios?

Se identifica las malas prácticas de desarrollo y diseño en la migración a MS ya que estos pueden convertir una arquitectura distribuida compleja, cuyos costos y riesgos pueden superar los beneficios de rendimiento y escalabilidad.

2.2. Fuentes de información

La metodología utilizada en esta investigación es sobre la literatura académica publicada en revistas o conferencias. Con base en los resultados de la búsqueda, se deben seleccionar los artículos que proponen la migración a MS y que proporcionen evidencia relevante relacionada con estos casos (Su, Li, & Taibi, 2023). A la pregunta principal de investigación de este estudio es: “**¿Cuáles son los tipos de riesgo asociados en la migración a MS cuando se aplican malas prácticas de diseño?**”. Para responder a esta pregunta, se emplea el método SRL, llevando a cabo un proceso de selección de artículos que cumplan con los criterios establecidos. Siguiendo el proceso SLR tradicional, se realizaron búsqueda basada en la compatibilidad con palabras clave definidas en bases de datos académicas relevantes IEEE Xplore, AcienceDirect, Scopus y Google Scholar. La búsqueda en Google Scholar se limitó a las primeras páginas de resultados, con el objetivo de abarcar un espectro amplio de fuentes relevantes. Este enfoque buscó maximizar la exhaustividad y profundidad de la revisión bibliográfica. Esta revisión permite identificar tendencias en los temas de investigación que han sido de gran interés para estudios previos, sirviendo como referencia para futuras investigaciones.

2.3. Estrategia de búsqueda

Como estrategia de búsqueda se emplea las palabras claves y frases específicas relacionadas tanto con el riesgo de migrar a MS como la migración de sistemas monolíticos. Aunque no se facilitan las cadenas de búsqueda exactas, se deduce que abarcan variaciones y sinónimos de términos básicos empleando operadores booleanos; “OR” y “AND”, donde las palabras clave sean utilizadas. Siguiendo las pautas de PRISMA 2020, utilizamos frases específicas para localizar y filtrar artículos académicos en las bases de datos seleccionadas. Las frases empleadas fueron las siguientes:

“RISK TYPE”, “RISK COMPONENT”, “RISK TYPE A MIGRATE TO MICROSERVICE”, “RISK COMPONENT A MIGRATE TO MICROSERVICE”, “MIGRATE MONOLITHIC SYSTEMS” y “MIGRATE TO MICROSERVICE”. Los artículos elegidos para su revisión debían mencionar “migración a microservicios” y “migración de sistemas monolíticos” en su título, resumen o enfoque de investigación. Se excluyeron del estudio libros, artículos de conferencias, informes comerciales, artículos de opinión y editoriales. Además, se excluye artículos que no incluían las palabras claves “tipo de riesgos” y “componentes de riesgo” como parte de su investigación o contexto. En la extracción de los datos requiere responder las preguntas solicitadas adaptando y fusionando las categorías proporcionadas por los artículos seleccionados (Su et al., 2023). La estrategia de búsqueda de literatura se basa en el uso de una herramienta de inteligencia de negocios (BI) de código abierto, conocida como Metabase, para gestionar la información. Se seleccionarán los artículos que coincidan con las palabras clave definidas como criterios de elegibilidad, mientras que aquellos que no sean relevantes serán excluidos y las entradas duplicadas e irrelevantes se eliminaron automáticamente. También se realizó una selección manual.

2.4. Proceso de selección de los estudios

Se seleccionaron y revisaron artículos relevantes para elaborar un resumen de cada uno. La recopilación incluyó publicaciones de los últimos diez años, desde 2015 hasta 2024, así como aquellos programados para publicarse en 2025, sumando un total de 802 artículos. En la Figura 1 se muestra la distribución anual de los 99 artículos seleccionados en el periodo de 2017 a 2025.

La búsqueda en las bases de datos bibliográficas (ver Tabla 1) identificó 802 títulos de artículos, los cuales se importaron al software Mendeley para su gestión. Durante este proceso, se eliminaron 77 duplicados, 362 artículos marcados como irrelevantes mediante herramientas automatizadas y 264 artículos que no contenían en sus títulos las palabras clave 'microservice', 'monolithic', 'migration', 'migrate to microservice', 'migrate monolithic systems' o 'migration of monolithic systems to microservice'.

Figura 1

Número de estudios realizado a lo largo de los últimos 10 años

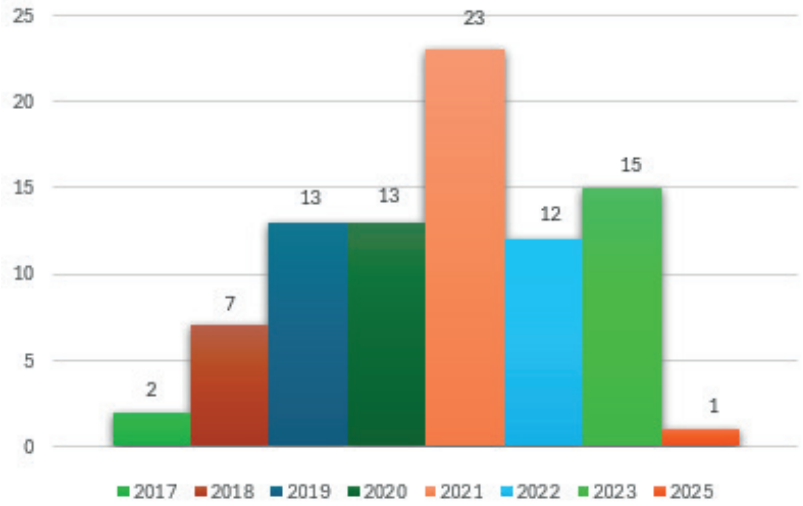


Tabla 1

Número de publicaciones por fuentes de bases de datos.

No.	Fuente de base de datos	# Publicaciones	# Títulos	# Resumen y palabra clave	# Seleccionados
1	IEEE Xplore	578	292	159	22
2	ScienceDirect	4,111	400	113	24
3	Scopus	1,074	89	73	43
4	Google Scholar	33	21	18	10
	Total	**5,796**	**802**	**363**	**99**

Este análisis, basado en el método de SLR, se enfocó en identificar investigaciones relacionadas con la migración de sistemas monolíticos a microservicios. Como resultado, se obtuvo un conjunto de 363 artículos seleccionados, los cuales se organizaron en una hoja de cálculo para su evaluación manual.

