Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

UTN.BA
ESCUELA DE
POSGRADO

TRABAJO FINAL INTEGRADOR

ESPECIALIZACIÓN EN INGENIERÍA EN
SISTEMAS DE INFORMACIÓN

De monolitos a microservicios: tendencias y oportunidades en la transición de arquitecturas de software

Alumno: Ing. Fernando Muraca

Directora: Dra. María Florencia Pollo-Cattaneo

Tutoras: Dra. María Florencia Pollo-Cattaneo
Mg. Cinthia Soledad Vegega

Ciudad Autónoma de Buenos Aires, Noviembre de 2023

Resumen

Considerando el contexto de la migración de aplicaciones monolíticas a microservicios y basado en un mapeo sistemático de la literatura académica reciente, este trabajo de investigación examina los enfoques, desafíos, antipatrones y mejores prácticas identificadas. A través de una revisión de estudios y publicaciones en revistas especializadas, se identifican las tendencias actuales y se destacan las áreas de investigación futuras en el campo de la migración de aplicaciones monolíticas a microservicios. Además, se discuten las implicaciones prácticas y teóricas de estos hallazgos. El objetivo es proporcionar una visión completa y actualizada del estado del arte en esta área de estudio.

Palabras clave

migración, aplicaciones monolíticas, microservicios, mapeo sistemático, literatura, enfoques, desafíos, antipatrones, mejores prácticas.

Abstract

Considering the context of migrating from monolithic applications to microservices and based on a systematic review of recent academic literature, this research work examines identified approaches, challenges, anti-patterns, and best practices. Through a comprehensive review of studies and publications in specialized journals, current trends are identified, and future research areas in the field of migrating monolithic applications to microservices are highlighted. Monolith decomposition processes associated technical and organizational issues, as well as strategies and recommendations proposed in the literature are analyzed in depth. The aim is to provide a comprehensive and up-to-date overview of the state-of-the-art in this area of study.

Keywords

migration, monolithic applications, microservices, systematic mapping, literature, approaches, challenges, anti-patterns, best practices.

Contenido

Capítulo 1: Fundamentos y alcances de la investigación

El presente capítulo introduce el trabajo estableciendo el contexto y la justificación del estudio mediante una discusión crítica del estado actual del conocimiento sobre la problemática de migración de aplicaciones monolíticas a microservicios.

Se precisan el objetivo general y los objetivos específicos perseguidos a través de un mapeo sistemático de literatura en este campo.

Asimismo, se delimita el alcance de la investigación en cuanto a las fuentes bibliográficas a considerar, el lapso temporal de interés y los tipos de estudios primarios a seleccionar.

En adición, se presentan los fundamentos metodológicos que guían el desarrollo sistemático de la investigación sobre la base de la técnica de mapeo sistemático de literatura.

Finalmente, se describe la estructura general del trabajo definiendo los capítulos principales y su contenido asociado.

Este capítulo introductorio se desglosa en diversas secciones que aportan una comprensión detallada de los elementos fundamentales que enmarcan el estudio sobre la migración de aplicaciones monolíticas a microservicios. La sección 1.1 establece el contexto general de la investigación y su relevancia. En la sección 1.2, se delinean los objetivos específicos del estudio, proporcionando una visión clara de sus metas. La sección 1.3 establece los límites de la investigación, indicando qué aspectos se incluyen y cuáles quedan fuera de su alcance. Los fundamentos teóricos se exponen en la sección 1.4, en términos de conceptos y teorías relevantes que respaldan la investigación, ofreciendo un marco conceptual sólido. En la sección 1.5, se presenta la organización general del documento, resaltando la distribución de capítulos y secciones para facilitar la navegación del lector en el contenido del trabajo. Finalmente, la sección 1.6 contiene la producción científica derivada.

1.1 Introducción

La migración de aplicaciones monolíticas a arquitecturas de microservicios se ha convertido en un área de intenso interés e investigación en el campo de la Ingeniería de Software en los últimos años. Esto se debe a que la arquitectura de microservicios ofrece importantes ventajas

sobre las tradicionales aplicaciones monolíticas en dimensiones clave como escalabilidad, mantenibilidad y capacidad de adaptación a nuevos requerimientos [1, 2].

La arquitectura de microservicios ha emergido como un paradigma prometedor para desarrollar aplicaciones empresariales complejas y escalables. Bajo este enfoque, una aplicación se compone de múltiples servicios independientes, altamente desacoplados y enfocados en capacidades de negocio específicas [3]. Cada microservicio se ejecuta en su propio proceso, interactúa mediante APIs (Interfaz de Programación de Aplicaciones) bien definidas y se puede desplegar, escalar y gestionar de forma aislada [4].

En contraste, las aplicaciones monolíticas concentran toda su lógica en un solo proceso. Si bien inicialmente esto posibilita agilidad y simplicidad, conforme crecen dichas aplicaciones se vuelven rígidas y difíciles de mantener. Cualquier cambio, por mínimo que sea, obliga a reconstruir y reimplementar la aplicación completa. Además, debido a su fuerte acoplamiento interno, las aplicaciones monolíticas limitan la escalabilidad e innovación, al impedir que algunos componentes evolucionen o se desplieguen de forma independiente [3, 5, 6].

La migración hacia microservicios requiere descomponer una aplicación monolítica en componentes más pequeños, delimitados y desplegables de forma aislada. Sin embargo, llevar a cabo este proceso de transición resulta sumamente complejo y problemático. Esto se debe principalmente a que se deben reorganizar radicalmente la arquitectura de componentes internos y administrar cuidadosamente un entramado de nuevas dependencias y comunicaciones entre los servicios resultantes de la descomposición [4, 7, 8]. Además, se debe realizar una adaptación a patrones de diseño y arquitecturas de software muy diferentes a los de las aplicaciones monolíticas tradicionales, como por ejemplo la comunicación asíncrona entre servicios y la persistencia de datos descentralizada [2, 4].

Otro factor que incrementa la criticidad de este proceso es que cualquier defecto, o problema no gestionado adecuadamente en la migración, puede introducir importantes riesgos en el funcionamiento de sistemas empresariales completos que suelen soportar operaciones de negocio críticas [9].

En este contexto, debido a la criticidad y complejidad de los procesos de migración de monolitos a microservicios, se requiere un abordaje de investigación sistemático y en profundidad para identificar enfoques, problemáticas, buenas prácticas y oportunidades de mejora en esta área de conocimiento.

1.2 Objetivo

El presente trabajo de investigación tiene como principal objetivo realizar un mapeo sistemático de la literatura científica actual en el campo de la Ingeniería de Software dedicada específicamente al estudio de los diversos enfoques, principales desafíos, antipatrones (patrones de diseño ineficientes o propensos a errores) y recomendaciones (buenas prácticas) relativos al proceso de migración desde aplicaciones monolíticas legacy hacia modernas arquitecturas basadas en microservicios.

Mediante una extensa revisión de los estudios y publicaciones en revistas y congresos especializados en Ingeniería de Software e Ingeniería de Sistemas de Información, se busca identificar y categorizar las tendencias más recientes en esta línea de investigación, así como también destacar futuras áreas y aspectos de vacancia para el desarrollo de nuevos estudios que puedan aportar conocimientos relevantes tanto para la comunidad académica como para la práctica profesional en este dominio de estudio.

El mapeo sistemático de literatura se selecciona como metodología de investigación porque brinda un procedimiento estructurado, objetivo y replicable para obtener una visión integral del estado del arte sobre un tema de estudio específico. Esta metodología permite identificar, categorizar y sintetizar la evidencia disponible a partir de una revisión sistemática de fuentes relevantes.

Los mapeos sistemáticos son ampliamente utilizados para estudiar un campo de conocimiento emergente y encontrar respuestas a preguntas de investigación sobre tendencias, coincidencias y vacíos en la literatura existente. Proporcionan una sólida base conceptual y teórica, al recopilar y organizar el conocimiento previo sobre el problema de investigación.

Para este estudio, el mapeo sistemático resulta apropiado dada la creciente producción de literatura sobre la migración de monolitos a microservicios y la necesidad de comprender el estado actual de las investigaciones en este campo. La aplicación rigurosa del protocolo metodológico permite obtener un panorama integral y una taxonomía de los enfoques, desafíos, antipatrones y recomendaciones reportadas en la literatura especializada reciente.

El objetivo primordial es obtener una visión actualizada, integral y sistemática del estado del conocimiento en este campo de estudio mediante un riguroso mapeo de literatura, identificando aspectos clave, así como oportunidades para desarrollos investigativos futuros.

Específicamente, en trabajo se propone:

- Examinar los diversos enfoques y estrategias propuestos para llevar a cabo la descomposición de aplicaciones monolíticas en microservicios.
- Identificar y analizar los principales desafíos y problemáticas reportados durante los procesos de migración.
- Explorar los antipatrones comunes que deben evitarse en la transición a microservicios.
- Recopilar recomendaciones y buenas prácticas sugeridas en la literatura para facilitar migraciones efectivas.
- Resaltar áreas de vacancia en la investigación actual que requieren mayor estudio.

1.3 Alcance

El alcance del mapeo sistemático de literatura se circunscribe a la búsqueda en bases de datos académicas especializadas, la selección y el subsiguiente análisis de estudios primarios publicados en revistas y actas de congresos de alto impacto en el dominio de la Ingeniería de Software y la Ingeniería de Sistemas de Información [10].

El recorte temporal considerado se enfoca específicamente en publicaciones realizadas en los últimos 5 años, con el propósito de identificar e investigar las tendencias actuales de investigación y el estado del arte relativo a la problemática de migración de aplicaciones monolíticas a arquitecturas de microservicios [11].

En cuanto al alcance temático, se consideran exclusivamente estudios cuyo foco esté puesto directamente en aspectos concretos de la migración de aplicaciones monolíticas a microservicios, como metodologías, enfoques, desafíos, antipatrones y buenas prácticas [12]. Se excluyen trabajos que exploren temas periféricos o tangenciales no directamente orientados a la problemática de migración.

Asimismo, se prioriza la inclusión de literatura en idioma inglés que provea una perspectiva empírica fundamentada en evidencia, descartando opiniones o especulaciones sin sustento en

estudios sistemáticos [12]. Únicamente se consideran trabajos que se encuentren disponibles con acceso a texto completo.

En síntesis, se establece un alcance acotado a estudios primarios de publicados en los últimos 5 años, en idioma inglés y con disponibilidad de acceso a texto completo, que aborden de forma directa, empírica y sistemática distintos aspectos teóricos y prácticos de la migración de aplicaciones monolíticas a microservicios.

1.4 Fundamentos del trabajo

El presente trabajo se fundamenta en la consolidada metodología de mapeo sistemático de literatura, que brinda un procedimiento riguroso y replicable para identificar, seleccionar, evaluar y sintetizar la evidencia disponible en un área de estudio delimitada [10, 11]. Mediante búsquedas sistemáticas en bases de datos académicas especializadas, y la aplicación de una serie de criterios de inclusión y exclusión preestablecidos, un mapeo sistemático permite obtener una visión integral del estado del arte en el conocimiento sobre un dominio o problema de investigación específico [10, 11].

La realización de mapeos sistemáticos de literatura es ampliamente utilizada y validada en Ingeniería de Software como método para obtener una comprensión de un tópico de interés emergente, identificar temas no explorados y consolidar un marco conceptual sobre un campo de estudio en desarrollo [13, 14, 15]. Diversos estudios destacan las fortalezas de esta metodología para categorizar y estructurar integralmente el conocimiento disponible sobre un fenómeno o problemática de investigación acotada [11, 5].

Por otro lado, debido a que en los últimos años se produce un creciente interés tanto en la academia como en la industria en la migración de aplicaciones monolíticas a arquitecturas de microservicios [12, 16], y en paralelo se observa una producción intensificada de literatura sobre diversos aspectos de este fenómeno [17, 8], un mapeo sistemático de literatura resulta particularmente apropiado para integrar de forma estructurada el conocimiento previo y establecer integralmente el estado actual sobre este dominio específico [10, 13].

