PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Mapeo sistemático de la literatura acerca de patrones de microservicios empleados en proyectos DevOps

Trabajo de investigación para obtener el grado académico de Maestra en Informática con mención en Ingeniería de Software que presenta:

Lourdes Mercedes Guzmán Guillén

Asesor:

*Eder **Ramiro** Quispe Vilchez*

Lima, 2024

Informe de Similitud

Yo, ***Eder Ramiro QUISPE VILCHEZ***, docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesor de la tesis titulada "Mapeo sistemático de la literatura acerca de patrones de microservicios empleados en proyectos DevOps" de la autora **Lourdes Mercedes Guzmán Guillén**, dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 21%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 1/08/2024.
- He revisado con detalle dicho reporte y la tesis, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha:

San Miguel, 1 de Agosto de 2024.

Apellidos y nombres del asesor / de la asesora: ***QUISPE VILCHEZ, Eder Ramiro***	
DNI: 42264307	Firma
ORCID: 0000-0003-1639-5134	

DEDICATORIA

A mi hija Luana, quien, sin saberlo, se ha convertido en mi más grande maestra de vida.

AGRADECIMIENTO

Me siento profundamente agradecida y bendecida por haber tenido el apoyo de mi familia durante este proceso de aprendizaje y crecimiento. Este logro no solo es mío, es también el reflejo de su sacrificio y su respaldo constante.

Además, quisiera agradecer a mi asesor, Eder Quispe, por su apoyo y orientación a lo largo de este trabajo de investigación. Las herramientas y conocimientos que me brindó fueron esenciales para superar los retos que se presentaron en el camino.

RESUMEN

En los últimos años, el software se ha vuelto fundamental para las empresas, impulsando el interés en tecnologías que mejoren la calidad, los tiempos de entrega y el costo del software. Esto ha impulsado la adopción de la arquitectura de microservicios y la cultura DevOps para afrontar estos desafíos. La mayoría de los desafíos en el desarrollo, monitoreo y pruebas de microservicios se presentan en la fase de diseño, y se pueden resolver aplicando patrones de microservicios.

El objetivo de esta investigación es identificar los patrones de microservicios más comunes en DevOps, destacar sus ventajas, su evolución en los últimos años y los sectores de la industria donde se emplean. Además, se explorarán las herramientas y prácticas DevOps aplicadas junto con estos patrones.

Este estudio consiste en la ejecución de un mapeo sistemático de la literatura encontrada en bases de datos académicas.

La sinergia entre DevOps y la arquitectura de microservicios ha experimentado una evolución a partir del 2015, con un notable crecimiento en 2022. De los 66 artículos seleccionados, el 80% son de conferencias, destacando las conferencias como "ACM International Conference Proceeding Series", "Communications in Computer and Information Science" y la "IEEE International Conference on Consumer Electronics". El patrón 'Service-per-container' predomina entre lo más implementados en DevOps, seguido por 'Remote Procedure Invocation (RPI)' y 'API Gateway'. Además, la 'Integración continua' se destaca como la práctica DevOps más implementada, seguida por la 'Entrega continua', 'Despliegue continuo' e 'Infraestructura como código', destacando la importancia de la automatización y la entrega eficiente.

Palabras clave:

DevOps, arquitectura de microservicios, patrones arquitectónicos de microservicios

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

CAPÍTULO I. INTRODUCCIÓN

1.1. Contexto

En los últimos años, la industria del software ha experimentado un crecimiento exponencial, donde las empresas compiten constantemente para transformar sus negocios a través del desarrollo de aplicaciones de software (Lazuardi et al., 2021). En este contexto, el interés por los microservicios ha ido aumentado tanto por parte de las empresas como de los investigadores (Filippone et al., 2021), motivando, además, la adopción de la cultura DevOps con el objetivo de conseguir un producto software que brinde una mejor experiencia de usuario, en menor tiempo, con mayor estabilidad y a un menor costo (Amaro et al., 2022).

Empresas líderes en la industria, como Netflix, Spotify y Amazon, han adoptado la arquitectura de microservicios, y muchas otras se encuentran en el proceso de migrar sus sistemas a esta nueva arquitectura (Taibi et al., 2018). Sin embargo, implementar una arquitectura distribuida, como es la de microservicios, plantea importantes desafíos, incluyendo la latencia, la tolerancia a fallos, la coherencia de datos y el balanceo de carga. Por lo tanto, es necesario el uso de nuevas tecnologías y prácticas que permitan abordar esta complejidad (Di Francesco et al., 2017).

Uno de los objetivos de los microservicios es desarrollar sistemas como un conjunto de pequeños servicios, cada uno desarrollado y desplegado de manera independiente, lo que implica una gestión más granular de sus recursos de infraestructura. DevOps proporciona la capacidad de llevar a cabo estas tareas vinculando el proceso de desarrollo con las operaciones (Pahl et al., 2018). Por ello, la implementación de la cultura DevOps para el desarrollo y despliegue de la arquitectura de microservicios adquiere una mayor importancia. Además, los microservicios facilitan una implementación eficaz de DevOps al promover equipos de trabajo pequeños y ágiles (Balalaie et al., 2016).

La compatibilidad inherente entre los microservicios y DevOps es uno de los motivos por los cuales las empresas adoptan esta arquitectura (Taibi et al., 2017). Investigaciones previas han demostrado que los equipos de desarrollo con una cultura DevOps bien establecida pueden beneficiarse significativamente

de la arquitectura de microservicios, por lo que su implementación debe considerar la adopción de la cultura DevOps (Schneider, 2016).

Según estudios recientes, se ha observado que la mayoría de los desafíos relacionados con el desarrollo, monitoreo y pruebas de microservicios se abordan en la fase de diseño (Waseem et al., 2022a). Es en esta fase donde los patrones de microservicios cobran relevancia, desempeñando un papel fundamental en la resolución de problemas conocidos en contextos particulares (Richardson, 2018). Dada la importancia de los patrones de microservicios, este trabajo de investigación se enfocará en su estudio, particularmente en su aplicación en un contexto DevOps.

El presente trabajo de investigación se organiza de la siguiente manera: el capítulo 1 presenta el contexto, la justificación y los objetivos del trabajo. El capítulo 2 describe el marco teórico, donde se aborda el tema de microservicios y sus patrones, y la cultura DevOps. En el capítulo 3, se revisan trabajos relacionados al tema de esta investigación. El capítulo 4 detalla la metodología utilizada para el mapeo sistemático de la literatura. Los resultados de este mapeo sistemático se presentan en el capítulo 5. Las conclusiones se exponen en el capítulo 6. Finalmente, en el capítulo 7 se plantean posibles temas para futuros trabajos de investigación.

1.2. Justificación

A pesar de la popularidad de la arquitectura de microservicios y la cultura DevOps, los desarrolladores suelen emplear los patrones de los que se tiene más documentación en internet, la cual no siempre es la más adecuada (Taibi et al., 2017). Los microservicios tienen su origen en la industria, y sólo recientemente los investigadores han empezado a explorarlos. Por ejemplo, los investigadores han propuesto metodologías para migrar sistemas hacia microservicios, han desarrollado herramientas de evaluación comparativa e identificación de microservicios, han estudiado los pros y los contras de los microservicios o han explorado las relaciones con otros campos de investigación, como la ingeniería de líneas de productos (Assunção et al., 2023).

El presente trabajo de investigación se realiza con la finalidad de proporcionar a los desarrolladores información relevante acerca de los patrones de microservicios utilizados en organizaciones donde se ha adoptado la cultura DevOps. Esta información será de utilidad en el proceso de adopción de microservicios, ya que brindará información que puede ayudar a seleccionar el patrón de microservicios más adecuado de acuerdo a las necesidades particulares de cada proyecto.

1.3. Objetivos

El objetivo general de la investigación es identificar los patrones de microservicios más comunes en un entorno DevOps, así como destacar sus ventajas más relevantes. Además, se explorarán las herramientas, prácticas o principios DevOps que se aplican junto con estos patrones.

Los objetivos específicos son:

- Identificar los patrones de microservicios que se aplican en proyectos donde se empleen las prácticas DevOps, y cuáles son los más usados en los últimos años.
- Conseguir un listado de las prácticas DevOps que se utilizan junto con la arquitectura de microservicios.
- Describir las ventajas que se pueden encontrar al implementar patrones de microservicios en proyectos DevOps.
- Determinar cómo ha evolucionado la frecuencia y el volumen de publicaciones sobre microservicios en proyectos DevOps con el tiempo, clasificándolas por su naturaleza o tipo de estudio.
- Explorar cómo los patrones de microservicios se aplican en diversos sectores de la industria bajo un enfoque DevOps.
- Identificar las principales plataformas académicas donde se publican estudios acerca de la aplicación de patrones de microservicios en un contexto DevOps.

CAPÍTULO II. MARCO TEÓRICO

2.1. Arquitectura de microservicios

La arquitectura de microservicios consiste en desarrollar una aplicación como un conjunto de pequeños servicios, cada uno ejecuta su propio proceso y se comunican mediante diversos mecanismos, usualmente APIs basadas en HTTP (Lewis, 2014). Los microservicios proponen descomponer a las aplicaciones en servicios independientes de acuerdo con las necesidades del negocio (Auer et al., 2021). Esta descomposición permite que los microservicios se desplieguen de manera independiente, tengan bajo acoplamiento, se organicen alrededor de las capacidades del negocio y que el equipo de desarrollo sea pequeño (Richardson, n.d.). Gracias a estas características los desarrolladores tienen una mayor flexibilidad en el desarrollo y mantenimiento del software. Esto permite que la aplicación se mantenga actualizada y se adapte mejor a las necesidades del cliente.

Las propiedades que poseen los microservicios permiten que cada uno se despliegue de manera independiente en un entorno virtualizado y lo haga a la velocidad necesaria, sin depender de otros componentes. Estas características hacen que los microservicios sean ideales para las prácticas modernas promovidas por la cultura DevOps (Ntentos et al., 2021).

2.2. Patrones de microservicios

Los patrones de diseño brindan una solución general a problemas recurrentes en el diseño de software, y se considera que mejoran la calidad del producto (Wedyan & Abufakher, 2020). Existen varias áreas en el diseño de microservicios, tales como, composición, orquestación, seguridad, descubrimiento de servicios, acceso a base de datos, etc (Waseem et al., 2022b). En cada área existen diferentes patrones que se pueden utilizar para implementar microservicios.

En (Richardson, 2018) se presenta un compilado de patrones de microservicios que se encuentran clasificados en áreas de acuerdo con el tipo de problema que resuelven. En este libro se han identificado 14 áreas que a su vez se clasifican en 3 capas:

- Patrones de Infraestructura: estos patrones resuelven temas relacionados a problemas de infraestructura que no competen al equipo de desarrollo.
- Patrones de infraestructura de la aplicación: estos abordan problemas de infraestructura que impacta al desarrollo.
- Patrones de Aplicación: estos afrontan los problemas que presentan los desarrolladores.

La clasificación antes mencionada se observa con mayor claridad en la Figura 2.1, la cual fue elaborada por Richardson, C. en (Richardson, n.d.-a).