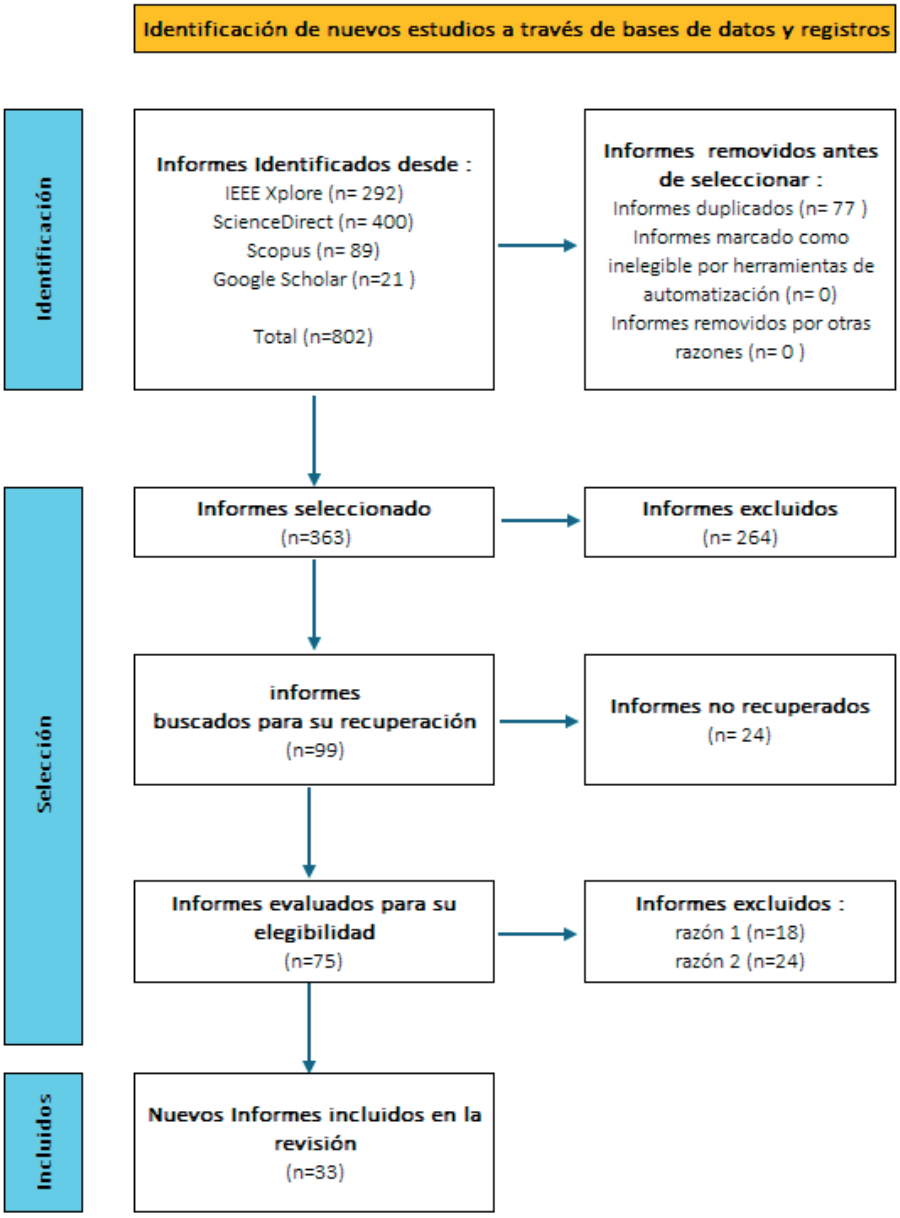
Tras analizar los títulos en función de nuestras preguntas y criterios de investigación, se descartaron los que no eran relevantes para el tema, quedando finalmente 99 artículos, de los cuales solo 75 estaban disponibles en texto completo (PDF). Los hallazgos de este análisis concluyente se detallarán en la sección de RESULTADO del documento. Como se indica en la Figura 2, los 42 artículos que se omitieron fueron excluidos por las siguientes razones: (1) Se excluyeron los artículos que se centraban exclusivamente en el uso de orquestadores de microservicios, lo que resultó en la exclusión de 18 documentos. (2) Se excluyeron 24 artículos adicionales porque se centraban principalmente en otros aspectos de los MS sin brindar un énfasis sustancial respecto a la migración de sistemas a MSA. Este criterio garantizó que se excluyeran los estudios que no se centraran sustancialmente en las técnicas de aprendizaje automático, manteniendo así el énfasis de la revisión en el impacto del aprendizaje automático en el desarrollo de ciudades inteligentes. La metodología PRISMA 2020 se muestra en la Figura 2, y el procedimiento detallado se describe en el proceso de selección.

2.5. Proceso de extracción de los datos

Se empleó herramientas de inteligencia artificial (IA); Wordtune Read y ChatGPT, mejorando el proceso de revisión y la obtención de resúmenes de los artículos seleccionados de forma más eficiente. Con el empleo de Wordtune Read se redujo los artículos a versiones de aproximadamente 2000 palabras, mientras que ChatGPT condensó aún más estos resúmenes en una versión de una sola página mediante el comando "crear un resumen de una página de este texto". Sin embargo, durante el

Figura 2

Diagrama de flujo PRISMA 2020, empleado para la búsqueda de artículos académicos sobre migración de sistemas monolíticos a microservicios.



proceso de selección se presentó ambigüedad en algunos resúmenes generados por IA que requerían una revisión manual más detallada. Además, la variabilidad en la calidad de los artículos también presentó retos, algunos de ellos carecían de datos para ser considerados relevantes. A pesar de ello, el enfoque sistemático ayudó a superar las dificultades y asegurar una selección precisa y significativa de literatura para nuestro estudio. El proceso de revisión manual involucró un análisis de cada resumen generado por la IA, comparándolo con el texto original para identificar posibles discrepancias o ambigüedades.

III. RESULTADO

Los MS representan un enfoque para desarrollar aplicaciones como un conjunto de servicios pequeños e independientes. Según Martin Fowler, cada servicio opera en su propio proceso y se comunica mediante mecanismos ligeros, comúnmente a través de APIs basadas en HTTP (Martin Fowler, 2014). Una revisión mediante el enfoque de literatura integradora PRISMA 2020 permite identificar los riesgos asociados con la migración de aplicaciones a una arquitectura de microservicios. El análisis de los 33 estudios seleccionados tras aplicar el

diagrama PRISMA 2020 reveló cuatro temas principales, cada uno de los cuales aporta al objetivo general de comprender los riesgos involucrados en el proceso de migrar a microservicios.

(1) Migración a microservicios: Durante la transformación de un sistema monolítico hacia una arquitectura basada en microservicios, la organización cambia su enfoque hacia un modelo más modular. Las aplicaciones se descomponen en componentes más pequeños e independientes, conocidos como microservicios, que pueden desarrollarse, implementarse y gestionarse de manera autónoma, lo que facilita la escalabilidad y la flexibilidad en el desarrollo.