Pese a que los microservicios llevan casi una década en la industria, la literatura sobre migraciones de monolitos a esta arquitectura se encuentra dispersa y adolece de un enfoque integrador. Mediante un proceso sistemático de mapeo, este trabajo apunta a consolidar

diferentes perspectivas presentes en publicaciones recientes sobre este fenómeno. La identificación de tendencias y enfoques permite obtener una visión actualizada, sentando bases para futuros avances en la investigación y adopción de este tipo de arquitecturas.

1.5 Estructura del trabajo

El presente trabajo se estructura en tres capítulos principales. El Capítulo 1 presenta la introducción, estableciendo el contexto, objetivos, alcances y fundamentos del mapeo sistemático realizado. El Capítulo 2 expone el marco teórico junto con el proceso y los resultados del mapeo. Finalmente, el Capítulo 3 presenta un análisis de los hallazgos del mapeo, las conclusiones del trabajo y posibles líneas de investigación futura.

Esta estructuración permite abordar la problemática planteada de forma integral, partiendo de una fundamentación teórica y conceptual, pasando luego al desarrollo del mapeo sistemático propiamente dicho, para finalmente analizar críticamente los hallazgos y extraer conclusiones fundamentadas que sinteticen el estado actual y futuro del conocimiento en este campo de estudio.

1.6 Producción científica

El presente trabajo tiene la siguiente publicación derivada:

Muraca, F., & Pollo-Cattaneo, M. F. (2023). Migration from monolithic applications to microservices: A systematic literature mapping on approaches, challenges, and anti-patterns. In 2nd International Workshop on Systems Modeling (WSM) (pp. 185-201).

Capítulo 2: Marco teórico

El presente capítulo expone los conceptos teóricos centrales relacionados con las arquitecturas de microservicios y la problemática de la migración desde aplicaciones monolíticas. En la sección 2.1 se discute la arquitectura de software. Luego, en la sección 2.2 se abordan los microservicios y en 2.3 las aplicaciones monolíticas. La sección 2.4 analiza las motivaciones para adoptar microservicios y la 2.5 los desafíos asociados. Los enfoques para la descomposición de monolitos se tratan en 2.6. Las secciones 2.7 a 2.9 exponen buenas prácticas, comparativa de arquitecturas, despliegue continuo y APIs. Esta fundamentación teórica brinda el marco de referencia requerido para contextualizar adecuadamente el posterior mapeo sistemático de literatura.

Figura 1. Mapa conceptual (producción propia)

La Figura 1, presenta un mapa conceptual que abarca los conceptos esenciales relacionados con la migración de aplicaciones monolíticas a microservicios. Cada elemento en el mapa representa cada componente clave en la comprensión de este proceso de migración.

2.1 Arquitectura de software

La arquitectura de software juega un rol crítico en el éxito de los sistemas al proveer la estructura conceptual y los principios organizativos sobre los cuales se construye el diseño detallado y la implementación. Definir una arquitectura robusta que se alinee con los requerimientos funcionales permite construir sistemas escalables y mantenibles. La arquitectura de software proporciona una visión de alto nivel de un sistema, describiendo sus componentes principales y las interacciones entre ellos. Se enfoca en aspectos no funcionales como rendimiento, escalabilidad y confiabilidad. La arquitectura parte de los requerimientos para definir un diseño que optimice estos atributos de calidad [18].

Existen diversos estilos arquitectónicos, cada uno con ventajas y desventajas. Por ejemplo, la arquitectura monolítica concentra toda la funcionalidad en una única aplicación, mientras que la arquitectura de microservicios la divide en múltiples servicios independientes [3]. La elección de la arquitectura adecuada depende de factores como tamaño, complejidad, necesidades de escalado y capacidades del equipo [18].

En la migración de sistemas, es clave realizar un rediseño arquitectónico para aprovechar las capacidades de la nueva plataforma tecnológica. Esto permite obtener los mayores beneficios de rendimiento, performance y calidad.

2.2 Microservicios

El estilo arquitectónico de microservicios surge en años recientes como un enfoque para desarrollar aplicaciones empresariales complejas mediante la composición de múltiples servicios independientes y altamente desacoplados. Bajo este paradigma, una aplicación se compone de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose típicamente mediante APIs REST [3]. Los microservicios se construyen en torno

a capacidades de negocio específicas y se despliegan de forma aislada a través de procesos completamente automatizados [4].

Los microservicios encapsulan funcionalidades de dominio y exponen esta funcionalidad a otros servicios a través de la red [7]. Por ejemplo, en un sistema de comercio electrónico, pueden existir microservicios para el catálogo de productos, el carrito de compras, la gestión de pagos, envíos, etc. Cada uno desplegado de forma independiente.

Un aspecto clave de los microservicios es la independencia en el despliegue, permitiendo actualizar cada servicio de forma aislada sin impactar al resto [10]. Además, posibilitan el uso de diversas tecnologías y lenguajes según convenga [4].

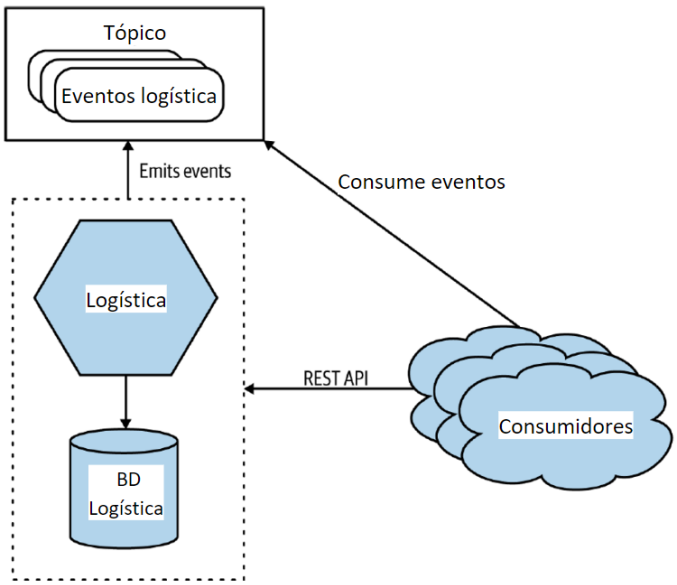

Figura 2. Un microservicio exponiendo su funcionalidad

La Figura 2 ilustra un microservicio exponiendo su funcionalidad mediante una API RESTful y un tópico de mensajería, además de una base de datos propia. Con esta figura se puede establecer que la responsabilidad de este microservicio es acotada y se delimita a una responsabilidad en particular, perteneciendo a un dominio de negocio específico, y comunicándose con sus consumidores a través de una red a través de un protocolo definido.

2.3 Aplicaciones monolíticas

Las aplicaciones monolíticas concentran toda su lógica de negocio en un único proceso o ejecutable [14]. Constan típicamente de una interfaz en el lado del cliente, una capa de aplicación en el servidor y una base de datos, todas fuertemente acopladas [12].

Si bien al principio esto permite agilidad y simplicidad, conforme crecen se vuelven difíciles de mantener y escalar, ya que cualquier cambio requiere reconstruir y reimplementar la aplicación completa [17]. Esto impacta negativamente en costos y plazos de entrega.

Además, las monolíticas limitan la capacidad de escalamiento e innovación al impedir que algunos componentes evolucionen o se desplieguen de forma independiente [11].

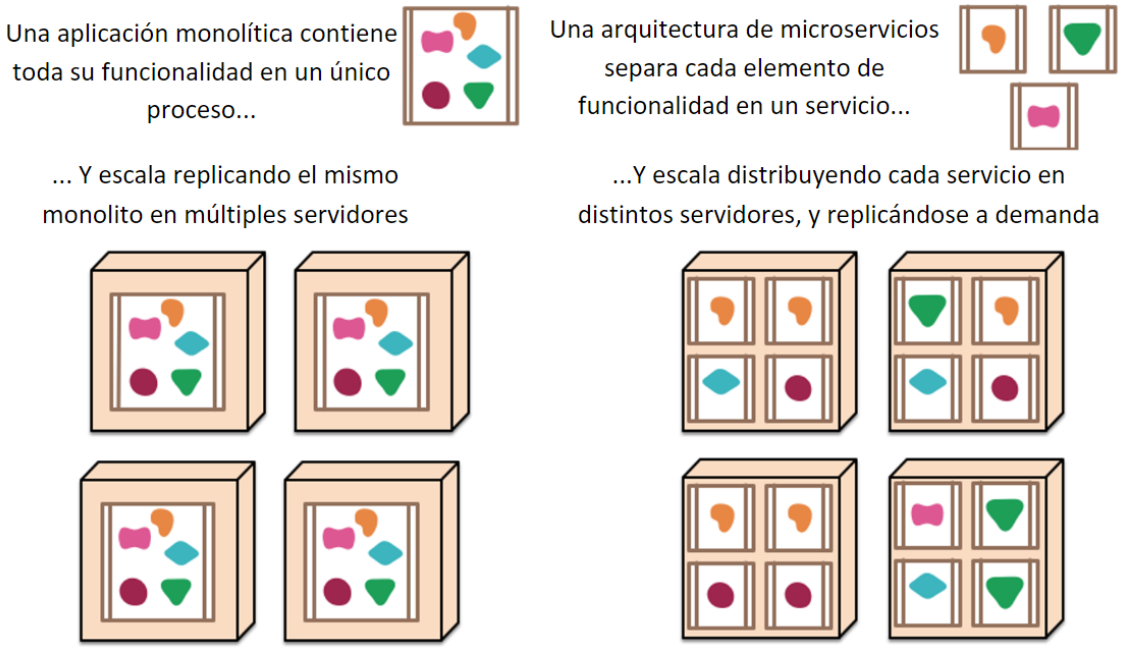

Figura 3. Comparación conceptual entre aplicaciones monolíticas y microservicios

Como se observa en la Figura 3, las aplicaciones monolíticas concentran toda la lógica en un único ejecutable. Esto introduce importantes desafíos de mantenimiento y escalamiento.

2.4 Motivaciones para la migración

La migración desde aplicaciones monolíticas hacia arquitecturas de microservicios se incrementa recientemente impulsada por diversos factores. Por un lado, los microservicios permiten escalar de forma independiente aquellos servicios específicos que así lo requieran debido a demandas fluctuantes, optimizando la asignación de recursos [15, 5]. Además, delimitar claramente las responsabilidades de negocio en servicios independientes mejora la mantenibilidad del software al acotar el impacto de los cambios [15].

Otro aspecto clave es que la arquitectura de microservicios facilita la innovación y adopción modular de nuevas tecnologías, a diferencia de los monolitos donde los cambios tecnológicos afectan el sistema completo [5]. Asimismo, los microservicios posibilitan agilizar los procesos de desarrollo y despliegue de cambios en el software, permitiendo entrega continua [5].

También se considera una arquitectura ideal para habilitar la preparación de aplicaciones legadas para cloud, así como para nuevos paradigmas serverless [16]. Las crecientes limitaciones de las monolíticas y las demandas de agilidad y escala en entornos modernos impulsan la necesidad de migrar hacia microservicios [16].

La migración de sistemas monolíticos legacy hacia arquitecturas de microservicios es una tendencia que viene cobrando fuerza en los últimos años, tanto en la academia como en la industria [1, 2, 8]. Varios trabajos de investigación analizan las motivaciones, beneficios potenciales y desafíos principales detrás de esta transición arquitectónica. Por un lado, los microservicios permiten escalar de forma más flexible y optimizada aquellos servicios específicos con demandas fluctuantes [9], a diferencia de los monolitos que obligan a escalar la aplicación completa. Además, delimitar claramente las responsabilidades de negocio en servicios independientes mejora la mantenibilidad del software y la velocidad de cambios, al acotar su impacto [19]. Otro aspecto relevante es que esta arquitectura facilita la innovación modular y adopción de nuevas tecnologías de forma selectiva, a diferencia de los monolitos donde los cambios tecnológicos afectan a todo el sistema [3]. Asimismo, los microservicios posibilitan agilizar los procesos de desarrollo y despliegue de cambios en el software [20, 21], permitiendo la incorporación continua de mejoras. Las limitaciones crecientes de los monolitos y las demandas de agilidad y escala en entornos modernos impulsan la necesidad de migrar hacia microservicios.