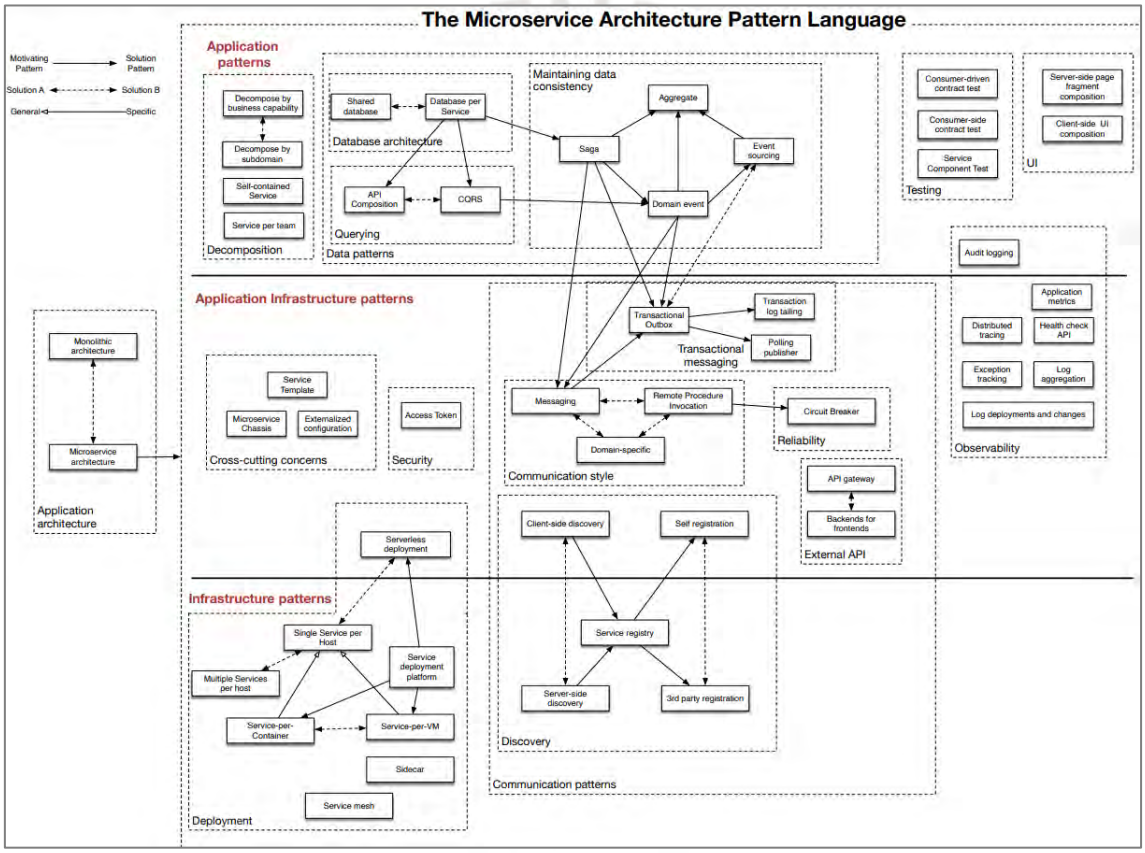

Figura 2.1. Vista de alto nivel de la clasificación de patrones de microservicios

2.3. DevOps

El movimiento DevOps se originó en el año 2007 cuando el consultor Patrick Debois estaba involucrado en un proyecto que consistía en realizar pruebas durante la migración de un datacenter. Durante este proceso, observó que los principales problemas surgían debido a las fricciones entre el equipo de desarrollo y operaciones. Este hallazgo lo motivó a buscar una solución a estos

inconvenientes. Un año más tarde, en colaboración con Andrew Shafer, quien había planteado anteriormente la idea de “Agile Infrastructure”, crearon un grupo de discusión. Este grupo se dedicó a explorar ideas para mejorar el trabajo entre los equipos de desarrollo y operaciones. Fue en el año 2009 cuando se empieza a usar el término DevOps (Watts, 2019).

DevOps se construye sobre los principios lean y ágiles (Ebert et al., 2016), y se basa en 4 pilares fundamentales: Infraestructura como código (IaC), Integración Continua (CI), Entrega continua (CD) y Despliegue Continuo (Aguiar Monteiro et al., 2020). Infraestructura como código se basa en la automatización de rutinas repetitivas para el aprovisionamiento y cambio de la configuración de sistemas (Morris, 2016). La integración continua consiste en que cada vez que se haga una actualización del código en el sistema de control de versiones, automáticamente inicie el proceso de compilación del código, se ejecuten las pruebas unitarias, se analice el código, se empaquete y se disponga en una ruta listo para ser liberado (Wouter de Kort, 2016). La entrega continua se basa en la puesta en producción de un software de manera rápida y eficiente, y con la posibilidad de realizarla en cualquier momento (Liang, 2015). Esto implica que las pruebas y la implementación del software en producción se pueda realizar de manera automática. Sin embargo, aún es necesaria la intervención de una persona para iniciar el pase a producción. Cuando el pase a producción se realiza de manera automática, estamos hablando de un despliegue continuo (Shahin et al., 2017).

Además de hacer uso de herramientas, prácticas y metodologías para implementar los 4 pilares antes mencionados, DevOps requiere un cambio en la cultura organizacional con la finalidad de conseguir la sinergia entre el equipo de desarrollo, calidad y operaciones (Ebert et al., 2016). Este aspecto es uno de los más importantes al implementar DevOps, tal como se señala en (Akbar et al., 2022), donde concluyen que construir una cultura de colaboración y compartir los objetivos es la práctica con mayor prioridad.

El artículo (Jabbari et al., 2016) realiza una síntesis de las prácticas DevOps que se han aplicado bajo ese contexto. Estas prácticas se encuentran listadas en la

Tabla 2.1. Esta lista de prácticas y su clasificación es la que utilizaremos como referencia para responder las preguntas de investigación.

Knowledge Area (KA)	Sub-Knowledge Area (Sub-KA)	Practice
Software engineering management	Software Project Planning	Continuous planning Feedback loop between developers and operators
	Software Project Enactment	Continuous monitoring Automated performance monitoring during test and continuous integration Automated feedback for performance models and performance predictions Application monitoring Automated dashboards
Software construction	Practical Considerations	Continuous integration
	Software Construction Fundamentals	Prototyping application
Software configuration management	Software Release Management and Delivery	Integrated deployment planning Continuous deployment Automated deployment Continuous delivery Cooperative application configurations Monitoring application and next development
	Management of the SCM Process	Staging application Integrated configuration management
	Software Configuration Control	Integrated change management Change management
Software testing	Test Techniques	Continuous testing Automated testing
Software Process	Process Definition	Process standardization Production support
Software quality	Practical Considerations	Use of data to guide QA
Software engineering tools and methods	Software Engineering Methods	Infrastructure as code Modeling & Simulation Measure performance metrics [in CI, Test & Ops] Continuous application performance
	Software Tools	DevOps maturity evaluation model Elasticity practice
Software requirements	Software Requirements Fundamentals	Defining requirements
	Requirements Process	Stakeholder participation
Software design	Software Structure and Architecture	Designing architecture

Tabla 2.1. Prácticas DevOps identificadas en (Jabbari et al., 2016)

CAPÍTULO III. TRABAJOS RELACIONADOS

En esta sección se revisarán investigaciones previas similares al presente trabajo de investigación y que estén relacionadas al tema de patrones de microservicios en un entorno DevOps.

Se identificaron tres investigaciones que llevaron a cabo mapeos sistemáticos de la literatura y dos que realizaron revisiones sistemáticas de la literatura con el propósito de identificar patrones de microservicios o analizar otros aspectos relacionados a los patrones como sus ventajas, problemas y atributos. Ninguno considera el enfoque DevOps al buscar la documentación relacionada a los patrones de microservicios.

La investigación “Architecting with microservices: A systematic mapping study” (Di Francesco et al., 2019), propone un marco de trabajo para la clasificación, comparación y evaluación de métodos y técnicas empleadas en microservicios, entre ellos considera los patrones de diseño. Así mismo, brinda una visión general del estado del arte de la arquitectura y evalúa su potencial de adopción en la industria.

En las investigaciones “Continuous Architecting with Microservices and DevOps: A Systematic Mapping Study” (Taibi et al., 2019) y “Architectural Patterns for Microservices: a Systematic Mapping Study” (Taibi et al., 2018), se identifican los patrones de microservicios más adoptados en la literatura, y sus respectivas ventajas, desventajas y lecciones aprendidas. Cabe mencionar que, si bien el primer artículo aborda el tema de microservicios y DevOps, en el proceso de selección de la literatura no considera la combinación de ambos temas al ejecutar las búsquedas.

En el artículo titulado “An Exploratory Study of Academic Architectural Tactics and Patterns in Microservices: A systematic literature review” (Osses et al., 2019), se identificaron 44 patrones de microservicios, siendo la mayoría de ellos relacionados a cinco atributos de calidad: escalabilidad, flexibilidad, desempeño, elasticidad y facilidad para la realización de pruebas. Además, el estudio reveló que la mayoría de los artículos acerca de microservicios estaban relacionados a DevOps e IoT.

El artículo “Taxonomical Classification and Systematic Review on Microservices” (Weerasinghe & Perera, 2022), brinda una visión general del estado del arte de la arquitectura de microservicios. Como parte de su investigación muestra en que áreas se utilizan con más frecuencia los patrones de microservicios y menciona cuales fueron los patrones más utilizados en la literatura.

Estos trabajos similares servirán como punto de partida para tener una visión general de la literatura existente en el campo de patrones de microservicios y DevOps, y comprender como contribuye este trabajo de investigación.

CAPÍTULO IV. MÉTODO DE INVESTIGACIÓN

La metodología que se va a emplear en este trabajo de investigación es el mapeo sistemático de la literatura, basada en la guía elaborada por Petersen, Vakkalanka y Kuzniarz (Petersen et al., 2015). En esta guía, proponen un proceso para realizar el mapeo que consiste en 3 pasos: planificación, conducción y muestra de resultados. El flujo de trabajo de la propuesta de Petersen se puede observar en la Figura 4.1.

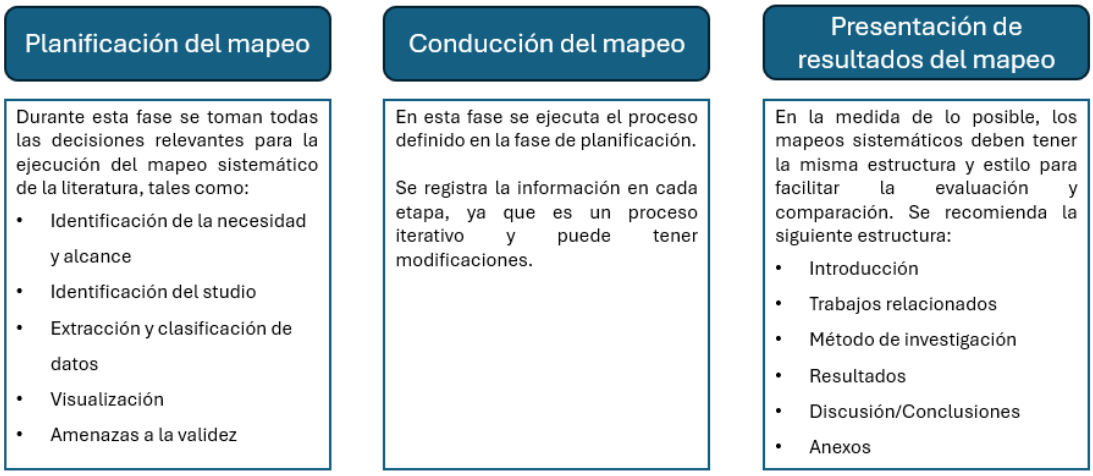

Figura 4.1. Proceso para realizar el mapeo sistemático de la literatura propuesto en (Petersen et al., 2015)

4.1. Preguntas bibliométricas

A continuación, se listan las preguntas bibliométricas que se han planteado para abordar el tema de investigación, así mismo se indica el objetivo de cada una de ellas.

PB1: ¿Cómo ha cambiado la frecuencia de publicación de estudios acerca de los patrones de microservicios en proyectos DevOps?

El objetivo es evaluar la tendencia y la evolución de la investigación sobre los patrones de microservicios en el contexto de DevOps a lo largo del tiempo.

PB2: ¿En qué conferencias o revistas se han publicado con mayor frecuencia estudios sobre los patrones de microservicios en proyectos DevOps?

El objetivo es identificar las plataformas académicas más relevantes para la publicación de estudios sobre patrones de microservicios en proyectos DevOps.

PB3: ¿Qué tipos de estudios y cuántos se han realizado acerca de la aplicación de patrones de microservicios en proyectos DevOps?

El objetivo es determinar la naturaleza, el alcance y la magnitud de la investigación sobre la implementación de patrones de microservicios en proyectos DevOps.

PB4: ¿Cuál es la cantidad de estudios que se han publicado en el ámbito académico e industrial, y en qué sectores de la industria?

El objetivo es cuantificar y categorizar la investigación sobre patrones de microservicios según su ámbito (académico o industrial) y el sector de la industria aplicada, con la finalidad de como la teoría se refleja en la práctica en diferentes campos.

4.2. Preguntas de investigación

El objetivo general de la investigación es presentar los patrones arquitectónicos de microservicios implementados, así como las ventajas y desventajas identificadas en proyectos ejecutados bajo un enfoque DevOps. Esto conduce al planteamiento de las siguientes preguntas de investigación:

PI1: ¿Cuáles son los patrones de microservicios más aplicados junto con alguna práctica o herramienta DevOps?

El objetivo es descubrir qué patrones de microservicios se integran con mayor frecuencia con prácticas o herramienta DevOps, lo que puede mostrar sobre las sinergias entre los patrones y la cultura DevOps.

PI2: ¿Cuáles son las práctica o herramientas DevOps más utilizados al implementar la arquitectura de microservicios?

El objetivo es identificar las prácticas o herramientas DevOps que son más comunes en la implementación de la arquitectura de microservicios, lo que puede destacar áreas de interés o mejores prácticas.

PI3: ¿En qué sectores de la industria se emplean los patrones de microservicios en proyectos ejecutados bajo un enfoque DevOps?

El objetivo es explorar la adopción de los patrones de microservicios en diferentes sectores de la industria dentro de proyectos DevOps, lo que puede mostrar como varían las aplicaciones de estos patrones en base a necesidades particulares de los sectores.