(2) Arquitectura de los microservicios: Las soluciones informáticas basadas en arquitecturas modernas, en contraste con la arquitectura monolítica, son ampliamente adoptadas por organizaciones que buscan mayor agilidad, flexibilidad y escalabilidad en sus aplicaciones. Entre estas, la MSA destaca por su enfoque de diseño, que divide una aplicación en un conjunto de servicios pequeños e independientes, los cuales se comunican entre sí, generalmente mediante APIs, permitiendo una gestión y desarrollo más eficiente.

(3) Límites de microservicios: Los límites en MS determinan cómo se dividen las funcionalidades del sistema en unidades independientes. Una definición adecuada de estos límites es esencial para aprovechar las ventajas de esta arquitectura, mientras que una definición deficiente puede resultar en un sistema complejo y difícil de gestionar.

(4) Riesgos migración / implementación de microservicios: Consiste en reconocer y documentar los posibles eventos o condiciones que podrían afectar negativamente un proyecto, proceso o actividad. Este análisis permite anticiparse a problemas potenciales, aumentando la probabilidad de éxito de los objetivos planteados. El resultado de la identificación de riesgos suele plasmarse en un registro o matriz de riesgos, donde se describen detalladamente los riesgos, sus posibles causas, consecuencias y las áreas afectadas.

Estos cuatro temas emergieron del análisis PRISMA 2020 como las áreas más críticas y recurrentemente tratadas en la literatura sobre la migración de aplicaciones monolíticas a MS. La revisión detallada identificó estos temas como pilares fundamentales para reconocer y clasificar los riesgos asociados con la implementación de soluciones basadas en microservicios. Cada tema aborda desafíos específicos en los proyectos de migración y resalta el potencial de los MS para desarrollar soluciones viables, escalables y resilientes.

3.1. Migración a microservicios

Según Saucedo y otros explican que en el área de la migración hacia microservicios, la mayoría de los estudios se han centrado en las técnicas o actividades de migración y los factores que motivan la migración hacia una MSA (Martínez Saucedo et al., 2025), esto se debe a que la migración a una MSA permite que cada componente se implemente, escale y pruebe de forma independiente, con un propósito específico para cada uno. Otras razones por las que las empresas migran a microservicio son:

1. Porque los MS tienen escalabilidad, los MS ofrecen la capacidad de escalar componentes individuales del sistema de manera independiente. Esto no solo mejora la eficiencia operativa, sino que también contribuye a la reducción de costos.

2. Porque al migrar a MS se tiene autonomía, los MS cuentan con la flexibilidad tecnológica, cada servicio puede ser desarrollado con la tecnología más adecuada para su función, sin depender de una única tecnología.

3. Porque los MS facilitan la actualización y mantenimiento sin afectar el resto de funcionalidades del negocio, Los MS suelen ser más pequeños y específicos en sus responsabilidades, lo que facilita su mantenimiento y mejora el entendimiento del código.

Como lo afirma Lenarduzzi y otros, la tendencia de las empresas a migrar hacia MS se basa en aprovechar sus beneficios, como el desarrollo, despliegue y escalado independientes de cada servicio. Esto permite al sistema abordar problemas de manera más eficiente, mejorar su calidad y facilitar el mantenimiento del software (Lenarduzzi, Lomio, Saarimäki, & Taibi, 2020a). Sin embargo, cuando una aplicación alcanza un nivel de complejidad que dificulta su mantenimiento y ralentiza la implementación de cambios, se vuelve evidente la necesidad de modernizarla. Otra opinión al respecto destaca Xili Wan y otros autores, los cuales estudiaron el problema de programación de MS con el objetivo de maximizar los ingresos de la implementación de MS,

diseñaron enfoques genéticos para determinar la cantidad de recursos asignados a cada microservicio y cómo escalar de manera eficiente mientras cambia la carga de trabajo (Wan, Guan, Wang, Bai, & Choi, 2018). Estos desafíos se agravan si las empresas buscan integrar tecnologías recientes para mantenerse competitivas. En este contexto, la migración a una MSA surge como una solución estratégica para mejorar la agilidad, escalabilidad y capacidad de adaptación de la aplicación. Sin embargo, no existe un marco conceptual para evaluar la calidad de estas aplicaciones descompuestas (Cojocaru, Oprescu, & Uta, 2019).

3.1.1. Deuda técnica

Según lo documenta Lenarduzzi y otros, El concepto de deuda técnica (TD) fue introducido por Cunningham en 1992, describiéndolo como el costo generado al acelerar el desarrollo de software, lo que conduce a deficiencias que aumentan los costos de mantenimiento. Posteriormente, McConnell en 2013 refinó esta definición, explicándola como un enfoque de diseño o construcción que, aunque conveniente a corto plazo, incrementa los costos técnicos a lo largo del tiempo. Más tarde, Avgeriou en 2016 ampliaron el concepto, definiendo la DT como una acumulación de decisiones de diseño o implementación que, aunque útiles inicialmente, dificultan o encarecen los cambios futuros (Lenarduzzi, Lomio, Saarimäki, & Taibi, 2020b).

El comentario de los autores abarca el origen de una metáfora, su función como respaldo a la refactorización y la corrección de interpretaciones erróneas frecuentes. En cuanto al impacto de la DT en MS, se sugiere que la autoadaptación podría ser una solución eficaz para reducirla en sistemas basados en esta arquitectura, especialmente al abordar los antipatrones y el código sucio que suelen estar presentes. Como respuesta a este desafío, se propone un enfoque para identificar la DT en sistemas monolíticos antes de su migración, con el objetivo de asegurar que el nuevo sistema basado en MS cumpla con los requisitos necesarios.

3.1.2. Antipatrones

En el diseño de sistema se puede evaluar de manera diferente y según la arquitectura y/o contexto en el que se aplique, lo que podría ser considerado un antipatrón en un sistema monolítico, donde las aplicaciones son grandes y rígidas, podría ser una práctica adecuada en arquitecturas más modernas, como MS o SOA (Arquitectura Orientada a Servicios), esto en el contexto de la evolución de las arquitecturas de software. Según Cerny y otros, para facilitar el desarrollo de sistemas, los profesionales e investigadores han identificado diversas soluciones comprobadas para problemas de diseño recurrentes y comunes. Estas soluciones son comúnmente conocidas como patrones de diseño (Cerny, Abdelfattah, Maruf, Janes, & Taibi, 2023).

Los antipatrones se presentan comúnmente a la falta de experiencia o conocimiento del equipo de trabajo, lo que puede llevar a tomar decisiones subóptimas durante el diseño o la implementación. Otra causa es la falta de planificación, que puede dar lugar a la adopción de soluciones apresuradas o mal fundamentadas. Además, el desconocimiento o la ignorancia de las mejores prácticas y principios establecidos en el ámbito del desarrollo de software o gestión de proyectos, como el uso adecuado de arquitecturas, patrones de diseño o metodologías ágiles, puede resultar en la implementación de soluciones que a largo plazo generen problemas, incrementen costos o dificulten la escalabilidad y mantenimiento del sistema. Tighilt presenta un análisis en el cual considera 16 antipatrones, estos fueron elegidos por él y sus colaboradores porque se pueden detectar en el código fuente de los sistemas basados en MS (Tighilt et al., 2023). En la Tabla 2 se listan estos antipatrones identificados por Tighilt.