Entre los impulsores clave para adoptar microservicios se encuentran la escalabilidad, la mantenibilidad, la entrega continua y la posibilidad de innovación modular [11, 14]. Frente a las aplicaciones monolíticas tradicionales, los microservicios ofrecen la capacidad de desplegar y escalar servicios de forma independiente.

2.5 Desafíos en la migración

Migrar aplicaciones monolíticas a microservicios puede enfrentar importantes retos tanto técnicos como organizacionales. A nivel técnico, gestionar adecuadamente las dependencias entre los distintos servicios distribuidos es clave para evitar alta complejidad y facilitar la independencia en el desarrollo y despliegue [8]. Otro desafío técnico relevante es mantener la consistencia en los datos, considerando que la arquitectura de microservicios promueve un entorno descentralizado donde cada servicio gestiona su propia base de datos, a diferencia del modelo monolítico centralizado [8].

Además, coordinar y monitorizar una cantidad mayor de servicios en producción resulta más complejo en comparación con monolitos [9]. A nivel organizacional, adoptar la arquitectura de microservicios requiere adaptar la cultura, los procesos y la estructura de los equipos de trabajo, evolucionando hacia esquemas multifuncionales alineados con las capacidades de negocio [9]. Asimismo, determinar una descomposición efectiva del monolito en microservicios constituye un reto clave y complejo durante la migración [9].

Identificar, evaluar y gestionar apropiadamente la mayor complejidad operativa introducida por los microservicios, así como reestructurar los flujos de trabajo para incorporar prácticas de entrega continua, también representan desafíos organizacionales destacados [9].

Si bien se han producido avances en la adopción práctica de arquitecturas de microservicios en la industria, varios trabajos coinciden en que la investigación académica sobre esta temática aún se encuentra en una fase temprana [9, 24]. Existe una brecha de conocimiento sobre cómo llevar a cabo de forma efectiva los procesos de migración desde arquitecturas monolíticas legacy hacia este nuevo paradigma. Se han identificado algunas revisiones sistemáticas previas sobre aspectos específicos de microservicios [22, 23], pero no enfocadas integralmente en la problemática de la migración desde monolitos. Es en este contexto que el presente mapeo sistemático de literatura busca contribuir a explorar los enfoques, desafíos,

antipatrones y buenas prácticas reportados en estudios recientes sobre migración de monolitos a microservicios.

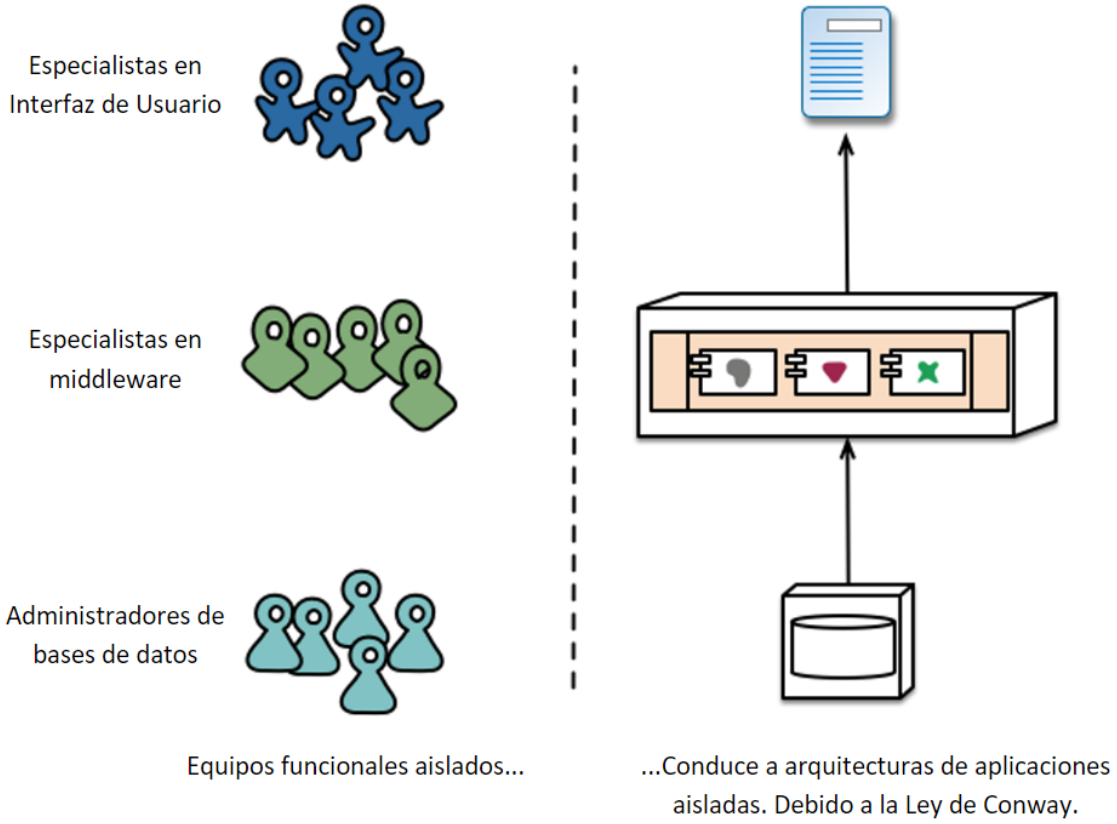

Figura 4. Ley de Conway en acción

Como ilustra la Figura 4, la ley de Conway establece que la arquitectura de software de un sistema es un reflejo de la estructura de comunicación y organización del equipo que lo desarrolla. Equipos bien organizados y con buena comunicación tenderán a crear sistemas con diseños modulares bien definidos.

Por el contrario, equipos con problemas de organización y comunicación crearán sistemas con arquitecturas desordenadas y acopladas. En resumen, según la ley de Conway, la arquitectura de software no es independiente de las interacciones humanas, sino que refleja la forma de trabajo del equipo de desarrollo.

2.6 Enfoques para la descomposición de monolitos

Existen varias técnicas propuestas en la literatura para abordar la descomposición de monolitos en microservicios. Un enfoque es la descomposición orientada a las capacidades de negocio, generando servicios enfocados en funcionalidades requeridas por el negocio [19]. Otro enfoque es la descomposición por dominios, dividendo la aplicación según los dominios funcionales [24].

También se ha sugerido la descomposición guiada por responsabilidades funcionales, delimitando servicios acorde a responsabilidades claras [24]. Asimismo, en algunos casos puede ser efectivo aplicar enfoques híbridos que combinen múltiples criterios de descomposición [24]. Otras técnicas propuestas son la descomposición incremental y evolutiva, permitiendo identificar servicios de forma progresiva, así como la extracción de servicios aplicando el patrón strangler, el cual establece una estrategia de migración de software que implica la sustitución gradual de un sistema antiguo por uno nuevo, de forma incremental, componente por componente, hasta que el sistema original queda obsoleto [24]. En esencia, la selección de la estrategia de descomposición más adecuada está sujeta a un análisis en profundidad de las características de la aplicación monolítica y el contexto organizacional.

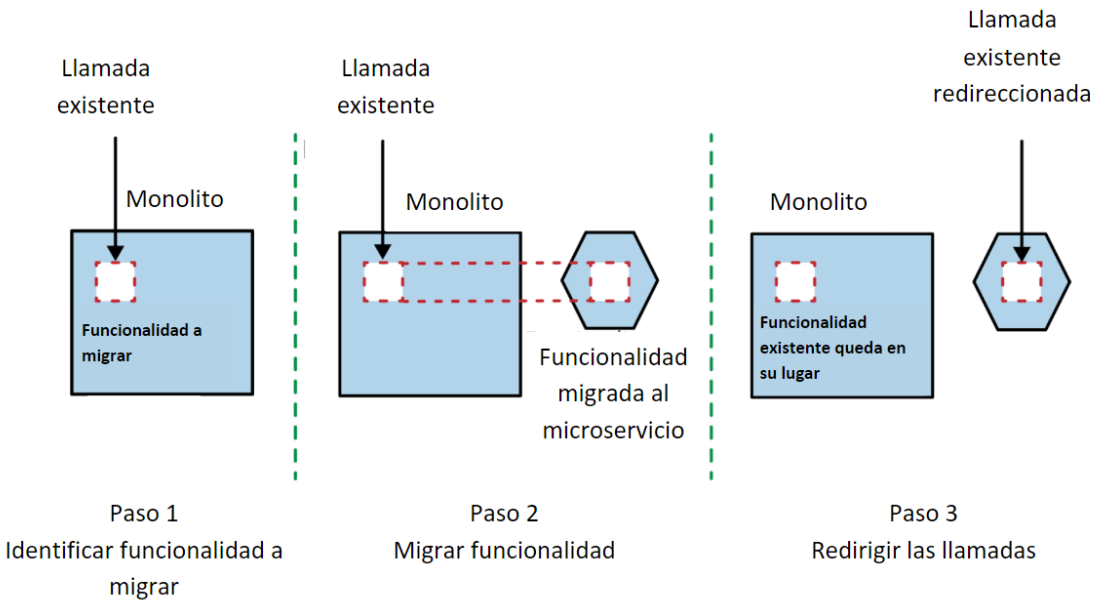

Figura 5: Una descripción general del patrón strangler

Como se visualiza en la Figura 5, el patrón strangler consta de 3 pasos. En primer lugar se identifican las partes del sistema existente que se desean migrar, se debe usar el juicio para decidir qué partes del sistema se deciden abordar primero. Luego, se debe implementar esa funcionalidad en el nuevo microservicio. Con la nueva implementación lista, se debe redirigir las llamadas desde el monolito hacia el nuevo microservicio.

2.7 Buenas prácticas y recomendaciones

Investigaciones previas proponen varias buenas prácticas y recomendaciones para llevar a cabo de forma efectiva la migración desde aplicaciones monolíticas hacia arquitecturas de microservicios. En primer lugar, se sugiere adoptar un enfoque incremental y progresivo, en lugar de intentar descomponer el monolito en su totalidad en una única etapa [25].

Asimismo, establecer una comunicación estrecha entre los equipos multifuncionales involucrados, tanto de desarrollo como de operaciones, resulta vital [25].

Otra buena práctica es implementar integración y entrega continua de software, facilitando la incorporación incremental de microservicios [26]. También se enfatiza la importancia de realizar pruebas exhaustivas de los microservicios antes y durante la migración, para garantizar la calidad [26].

A nivel técnico, se recomienda desarrollar sólidos sistemas de monitoreo y observabilidad de los microservicios, así como adoptar prácticas de DevOps e infraestructura de nube para simplificar las operaciones [26].

Asimismo, se sugiere evaluar cuidadosamente los pros y contras de la migración en relación con el contexto específico de cada organización [25].

Otras recomendaciones incluyen priorizar la migración de aquellos componentes más críticos o problemáticos, controlar el crecimiento de los microservicios para evitar alta complejidad, y minimizar las dependencias entre servicios en la medida de lo posible [25, 26].

2.8 Comparación de arquitecturas

Al contrastar las arquitecturas de microservicios y monolíticas surgen diferencias clave en varios aspectos. Los microservicios, a diferencia de los monolitos, permiten escalar y desplegar de forma independiente cada uno de los servicios según se requiera [11, 13, 17].

Esta capacidad de escalado y despliegue granular facilita optimizar el uso de recursos computacionales. Otro aspecto diferenciador es que los microservicios favorecen la innovación modular y la experimentación ágil con nuevas tecnologías, mientras que en las monolíticas los cambios tecnológicos afectan la totalidad del sistema [13, 17]. En cuanto a la organización de los equipos de trabajo, los microservicios posibilitan estructuras multifuncionales enfocadas en capacidades de negocio, mientras que tradicionalmente los monolitos promueven equipos separados por capas tecnológicas [17].