PI4: ¿Qué ventajas se obtienen al implementar los patrones de microservicios en proyectos ejecutados bajo un enfoque DevOps?

El objetivo es analizar las ventajas de aplicar patrones de microservicios en proyectos DevOps, con la finalidad de comprender mejor los beneficios que se pueden encontrar al implementar estos patrones.

PI5: ¿Cómo ha variado en el tiempo el uso de los diferentes patrones de microservicios en proyectos ejecutados bajo un enfoque DevOps?

El objetivo es examinar cómo ha cambiado la adopción de diferentes patrones de microservicios en proyectos DevOps a lo largo del tiempo, lo que podría mostrar tendencias emergentes, prácticas que se están dejando de realizar o la madurez de ciertos patrones.

4.3. Protocolo de búsqueda

La estrategia para identificar los términos para la búsqueda está basada en el método PICOC, el cual fue propuesto por Kitchenham y Charters desde el punto de vista de la ingeniería de software (Kitchenham et al., 2007). En el presente trabajo de investigación se emplean 3 criterios: población, intervención y contexto, tal como se puede observar en la Tabla 4.1.

PICOC	Términos
Población	Arquitectura de microservicios, patrones arquitectónicos de microservicios
Intervención	Proyectos u organizaciones bajo un enfoque DevOps Prácticas, principios o herramientas DevOps
Comparación	
Resultado	
Contexto	Ámbito académico, caso de estudio o aplicación en la industria.

Tabla 4.1: PICOC para las preguntas de investigación

Se utilizaron 4 fuentes de datos: Scopus, IEEE Xplore, ACM y Web of science. En base a los términos identificados usando PICO, se ejecutaron las cadenas de búsqueda que se muestran en la Tabla 4.2. Los términos se buscaron en los títulos, resúmenes y/o palabras clave de la literatura disponible en cada base de datos, dando como resultado la cantidad de artículos que se muestran en la Tabla 4.3.

Base de datos	Cadena de búsqueda
Scopus	(microservice* OR "micro-service*") AND (pattern* OR CQRS OR "API composition" OR "service mesh" OR "api gateway" OR "api services" OR "api rest" OR grpc OR orchestration OR decompos* OR container*) AND (devops OR "continuous delivery" OR "continuous integration" OR "continuous deploy*" OR "automated deploy*" OR "automated test*" OR "continuous planning" OR "continuous monitoring" OR "continuous testing" OR "infrastructure as code" OR "CI/CD")
IEEE	("Abstract": microservice* OR "Abstract": "micro-service*" OR "Document Title": microservice* OR "Document Title": "micro-service*") AND ("Abstract": pattern* OR "Abstract": CQRS OR "Abstract": "API composition" OR "Abstract": "service mesh" OR "Abstract": "api gateway" OR "Abstract": "api services" OR "Abstract": "api rest" OR "Abstract": grpc OR "Abstract": orchestration OR "Abstract": decompos* OR "Abstract": container* OR "Document Title": pattern* OR "Document Title": CQRS OR "Document Title": "API composition" OR "Document Title": "service mesh" OR "Document Title": "api gateway" OR "Document Title": "api services" OR "Document Title": "api rest" OR "Document Title": grpc OR "Document Title": orchestration OR "Document Title": decompos* OR "Document Title": container*) AND ("Abstract": devops OR "Abstract": "continuous delivery" OR "Abstract": "continuous integration" OR "Abstract": "continuous deploy*" OR "Abstract": "automated deploy*" OR "Abstract": "automated test*" OR "Abstract": "continuous planning" OR "Abstract": "continuous monitoring" OR "Abstract": "continuous testing" OR "Abstract": "infrastructure as code" OR "Abstract": "CI/CD" OR "Document Title": devops OR "Document Title": "continuous delivery" OR "Document Title": "continuous integration" OR "Document Title": "continuous deploy*" OR "Document Title": "automated deploy*" OR "Document Title": "automated test*" OR "Document Title": "continuous planning" OR "Document Title": "continuous monitoring" OR "Document Title": "continuous testing" OR "Document Title": "infrastructure as code" OR "Document Title": "CI/CD")
ACM	(Title:(microservice* OR micro-services) OR Abstract:(microservice* OR micro-services)) AND (Title:(pattern* OR CQRS OR "API composition" OR "service mesh" OR "api gateway" OR "api services" OR "api rest" OR grpc OR orchestration OR decompos* OR container*) OR Abstract:(pattern* OR CQRS OR "API composition" OR "service mesh" OR "api gateway" OR "api services" OR "api rest" OR grpc OR orchestration OR decompos* OR container*)) AND (Title:(devops OR "continuous delivery" OR "continuous integration" OR "continuous deploy*" OR "automated deploy*" OR "automated test*" OR "continuous planning" OR "continuous monitoring" OR "continuous testing" OR "infrastructure as code" OR "CI/CD") OR Abstract:(devops OR "continuous delivery" OR "continuous integration" OR "continuous deploy*" OR "automated deploy*" OR "automated test*" OR "continuous planning" OR "continuous monitoring" OR "continuous testing" OR "infrastructure as code" OR "CI/CD"))

Web of science	(TI=(microservice* OR "micro-service*") OR AB=(microservice* OR "micro-service*")) AND (TI=(pattern* OR CQRS OR "API composition" OR "service mesh" OR "api gateway" OR "api services" OR "api rest" OR grpc OR orchestration OR decompos* OR container*) OR AB=(pattern* OR CQRS OR "API composition" OR "service mesh" OR "api gateway" OR "api services" OR "api rest" OR grpc OR orchestration OR decompos* OR container*)) AND (TI=(devops OR "continuous delivery" OR "continuous integration" OR "continuous deploy*" OR "automated deploy*" OR "automated test*" OR "continuous planning" OR "continuous monitoring" OR "continuous testing" OR "infrastructure as code" OR "CI/CD") OR AB=(devops OR "continuous delivery" OR "continuous integration" OR "continuous deploy*" OR "automated deploy*" OR "automated test*" OR "continuous planning" OR "continuous monitoring" OR "continuous testing" OR "infrastructure as code" OR "CI/CD"))

Tabla 4.2: Cadena de búsqueda por base de datos

Base de datos	Resultado de búsqueda
Scopus	192
IEEE	64
ACM	57
Web of science	20
TOTAL	**333**

Tabla 4.3: Resultados de búsqueda por base de datos

4.4. Selección de estudios

La selección de los estudios se realizó en 4 fases. En cada una se aplicaron criterios de inclusión y/o exclusión. Los criterios de inclusión están representados por “CI” y los de exclusión por “CE”. A continuación, se indican los criterios aplicados en cada fase.

i. La primera fase consistió en aplicar los siguientes criterios a todos los estudios encontrados al ejecutar las cadenas de búsqueda de la Tabla 4.2.

CI1: Se presenta en idioma inglés o español.

CE1: Se encuentra duplicado.

CE2: Es un resumen de conferencia/editorial o capítulo de libro.

ii. En la segunda fase, se revisaron los títulos y resúmenes de los documentos que no fueron excluidos en la primera fase, a los cuales se les aplicó los siguientes criterios:

CI2: Se menciona alguna referencia a la aplicación de un patrón de microservicios.

CI3: El artículo indica que fue desarrollado bajo una organización o proyecto DevOps, o menciona la implementación de un principio, práctica o herramienta DevOps.

CE3: Es un estudio secundario o terciario.

CE4: Los estudios mencionan el término de microservicios como parte de la arquitectura SOA (Service-Oriented Architecture)

iii. En la tercera fase, se aplicó el siguiente criterio a los artículos que no fueron excluidos en la primera ni segunda fase:

CI4: Se dispone del acceso completo al estudio.

iv. En la cuarta fase, se realizó una lectura completa de los artículos que no fueron excluidos en las fases previas para determinar si cumplían con los siguientes criterios:

CI5: El artículo menciona, mediante un ejemplo, experimento o caso de uso concreto, la implementación de por lo menos un patrón de microservicios en una organización o proyecto donde se haya aplicado algún principio, práctica o herramienta DevOps.

CI6: El artículo presenta información clara relacionada a, por lo menos, una pregunta de investigación.

Se encontró un total de 333 artículos en las búsquedas realizadas en las 4 bases de datos. Luego de aplicarse los criterios de inclusión y exclusión de cada una de las cuatro fases, se obtuvo un total de 90 artículos. En la Figura 4.2, se puede observar la cantidad de artículos que se conservaron luego de aplicar los criterios de cada fase.

		Artículos excluidos en cada fase				
Base de datos	**Resultado de búsqueda**	**Fase 1**	**Fase 2**	**Fase 3**	**Fase 4**	**Artículos seleccionados**
Scopus	192	30	69	8	22	**64**
IEEE	64	57	3	1	0	**2**
ACM	57	21	34	1	0	**1**
Web of science	20	19	0	1	0	**0**
Total	**333**	**127**	**107**	**11**	**22**	**66**

Figura 4.2. Resultado del proceso de selección de artículos

La relación de los artículos seleccionados se puede encontrar en la Tabla 4.4.

Ref.	Titulo de la publicación	Autor	Fuente
[1]	5G-Kube: Complex Telco Core Infrastructure Deployment Made Low-Cost	Scotece D.; Noor A.; Foschini L.; Corradi A.	Scopus
[2]	Load balancing and service discovery using Docker Swarm for microservice based big data applications	Singh N.; Hamid Y.; Juneja S.; Srivastava G.; Dhiman G.; Gadekallu T.R.; Shah M.A.	Scopus
[3]	µDetector: Automated Intrusion Detection for Microservices	J. Flora; M. Teixeira; N. Antunes	IEEE
[4]	Implementing a Microservices Architecture for Predicting the	Guerdoux G.; Audeh B.; Tiffet T.; Bousquet C.	Scopus

	Opinion of Twitter Users on COVID Vaccines		
[5]	Microservices Integrated Performance and Reliability Testing	Camilli M.; Guerriero A.; Janes A.; Russo B.; Russo S.	Scopus
[6]	Record linkage based patient intersection cardinality for rare disease studies using Mainzelliste and secure multi-party computation	Kussel T.; Brenner T.; Tremper G.; Schepers J.; Lablans M.; Hamacher K.	Scopus
[7]	Containerized Architecture Performance Analysis for IoT Framework Based on Enhanced Fire Prevention Case Study: Rwanda	Hitimana E.; Bajpai G.; Musabe R.; Sibomana L.; Jayavel K.	Scopus
[8]	Moving Toward the Obsolescence of Obsolescence: A Walk in the Clouds	Griffin R.; Henson N.	Scopus
[9]	Enabling rank-based distribution of microservices among containers for green cloud computing environment	Saboor A.; Mahmood A.K.; Omar A.H.; Hassan M.F.; Shah S.N.M.; Ahmadian A.	Scopus
[10]	An Extensible and Secure Architecture based on Microservices	De O. Junior R.S.; Da Silva R.C.A.; Santos M.S.; Albuquerque D.W.; Almeida H.O.; Santos D.F.S.	Scopus
[11]	Automated Testing and Resilience of Microservice's Network-link using Istio Service Mesh	Karn R.R.; Das R.; Pant D.R.; Heikkonen J.; Kanth R.	Scopus
[12]	Automated Grey-Box Testing of Microservice Architectures	Giamattei L.; Guerriero A.; Pietrantuono R.; Russo S.	Scopus

[13]	mira: an Application Containerisation Pipeline for Small Software Development Teams in Low Resource Settings	Mwotil A.; Bainomugisha E.; Araka S.G.M.	Scopus
[14]	Abstractions of Abstractions: Metadata to Infrastructure-as-Code	Deslauriers J.; Kovacs J.; Kiss T.	Scopus
[15]	Bunk8s: Enabling Easy Integration Testing of Microservices in Kubernetes	Reile C.; Chadha M.; Hauner V.; Jindal A.; Hofmann B.; Gerndt M.	Scopus
[16]	Improving Business deliveries using Continuous Integration and Continuous Delivery using Jenkins and an Advanced Version control system for Microservices-based system	Singh A.; Singh V.; Aggarwal A.; Aggarwal S.	Scopus
[17]	Expert System for Kubernetes Cluster Autoscaling and Resource Management	Hettiarachchi L.S.; Jayadeva S.V.; Bandara R.A.V.; Palliyaguruge D.; Samaratunge Arachchillage U.S.S.; Kasthurirathna D.	Scopus
[18]	A Delivery Robot Cloud Platform Based on Microservice	Yin Z.; Liu J.; Chen B.; Chen C.	Scopus
[19]	From DevOps to NoOps: Is It Worth It?	Jindal A.; Gerndt M.	Scopus
[20]	A framework for microservices synchronization	De Iasio A.; Zimeo E.	Scopus
[21]	MiCADO-Edge: Towards an Application-level Orchestrator for the Cloud-to-Edge Computing Continuum	Ullah A.; Dagdeviren H.; Ariyattu R.C.; DesLauriers J.; Kiss T.; Bowden J.	Scopus