3.2. Arquitectura de los microservicios

La MSA se plantea como una solución eficaz para escalar aplicaciones tradicionales y se basan en conceptos de la arquitectura orientada a servicios (SOA), cuyo enfoque principal consiste en dividir los servicios en relación con las capacidades comerciales (Cojocaru et al., 2019). En este enfoque, cada funcionalidad del negocio se encapsula en un MS independiente, siguiendo el principio de responsabilidad única. Aunque no siempre es obligatorio, se recomienda que cada MS gestione su propia base de datos, promoviendo una mayor autonomía y desacoplamiento entre los componentes. Según Velepucha y Flores una arquitectura MS también presenta desventajas, como un aumento en la complejidad a medida que crece el número de MS. Además, realizar pruebas funcionales y técnicas se vuelve más complicado, ya que una solicitud puede atravesar múltiples microservicios. Esto, a su vez, puede generar mayores tiempos de respuesta y aumentar la latencia en la red (Velepucha & Flores, 2021). También, menciona las complejidades en las que se ve expuesta al emplear MS como parte de una solución a las necesidades de cambio. Sin embargo, este enfoque también

Tabla 2

Listado de los 16 antipatrones identificados por Rafik Tighilt y otros autores en su artículo; On the maintenance support for microservice-based systems through the specification and the detection of microservice antipatterns (Tighilt et al., 2023).

Nro	Descripción Español	Descripción Inglesa
1	Cortes incorrectos	Wrong Cuts
2	Dependencias cíclicas	Cyclic Dependencies
3	Megaservicio	Mega Service
4	Nanoservicio	Nano Service
5	Bibliotecas compartidas	Shared Libraries.
6	Puntos finales codificados de forma rígida	Hard-Coded Endpoints
7	Configuración manual	Manual Configuration
8	Sin integración continua	No Continuous Integration
9	Sin API Gateway	No API Gateway
10	Tiempos de espera	Timeouts
11	Varias instancias de servicio por host	Multiple Service Instances Per Host
12	Persistencia compartida	Shared Persistence
13	Sin control de versiones de API	No API Versioning
14	Sin comprobación de estado	No Health Check
15	Registro local	Local Logging
16	Supervisión insuficiente	Insufficient Monitoring

puede generar una mayor cantidad de MS, lo que aumenta la probabilidad de fallos en la integración, debido a la complejidad de mantener disponibles de forma instantánea numerosos servicios, lo que podría afectar la disponibilidad general del sistema (Dragoni et al., 2016).

La realización de pruebas funcionales y técnicas también se vuelve más desafiante, ya que una solicitud puede atravesar varios microservicios. Esto, a su vez, puede ocasionar tiempos de respuesta más largos y una mayor latencia en la red. Según Cojocaru, la MSA se basa principalmente en el diseño basado en el dominio, una técnica en la que cada MS debe ofrecer una funcionalidad específica y limitada que refleje la capacidad empresarial especificada a través de límites claros del servicio (Cojocaru et al., 2019). Al respecto debemos considerar que con la aplicabilidad de técnicas de descomposición que permite controlar la granularidad y los atributos de calidad podemos manejar los principios de cohesión y acoplamiento presentes en la arquitectura. Según Taibi y otros, la migración a MS requiere desarrolladores más experimentados (Taibi, Lenarduzzi, & Pahl, 2017).

3.2.1. Seguridad

La MSA requiere implementar medidas de seguridad robustas, ya que estos sistemas pueden ser vulnerables a la inyección de código malicioso, lo que podría provocar bloqueos o comprometer su integridad. Hannousse y otros, mencionan que en el intercambia de datos entre MS a través de buses de eventos pueden ser interceptados y alterados por personas malintencionadas. Por lo tanto, proteger los canales de comunicación entre MS es obligatorio para proteger los sistemas basados en MS (Hannousse & Yahiouche, 2021). Es fundamental adoptar estrategias que garanticen la ejecución exclusiva de MS confiables y protejan tanto contra ataques internos como externos. En cuanto a las conexiones entre MS, su pérdida o la configuración insegura de un solo servicio puede comprometer la seguridad del sistema completo. Además, los ataques dirigidos a las APIs tienen el potencial de eludir las medidas de seguridad tradicionales proporcionadas por las puertas de enlace, lo que podría permitir a usuarios malintencionados acceder y controlar activos críticos del sistema.

3.2.2. Contenedores

En la industria del software basado en plataformas modernas, garantizar la portabilidad y consistencia de las aplicaciones es una tarea esencial, especialmente en entornos complejos o distribuidos. Según Sing, hay dos enfoques principales para la virtualización: máquinas virtuales (VM) y contenedores. Los contenedores son fáciles de administrar debido a su peso ligero; superan a las VM en muchos aspectos (Vindeep Sing, 2017). Entonces, los contenedores, se han convertido en una solución fundamental al implementar microservicios, hace que estas unidades ejecutables de software encapsulan el código de

una aplicación junto con todas sus bibliotecas y dependencias necesarias, proporcionando un entorno uniforme para su ejecución en cualquier plataforma. Según lo investigado por Bentaleb, potenciar el uso de tecnologías de contenedores sobre infraestructura de computación distribuida, hace necesario habilitar una tubería de paralelismo para su ejecución (Bentaleb, Sebaa, Kalli, & Belloum, 2022). Permitiendo a los desarrolladores simplificar procesos como la prueba y el despliegue, reduciendo las discrepancias entre entornos.

3.2.3. Orquestadores / Coreografía

Los orquestadores de MS son herramienta o framework que gestiona, organiza y coordina la ejecución de múltiples MS dentro de una arquitectura basada en este enfoque. Su objetivo principal es optimizar y automatizar el despliegue, escalado, comunicación, monitoreo y recuperación de fallos de los MS. Stutz señala que; además de la orquestación, existe un segundo método de asociación, la coreografía. El enfoque de esta contribución es la cuestión de cómo se puede aplicar la coreografía en el contexto de la automatización (Stutz, Fay, Barth, & Maurmaier, 2020).