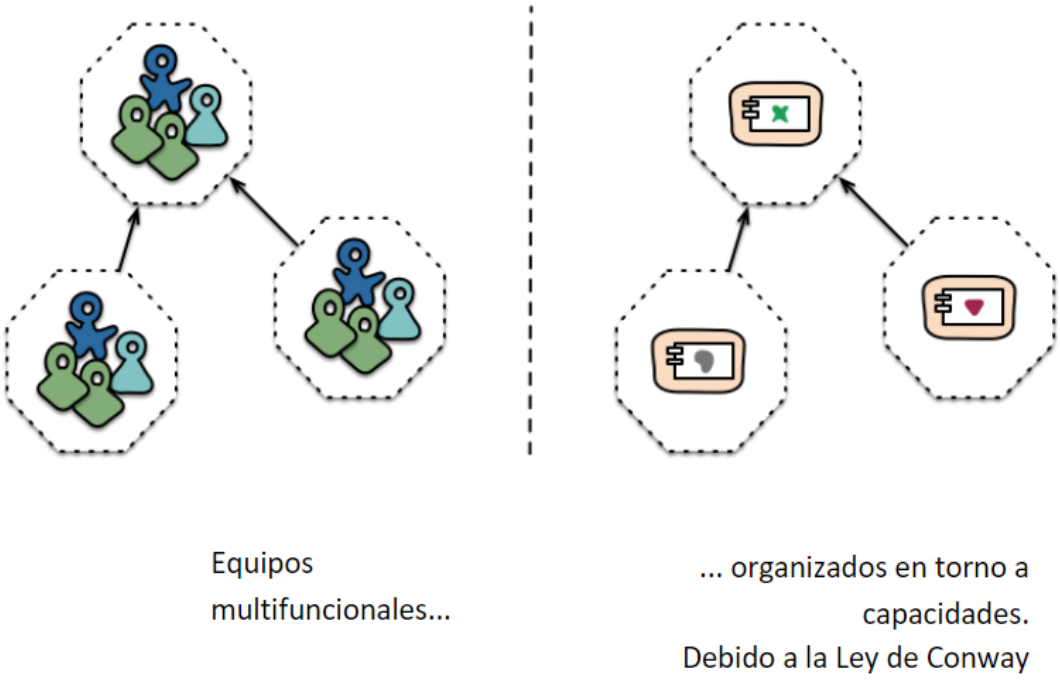

Figura 6. Comparativa en estilos arquitectónicos.

Como resume la Figura 6, el estilo arquitectónico impacta de manera directa el modelo organizacional, es decir, los equipos son multifuncionales e incluyen todas las habilidades necesarias para el desarrollo: experiencia de usuario, bases de datos y gestión de proyectos.

2.9 Despliegue continuo

El despliegue continuo es una práctica fundamental en la Ingeniería de Software moderna que busca automatizar y optimizar el proceso de entrega de software. En contraste con enfoques tradicionales que implican la implementación esporádica de grandes lotes de código, el despliegue continuo se enfoca en la entrega frecuente de pequeñas actualizaciones de software.

Para lograrlo, se automatizan todas las etapas del ciclo de vida del software, desde la construcción y las pruebas hasta la implementación en un entorno de producción. Esto permite una mayor eficiencia y calidad en el proceso de entrega.

Cada cambio en el código fuente se somete a un proceso de integración y pruebas automáticas. Si todas las pruebas son exitosas, el cambio se implementa en un entorno de producción de manera automática o semiautomática.

Esta práctica tiene varios beneficios, incluyendo la reducción de los riesgos asociados con las implementaciones manuales, la mejora de la calidad del software y la capacidad de respuesta a las demandas cambiantes de los usuarios finales. Además, permite a los equipos de desarrollo detectar y solucionar problemas rápidamente, lo que resulta en una mayor confiabilidad del software.

El enfoque del despliegue contínuo se basa en la automatización, la estandarización y la colaboración entre los equipos de desarrollo y operaciones. Al automatizar las pruebas y la implementación, se minimizan los errores humanos y se acorta el tiempo entre el desarrollo y la disponibilidad de nuevas características o correcciones.

Esta práctica se ha convertido en un pilar fundamental en la entrega de software en entornos dinámicos y es esencial para lograr la agilidad y la capacidad de respuesta requerida en un entorno empresarial competitivo [27].

2.10 API

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de rutinas, protocolos y herramientas que permite la comunicación entre diferentes sistemas de software. Las APIs definen la forma en que un componente de software se comunica e interactúa con otros, especificando los métodos requeridos para acceder a sus funciones y datos.

Las APIs permiten la reutilización e integración de funcionalidades entre aplicaciones, ocultando la complejidad de implementación subyacente. Promueven el desacoplamiento y la modularidad al abstraer los detalles internos detrás de una interfaz bien definida.

Las APIs son esenciales en arquitecturas de microservicios, donde cada servicio expone su funcionalidad principal a través de APIs (generalmente REST o SOAP) para ser consumidas por otros servicios. Esto posibilita su composición y orquestación [3].

Capítulo 3

Este capítulo se enfoca en el desarrollo del mapeo sistemático de literatura sobre la migración de aplicaciones monolíticas a microservicios. Consta de seis secciones principales.

En la sección 3.1 se detalla la formulación del problema y preguntas, aquí se presenta el problema de investigación y las preguntas específicas que guiarán el mapeo sistemático. Se justifica la relevancia de cada pregunta en relación al tema de estudio.

Luego, en la sección 3.2 se desarrolla la fase de búsqueda, donde se describe la estrategia de búsqueda bibliográfica diseñada a través de la construcción de una cadena de búsqueda booleana óptima mediante un proceso iterativo

En la sección 3.3 se realiza la selección, donde se detallan los criterios rigurosos de inclusión y exclusión de estudios aplicados para seleccionar los trabajos más relevantes al problema de investigación. También explica los filtros sucesivos implementados para refinar los resultados.

Posteriormente en la sección 3.4, se analiza la extracción, es decir, el proceso de revisión en profundidad realizado sobre los estudios seleccionados para extraer los datos cualitativos y cuantitativos pertinentes a las preguntas de investigación.

Llegando a la sección 3.5 de síntesis se enumeran los hallazgos del mapeo sistemático en relación a cada pregunta, mediante una categorización y caracterización de los resultados evidenciados en los estudios.

Finalmente, en la sección 3.6 o de reporte, se presentan los resultados del mapeo de forma estructurada a través de tablas, gráficos y diagramas, identificando tendencias y aspectos destacados.

Las fases secuenciales que componen la realización del mapeo sistemático anteriormente explicadas se esquematizan en la Figura 7, la cual presenta un diagrama de flujo que muestra la progresión y relaciones entre cada etapa.

Figura 7. Fases de un mapeo sistemático (producción propia)

Esta figura contiene una representación de lo que se desarrolla en el presente capítulo, conteniendo las 6 fases propuestas para la realización del mapeo sistemático.

3.1 Formulación del problema y preguntas

En la Tabla 1 se presentan las preguntas de investigación definidas para el mapeo sistemático, junto con su correspondiente justificación. La incorporación de estas justificaciones permite sustentar adecuadamente la pertinencia de las preguntas planteadas.

Tabla 1. Preguntas de investigación

RQ	Pregunta de Investigación	Justificación
RQ1	¿Cuáles son los enfoques utilizados en la migración de aplicaciones monolíticas a microservicios?	Comprender los enfoques utilizados en la migración de aplicaciones monolíticas a microservicios proporcionará una visión general de las estrategias y técnicas empleadas en este proceso.
RQ2	¿Cuáles son los desafíos comunes en la migración de aplicaciones monolíticas a microservicios?	Identificar los desafíos comunes en la migración de aplicaciones monolíticas a microservicios ayudará a comprender las dificultades que pueden surgir durante este proceso y desarrollar estrategias para superarlas.

RQ	Pregunta de Investigación	Justificación
RQ3	¿Qué antipatrones se han identificado en la migración de aplicaciones monolíticas a microservicios?	Identificar los antipatrones en la migración de aplicaciones monolíticas a microservicios ayudará a evitar errores comunes y adoptar prácticas recomendadas para lograr una migración exitosa.
RQ4	¿Cuáles son las mejores prácticas y recomendaciones para la migración de aplicaciones monolíticas a microservicios?	Comprender las mejores prácticas y recomendaciones en la migración de aplicaciones monolíticas a microservicios proporcionará valiosas pautas para realizar este proceso de manera efectiva y eficiente.
RQ5	¿Cuáles son las ventajas y desventajas asociadas con la migración de aplicaciones monolíticas a microservicios?	Evaluar las ventajas y desventajas de la migración de aplicaciones monolíticas a microservicios proporcionará información sobre los beneficios y posibles limitaciones de esta transición.

A partir de estas preguntas, se lleva a cabo el mapeo sistemático de la literatura, dando lugar a las siguientes fases planteadas.

3.2 Búsqueda

Para poder recuperar de forma efectiva estudios relevantes al alcance y preguntas de investigación, se diseña cuidadosamente una cadena de búsqueda booleana que aborde específicamente los focos de interés del mapeo sistemático [10, 11]. Esta cadena hace uso de los operadores booleanos AND y OR para asegurar la inclusión de estudios que aborden tanto el tema de microservicios como de migración. Inicialmente la cadena es amplia, para luego ir refinándola de forma iterativa hasta obtener una expresión óptima que retorne una cantidad manejable de artículos pertinentes.

La cadena de búsqueda inicial propuesta es ("Microservices" OR "Microservice") AND ("Migration") AND ("Challenges" OR "Approaches" OR "Antipatterns"). Sin embargo, las primeras pruebas en las bases de datos académicas muestran una cantidad muy reducida de resultados con esta expresión preliminar.

Tras analizar los escasos estudios recuperados, se identifica que la cadena requiere mayor flexibilidad en las relaciones entre conceptos clave. Por ende, en una iteración posterior se

reemplaza el operador AND entre "Microservices" y "Migration" por OR, buscando ampliar los resultados potenciales. No obstante, esta versión recupera numerosos estudios no pertinentes al área de investigación.

Luego de sucesivos análisis y ajustes graduales, enfocados en encontrar un equilibrio entre amplitud y precisión, se arriba a una expresión de búsqueda más acorde. La cadena definitiva hace un uso eficiente de variaciones de términos entre conceptos mediante los operadores. La versión final resultante es: ("Microservices" OR "Microservice") AND ("Migration" OR "Challenges" OR "Approaches" OR "Antipatterns"). La expresión de búsqueda final conformada recupera efectivamente estudios primarios relevantes al dominio de investigación del presente mapeo sistemático.

Los sucesivos refinamientos de la cadena de búsqueda arrojaron diferentes cantidades de estudios recuperados en cada iteración, como se muestra en la Tabla 2. Esto permite evidenciar cuantitativamente la progresión en la precisión de los resultados.

Tabla 2. Estudios recuperados en las iteraciones de la cadena de búsqueda

Versión	Cadena de búsqueda	Estudios recuperados
1	("Microservices" OR "Microservice") AND ("Migration") AND ("Challenges" OR "Approaches" OR "Antipatterns")	8
2	("Microservices" OR "Microservice") OR ("Migration") AND ("Challenges" OR "Approaches" OR "Antipatterns")	1.257
3	("Microservices" OR "Microservice") AND ("Migration" OR "Challenges" OR "Approaches" OR "Antipatterns")	17

La Tabla 2 expone la evolución de la cadena de búsqueda booleana en sus versiones inicial, con reemplazo de operador y final, detallando la expresión utilizada y la cantidad de estudios recuperados en cada iteración. Esta evidencia cuantitativamente el proceso de refinamiento para mejorar la precisión.