[22]	TORCH: a TOSCA-Based Orchestrator of Multi-Cloud Containerised Applications	Tomarchio O.; Calcaterra D.; Di Modica G.; Mazzaglia P.	Scopus
[23]	Analysis of Performance in Docker Net deployed in AWS cloud	Romanov O.; Mankivskyi V.; Veres L.; Saychenko I.	Scopus
[24]	An Advanced DevOps Environment for Microservice-based Applications	Throner S.; Hutter H.; Sanger N.; Schneider M.; Hanselmann S.; Petrovic P.; Abeck S.	Scopus
[25]	Performance and Security Evaluation in Microservices Architecture Using Open Source Containers	Castillo Rivas D.A.; Guamán D.	Scopus
[26]	Automatic Deployment Mechanism for Smart Communication Service Based on Container Technology	Chen S.-L.; Chuang H.-Y.	Scopus
[27]	XERIS/APEX	Wai, Richard	ACM
[28]	Container mapping and its impact on performance in containerized cloud environments	Ambrosino G.; Fioccola G.B.; Canonico R.; Ventre G.	Scopus
[29]	Nonlinear adaptive robust motion control for hydraulic winch in oil and gas wireline operation	Bu F.	Scopus
[30]	RADON: rational decomposition and orchestration for serverless computing	Casale G.; Artač M.; van den Heuvel W.-J.; van Hoorn A.; Jakovits P.; Leymann F.; Long M.; Papanikolaou V.; Presenza D.; Russo A.; Srirama S.N.; Tamburri D.A.; Wurster M.; Zhu L.	Scopus

[31]	MisMesh: Security issues and challenges in service meshes	Hahn D.A.; Davidson D.; Bardas A.G.	Scopus
[32]	Prevant (Preview servant): Composing microservices into reviewable and testable applications	Schreiber M.	Scopus
[33]	Dogfooding: Using IBM cloud services to monitor IBM cloud infrastructure	Pourmajidi W.; Miranskyy A.; Steinbacher J.; Erwin T.; Godwin D.	Scopus
[34]	Comparison of zero downtime based deployment techniques in public cloud infrastructure	Rudrabhatla C.K.	Scopus
[35]	Deriving Microservice Code from Underspecified Domain Models Using DevOps-Enabled Modeling Languages and Model Transformations	Rademacher F.; Sachweh S.; Zundorf A.	Scopus
[36]	Automation of Microservices Application Deployment Made Easy By Rundeck and Kubernetes	Rajavaram H.; Rajula V.; Thangaraju B.	Scopus
[37]	Comparison of different CI/CD Tools integrated with cloud platform	Singh C.; Gaba N.S.; Kaur M.; Kaur B.	Scopus
[38]	Kuksa: A Cloud-Native Architecture for Enabling Continuous Delivery in the Automotive Domain	Banijamali A.; Jamshidi P.; Kuvaja P.; Oivo M.	Scopus
[39]	Test Automation Framework as a Service (TAFaaS) - scale test automation & devops practices with cloud, containers, and microservice.	Sivanandan S.	Scopus

[40]	Towards multi-party policy-based access control in federations of cloud and edge microservices	Preuveneers D.; Joosen W.	Scopus
[41]	Container Orchestration Engines: A Thorough Functional and Performance Comparison	Al Jawarneh I.M.; Bellavista P.; Bosi F.; Foschini L.; Martuscelli G.; Montanari R.; Palopoli A.	Scopus
[42]	Agile Network Access Control in the Container Age	C. Diekmann; J. Naab; A. Korsten; G. Carle	IEEE
[43]	Workload performance and interference on containers	Garg S.K.; Lakshmi J.	Scopus
[44]	A Vision for the Next Generation Platform-as-a-Service	Van Rossem S.; Sayadi B.; Roullet L.; Mimidis A.; Paolino M.; Veitch P.; Berde B.; Labrador I.; Ramos A.; Tavernier W.; Ollora E.; Soler J.	Scopus
[45]	Understanding automated continuous integration for containerized smart energy IoT-cloud service	Kim C.; Kim S.; Kim J.	Scopus
[46]	Microservice-based cloud robotics system for intelligent space	Xia C.; Zhang Y.; Wang L.; Coleman S.; Liu Y.	Scopus
[47]	Architecting Enterprise Applications for the Cloud: The Unicorn Universe Cloud Framework	Beranek M.; Stastny M.; Kovar V.; Feuerlicht G.	Scopus
[48]	Identity and Access Control for micro-services based 5G NFV platforms	Guija D.; Siddiqui M.S.	Scopus
[49]	Building lean continuous integration and delivery pipelines	Debroy V.; Miller S.; Brimble L.	Scopus

	by applying devops principles: A case study at varidesk		
[50]	A microservice architecture for the industrial internet-of-things	Dobaj J.; Krisper M.; Iber J.; Kreiner C.	Scopus
[51]	A reference architecture for the container ecosystem	Syed M.H.; Fernandez E.B.	Scopus
[52]	If Docker is the Answer, What is the Question?	Zhu H.; Bayley I.	Scopus
[53]	AUTOGENIC: Automated generation of self-configuring microservices	Kehrer S.; Blochinger W.	Scopus
[54]	A web-based service deployment method to edge devices in smart factory exploiting Docker	Ha J.; Kim J.; Park H.; Lee J.; Jo H.; Kim H.; Jang J.	Scopus
[55]	Delivering Elastic Containerized Cloud Applications to Enable DevOps	Barna C.; Khazaei H.; Fokaefs M.; Litoiu M.	Scopus
[56]	Enabling Enterprise Blockchain AppDev Teams	Yammanuru P.R.; Jain A.; Vinayakaram H.	Scopus
[57]	Requirements reconciliation for scalable and secure microservice (de)composition	Ahmadvand M.; Ibrahim A.	Scopus
[58]	Container-based microservice architecture for cloud applications	Singh V.; Peddoju S.K.	Scopus
[59]	RAML-based mock service generator for microservice applications testing	Ashikhmin N.; Radchenko G.; Tchernykh A.	Scopus
[60]	Microservice architectures for scalability, agility and reliability in e-commerce	Hasselbring W.; Steinacker G.	Scopus
[61]	PCCTE: A portable component conformance test environment	Lv K.; Zhao Z.; Rao R.; Hong P.; Zhang X.	Scopus

	based on container cloud for avionics software development		
[62]	SAVI-IoT: A self-managing containerized IoT platform	Khazaei H.; Bannazadeh H.; Leon-Garcia A.	Scopus
[63]	Containerized development and microservices for self-driving vehicles: Experiences & best practices	Berger C.; Nguyen B.; Benderius O.	Scopus
[64]	Container and microservice driven design for cloud infrastructure DevOps	Kang H.; Le M.; Tao S.	Scopus
[65]	Automated Deployment of a Microservice-based Monitoring Infrastructure	Ciuffoletti A.	Scopus
[66]	Microservices validation: Mjolnirr platform case study	Savchenko D.I.; Radchenko G.I.; Taipale O.	Scopus

Tabla 4.4. Estudios seleccionados

4.5. Extracción de datos

Con el objetivo de responder las preguntas bibliométricas y las de investigación se elaboró el formulario de extracción de datos que se encuentra en la Tabla 4.5. Este formulario se aplicó a todos los estudios de la Tabla 4.4.

DATO	PREGUNTA DE INVESTIGACIÓN
Título del artículo	
Autor	
Año de publicación	PB1, PI5
Título de la publicación de origen	PB2
Tipo de publicación	PB3
Patrón de microservicios aplicado	PI1, PI3, PI5
Ventajas de implementar los patrones de microservicios	PI4
Práctica, principio o herramienta DevOps empleada	PI2
Industria o entorno académico Caso de estudio, experimento o uso en industria	PB4, PI3

Tabla 4.5. Formulario para la extracción de datos

CAPÍTULO V. RESULTADOS DEL MAPEO SISTEMÁTICO

5.1. Preguntas bibliométricas

PB1: ¿Cómo ha cambiado la frecuencia de publicación de estudios acerca de los patrones de microservicios en proyectos DevOps?

La cultura DevOps empezó a consolidarse alrededor de 2009, mientras que la arquitectura de microservicios empezó a ganar relevancia en la industria y en publicaciones en internet, como blogs, a partir del 2014. Por lo tanto, es coherente encontrar artículos que mencionen ambos conceptos a partir del 2015, tal como se muestra en la Figura 5.1. Inicialmente, se observa un número bajo de publicaciones, lo que refleja el inicio de la adopción combinada de ambos conceptos, seguido por un incremento en los años posteriores, lo cual indica un aumento en el interés y adopción de ambas prácticas en la industria.

Hay una tendencia ligeramente ascendente desde el 2015 hasta el 2021, con pequeñas fluctuaciones en el número de artículos. Sin embargo, en el 2022 se observa un aumento significativo en la cantidad de publicaciones. En la Tabla 5.1, se encuentra la relación de los artículos que se publicaron en cada año.

Año	Publicaciones
2015	[65], [66]
2016	[64]
2017	[54], [55], [56], [57], [58], [59], [60], [61], [62], [63]
2018	[43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53]
2019	[36], [37], [38], [39], [40], [41], [42]
2020	[28], [29], [30], [31], [32], [33], [34], [35]
2021	[18], [19], [20], [21], [22], [23], [24], [25], [26], [27]
2022	[10], [11], [12], [13], [14], [15], [16], [17], [4], [5], [6], [7], [8], [9]
2023	[1], [2], [3]

Tabla 5.1. Artículos publicados por año

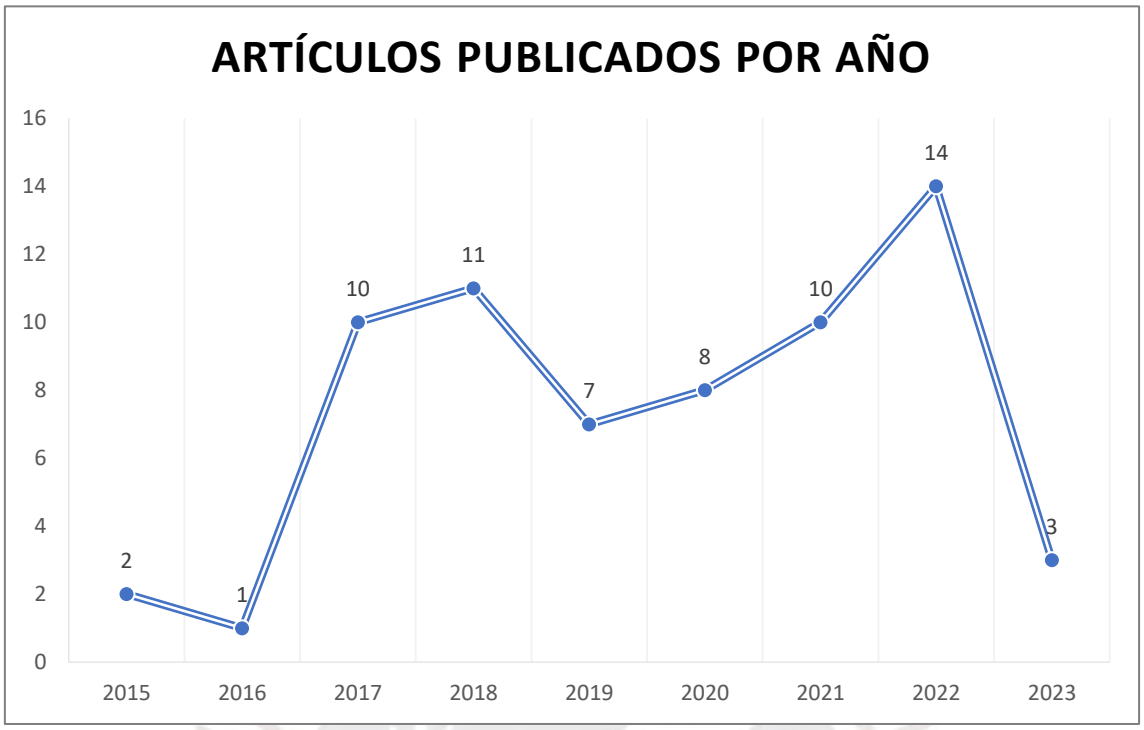

Figura 5.1. Cantidad de artículos publicados por año

PB2: ¿En qué conferencias o revistas se han publicado con mayor frecuencia estudios sobre los patrones de microservicios en proyectos DevOps?

Los 3 canales de publicación que cuentan con más aportes relacionados al tema de investigación son “ACM International Conference Proceeding Series”, “Proceedings - IEEE International Symposium on Service-Oriented System Engineering, SOSE” y “Communications in Computer and Information Science”, contando con 4, 3 y 3 publicaciones respectivamente. Cabe mencionar que una de las conferencias con mayor cantidad de publicaciones es una especializada en el tema de sistemas orientados a servicios: “IEEE International Conference on Service-Oriented System Engineering, SOSE”.