Una de las desventajas de los orquestadores en los MS es su dependencia, lo que limita su autonomía y flexibilidad de los servicios. Además, un fallo en el orquestador puede comprometer todo el sistema, convirtiéndose en un posible cuello de botella si no se escala correctamente. Con el tiempo, el orquestador puede ganar complejidad, dificultando su gestión y evolución. Mientras en la coreografía, cada MS interactúa de manera independiente mediante eventos, reaccionando a los cambios del sistema. Una de sus desventajas es en el flujo del sistema el cual es difícil de seguir, ya que no existe un control central que coordine los servicios. Diseñar un sistema coreografiado puede requerir una planificación cuidadosa y herramientas específicas para gestionar los eventos.

3.3. Límites de microservicios

Los límites de los MS se refieren a las fronteras que separan las funcionalidades o componentes individuales en una arquitectura de microservicios. Según Waseem y otros, observa que los profesionales de MSA consideran que "definir claramente los límites de los microservicios" es el desafío más crítico al diseñar sistemas de MS (Waseem, Liang, Shahin, Di Salle, & Márquez, 2021). Por lo tanto; definir correctamente estos límites es crucial para garantizar una arquitectura modular, escalable y fácil de mantener. Cada MS debe ser responsable de una funcionalidad o conjunto de funcionalidades específicas y autónomas, esto se basa en el concepto de contextos delimitados del diseño orientado a dominios (DDD), donde cada MS opera dentro de un dominio bien acotado. Singh y Peddoju definen que cualquier actualización de un MS o la incorporación de cualquier nuevo MS a la aplicación debe integrarse con la aplicación con un mínimo esfuerzo y un tiempo de inactividad mínimo para la aplicación (Vindeep Sing, 2017). Consecuente con ello, los MS deben ser independientes en términos de desarrollo, despliegue y escalado, sus límites garantizan que no dependan directamente de la implementación interna de otros servicios, interfaces bien definidas, como APIs o sistemas de mensajería.

3.4. Riesgos migración / implementación de microservicios

El propósito de emplear una metodología para identificar, evaluar y mitigar los riesgos asociados con la migración de aplicaciones monolíticas a una MSA es garantizar una transición controlada, eficiente y alineada con la estabilidad, seguridad o rendimiento de las aplicaciones. En la Figura 3 se presenta la metodología para la gestión del riesgo en la migración a MS, de manera efectiva, asegurando una transición más fluida y controlada. Según Rosado, la mayoría de los enfoques tratan de integrar nuevos artefactos para los modelos de procesos de negocio para apoyar el análisis de riesgos, pero a veces, la notación puede aumentar la complejidad, lo que dificulta tener una herramienta de gestión de riesgos para apoyar el análisis (Rosado et al., 2024).

El riesgo en un proyecto de migración es identificado por cualquier miembro involucrado, y su declaración debe incluir tanto una condición como una consecuencia, la posibilidad de reutilizar el conocimiento adquirido durante los análisis de riesgos previos, siendo imprescindible contar con herramientas de apoyo que automaticen tareas y faciliten el cumplimiento de las metodologías aplicadas (Rosado et al., 2024).

El análisis de riesgo se basa en tres indicadores; la probabilidad de ocurrencia, el impacto y el nivel de exposición (producto de la probabilidad por el impacto), utilizando una escala numérica del 1 al 10. Basado en el trabajo Maniah, el presente estudio identificará los tipos de riesgos y componentes de riesgo en la migración utilizando el método SRL (Maniah, Soewito, Lumban Gaol, & Abdurachman, 2022). Enfocado en ello, la planificación de riesgos implica definir acciones de mitigación (antes del

riesgo) y contingencia (después del riesgo), asignando responsables para cada acción.

En la Tabla 3, se presenta una matriz con la identificación de riesgos tipificando e identificando sus disparadores y en la Figura 4 se muestra su agrupación por tipo. El seguimiento se centra en evaluar y actualizar la probabilidad y el impacto de los riesgos, mientras que el control revisa la ejecución de las acciones para mitigar los riesgos o manejar las contingencias, lo que puede llevar a eliminar ciertos riesgos de la lista de administración. Rosado propone considerar una metodología de gestión de riesgos que incluye tres fases: identificación, evaluación y gestión de riesgos (Rosado et al., 2024). En la Tabla 4, se realiza una evaluación de los riesgos identificados.

Figura 3

Se muestra la metodología empleada para identificar y gestionar riesgo en la migración a MS. Basada en las mejores prácticas de ITIL 4, que aporta un elemento para una correcta gestión de riesgos (Sarah Vilarinho, 2013).

Escalas para la probabilidad

Probabilidad	Valores
Muy alto	9
Alto	7
Medio	5
Bajo	3
Muy bajo	1

Escalas para el impacto

Impacto	Valores
Muy alto	9
Alto	7
Medio	5
Bajo	3
Muy bajo	1

Probabilidad	Amenazas				
9	9	27	45	63	81
7	7	21	35	49	63
5	5	15	25	35	45
3	3	9	15	21	27
1	1	3	5	7	9
Impacto	**1**	**3**	**5**	**7**	**9**

Riesgo Alto
Riesgo Medio
Riesgo Bajo

$$\text{Riesgo de Migración} = \sum_{i=1}^{n} \frac{Pi \times Li}{n}$$

Donde:

Pi: Probabilidad del riesgo i
Li: Impacto del riesgo i

Tabla 3

Matriz de identificadores de riesgo al migrar y/o implementar MS

Cod.	Tipo	Disparador del riesgo	Descripción de Riesgo
R1	Migración	Al migrar a Microservicios	Aumento de la complejidad a medida que se incrementa el número de MS (Velepucha & Flores, 2021).
R2	Migración	Migración a Microservicios	Migrar a microservicios, requiere organizarse de manera que los diferentes equipos de trabajo ahora sean responsables de los servicios (Velepucha & Flores, 2021).
R3	Migración	Migración a Microservicios	Debido a la naturaleza altamente distribuida de los microservicios, a medida que aumenta su número, se vuelven más difíciles de administrar y monitorear que las aplicaciones monolíticas (Li et al., 2019).
R4	Monolítico	Descomposición de sistemas monolíticos	Un desafío clave al descomponer un sistema monolítico es determinar las particiones adecuadas, ya que la granularidad de la MSA puede influir notablemente en diversos atributos de calidad del sistema, como el rendimiento y la capacidad de ser probado (Li et al., 2019).
R5	Limitaciones	Limitaciones de escalabilidad y rendimiento de los lenguajes de programación	Muchos lenguajes de programación no están optimizados para los requisitos de rendimiento y escalabilidad de la arquitectura de microservicios, o no tienen marcos escalables o de alto rendimiento que permitan que los MS procesen tareas de manera eficiente (Susan J. Fowler, n.d.)
R6	Microservicios	lenguajes de programación optimizados	Al escribir un nuevo microservicio, asegúrese de que el lenguaje en el que se escribe el servicio no introduzca restricciones de escalabilidad y rendimiento en los MS (Susan J. Fowler, n.d.)
R7	Datos	*Manejo de los datos de una manera escalable y eficiente*	La forma en que un MS almacena y maneja los datos puede convertirse fácilmente en la limitación o restricción más importante que le impide ser escalable y eficiente (Susan J. Fowler, n.d.).
R8	Conectividad	Tener cuidado con las conexiones de bases de datos	Asegúrese de que todas las conexiones se cierren de manera adecuada para evitar comprometer la disponibilidad de un servicio y la disponibilidad de la base de datos para todos los MS que la usan (Susan J. Fowler, n.d.).
R9	Arquitectura	Implementación de una arquitectura de microservicio	Una MSA presenta desventajas, como un aumento en la complejidad a medida que crece el número de MS. Además, realizar pruebas funcionales y técnicas se vuelve más complicado, ya que una solicitud puede atravesar múltiples MS. Esto, a su vez, puede generar mayores tiempos de respuesta y aumentar la latencia en la red (Velepucha & Flores, 2021).
R10	Arquitectura	Adopción de la arquitectura de microservicio al migrar sistemas monolíticos.	Los MS necesitan comunicarse e intercambiar datos entre sí. Aunque la MSA ayuda a reducir la complejidad de la unidad al permitir una alta cohesión y un bajo acoplamiento, requiere plataformas de comunicación y mecanismos de implementación debido a la naturaleza distribuida (Karabey Aksakalli, Çelik, Can, & Tekinerdoğan, 2021).
R11	Arquitectura	Complejidad arquitectónica	Implementación de sistemas complejos con MS es un problema grave debido a la necesidad de implementar una gran cantidad de paquetes, interdependencias de servicios y la necesidad de que los componentes de software asignen los recursos computacionales disponibles de manera óptima (Karabey Aksakalli et al., 2021).
R12	Orquestación	Orquestación de servicios	Dificultad al descubrir el servicio apropiado en un sistema donde el número de servicios ha aumentado sustancialmente. Por lo tanto, se necesitan sistemas de orquestación de servicios efectivos (Karabey Aksakalli et al., 2021).
R13	Microservicios	Características de los sistemas de microservicios	Los sistemas de MS son cada vez más complejos y necesitan un monitoreo y pruebas más intensivos que otros tipos de sistemas (Waseem et al., 2021).
R14	Migración	En la migración de un sistema monolítico a microservicios	Se prioriza el desarrollo de nuevas funcionalidades por sobre la refactorización del código, lo que genera deuda técnica (TD) cada vez que se pospone una actividad y aumenta el costo de mantenimiento del software (Lenarduzzi et al., 2020b).
R15	Migración	En el proceso de migración	En los MS, al desarrollar nuevas aplicaciones, existen MS monolíticos construidos como una sola unidad, lo que endurece su desarrollo, implementación y mantenimiento (Selmadji et al., 2020).
R16	Microservicios	Implementación automática de microservicios.	Con una pequeña cantidad de MS, podría ser aceptable aprovisionar manualmente máquinas para implementarlos. Sin embargo, si esta cantidad aumenta, en algún momento, el uso de un enfoque manual podría no ser posible. Por lo tanto, se requiere una implementación automática (Selmadji et al., 2020).
R17	Acoplamiento	Microservicios altamente Acoplados	Presencia de cortes incorrectos en MS aumenta potencialmente el acoplamiento entre MS asociados, riesgo de exhibir antipatrones (Tighilt et al., 2023).
R18	Migración	Cuando migrar a microservicios	Momento inadecuado para que las empresas migren a MS sin evaluar las características del modelo de empresas o del software para beneficiarse de las ventajas de los MS (F Auer, Lenarduzzi, Felderer, & Taibi, 2021).
R19	Seguridad	Niveles de aplicación de seguridad de microservicios	La adopción de una MSA como diseño arquitectónico de aplicaciones distribuidas introduce vulnerabilidades de seguridad en diferentes capas arquitectónicas (Hannousse & Yahiouche, 2021).
R20	Costos Generales	Estimaciones utilizando una arquitectura basada en microservicios	Las MSA requieren maquinaria adicional, lo que puede suponer costos sustanciales. El desarrollo de un sistema basado en MS tiene costos iniciales más altos que los de desarrollar un sistema tradicional (Taibi et al., 2017).
R21	Microservicio	Actualizaciones de servicio con cero tiempo de inactividad	Los MS deben actualizarse constantemente por muchas razones, como correcciones de errores, incorporación de nuevas funciones, cambios en la lógica, etc (Vindeep Sing, 2017).