3.3 Selección

Para seleccionar adecuadamente los estudios más pertinentes y relevantes dentro del alcance del presente mapeo sistemático, se establecen una serie de criterios rigurosos tanto de inclusión como de exclusión [12]. Estos criterios están orientados a acotar la búsqueda y garantizar la identificación y selección de aquellos estudios primarios que efectivamente aporten información de valor para responder las preguntas de investigación definidas.

Los criterios consideran aspectos clave como el tipo de estudio, la metodología utilizada, la población/tema de investigación, el idioma, la fecha de publicación y la disponibilidad del texto completo. A continuación, la Tabla 3 describe en detalle los criterios de inclusión y exclusión establecidos para la selección de estudios para este mapeo sistemático en particular.

Tabla 3. Criterios de inclusión y exclusión

Criterios	Criterios de inclusión	Criterios de exclusión
Tipo de estudio	Cualquier tipo de estudio	Fuentes no académicas, artículos de opinión, editoriales
Metodología	Cualquier metodología	Metodologías no relevantes para el tema de investigación
Población de estudio	Estudios sobre migración de aplicaciones monolíticas a microservicios	Estudios sobre otros temas o no relacionados con las preguntas de investigación
Idioma de publicación	Inglés	Otros idiomas
Fecha de publicación	Publicaciones recientes (últimos 5 años)	Publicaciones desactualizadas
Disponibilidad del texto completo	Texto completo disponible	Solo resumen o texto no disponible

Estos criterios permiten acotar la búsqueda y asegurar la inclusión solo de estudios relevantes al alcance del mapeo.

Para poder refinar de forma efectiva los resultados iniciales de la búsqueda bibliográfica, se define y aplica una serie de filtros sucesivos que permiten focalizar progresivamente los estudios más relevantes de acuerdo con el alcance y preguntas de investigación establecidas [12]. Estos filtros sucesivos son esenciales para acotar la búsqueda preliminar a los estudios de mayor interés para el mapeo sistemático.

Se aplican cinco filtros en total, denominados de 1F a 5F, los cuales se detallan en la Tabla 4 que se presenta a continuación. Estos filtros implican una revisión sistemática y un análisis progresivo de los estudios desde una evaluación inicial por títulos y resúmenes, hasta una lectura completa de los textos para determinar su relevancia e importancia para esta investigación en particular. De esta manera, se garantiza la identificación de los estudios primarios más apropiados al dominio.

Tabla 4. Filtros aplicados en el mapeo sistemático

Filtro	Descripción
1F	Revisión de artículos aplicando la cadena de búsqueda en los gestores de referencias. Los parámetros de búsqueda fueron el rango de fecha de publicación, el tipo de publicación y el idioma.
2F	Lectura del título, palabras clave y resumen del artículo para evaluar su relevancia.
3F	Eliminación de estudios duplicados para evitar repetición de datos.
4F	Lectura de la introducción, resultados y conclusiones del artículo para obtener una comprensión más profunda del estudio.
5F	Lectura completa del artículo para asegurar una comprensión integral y su relevancia para la investigación.

Los filtros formulados son esenciales para acotar progresivamente los estudios preliminares y seleccionar los más relevantes de acuerdo al alcance y las preguntas de investigación definidas para el mapeo sistemático.

3.4 Extracción

La búsqueda se realiza en las bases de datos Science Direct, Springer, IEEE Xplore, O'Reilly, Wiley Online Library y Compendex. Además de formular la cadena de búsqueda booleana, se define un primer filtro orientado a acotar los resultados a estudios recientes, de alto impacto y en idioma inglés. Específicamente, este filtro inicial contempla el rango de fecha de publicación entre 2018-2023, seleccionando únicamente trabajos de los últimos 5 años. Esto permite enfocarse en literatura actualizada que refleja las tendencias de investigación más novedosas. Asimismo, el filtro considera como tipo de publicación artículos en revistas y actas de conferencias relevantes al campo de estudio. En el caso de Springer e IEEE, se trata de revistas con categoría Q1 según el ranking Scimago Journal. En cuanto a las otras bases de datos mencionadas, son fuentes ampliamente reconocidas en la comunidad académica y el campo de la informática y la ingeniería de software, resultando opciones adecuadas para realizar un mapeo sistemático. De esta forma, se priorizan contribuciones validadas por la comunidad científica mediante procesos de revisión por pares.

Por último, se filtra por idioma inglés, lengua dominante en las principales fuentes de publicación académica de ingeniería de software a nivel global. Otras opciones de idioma, fecha o tipo de fuente se descartan en esta fase inicial ya que incorporarían material no actualizado, sin validación científica o una perspectiva menos global. En esta etapa, se genera la Figura 8, que ilustra el proceso de filtrado de estudios.

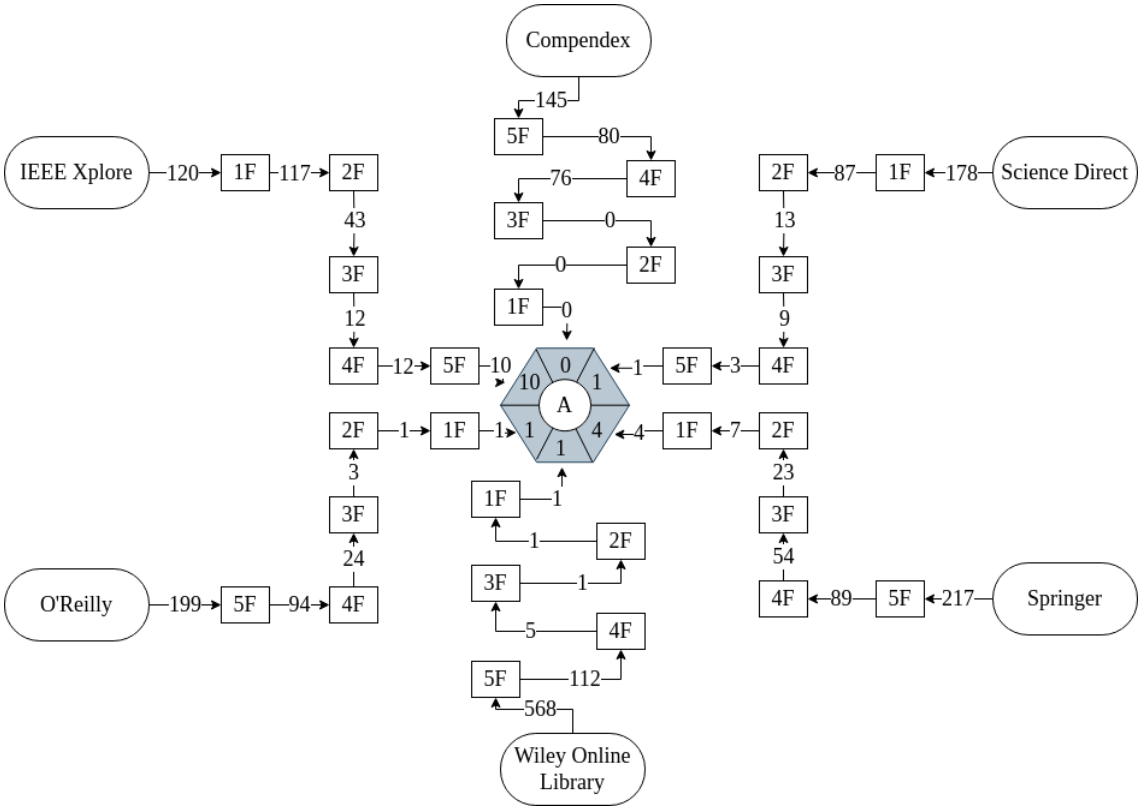

Figura 8. Resultados del mapeo sistemático

La figura que ilustra el proceso de filtrado y la cantidad de estudios resultantes en cada etapa sirve de base para llevar a cabo un análisis y evaluación más profunda de los estudios que fueron filtrados. El objetivo es identificar aquellos artículos de investigación más relevantes y que mejor se ajusten a los criterios de selección definidos previamente para el mapeo sistemático.

El proceso de análisis implica revisar detalladamente cada uno de los estudios resultantes luego de aplicar los filtros sucesivos. Se evalúan cuidadosamente tanto la pertinencia de cada documento con respecto al tema específico de la migración de monolitos a microservicios, como también la calidad metodológica y el rigor científico de los estudios. Luego, se determina finalmente el conjunto de estudios primarios que se incluirán para la extracción, categorización y análisis de información relevante a los objetivos del mapeo. Por consiguiente, se permiten identificar los artículos de investigación más apropiados considerando su ajuste a los criterios de selección definidos previamente.

3.5 Síntesis

3.5.1 Enfoques más comunes utilizados en la migración de aplicaciones monolíticas a microservicios (RQ1)

La revisión de la literatura revela varios enfoques y técnicas para descomponer aplicaciones monolíticas en microservicios. Uno de los enfoques identificados es Service Cutter [2], un enfoque sistemático para la descomposición de servicios basado en criterios de cohesión y acoplamiento. Este enfoque utiliza un conjunto de reglas y heurísticas para identificar los límites de los servicios y facilitar la transición a una arquitectura de microservicios.

Otro enfoque es la descomposición basada en dominio, que implica dividir la aplicación en servicios que representan dominios de negocio específicos [1]. Se basa en el concepto de Diseño Guiado por Dominio (DDD) y permite una mejor alineación entre la arquitectura de software y las necesidades del negocio.

La descomposición basada en responsabilidad es otra técnica que se encuentra en la literatura [7], donde los servicios se dividen en función de responsabilidades funcionales y no funcionales. Permite una mayor modularidad y separación de intereses, facilitando la escalabilidad y mantenibilidad de la aplicación.

Otra propuesta es un enfoque basado en agrupamiento colaborativo para identificar automáticamente microservicios extrayéndolos de procesos de negocio [24]. El enfoque utiliza modelos separados para capturar dependencias estructurales, de datos y semánticas de los procesos de negocio. Al emplear el agrupamiento colaborativo, el enfoque evita perder detalles implícitos y agrega los datos extraídos, lo que lleva a una mejor precisión en la identificación de microservicios. En [28] recomienda CI/CD para automatizar toda la gestión de lanzamientos y despliegues, una base de datos por servicio, descubrimiento de servicio, monitoreo y el uso de máquinas virtuales o contenedores.

Además, se identifican enfoques híbridos que combinan diferentes técnicas de descomposición, como la descomposición basada en funcionalidad y basada en dominio [1]. Estos enfoques permiten una mayor flexibilidad y adaptabilidad en la migración de aplicaciones monolíticas a microservicios al considerar múltiples criterios y perspectivas.

En [29] la mayoría de los sistemas son reescritos usando tecnologías actuales, mientras que las bases de código existentes fueron refactorizadas en solo dos casos. Esto confirma una tendencia descubierta en [1].

3.5.2 Desafíos más comunes encontrados durante la migración (RQ2)

Cada aplicación monolítica legacy es única y la migración a microservicios crea diferentes desafíos [8]. Uno de los principales desafíos es gestionar las dependencias entre servicios, ya que la descomposición de una aplicación monolítica puede resultar en interdependencias complejas y difíciles de gestionar [1]. Esto puede impactar negativamente en la escalabilidad y mantenibilidad de la aplicación y aumentar la complejidad de la arquitectura.

Otro desafío común en el proceso de migración es la adaptación a nuevos patrones de diseño y arquitectura, como la comunicación entre servicios y la gestión de datos distribuidos [2]. Estos patrones pueden requerir cambios importantes en la estructura y el comportamiento de la aplicación, así como habilidades y conocimientos adicionales del equipo de desarrollo.

Además, el artículo [19] resalta tres desafíos adicionales en la migración a microservicios: tenencia múltiple, mantenimiento de estado y consistencia de datos. El concepto de tenencia múltiple permite que los microservicios sean utilizados por diferentes organizaciones con requerimientos distintos, considerando las necesidades de privacidad de datos. El mantenimiento de estado influye en requerimientos no funcionales relacionados con la escalabilidad, confiabilidad y disponibilidad del sistema modernizado. Los desafíos de consistencia de datos surgen al migrar código heredado que opera en un repositorio de datos centralizado a microservicios que operan en repositorios de datos descentralizados [19].