En la Figura 5.2, se pueden observar las conferencias o revistas que tienen más de 2 publicaciones, y en la Tabla 5.2 se pueden encontrar las referencias de cada una de estas publicaciones. Las revistas o conferencias que publicaron solo uno de los estudios seleccionados suma un total de 48, lo cual representa la mayor parte de los estudios seleccionados. Por lo tanto, esto sugiere una

concentración de trabajos en un pequeño conjunto de canales, lo cual podría indicar una preferencia de la comunidad académica hacia estos foros.

Figura 5.2. Revistas o conferencias con más de una publicación relacionada al tema de investigación

Conferencia /Revista	Publicaciones
ACM International Conference Proceeding Series	[13], [48], [50], [51]
Communications in Computer and Information Science	[19], [25], [59]
IEEE International Conference on Consumer Electronics	[10], [26]
Journal of Grid Computing	[21], [22]
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	[38], [47]
Proceedings - IEEE International Conference on Service-Oriented System Engineering, SOSE	[24], [28], [52]
Proceedings - IEEE International Conference on Software Architecture Workshops, ICSAW: Side Track Proceedings	[60], [62]

Tabla 5.2. Revistas o conferencias con más de una publicación relacionada al tema de investigación

PB3: ¿Qué tipos de estudios y cuántos se han realizado acerca de la aplicación de patrones de microservicios en proyectos DevOps?

Los artículos seleccionados corresponden a dos tipos de publicaciones: artículos de revistas o artículos de conferencias. La Figura 5.3 muestra que hay una predominancia de los trabajos presentados en conferencias, los cuales representa el 80% del total, mientras que las publicaciones en revistas constituyen un %. Esto indica que hay una tendencia hacia la presentación de estudios en eventos de conferencia. En la Tabla 5.3, se listan las referencias de las publicaciones clasificadas por cada tipo.

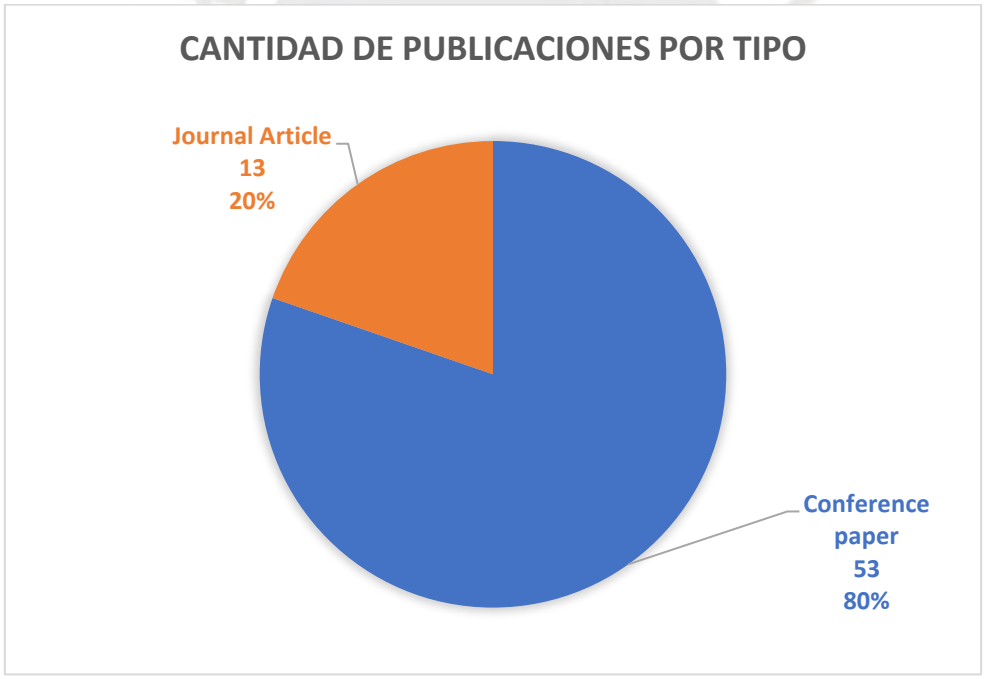

Figura 5.3. Cantidad de publicaciones por tipo

Tipo de publicación	Publicaciones
Conference paper	[10], [11], [12], [13], [14], [15], [16], [17], [19], [23], [24], [25], [26], [28], [29], [3], [30], [31], [32], [33], [34], [35], [36], [37], [38], [4], [40], [41], [43], [44], [45], [47], [48], [49], [5], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [8]
Journal Article	[1], [18], [2], [20], [21], [22], [27], [39], [42], [46], [6], [7], [9]

Tabla 5.3. Listado de publicaciones por tipo

PB4: ¿Cuál es la cantidad de estudios que se han publicado en el ámbito académico e industrial, y en qué sectores de la industria?

La Figura 5.4 muestra que las investigaciones en el ámbito académico representan el 64% del total de artículos seleccionados, superando a las realizadas en el sector industrial. Esto sugiere una tendencia hacia la exploración de nuevas teorías y desarrollo de nuevos conocimientos relacionados a la arquitectura de microservicios en un contexto de DevOps. En la Tabla 5.4 se encuentran las referencias de las publicaciones por cada ámbito de estudio.

Dentro del contexto industrial se identificaron los sectores en donde más se han aplicado los conceptos de patrones de microservicios en un contexto DevOps, los cuales se pueden observar en la Figura 5.5. La mayoría de los artículos que se encuentran en este grupo, han aplicado los conceptos estudiados en los sectores de telecomunicaciones, automotriz, retail y servicios de TI. Así mismo, se puede obtener la relación de publicaciones por cada sector de la industria en la Tabla 5.5.

Figura 5.4. Cantidad de publicaciones por ámbito de investigación

Contexto de la investigación	Publicaciones
Académico	[10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [2], [20], [21], [23], [24], [26], [27], [28], [3], [30], [31], [34], [35], [36], [41], [42], [43], [45], [47], [5], [51], [52], [53], [55], [57], [58], [59], [62], [64], [65], [66], [9]
Industria	[1], [22], [25], [29], [32], [33], [37], [38], [39], [4], [40], [44], [46], [48], [49], [50], [54], [56], [6], [60], [61], [63], [7], [8]

Tabla 5.4. Relación de publicaciones por contexto en el que se realizó la investigación

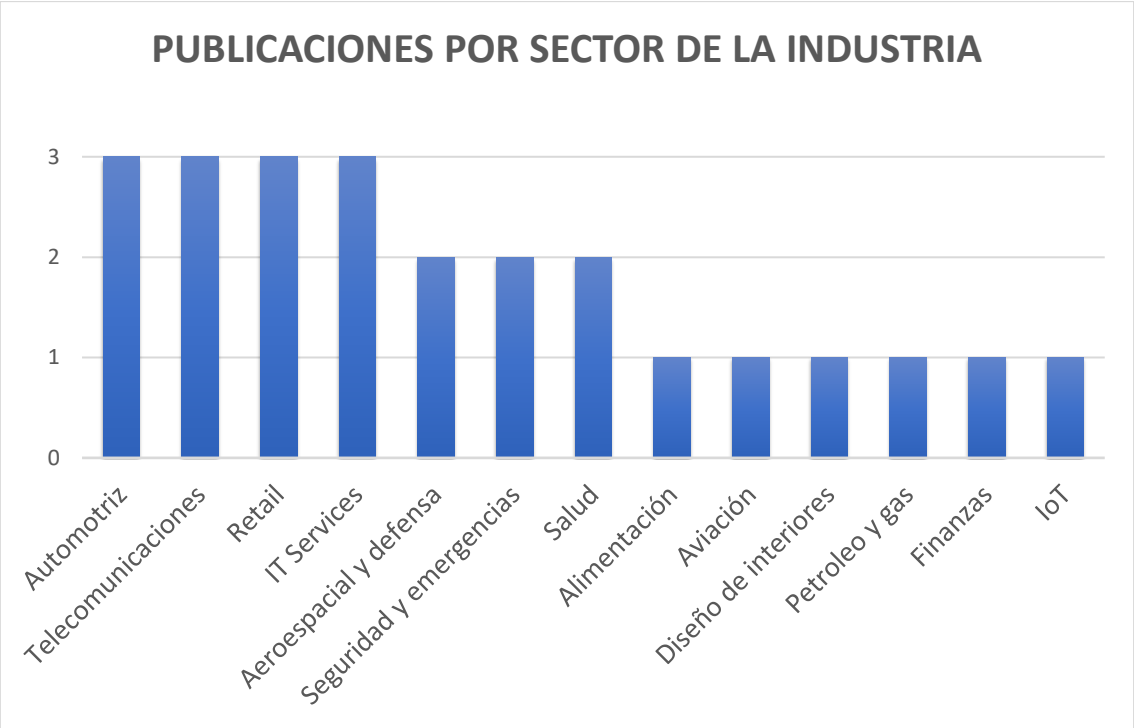

Figura 5.5. Cantidad de publicaciones por sector de la industria

Sector de la industria	Artículos
Telecomunicaciones	[1], [44], [48]
Automotriz	[38], [50], [63]
Retail	[22], [39], [60]
IT Services	[32], [33], [37]
Salud	[4], [6]
Seguridad y emergencias	[40], [7]
Aeroespacial y defensa	[46], [8]
Alimentación	[25]
Aviación	[61]
Diseño de interiores	[49]
Finanzas	[56]

IoT	[54]
Petroleo y gas	[29]

Tabla 5.5. Relación de artículos por sector de la industria

5.2. Preguntas de investigación

PI1: ¿Cuáles son los patrones de microservicios más aplicados junto con alguna práctica o herramienta DevOps?

Los patrones de microservicios se han clasificado según el área de diseño en la que se aplican, como se muestra en Figura 5.6. Esta figura destaca que los patrones de microservicios más frecuentes están relacionados al área de diseño de “Despliegue”. En la Figura 5.7, se puede encontrar de manera más detallada la cantidad de menciones por cada patrón. Así mismo, la Tabla 5.6 proporciona la relación de artículos que mencionan cada patrón.

Dentro del contexto DevOps, el patrón de microservicios “Service-per-container” predomina con una presencia en 53 artículos. El “Remote Procedure Invocation (RPI)” es el segundo patrón más utilizado en conjunto con DevOps, con un total de 15 menciones. El tercer patrón más empleado es el “API Gateway”, con 7 menciones. La frecuencia de estos 3 patrones más empleados, así como los otros que se identificaron en los artículos seleccionados, se puede encontrar en la Figura 5.7.

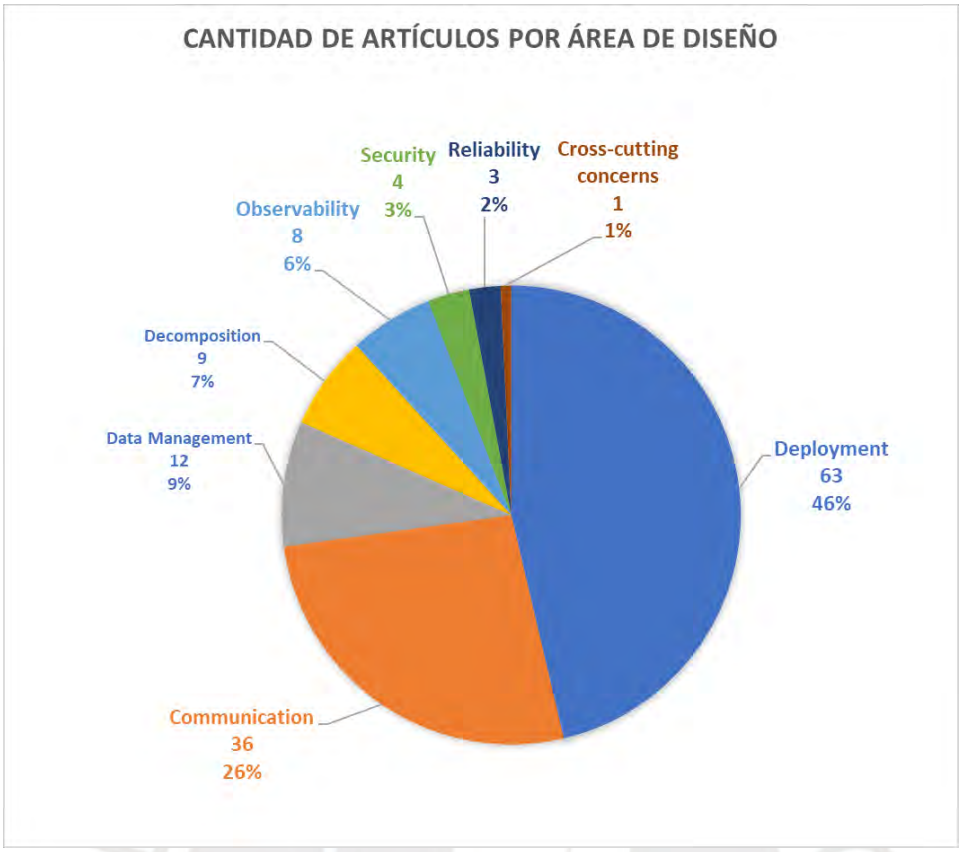

Figura 5.6. Cantidad de artículos que abordan un área de diseño de microservicios