Figura 4

Agrupación por tipos de riesgos

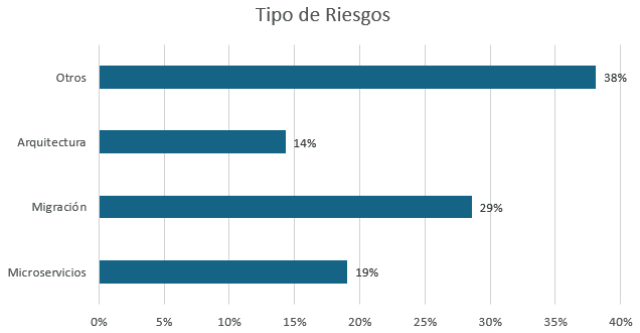


Tabla 4

Matriz de exposición del riesgo al migrar y/o implementar microservicios

Cod.	Probabilidad	Impacto	Nivel Exposición	Mitigación del Riesgo
R1	7	7	49	Implementar un diseño modular y estandarizado para los microservicios, acompañado de herramientas de monitoreo y gestión centralizada.
R2	9	5	45	Se recomienda implementar una estrategia clara de responsabilidad y coordinación, definiendo roles y responsabilidades y una cultura de colaboración interequipos.
R3	9	7	63	Diseñar arquitecturas con registros centralizados y observabilidad desde el inicio, incluyendo trazabilidad distribuida y métricas para detectar y resolver problemas.
R4	5	9	45	Implementar un análisis exhaustivo de los dominios del negocio y sus dependencias, utilizando técnicas como Domain-Driven Design y mapeo de contextos delimitados.
R5	5	5	25	Uso de lenguajes de programación y framework optimizados para la arquitectura de microservicios, priorizando aquellos que ofrezcan alto rendimiento y escalabilidad.
R6	3	5	15	Evalúe el impacto del lenguaje en el rendimiento general del sistema y realice pruebas de carga tempranas para identificar posibles limitaciones.
R7	5	5	25	Uso de bases de datos desacopladas, particionamiento, estrategias de caché, y garantizar que cada MS tenga autonomía en el manejo de sus datos.
R8	5	9	45	Implementar un mecanismo de gestión de conexiones adecuado que asegure el cierre correcto de conexiones a la base de datos (connection pooling y timeouts).
R9	9	7	63	Diseño modular y escalable, herramientas de pruebas automatizadas, monitoreo que detecten cuellos de botella y optimicen la latencia y la comunicación entre servicios.
R10	3	9	27	Implemente plataformas de mensajería robustas y mecanismos de integración eficientes, colas de mensajes o APIs RESTful con comunicación segura y escalable.
R11	7	5	35	Se debe realizar una arquitectura bien definida, con una correcta segmentación de servicios, minimizando dependencias innecesarias.
R12	7	7	49	Integrar herramientas que permitan una gestión centralizada y automatizada de los servicios, optimizando la búsqueda y selección del servicio adecuado.
R13	5	3	15	Establecer un sistema de pruebas automatizadas, Desarrollo iterativo y escalable e Infraestructura resiliente y monitoreo avanzado con visibilidad en tiempo real.
R14	7	9	63	Implemente un proceso de gestión técnica que contemple la refactorización continua como parte de la planificación de cada iteración del proyecto.
R15	3	5	15	Diseñe MS como una unidad autónoma, interfaz clara y responsabilidades específicas, asegurando independencia en términos de desarrollo y mantenimiento.
R16	7	9	63	A medida que la escala crece, la automatización se convierte en una necesidad para garantizar eficiencia, consistencia y capacidad de manejo del incremento de recursos.
R17	5	9	45	Cada microservicio tenga una única responsabilidad y esté lo más desacoplado posible de otros servicios. APIs robustas que minimicen el impacto de cambios.
R18	3	7	21	Evaluar la necesidad, migración alineada con objetivos estratégicos, asegurando que se cuente con la infraestructura adecuada y el equipo capacitado para llevarla a cabo.
R19	5	5	25	Autenticación y autorización robustas, comunicación segura mediante protocolos seguros TLS/SSL para prevenir ataques de interceptación y manipulación de datos.
R20	5	7	35	Se debe evaluar los costos a largo plazo y la viabilidad de utilizar recursos compartidos y automatización para reducir el impacto de la infraestructura adicional.
R21	5	9	45	Debido a correcciones de errores, nuevas funciones o cambios en la lógica, se recomienda implementar un proceso de gestión de versiones robusto.

IV. DISCUSIÓN

Esta revisión sistemática, guiada por el marco PRISMA 2020, exploró el universo de los MS y la migración de sistemas monolíticos. El análisis reveló cuatro temas destacados: migración a microservicios, arquitectura de microservicios, límites de MS y riesgos al migrar y/o implementar MS. Aunque la revisión ofrece valiosas perspectivas sobre los beneficios actuales al emplear estas tecnologías, surgen los riesgos que expone el uso de esta tecnología y que deberán ser analizadas futuras investigaciones. En la implementación de una migración empleado MS es inherentemente compleja y abarca múltiples aspectos tanto técnicos como organizacionales. Entre los principales desafíos se encuentra la necesidad de implementar tanto soluciones de orquestación como herramientas avanzadas de observabilidad, que permitan gestionar latencias, fallos en la comunicación y la sincronización de datos entre servicios, no obstante, esta arquitectura presenta algunas desventajas, como el aumento de la complejidad a medida que se incrementa el número de MS. Además, se requiere una infraestructura robusta para automatizar y gestionar despliegues frecuentes de manera eficiente, ya que con la identificación de estos problemas abordados resaltan la importancia de una planificación adecuada al migrar o desarrollar una arquitectura basada en MS. El proceso de migración implica varios pasos clave, como el análisis de la arquitectura existente, la identificación de límites funcionales, la adopción de herramientas de contenedores (como Docker) y plataformas de orquestación (como Kubernetes), y la implementación de prácticas de DevOps para optimizar la gestión del ciclo de vida de los servicios. El análisis de los riesgos asociados al uso de tecnologías modernas, detallados en la Tabla 4 y la Tabla 5, se fundamenta en las opiniones de los autores de los artículos revisados. Este enfoque ha sido clave para llevar a cabo la presente investigación, permitiendo identificar aspectos críticos a considerar en la implementación de MS y aportando un marco integral para evaluar sus implicancias en entornos tecnológicos.

V. CONCLUSIONES

El presente trabajo de investigación tuvo como objetivo analizar los riesgos asociados con la migración hacia arquitecturas basadas en MS, con el fin de facilitar y agilizar la identificación de riesgos en el diseño de aplicaciones y en la implementación de MSA. Para ello, se desarrolló una matriz de riesgos que permite identificar y evaluar los riesgos inherentes a este proceso de migración, así como las oportunidades relacionadas con su implementación de MS. Los resultados obtenidos en este estudio constituyen un aporte valioso para los usuarios y profesionales que trabajan con MS, proporcionando un marco de referencia práctico y útil para la gestión de este tipo de migraciones. La experimentación realizada resalta la relevancia de los MS desde la perspectiva propuesta. No obstante, se considera esencial fortalecer estos hallazgos mediante revisiones adicionales aplicadas a migraciones de mayor escala. Asimismo, se plantea reforzar la validación de los resultados mediante la comparación con herramientas de última generación y metodologías avanzadas o framework especializados en la implementación de arquitecturas basadas en MS.

REFERENCIAS

[1] Al-Ruithe, M. . B. E. & H. K. A. (2019). A systematic literature review of data governance and cloud data governance.

[2] Auer, F, Lenarduzzi, V., Felderer, M., & Taibi, D. (2021). From monolithic systems to Microservices: An assessment framework. *Information and Software Technology*, *137*. https://doi.org/10.1016/j.infsof.2021.106600

[3] Auer, Florian, Lenarduzzi, V., Felderer, M., & Taibi, D. (2021). From monolithic systems to Microservices: An assessment framework. *Information and Software Technology*, *137*, 106600. https://doi.org/10.1016/J.INFSOF.2021.106600

[4] Bentaleb, O., Sebaa, A., Kalli, S., & Belloum, A. S. Z. (2022). Deployment of a programming framework based on microservices and containers with application to the astrophysical domain. *Astronomy and Computing*, *41*, 100655. https://doi.org/10.1016/J.ASCOM.2022.100655

[5] Cerny, T., Abdelfattah, A. S., Bushong, V., Al Maruf, A., & Taibi, D. (2022). Microservice Architecture Reconstruction and Visualization Techniques: A Review. *Proceedings - 16th IEEE International Conference on Service-Oriented System Engineering, SOSE 2022*, 39–48. https://doi.org/10.1109/SOSE55356.2022.00011

[6] Cerny, T., Abdelfattah, A. S., Maruf, A. Al, Janes, A., & Taibi, D. (2023). Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. *Journal of Systems and Software*, *206*. https://doi.org/10.1016/j.jss.2023.111829

[7] Cojocaru, M.-D., Oprescu, A., & Uta, A. (2019). Attributes assessing the quality of microservices automatically decomposed from monolithic applications. In *Proceedings - 2019 18th International Symposium on Parallel and Distributed Computing, ISPDC 2019* (pp. 84–93). https://doi.org/10.1109/ISPDC.2019.00021