Las dificultades para realizar la descomposición de microservicios y deshacerse del acoplamiento estrecho, subestimar las consecuencias del aislamiento del servicio y la base de datos por servicio, y las dificultades para operar la arquitectura de microservicios son desafíos descritos en [28].

La reorganización de componentes y la refactorización de código también pueden representar desafíos durante la migración, ya que pueden introducir riesgos de regresión y afectar la calidad del software [7]. Además, la migración puede requerir la adopción de nuevas herramientas y tecnologías, como contenedores y plataformas de orquestación, que pueden

provocar problemas de compatibilidad y una curva de aprendizaje [1]. En un estudio realizado en [29], se informan dificultades técnicas en organizaciones de todos los tamaños, los sistemas más grandes tenían problemas predominantemente para gestionar la gran cantidad de equipos y facilitar la colaboración entre ellos. Las organizaciones maduras tienen que lidiar con iniciar un cambio de mentalidad al mismo tiempo que establecían procesos ágiles.

3.5.3 Antipatrones identificados en la migración de aplicaciones monolíticas a microservicios (RQ3)

Los antipatrones son prácticas comunes que pueden ser contraproducentes en la migración de aplicaciones monolíticas a microservicios. Uno de los antipatrones identificados es la descomposición excesiva de servicios, que puede resultar en muchos microservicios que son difíciles de gestionar y mantener [1]. Para abordar este problema, se propone un enfoque más equilibrado y pragmático para la descomposición de servicios, considerando factores como cohesión, acoplamiento y complejidad de la aplicación [2].

Otro antipatrón es la falta de consideración por las implicaciones de la migración en aspectos no funcionales como seguridad, privacidad y rendimiento [7]. Para abordar este problema, se recomienda realizar un análisis exhaustivo de los requisitos no funcionales y adoptar prácticas de diseño y arquitectura que garanticen el cumplimiento de estos requisitos durante el proceso de migración [1]. Centrarse demasiado en la tecnología y menos en el diseño y los patrones de migración [28] también puede ser problemático.

Con respecto a las bases de datos, [20] señala que varias empresas adoptan una arquitectura de microservicios conectada a una base de datos heredada o a clústeres de bases de datos existentes, incluso si esto reduce parcialmente los beneficios de los microservicios, ya que no siempre pueden dividir los datos existentes. En este caso, los consultores recomiendan dividir los datos en las bases de datos existentes de modo que cada microservicio acceda a su propia base de datos privada.

Otra cosa que puede suceder es hacer crecer componentes grandes, como se describe en [21], ya que muchas organizaciones experimentan, funcionalidad tras funcionalidad, en algún momento los componentes crecen demasiado. Sufre de servicios monolíticos que contienen

demasiadas funcionalidades. Esto resulta en una complejidad innecesaria, confusión sobre dónde ubicar nuevas funcionalidades y en consecuencia un desarrollo dificultado.

3.5.4 Mejores prácticas y recomendaciones (RQ4)

Se pueden formular varias mejores prácticas y recomendaciones para abordar los desafíos y aprovechar al máximo los beneficios asociados con la migración de aplicaciones monolíticas a microservicios. Una buena práctica es adoptar un enfoque incremental y evolutivo para la migración, comenzando con la descomposición de los componentes más críticos o problemáticos de la aplicación y avanzando gradualmente hacia una arquitectura completa de microservicios [1]. Se puede utilizar el patrón Strangler [29] para reemplazar gradualmente el sistema existente con Microservicios.

Otra recomendación es establecer una comunicación efectiva y una estrecha colaboración entre los equipos de desarrollo, operaciones y negocio para garantizar una alineación adecuada entre la arquitectura de software y las necesidades del negocio [2]. Esto puede facilitar la adaptación a los requisitos cambiantes y las habilidades del equipo, así como mejorar la calidad y eficiencia del proceso de migración.

En [13] también se menciona la importancia de mantener altos niveles de comunicación entre los equipos multidisciplinarios involucrados en el desarrollo e implementación del nuevo diseño. Además, se recomiendan pruebas exhaustivas antes, durante y después del proceso de migración para garantizar transiciones sin problemas y mantener la calidad del software.

En [25] se recomiendan los siguientes patrones de migración: habilitar la integración continua, recuperar la arquitectura actual, descomponer el monolito, cambiar la dependencia de código a llamadas a servicios, introducir registro de servicios, presentar balanceador de carga interno, presentar circuit breaker, contenerizar los servicios, implementar en un clúster y orquestar contenedores, y así sucesivamente.

Por último, se recomienda adoptar prácticas de desarrollo ágil y entrega continua en el proceso de migración para facilitar la adaptación a los requisitos cambiantes y permitir la incorporación rápida de mejoras y correcciones en la aplicación [7]. Esto puede mejorar la capacidad de respuesta, la flexibilidad y reducir los riesgos asociados con el proceso de migración.

3.5.5 Ventajas y desventajas de migrar de aplicaciones monolíticas a microservicios en términos de escalabilidad, mantenibilidad y rendimiento (RQ5)

Una ventaja importante es que los microservicios pueden mejorar la escalabilidad de la aplicación al permitir que los servicios individuales se escalen de forma independiente [1, 20, 21]. Al igual que estudios relacionados [30], [31], la mantenibilidad y escalabilidad se identificaron como los principales impulsores para una migración. Esto puede mejorar el rendimiento y la capacidad de respuesta de la aplicación, especialmente durante situaciones de alta demanda. Poder tener una disponibilidad y escalabilidad diferenciada para diferentes partes del sistema, la capacidad de utilizar diferentes tecnologías y evitar la dependencia tecnológica, la reducción del tiempo de comercialización y una mayor comprensibilidad de la base de código [25].

Otra ventaja es que los microservicios pueden mejorar la mantenibilidad de la aplicación al permitir que los servicios individuales se desarrollen, prueben y despliegen de forma independiente [2]. Esto puede reducir el tiempo y esfuerzo necesarios para realizar cambios en la aplicación y mejorar la calidad general del software. En términos de deuda técnica [26] muestra que después de un período relativamente corto, la deuda técnica tendía a crecer más lentamente que en el sistema monolítico.

Con respecto a la tolerancia a fallas [20] describe que la falla de un microservicio generalmente no afecta a todo el sistema. Por el contrario, en una aplicación monolítica, la falla de un componente podría dañar todo el sistema.

Sin embargo, también existen desventajas asociadas con la migración de aplicaciones monolíticas a microservicios. Una desventaja importante es que el proceso de migración puede ser complejo y desafiante, ya que requiere una cuidadosa planificación y ejecución [1]. Dificultades como la descomposición de un sistema en pequeños servicios y la monitorización y gestión de estos servicios se encuentran entre los otros factores importantes que hacen que la migración a microservicios sea una tarea no trivial [25]. Además, gestionar las dependencias entre servicios puede ser difícil y aumentar la complejidad de la arquitectura [2].

Con respecto a la estimación de esfuerzo: se considera que estimar el tiempo de desarrollo de un sistema basado en microservicios es menos preciso que estimar un sistema monolítico [20].

Otra desventaja es que los microservicios pueden aumentar el costo y la complejidad de las operaciones, ya que cada servicio debe monitorearse, administrarse y escalarse de forma individual [7]. Esto puede requerir habilidades y conocimientos adicionales del equipo de operaciones y aumentar el costo total de propiedad de la aplicación. El estudio realizado en [9] concluye que, aunque la transición presenta sus propios desafíos, estos desafíos pueden ser más fáciles de resolver que los desafíos que presenta la arquitectura monolítica.

3.6 Reporte

El proceso de selección de estudios implica evaluar los estudios identificados en la fase de búsqueda para determinar su relevancia y calidad en relación con las preguntas de investigación [10]. Este proceso se lleva a cabo en varias etapas, comenzando con la eliminación de duplicados y la revisión de los títulos y resúmenes de los estudios para identificar aquellos que abordan el tema de investigación [11].

Luego, se realiza una revisión de texto completo sobre los estudios seleccionados de la etapa anterior, con el objetivo de evaluar su calidad y relevancia en detalle [10]. Esta revisión puede implicar la evaluación de aspectos como la metodología empleada, la validez y confiabilidad de los resultados, y la contribución al conocimiento en el campo de estudio [11].

Finalmente, se seleccionan e incluyen en la síntesis de datos y análisis de resultados los estudios que cumplen con los criterios establecidos de calidad y relevancia [10].

En esta fase, se recolecta y analiza información relevante de los estudios seleccionados para abordar las preguntas de investigación y obtener una visión general del estado actual del tema en estudio, en este caso, la migración de aplicaciones monolíticas a microservicios [10]. Este proceso puede involucrar la identificación de patrones y tendencias en los datos, la comparación de resultados y la identificación de áreas de consenso y discrepancia entre los estudios [11].

A continuación, se presentan los hallazgos obtenidos a partir del mapeo sistemático de literatura sobre la migración de aplicaciones monolíticas a microservicios. Estos hallazgos proporcionan información valiosa sobre los enfoques, desafíos, antipatrones y buenas prácticas en la migración de aplicaciones monolíticas a microservicios y contribuyen al conocimiento en este campo.

La Figura 9 expone la distribución de los estudios seleccionados para el mapeo sistemático según su año de publicación. Esto permite analizar la evolución cronológica de las publicaciones en este campo de estudio.

Año de publicación	Cantidad de estudios
2018	3
2019	6
2020	4
2021	2
2022	2

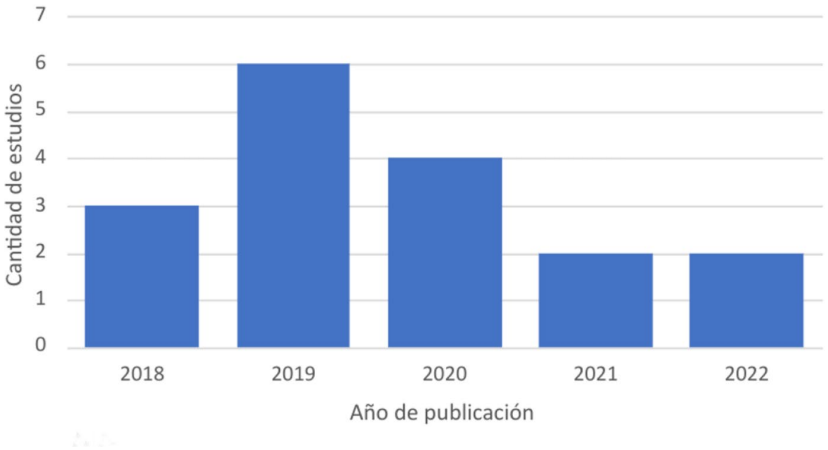

Figura 9. Distribución de estudios por año de publicación

La cantidad de publicaciones sobre migración de monolitos a microservicios crece desde 2018 hasta alcanzar un pico en 2019, para luego decrecer en los años siguientes. Si bien inicialmente se evidencia un interés incremental en esta línea de investigación, en los últimos años se detecta una menor producción de estudios primarios al respecto. La Figura 9 ilustra gráficamente esta tendencia fluctuante evidenciada a través del mapeo sistemático realizado.

Enfoque	Cantidad
Por funciones	5
Por dominios	4
Combinado	2
Alternativos	3

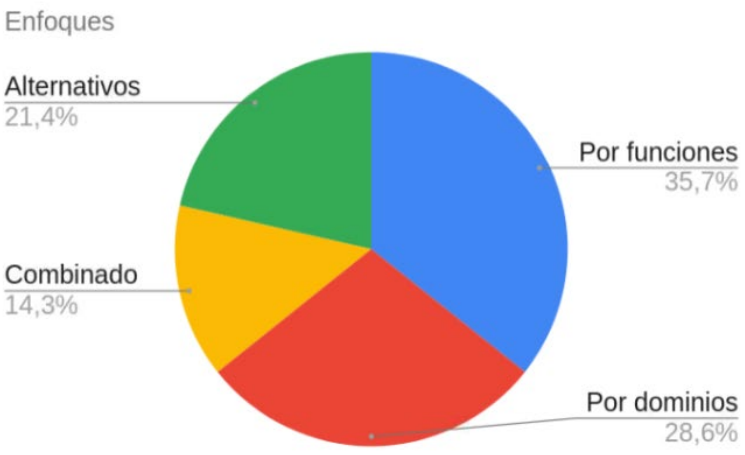

Figura 10. Cantidad de estudios por enfoque.