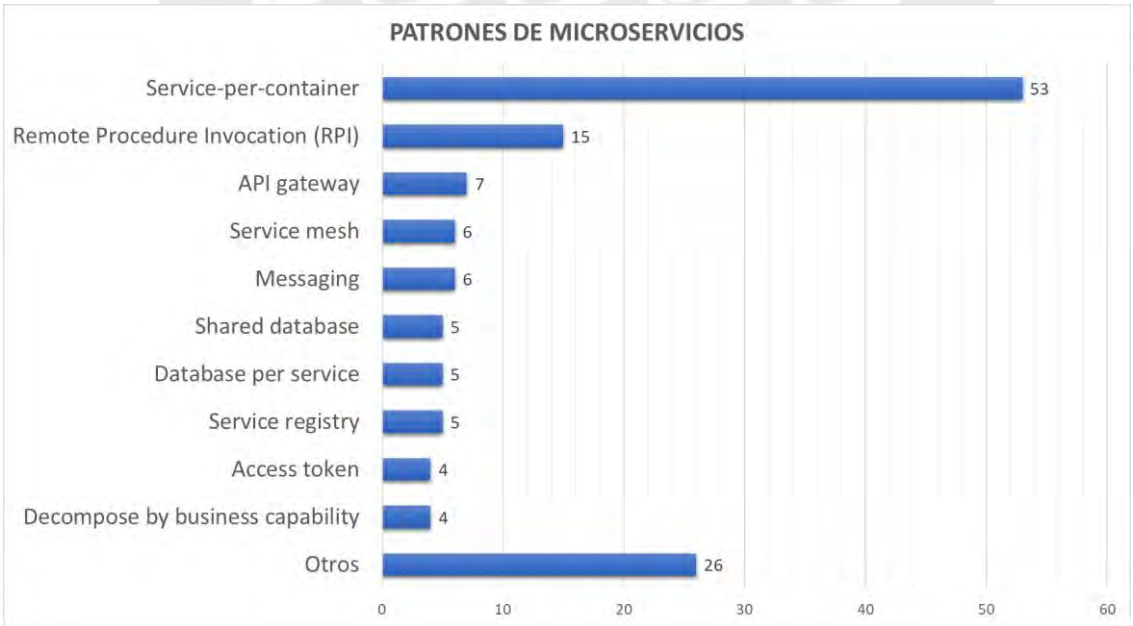

Figura 5.7. Patrones de microservicios empleados en los estudios seleccionados

Área de diseño	Patrón de microservicios	Artículos
Communication	API gateway	[10], [18], [25], [39], [46], [57], [65]
	Client-side discovery	[37]
	Mediator	[48]
	Messaging	[22], [28], [31], [32], [37], [54]
	Remote Procedure Invocation (RPI)	[15], [16], [19], [25], [29], [3], [32], [34], [4], [44], [46], [5], [57], [69], [8]
	Service discovery	[44]
	Service registry	[10], [18], [25], [37], [44]
Cross-cutting concerns	Externalized configuration	[37]
Data Management	Database per service	[22], [25], [39], [48], [65]
	Domain event	[48]
	Saga	[48]
	Shared database	[19], [31], [32], [33], [44]
Decomposition	Decompose by business capability	[4], [42], [58], [65]
	Decompose by subdomain	[34], [37], [48]
	Decomposition by business capability	[45]
	Decomposition by non-functional requirement	[55]
Deployment	Multiple Services per host	[23]
	Serverless deployment	[29], [32]
	Service mesh	[10], [11], [12], [17], [25], [30]

	Service-per-container	[1], [10], [12], [13], [14], [15], [16], [17], [18], [19], [2], [21], [22], [23], [24], [25], [26], [27], [28], [3], [31], [33], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [47], [49], [5], [50], [51], [52], [53], [56], [58], [59], [61], [62], [63], [64], [66], [67], [69], [7], [8], [9]
	Sidecar	[12]
Observability	Application metrics	[17], [21]
	Distributed tracing	[11], [12]
	Health check API	[38]
	Log aggregation	[18], [37], [44]
Reliability	Circuit Breaker	[11], [37]
	Synchronizer	[20]
Security	Access token	[10], [39], [44], [46]

Tabla 5.6. Relación de artículos que emplean un patrón de microservicios

PI2: ¿Cuáles son las práctica o principio DevOps más utilizados al implementar la arquitectura de microservicios?

La Figura 5.8 presenta las prácticas más relevantes de DevOps asociadas a la implementación de la arquitectura de microservicios. La “Integración continua” (Continuous integration) se destaca, habiéndose empleado en un 29% de los artículos evaluados. Le sigue la “Entrega continua” (Continuous delivery), con un 14%. Las siguientes dos prácticas que cuentan con un 11%, son “Despliegue continuo” (Continuous deployment) e “Infraestructura como código” (Infrastructure as code).

Figura 5.8. Prácticas DevOps empleadas en los artículos seleccionados

La frecuencia de uso de las herramientas DevOps se puede observar en la Figura 5.9. En este gráfico, predomina el uso de Docker, reflejando la importancia del uso de contenedores en el despliegue de la arquitectura de microservicios. Así mismo, la segunda herramienta más utilizada es Kubernetes, lo cual refuerza la tendencia hacia la contenerización porque tiene un rol esencial en la orquestación de los contenedores. La tercera herramienta más utilizada es Jenkins, la cual es utilizada en la práctica de “Integración Continua” por lo tanto se alinea a la predominación de esta práctica en la adopción de la cultura DevOps.

La relación de artículos que emplean cada una de las prácticas y herramientas DevOps se puede encontrar en la Tabla 5.7.

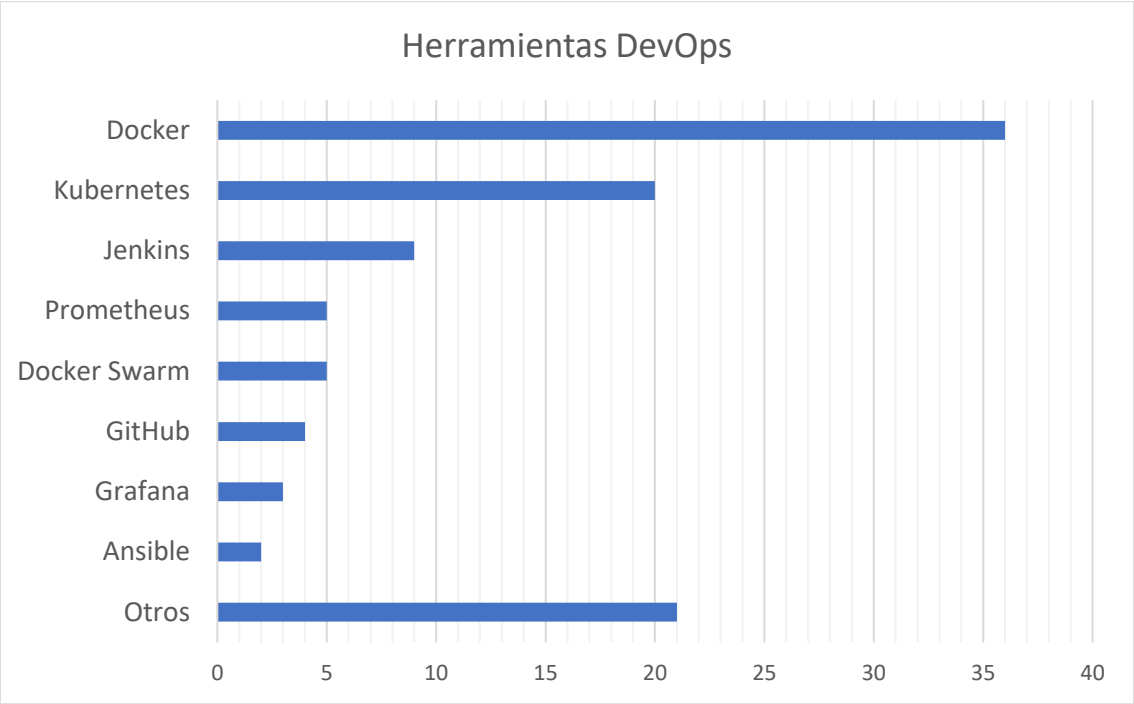

Figura 5.9. Herramientas DevOps empleadas en los artículos seleccionados

Tipo	Práctica o herramienta	Artículos
Herramienta	Ansible	[14], [21]
	Apache Jmeter	[20]
	Apache Mesos	[41]
	Azure Monitor	[16]
	cAdvisor	[7]
	Chef	[39]
	Docker	[13], [14], [15], [17], [18], [2], [21], [22], [23], [24], [25], [26], [28], [29], [32], [34], [36], [37], [38], [39], [4], [40], [41], [42], [43], [45], [46], [47], [49], [52], [54], [56], [59], [6], [61], [7]
	Docker compose	[36]
	Docker Hub	[36], [38]
	Docker Swarm	[2], [22], [41], [51], [7]

	Elastic Container Registry	[37]
	Git	[36], [39]
	GitHub	[25], [36], [45], [49]
	GitLab	[24], [37]
	Grafana	[16], [17], [7]
	Jenkins	[16], [18], [20], [36], [37], [38], [39], [45], [61]
	Kiali	[17]
	Kubernetes	[1], [14], [15], [16], [17], [18], [19], [20], [21], [22], [24], [25], [26], [28], [36], [41], [46], [49], [51], [61]
	Mesosphere Marathon	[51]
	Prometheus	[14], [17], [21], [22], [7]
	Puppet	[39], [46]
	Rancher	[41]
	Rundeck	[36]
	Selenium	[59]
	Terraform	[14]
Práctica	Application monitoring	[17]
	Automated deployment	[17], [18]
	Automated monitoring	[33]
	Automated testing	[11], [12], [15], [32], [39], [60], [8], [9]
	Autonomous scaling	[27]
	Autonomous self-configuration	[27]
	Colaboración entre el equipo de desarrollo y operaciones	[13], [16], [33], [35], [49], [55], [7]

	Continuous delivery	[18], [20], [24], [30], [32], [36], [37], [38], [44], [46], [48], [49], [56], [58], [8]
	Continuous deployment	[13], [15], [16], [18], [20], [25], [27], [50], [52], [6], [60], [9]
	Continuous integration	[13], [15], [16], [18], [2], [20], [24], [25], [27], [29], [30], [32], [36], [37], [38], [39], [44], [45], [48], [49], [50], [52], [56], [58], [59], [6], [60], [61], [66], [8], [9]
	Continuous monitoring	[1], [10], [14], [33], [38], [44], [46], [48], [5], [7]
	Continuous testing	[36], [5]
	GitOps	[1]
	Infrastructure as a service	[28]
	Infrastructure as code	[1], [14], [17], [21], [26], [30], [34], [36], [49], [59], [62], [64]
	Log file analysis	[52]
	MLOps	[4]

Tabla 5.7. Relación de artículos donde se emplea alguna prácticas o herramienta DevOps

PI3: ¿En qué sectores de la industria se emplean los patrones de microservicios en proyectos ejecutados bajo un enfoque DevOps?

Los sectores de la industria que emplean patrones de microservicios se listan en la Tabla 5.5. En esta tabla, se clasifican los patrones de microservicios aplicados en cada sector de la industria. El uso del patrón “Service-per-container” se identificó en todos los sectores, con excepción del de Finanzas.

Sector de la industria	Patrón de microservicios	Frecuencia
Aeroespacial y defensa	Access token	1
	Log aggregation	1
	Remote Procedure Invocation (RPI)	2
	Service discovery	1

	Service registry	1
	Service-per-container	2
	Shared database	1
Alimentación	API gateway	1
	Database per service	1
	Remote Procedure Invocation (RPI)	1
	Service mesh	1
	Service registry	1
	Service-per-container	1
Automotriz	Circuit Breaker	1
	Client-side discovery	1
	Database per service	1
	Decompose by subdomain	2
	Domain event	1
	Externalized configuration	1
	Log aggregation	1
	Mediator	1
	Messaging	1
	Saga	1
	Service registry	1
	Service-per-container	2
Aviación	Service-per-container	1
Diseño de interiores	Service-per-container	1
Finanzas	Messaging	1
IoT	Service-per-container	1
IT Services	Messaging	2
	Remote Procedure Invocation (RPI)	1
	Serverless deployment	1
	Service-per-container	2
	Shared database	2
Retail	Database per service	1
	Decompose by business capability	1
	Health check API	1

	Messaging	1
	Service-per-container	3
Salud	Remote Procedure Invocation (RPI)	2
	Service-per-container	2
Seguridad y emergencias	Access token	1
	API gateway	1
	Database per service	1
	Service-per-container	2
Telecomunicaciones	Access token	1
	API gateway	1
	Decompose by business capability	1
	Remote Procedure Invocation (RPI)	1
	Service-per-container	2
Energía	Messaging	1
	Service-per-container	1

Tabla 5.8. Patrones de microservicios empleados por sector de la industria

La Figura 5.10 representa la popularidad de los patrones de microservicios en cada sector de la industria. Se observa que el patrón de microservicios “Service-per-container” lidera en adopción, lo que sugiere que la contenerización es una estrategia preferida en múltiples industrias. Siguiendo en popularidad se encuentra el patrón “Messaging” y “Remote Procedure Invocation” destacando la importancia de la comunicación entre microservicios en varios sectores de la industria.