[8] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2016). Microservices: yesterday, today, and tomorrow. Retrieved from http://arxiv.org/abs/1606.04036

[9] Esparza-Peidro, J., Muñoz-Escoí, F. D., & Bernabéu-Aubán, J. M. (2024). Modeling microservice architectures. *Journal of Systems and Software*, *213*, 112041. https://doi.org/10.1016/J.JSS.2024.112041

[10] Hannousse, A., & Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study. *Computer Science Review*, *41*, 100415. https://doi.org/https://doi.org/10.1016/j.cosrev.2021.100415

[11] Karabey Aksakalli, I., Çelik, T., Can, A. B., & Teknerdoğan, B. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Systems and Software*, *180*, 111014. https://doi.org/https://doi.org/10.1016/j.jss.2021.111014

[12] Lelovic, L., Huzinga, A., Goulis, G., Kaur, A., Boone, R., Muzrapov, U., … Cerny, T. (2024). Change impact analysis in microservice systems: A systematic literature review. *Journal of Systems and Software*, 112241. https://doi.org/10.1016/J.JSS.2024.112241

[13] Lenarduzzi, V., Lomio, F., Saarimäki, N., & Taibi, D. (2020a). Does migrating a monolithic system to microservices decrease the technical debt? *Journal of Systems and Software*, *169*, 110710. https://doi.org/10.1016/J.JSS.2020.110710

[14] Lenarduzzi, V., Lomio, F., Saarimäki, N., & Taibi, D. (2020b). Does migrating a monolithic system to microservices decrease the technical debt? *Journal of Systems and Software*, *169*, 110710. https://doi.org/https://doi.org/10.1016/j.jss.2020.110710

[15] Lercher, A., Glock, J., Macho, C., & Pinzger, M. (2024). Microservice API Evolution in Practice: A Study on Strategies and Challenges. *Journal of Systems and Software*, *215*, 112110. https://doi.org/10.1016/J.JSS.2024.112110

[16] Li, S., Zhang, H., Jia, Z., Li, Z., Zhang, C., Li, J., … Shan, Z. (2019). A dataflow-driven approach to identifying microservices from monolithic applications. *Journal of Systems and Software*, *157*, 110380. https://doi.org/https://doi.org/10.1016/j.jss.2019.07.008

[17] Maniah, Soewito, B., Lumban Gaol, F., & Abdurachman, E. (2022). A systematic literature Review: Risk analysis in cloud migration. *Journal of King Saud University - Computer and Information Sciences*, *34*(6), 3111–3120. https://doi.org/10.1016/J.JKSUCI.2021.01.008

[18] Martin Fowler. (2014, March 25). Microservices.

[19] Martínez Saucedo, A., Rodríguez, G., Gomes Rocha, F., & Santos, R. P. dos. (2025). Migration of monolithic systems to microservices: A systematic mapping study. *Information and Software Technology*, *177*, 107590. https://doi.org/10.1016/J.INFSOF.2024.107590

[20] Matias, M., Ferreira, E., Mateus-Coelho, N., & Ferreira, L. (2024). Enhancing Effectiveness and Security in Microservices Architecture. *Procedia Computer Science*, *239*, 2260–2269. https://doi.org/10.1016/J.PROCS.2024.06.417

[21] Rosado, D. G., Sánchez, L. E., Varela-Vaca, Á. J., Santos-Olmo, A., Gómez-López, M. T., Gasca, R. M., & Fernández-Medina, E. (2024). Enabling security risk assessment and management for business process models. *Journal of Information Security and Applications*, *84*, 103829. https://doi.org/10.1016/J.JISA.2024.103829

[22] Sarah Vilarinho, M. M. da S. (2013). Risk Management Model in ITIL. *Sociotechnical Enterprise Information Systems Design and Integration*. Retrieved from https://www.igi-global.com/gateway/chapter/75882

[23] Selmadji, A., Seriai, A.-D., Bouziane, H. L., Oumarou Mahamane, R., Zaragoza, P., & Dony, C. (2020). From monolithic architecture style to microservice one based on a semi-automatic approach. In *Proceedings - IEEE 17th International Conference on Software Architecture, ICSA 2020* (pp. 157–168). https://doi.org/10.1109/ICSA47634.2020.00023

[24] Stutz, A., Fay, A., Barth, M., & Maurmaier, M. (2020). Orchestration vs. Choreography Functional Association for Future Automation Systems. *IFAC-PapersOnLine*, *53*(2), 8268–8275. https://doi.org/10.1016/J.IFACOL.2020.12.1961

[25] Su, R., Li, X., & Taibi, D. (2023). Back to the Future: From Microservice to Monolith. Retrieved from http://arxiv.org/abs/2308.15281

[26] Susan J. Fowler. (n.d.). Production-Ready Microservice. Retrieved December 3, 2024, from https://www.oreilly.com/library/view/production-ready-microservices/9781491965962/ch04.html

[27] Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. *IEEE Cloud Computing*, *4*(5), 22–32. https://doi.org/10.1109/MCC.2017.4250931

[28] Tighilt, R., Abdellatif, M., Trabelsi, I., Madern, L., Moha, N., & Guéhéneuc, Y. G. (2023). On the maintenance support for microservice-based systems through the specification and the detection of microservice antipatterns. *Journal of Systems and Software*, *204*, 111755. https://doi.org/10.1016/J.JSS.2023.111755

[29] Velepucha, V., & Flores, P. (2021). Monoliths to microservices-migration problems and challenges: A SMS. In *2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)* (pp. 135–142).

[30] Vindeep Sing, S. K. P. (2017). Container-based Microservice Architecture for Cloud Applications. In *International Conference on Computing, Communication and Automation (ICCCA2017)*.

[31] Wan, X., Guan, X., Wang, T., Bai, G., & Choi, B.-Y. (2018). Application deployment using Microservice and Docker containers: Framework and optimization. *Journal of Network and Computer Applications*, *119*, 97–109. https://doi.org/https://doi.org/10.1016/j.jnca.2018.07.003

[32] Waseem, M., Liang, P., Shahin, M., Di Salle, A., & Márquez, G. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software*, *182*, 111061. https://doi.org/https://doi.org/10.1016/j.jss.2021.111061

[33] Yepes-Nuñez, J. J., Urrútia, G., Romero-García, M., & Alonso-Fernández, S. (2021). Declaración PRISMA 2020: una guía actualizada para la publicación de revisiones sistemáticas. *Revista Española de Cardiología*, *74*(9), 790–799. https://doi.org/10.1016/J.RECESP.2021.06.016

Financiamiento:

Propio.

Conflictos de interés:

El autor declara no tener conflicto de interés.