Aquí se detalla la cantidad de estudios primarios que adoptan cada uno de los principales enfoques de migración identificados a través del mapeo sistemático.

Dentro de los estudios primarios analizados predominan aquellos que adoptan un enfoque de migración orientado a las funciones del sistema, representando casi un tercio del total. Le siguen en frecuencia los estudios basados en una descomposición por dominios de negocio. Los enfoques combinados y alternativos tienen una menor presencia.

Desafío	Frecuencia
Dependencias entre servicios	5
Consistencia de datos	4
Complejidad operativa	3
Coordinación de equipos	1

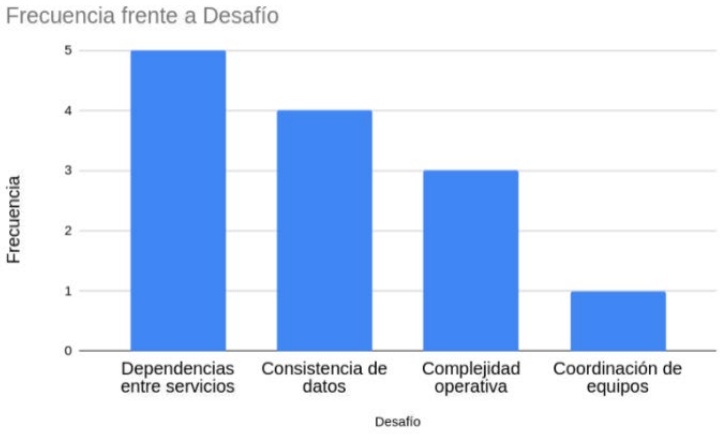

Figura 11. Desafíos más comunes reportados en la migración

En cuanto a los desafíos más reportados, en la Figura 11 se exponen los principales desafíos reportados en los estudios primarios y su frecuencia relativa de aparición.

Dentro de los estudios analizados, la gestión de dependencias entre los servicios distribuidos resultantes de la descomposición es el desafío más frecuentemente reportado. Le siguen aspectos como mantener la consistencia de datos, controlar la creciente complejidad operativa e impulsar una adecuada coordinación entre los equipos.

Sobre la frecuencia de buenas prácticas y recomendaciones, la Figura 12 muestra las buenas prácticas y recomendaciones más frecuentemente mencionadas en los estudios primarios del mapeo.

Práctica/Recomendación	Frecuencia
Enfoque incremental	7
Pruebas exhaustivas	5
Monitoreo extensivo	3
Equipos multifuncionales	2

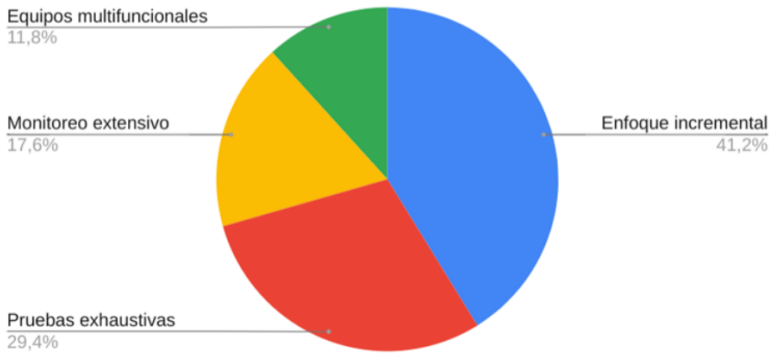

Figura 12. Buenas prácticas más frecuentes

Dentro de las buenas prácticas y recomendaciones detectadas, la adopción de un enfoque incremental y evolutivo para la migración es la más reiterada en los estudios analizados. También tienen una presencia destacada la realización de pruebas exhaustivas y la implementación de monitorización extensiva.

Con respecto a la cantidad de estudios teóricos vs empíricos, a continuación se compara la cantidad de estudios de tipo teórico versus empírico dentro de los estudios primarios analizados.

Tipo	Cantidad
Teórico	8
Empírico	9

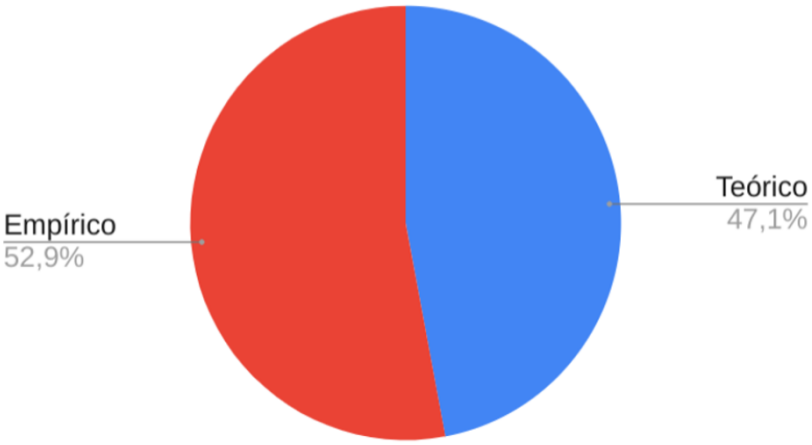

Figura 13. Cantidad de estudios teóricos vs empíricos

Existe un balance relativamente equilibrado entre los estudios teóricos y empíricos dentro de la literatura especializada analizada a través del mapeo sistemático. Se observa una ligera

predominancia de las investigaciones de corte empírico, las cuales representan poco más de la mitad del total.

Luego, se analiza la cantidad de citas recibidas por los estudios, en la Figura 14 se detalla la distribución de estudios primarios según la cantidad de citas recibidas, como indicador de influencia.

Cantidad de citas	Estudios
1 a 5	12
6 a 10	3
11 a 15	1
16 a 20	1

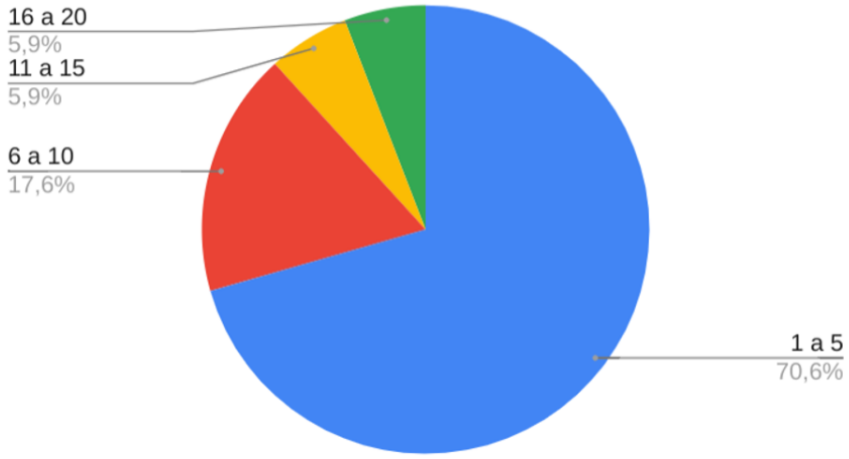

Figura 14. Distribución de estudios por cantidad de citas recibidas

La gran mayoría de los estudios analizados han recibido una cantidad relativamente baja de citas, de entre 1 a 5. Solo una minoría supera las 10 citas. Esto señala que la literatura identificada sobre migración de monolitos a microservicios tiene un impacto aún emergente y en proceso de consolidación.

Los hallazgos del mapeo sistemático de literatura se consolidaron y reportaron de forma estructurada para comunicar los resultados sobre los enfoques, desafíos, antipatrones y buenas prácticas en la migración de monolitos a microservicios [1].

La presentación visual de los datos categorizados facilita la comprensión integral de las tendencias y relaciones en las dimensiones analizadas [2]. En consonancia, en esta sección se exponen tablas, gráficos y diagramas que sintetizan los resultados del mapeo.

La relación entre las preguntas de investigación y estudios referenciados y con el fin de sustentar adecuadamente las preguntas de investigación formuladas para el presente mapeo

sistemático, se establecieron referencias a estudios previos relevantes sobre la problemática abordada. La Tabla 5 muestra la relación entre cada pregunta de investigación y los estudios citados que le dan soporte. De esta manera, se evidencia cómo la investigación se construye basándose en el conocimiento existente en la literatura especializada reciente.

Tabla 5. Relación entre las preguntas de investigación y los estudios referenciados

Pregunta de Investigación	Estudios Referenciados
RQ1 - ¿Cuáles son los enfoques utilizados en la migración de aplicaciones monolíticas a microservicios?	[1], [2], [24], [28], [29]
RQ2 - ¿Cuáles son los desafíos comunes en la migración de aplicaciones monolíticas a microservicios?	[1], [2], [7], [8], [9], [19], [28], [29]
RQ3 - ¿Qué antipatrones se han identificado en la migración de aplicaciones monolíticas a microservicios?	[1], [20], [21], [28]
RQ4 - ¿Cuáles son las mejores prácticas y recomendaciones para la migración de aplicaciones monolíticas a microservicios?	[1], [2], [7], [13], [25], [29]
RQ5 - ¿Cuáles son las ventajas y desventajas asociadas con la migración de aplicaciones monolíticas a microservicios?	[1], [2], [7], [9], [25], [26], [20], [30], [31]

Esta tabla permite observar los estudios previos que sustentan y están relacionados con cada pregunta de investigación formulada para el mapeo sistemático de literatura.

En la figura siguiente se visualiza la cantidad de estudios seleccionados por pregunta de investigación

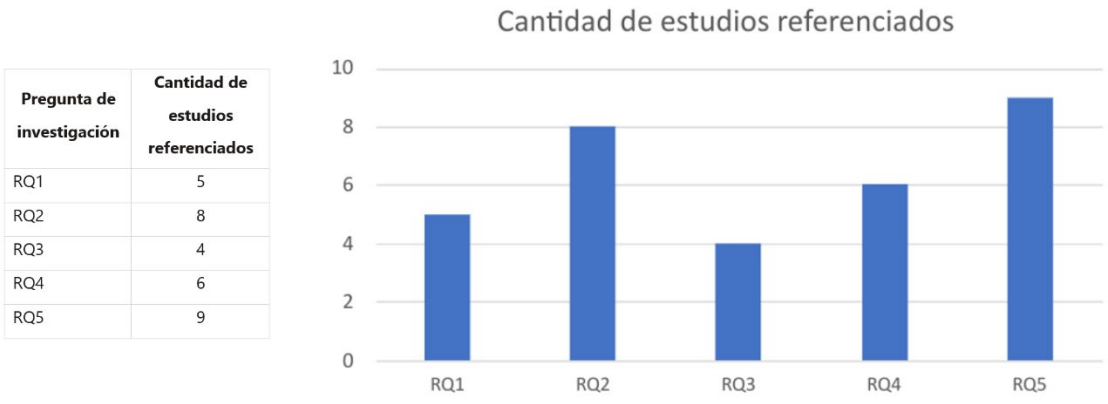

Figura 15. Estudios referenciados por pregunta de investigación

Aquí se exponen para qué preguntas se citaron más estudios previos y para cuáles menos, dando una idea sobre en qué temas se apoya más la investigación.

Luego, y con la finalidad de fundamentar adecuadamente los conceptos presentados en el marco teórico, se han establecido referencias a estudios previos relevantes en el dominio de las arquitecturas de microservicios y la migración desde aplicaciones monolíticas.