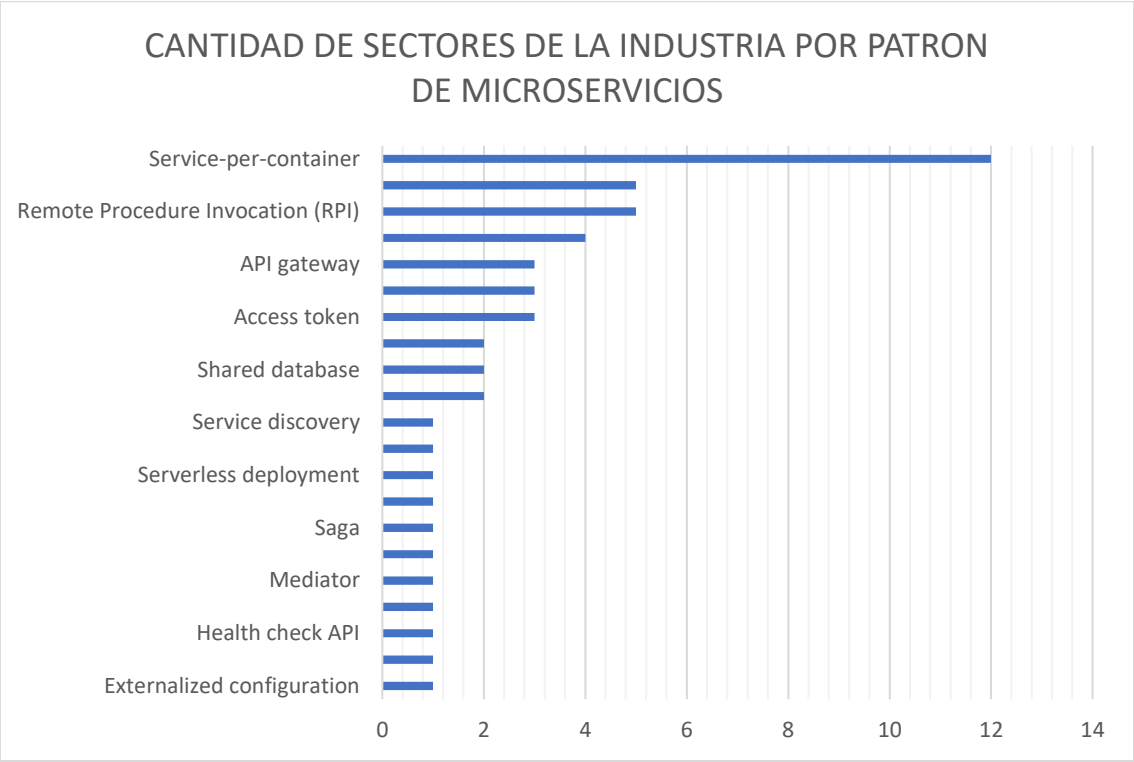

Figura 5.10. Cantidad de sectores de la industria donde se ha empleado un patrón de microservicios

PI4: ¿Qué ventajas se obtienen al implementar los patrones de microservicios en proyectos ejecutados bajo un enfoque DevOps?

La Figura 5.11, muestra las 11 ventajas más relevantes al implementar patrones de microservicios en los artículos seleccionados. Se observa que las ventajas "Mejora de la escalabilidad y flexibilidad" e "Independencia de microservicios", tienen el mayor número de menciones, con un total de 21 cada una. La tercera ventaja más frecuente es "Despliegue eficiente", con un total de 13 menciones. En la Tabla 5.9, se puede encontrar la relación de artículos que mencionan las ventajas que se muestran en la Figura 5.11.

Figura 5.11. Ventajas más relevantes al implementar patrones de microservicios

Ventaja	Artículos
Reducción de Sobrecarga de Rendimiento	[2]
Flexibilidad en la implementación	[3], [6], [4]
Compatibilidad con CI/CD	[24], [31], [3]
Flexibilidad en el Uso de Lenguajes	[33], [8], [46]
Compatibilidad con la Computación en la Nube	[8], [52], [22], [50]
Escalamiento automático	[38], [8], [14], [33]
Mejora la seguridad	[40], [57], [48], [8], [27], [42]
Tolerancia a Fallos	[38], [10], [1], [50], [52], [13], [3]
Despliegue Eficiente	[44], [58], [51], [15], [1], [17], [46], [19], [53], [20], [63], [22], [41]

Independencia de Microservicios	[39], [60], [49], [17], [19], [56], [2], [66], [22], [46], [23], [53], [24], [57], [25], [61], [37], [8], [10], [44]
Mejora de la Escalabilidad y Flexibilidad	[54], [48], [7], [17], [51], [23], [6], [3], [1], [30], [50], [33], [52], [36], [58], [39], [60], [41], [8], [42], [46]

Tabla 5.9. Relación de artículos por cada ventaja identificada en los artículos seleccionados

PI5: ¿Cómo ha variado en el tiempo el uso de los diferentes patrones de microservicios en proyectos ejecutados bajo un enfoque DevOps?

Para responder a la pregunta de investigación 5, se evaluó la tendencia anual de implementación de los tres patrones de microservicios más usados identificados en la PI1: “Service-per-container”, “Remote Procedure Invocation” y “API Gateway”. Según la Figura 5.12, el patrón “Service-per-container” se ha empleado consistentemente desde el inicio de los registros sobre el tema, es decir desde el 2015 hasta el 2023, manteniendo un porcentaje igual o mayor al 50%.

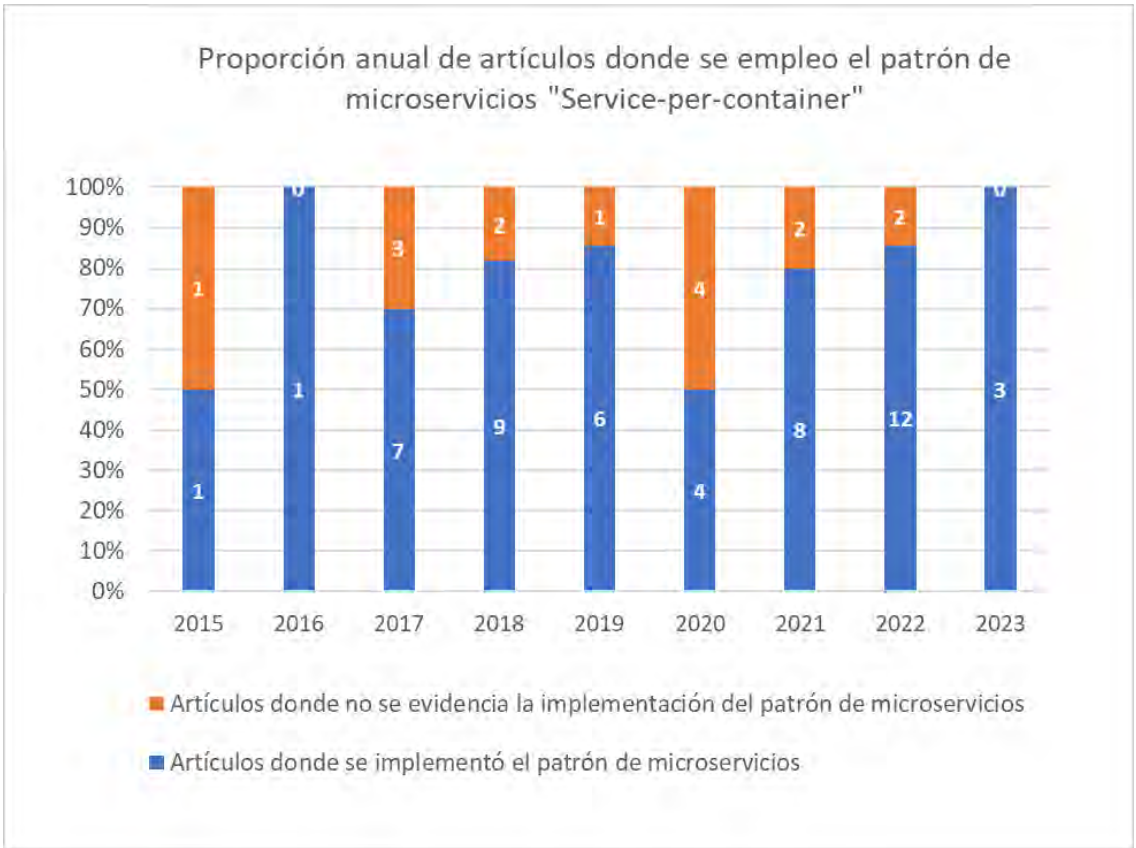

Figura 5.12. Proporción anual de artículos donde se empleó el patrón de microservicios "Service-per-container"

La Figura 5.13 muestra la proporción anual de artículos que aplicaron el patrón de microservicios "Remote Procedure Invocation" en comparación con aquellos que no reportan su uso. En los primeros años, 2015 y 2016, no se identificó ningún artículo donde se haya empleado. A partir del 2017, se evidencia su adopción, manteniendo una tendencia ascendente, a excepción del año 2019 donde no se identificó ningún artículo donde se haya empleado. En el año 2022 se observa la mayor cantidad de artículos donde se aplica este patrón y representa más del 40% de las publicaciones de ese año.

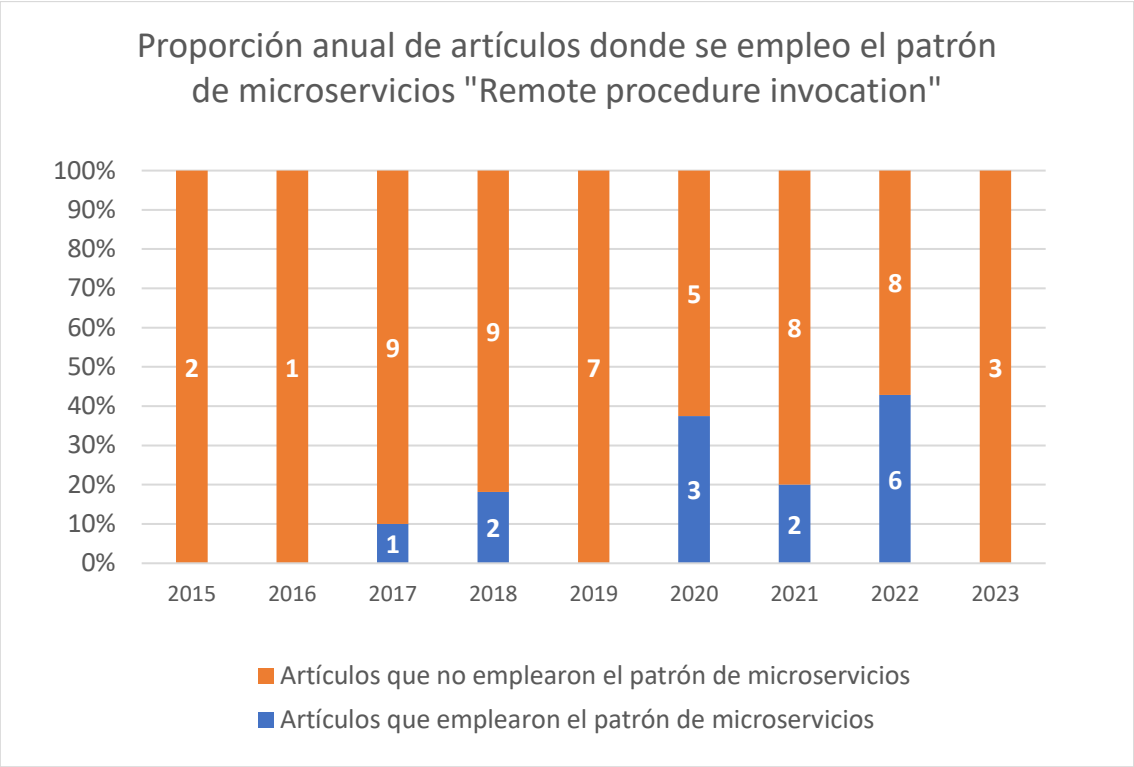

Figura 5.13. Proporción anual de artículos donde se empleó el patrón de microservicios "Remote Procedure Invocation"

La Figura 5.14 muestra la proporción anual de artículos que aplicaron el patrón de microservicios "API Gateway" en comparación con aquellos que no reportan su uso. Se puede observar que el patrón se implementaba desde el 2015, sin embargo, en los años 2016, 2020 y 2023 no se evidencia su aplicación. En el año 2022, se observa el mayor número de artículos que aplicaron este patrón, aunque representa solo el 20% del total de artículos de ese año.

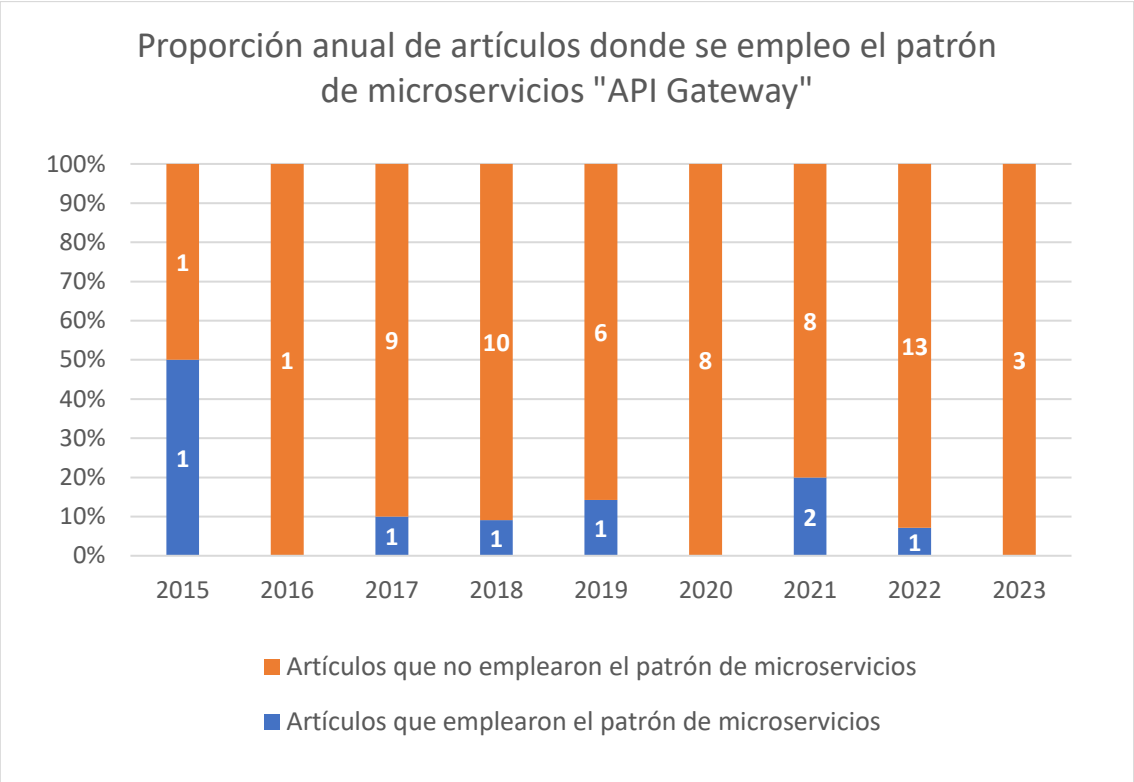

Figura 5.14. Proporción anual de artículos donde se empleó el patrón de microservicios "API Gateway"

CAPÍTULO VI. CONCLUSIONES

La sinergia de la cultura DevOps y la arquitectura de microservicios ha mostrado una evolución desde sus inicios. Aunque DevOps comenzó a consolidarse en 2009 y los microservicios ganaron relevancia en 2014, no fue sino hasta 2015 que se evidencia la combinación de ambos conceptos en la literatura, y a partir de ahi se evidencia un creciente interés y adopción de estas prácticas. Este aumento se mantuvo con variaciones menores hasta 2021, pero en 2022 se aprecia un aumento significativo, lo que podría indicar un fortalecimiento de la implementación de microservicios junto con la adopción de DevOps. De todas las publicaciones seleccionadas en el trabajo de investigación, el 80% son artículos de conferencia. Las tres conferencias más populares para este tipo de publicaciones son la "ACM International Conference Proceeding Series", "Communications in Computer and Information Science" y la "IEEE International Conference on Consumer Electronics".

En el contexto de la cultura DevOps, se ha identificado que el patrón de microservicios “Service-per-container” predomina como el patrón más aplicado. La popularidad de este patrón, que se considera como parte del área de diseño de “Despliegue”, refleja una fuerte preferencia por la contenerización en la implementación de microservicios, en concordancia con las tendencias actuales de desarrollo de software. Por otro lado, el patrón “Remote Procedure Invocation (RPI)”, que ocupa el segundo lugar en términos de uso junto a DevOps, se asocia con el área de “Comunicación” y destaca por incorporar la tecnología REST, lo cual es un factor significativo en la popularidad de este patrón. El tercer patrón más empleado, que también pertenece al área de “Comunicación”, es el patrón “API Gateway”. De estos tres, el patrón “Remote Procedure Invocation” ha experimentado un aumento en su implementación en los últimos años, mientras que la popularidad de los otros dos ha permanecido relativamente estable. Las ventajas más significativas al implementar estos patrones de microservicios son: “Mejora en la escalabilidad y flexibilidad” e “Independencia de microservicios”.

Los resultados del mapeo sistemático indican una adopción significativa de prácticas DevOps, siendo la “Integración continua” (Continuous integration) la práctica más aplicada en proyectos donde se implementa la arquitectura de

microservicios, lo cual indica el papel fundamental de la automatización en el proceso de desarrollo de software. Le sigue la “Entrega continua” (Continuous delivery), destacando su importancia para la entrega rápida y eficiente del software. Las siguientes dos prácticas, que cuentan con la misma relevancia, son “Despliegue continuo” (Continuous deployment) e “Infraestructura como código” (Infrastructure as code). El predominio de estas 4 prácticas confirma su importancia en la adopción de la cultura DevOps.

CAPÍTULO VII. TRABAJO FUTURO

Para futuras investigaciones, se propone desarrollar un estudio que evalúe la efectividad de la implementación de los patrones de microservicios en entornos DevOps, utilizando métricas de rendimiento, mantenibilidad y escalabilidad.

Por otro lado, se sugiere profundizar en las sinergias entre patrones de microservicios y prácticas DevOps, con el fin de identificar y documentar buenas prácticas o configuraciones eficientes.

REFERENCIAS BIBLIOGRÁFICAS

Aguiar Monteiro, L. De, Pessoa Monteiro, D. S. M., Carvalho Almeida, W. H., Cavalcanti De Lima, A., & Sette, I. S. (2020). Methods of Implementation, Maturity Models and Definition of Roles in DevOps Frameworks: A Systematic Mapping. *Proceedings - 2020 International Conference on Computational Science and Computational Intelligence, CSCI 2020*, 1766–1773. https://doi.org/10.1109/CSCI51800.2020.00327

Akbar, M. A., Rafi, S., Alsanad, A. A., & Qadri, S. F. (2022). Toward Successful DevOps : A Decision-Making Framework. *IEEE Access*, *10*, 51343–51362.

Amaro, R. M. D., Pereira, R., & Mira da Silva, M. (2022). Capabilities and Practices in DevOps: A Multivocal Literature Review. *IEEE Transactions on Software Engineering*, *5589*(c), 1–1. https://doi.org/10.1109/tse.2022.3166626

Assunção, W. K. G., Krüger, J., Mosser, S., & Selaoui, S. (2023). How do microservices evolve? An empirical analysis of changes in open-source microservice repositories. *Journal of Systems and Software*, *204*. https://doi.org/10.1016/j.jss.2023.111788

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

Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). *Microservices Architecture Enables DevOps Migration to a Cloud-Native Architecture*. www.pegahtech.ir

Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, *150*, 77–97. https://doi.org/10.1016/j.jss.2019.01.001

Di Francesco, P., Malavolta, I., & Lago, P. (2017). Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, 21–30. https://doi.org/10.1109/ICSA.2017.24

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE SOFTWARE*.

Filippone, G., Autili, M., Rossi, F., & Tivoli, M. (2021). Migration of Monoliths through the Synthesis of Microservices using Combinatorial Optimization. *Proceedings - 2021 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2021*, 144–147. https://doi.org/10.1109/ISSREW53611.2021.00056

Jabbari, R., Ali, N. Bin, Petersen, K., & Tanveer, B. (2016). What is DevOps? A systematic mapping study on definitions and practices. *ACM International Conference Proceeding Series*, *24-May-201*. https://doi.org/10.1145/2962695.2962707

Kitchenham, B., Stuart, C., David, B., & Brereton, P. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. In *EBSE Technical Report*.

Lazuardi, M., Raharjo, T., Hardian, B., & Simanungkalit, T. (2021). Perceived Benefits of DevOps Implementation in Organization: A Systematic Literature Review. *ACM International Conference Proceeding Series*, 10–16. https://doi.org/10.1145/3512716.3512718

Lewis, J. (2014). *Microservices*. https://www.martinfowler.com/articles/microservices.html

Liang, Q. (2015). Continuous Delivery Huge Benefits, but Challenges Too. *IEEE SOFTWARE*. https://doi.org/10.1201/9781003221579

Morris, K. (2016). *Infrastructure as Code* (2nd ed.). O'Reilly Media, Inc.

Ntentos, E., Zdun, U., Plakidas, K., & Geiger, S. (2021). Semi-automatic Feedback for Improving Architecture Conformance to Microservice Patterns and Practices. *Proceedings - IEEE 18th International Conference on Software Architecture, ICSA 2021*, 36–46. https://doi.org/10.1109/ICSA51549.2021.00012

Osses, F., Márquez, G., & Astudillo, H. (2019). *An Exploratory Study of Academic Architectural Tactics and Patterns in Microservices: A systematic literature review*. https://www.researchgate.net/publication/331022065

Pahl, C., Jamshidi, P., & Zimmermann, O. (2018). Architectural principles for cloud software. *ACM Transactions on Internet Technology*, *18*(2). https://doi.org/10.1145/3104028

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Richardson, C. (n.d.-a). *A pattern language for microservices*. Retrieved September 2, 2023, from https://microservices.io/patterns/

Richardson, C. (n.d.-b). *Microservice Architecture*. What Are Microservices? Retrieved September 2, 2023, from https://microservices.io/

Richardson, C. (2018). *Microservices patterns*. Manning.

Schneider, T. (2016). *Achieving Cloud Scalability with Microservices and DevOps in the Connected Car Domain*.

Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, *5*(Ci), 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). *Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation*. http://tomcat.apache.org/

Taibi, D., Lenarduzzi, V., & Pahl, C. (2018). Architectural patterns for microservices: A systematic mapping study. *CLOSER 2018 - Proceedings of the 8th International Conference on Cloud Computing and Services Science*, *2018-January*, 221–232. https://doi.org/10.5220/0006798302210232

Taibi, D., Lenarduzzi, V., & Pahl, C. (2019). Continuous architecting with microservices and DevOps: A systematic mapping study. *Communications in Computer and Information Science*, *1073*, 126–151. https://doi.org/10.1007/978-3-030-29193-8_7

Waseem, M., Liang, P., Ahmad, A., Shahin, M., Khan, A. A., & Márquez, G. (2022a). *Decision models for selecting patterns and strategies in microservices systems and their evaluation by practitioners*. 135–144. https://doi.org/10.1145/3510457.3513079

Waseem, M., Liang, P., Ahmad, A., Shahin, M., Khan, A. A., & Márquez, G. (2022b). Decision Models for Selecting Patterns and Strategies in Microservices Systems and their Evaluation by Practitioners. *National Key R&D Program of China*, 39–40. https://doi.org/10.1145/XXXXXXX.XXXXXXX

Watts, S. (2019). *A Brief History of DevOps*. BMC DevOps Guide. https://www.bmc.com/blogs/devops-history/

Wedyan, F., & Abufakher, S. (2020). Impact of design patterns on software quality: A systematic literature review. In *IET Software* (Vol. 14, Issue 1, pp. 1–17). Institution of Engineering and Technology. https://doi.org/10.1049/iet-sen.2018.5446

Weerasinghe, S., & Perera, I. (2022). Taxonomical Classification and Systematic Review on Microservices. In *International Journal of Engineering Trends and Technology* (Vol. 70, Issue 3, pp. 222–233). Seventh Sense Research Group. https://doi.org/10.14445/22315381/IJETT-V70I3P225

Wouter de Kort. (2016). *DevOps on the Microsoft Stack* (Apress, Ed.).