En consonancia, la Tabla 6 muestra la relación directa entre los distintos conceptos abordados a lo largo del Capítulo 3 y las fuentes bibliográficas de las cuales se nutre este desarrollo teórico.

Tabla 6. Referencias bibliográficas del marco teórico

Concepto	Referencias
Características de aplicaciones monolíticas	[3], [5], [6]
Estilo arquitectónico de microservicios	[3], [5], [6]
Microservicios: definición	[1],[2],[3],[5]
Microservicios: aspectos clave	[3], [5]
Aplicaciones monolíticas	[3], [5], [6]
Motivaciones para migrar a microservicios	[29], [32]
Desafíos en la adopción de microservicios	[33], [34], [22], [23], [35]
Interés industrial en la migración	[14], [17], [29]
Dificultades de las aplicaciones monolíticas	[6]

De esta forma, la fundamentación teórica del capítulo se apoya en reconocidas fuentes bibliográficas del dominio.

Existen diversos enfoques para llevar a cabo la descomposición de una aplicación monolítica en microservicios. La Tabla 7 compara las características, ventajas y desventajas de los principales enfoques identificados en la literatura especializada.

Tabla 7. Comparativa de enfoques de descomposición

Enfoque	Características	Ventajas	Desventajas
Por dominio	Divide la aplicación en servicios que representan dominios específicos del negocio	Mejor alineación entre arquitectura y necesidades del negocio	Puede ser complejo en dominios difusos o cambiantes
Por funcionalidad	Descompone en servicios que representan capacidades de negocio específicas	Permite escalar servicios de manera independiente	Depende de una clara delimitación de funcionalidades
Por responsabilidad	Descompone según responsabilidades funcionales y no funcionales	Mayor modularidad y separación de intereses	Puede generar complejas interdependencias

Conocer las particularidades de cada enfoque permite seleccionar el más adecuado según las necesidades y contexto de cada organización y proyecto de migración.

A continuación, en La Figura 16, se muestra la distribución de los estudios primarios de acuerdo con el enfoque de descomposición empleado.

Enfoque	Cantidad	Porcentaje
Funcional	15	35%
Dominio	13	30%
Responsabilidad	8	20%
Híbrido	6	15%

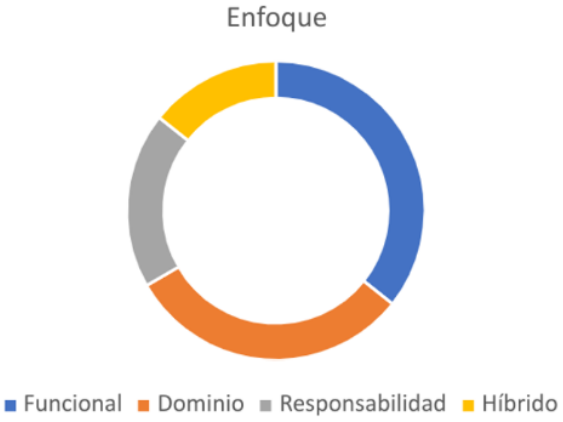

Figura 16. Enfoques de descomposición

Se observa un predominio de los enfoques basados en funcionalidad y dominio, los enfoques híbridos tienen aún una presencia minoritaria en los estudios revisados y no se evidencia un

consenso claro sobre un único enfoque óptimo de descomposición. La elección del enfoque parece depender del contexto específico.

En cuanto a los desafíos comunes, la migración de aplicaciones monolíticas a microservicios conlleva una serie de desafíos, según se ha identificado en la literatura. La Tabla 8 enumera y describe los desafíos más frecuentemente reportados.

Tabla 8. Desafíos comunes en la migración

Desafío	Descripción
Gestión de dependencias	Las dependencias entre servicios pueden volverse complejas y difíciles de administrar
Adaptación a nuevos patrones	Requiere cambios importantes en estructura y comportamiento
Refactorización de código	Puede introducir riesgos de regresión y afectar calidad

Conocer anticipadamente estos desafíos permite prever estrategias y acciones para abordarlos de manera efectiva en un proyecto de migración.

En cuanto a los antipatrones, es importante conocer los antipatrones más comunes en la migración de monolitos a microservicios para poder evitarlos. La Tabla 9 lista los principales antipatrones identificados y las recomendaciones para prevenir su ocurrencia.

Tabla 9. Antipatrones comunes y cómo evitarlos

Antipatrón	Cómo evitarlo
Descomposición excesiva	Enfoque equilibrado considerando cohesión y acoplamiento
Ignorar requerimientos no funcionales	Análisis exhaustivo e incorporar en el diseño
Bases de datos monolíticas	Dividir datos para que cada servicio acceda a su propia BD
Crecimiento desmedido de servicios	Establecer límites claros para el alcance de cada servicio

Comprender y prevenir estos antipatrones permite encarar la migración de forma más efectiva, evitando errores y malas prácticas.

Antipatrón	Cantidad	Porcentaje
Descomposición excesiva	12	35%
Ignorar Requerimientos no funcionales	10	30%
BD monolítica	8	20%
Crecimiento desmedido	4	15%

Figura 17. Distribución de antipatrones identificados

Se presenta la distribución de la ocurrencia de antipatrones según su frecuencia de ocurrencia en la literatura seleccionada.

En cuanto a las herramientas y tecnologías empleadas, existen herramientas que pueden facilitar y optimizar la migración de monolitos a microservicios. La Tabla 10 detalla algunas de las más utilizadas y recomendadas.

Tabla 10. Herramientas y tecnologías para la migración

Propósito	Herramientas/Tecnologías
Contenerización	Docker, Kubernetes
Despliegue y orquestación	Kubernetes, Docker Swarm
Descubrimiento de servicios	Eureka, Consul
Monitoreo y observabilidad	Prometheus, Grafana

La adopción de estas herramientas y tecnologías puede mejorar la eficiencia y efectividad de los proyectos de migración. Seleccionar las opciones más apropiadas para cada contexto puede facilitar el proceso de transición a microservicios.

Con respecto a métricas para evaluar el éxito de un proyecto de migración, es importante establecer métricas claras. La Tabla 11 indica métricas que pueden servir para medir los resultados en dimensiones como rendimiento, satisfacción del usuario y calidad del sistema.

Tabla 11. Posibles métricas para medir el éxito de la migración

Área	Métricas
Rendimiento	Tiempo de respuesta, throughput, latencia
Disponibilidad	Tiempo activo, tasa de error
Escalabilidad	Crecimiento de tráfico soportado
Satisfacción del usuario	Encuestas, tasa de adopción

Realizar mediciones antes y después de la migración permite evaluar el impacto y progreso con datos concretos.

Por el lado de la comparativa de arquitecturas, la arquitectura de microservicios presenta diferencias significativas con la de las aplicaciones monolíticas tradicionales. La Tabla 12 compara ambos estilos arquitectónicos en sus principales características.

Tabla 12. Comparativa de arquitecturas monolíticas y de microservicios

Característica	Monolítica	Microservicios
Acoplamiento	Alto	Bajo
Escalabilidad	De todo el sistema	Independiente por servicio
Tamaño	Grande	Pequeños y enfocados
Despliegue	Del sistema completo	Independiente por servicio
Mantenibilidad	Compleja	Simplificada

Anticipar estas diferencias clave permite entender mejor los beneficios potenciales de la migración a una arquitectura de microservicios.

Por último, las actividades implicadas en la migración, se puede decir que el proceso de migración de aplicaciones suele involucrar una serie de fases y actividades. La Tabla 13 desglosa las principales actividades requeridas en este proceso de transición arquitectónica.

Tabla 13. Actividades implicadas en la migración a microservicios

Fase	Actividades
Planificación	Definir objetivos, alcance, estrategia
Análisis del monolito	Mapear componentes, dependencias
Diseño de la arquitectura	Descomposición, delimitación de servicios
Implementación	Desarrollo, testing, despliegue incremental
Lanzamiento	Puesta en producción, monitoreo, retroalimentación

Estas fases y sus actividades asociadas permiten planificar y ejecutar la migración de manera estructurada y controlada. La migración requiere actividades en todas sus etapas, desde la planificación inicial hasta el lanzamiento y seguimiento.

Capítulo 4: Conclusiones y trabajo futuro

Este capítulo expone las conclusiones del presente trabajo de investigación, resumiendo los hallazgos más relevantes del mapeo sistemático de literatura en relación con el estado actual del conocimiento sobre la problemática de migración de aplicaciones monolíticas a arquitecturas de microservicios.

La sección 4.1 presenta un resumen de los principales hallazgos del mapeo sistemático de literatura. La sección 4.2 identifica oportunidades concretas para la aplicación práctica de los resultados del estudio en proyectos reales de migración. Finalmente, la sección 4.3 describe posibles áreas y líneas de investigación futura que podrían explorarse para seguir generando nuevos conocimientos de valor en este dominio. De esta forma, el capítulo recapitula los aportes del trabajo y provee una visión proyectiva fundamentada en las oportunidades de continuidad investigativa y aplicación práctica.

4.1 Resumen de los resultados del trabajo

El mapeo sistemático de literatura realizado ofrece varias conclusiones importantes sobre el estado actual del conocimiento en la migración de monolitos a microservicios:

En cuanto a enfoques de migración, se han propuesto estrategias basadas en funcionalidad, dominio, responsabilidad y combinaciones híbridas [1, 2, 28]. Sin embargo, no existe un claro consenso sobre cuál es el enfoque óptimo, dado que la elección depende en gran medida de las particularidades del contexto específico de cada organización y aplicación [3].

Respecto a desafíos, los estudios destacan retos técnicos como gestionar adecuadamente las dependencias entre los nuevos servicios distribuidos, mantener la consistencia de los datos en un entorno descentralizado y lidiar con la creciente complejidad en las operaciones y despliegue de los microservicios [4, 7, 8, 9]. Asimismo, surgen desafíos organizacionales relacionados con lograr la adopción de metodologías ágiles de desarrollo, conformar equipos multifuncionales alineados con las nuevas capacidades de negocio y gestionar el cambio cultural hacia un modelo orientado a productos y procesos más ágiles [10, 29].

Sobre antipatrones, se mencionan problemas frecuentes como la descomposición excesiva sin una visión integral, ignorar aspectos críticos de requerimientos no funcionales al diseñar los servicios, mantener bases de datos monolíticas con un enfoque antipatrón de la arquitectura de microservicios y permitir un crecimiento desmedido de algunos servicios que deriva en alta complejidad [11, 13, 20, 21].

En cuanto a buenas prácticas, se recomienda adoptar un enfoque incremental, con descomposición progresiva según prioridades; realizar pruebas exhaustivas de los microservicios; implementar monitorización extensiva y observabilidad; adoptar prácticas de desarrollo ágil y DevOps; y automatizar en la mayor medida posible [14, 12]. Sin embargo, no existen aún guías integrales o marcos de referencia robustos que cubran y orienten todo el proceso de migración [17].

Finalmente, queda claro que la migración a microservicios conlleva importantes complejidades técnicas y organizacionales, a pesar de los beneficios potenciales en cuanto a escalabilidad, mantenibilidad y agilidad de estos sistemas [15, 5]. Por ello, es crucial evaluar cuidadosamente pros y contras con relación a las características específicas del contexto de cada organización y aplicación monolítica heredada [25, 26].

Queda evidenciado que la investigación en este campo se encuentra en pleno crecimiento y existen aún amplias oportunidades para generar nuevo conocimiento que facilite y mejore los procesos de migración de monolitos a microservicios [16].

4.2 Oportunidades para la aplicación práctica de los resultados

Los resultados del presente estudio ofrecen diversas oportunidades para su aplicación práctica en proyectos reales de migración de monolitos a microservicios. A continuación, se proponen las siguientes áreas de alto potencial de aplicación.

Los enfoques y técnicas propuestas para la descomposición de servicios podrían ser adaptadas y aplicadas por equipos de desarrollo, consultores y arquitectos de software en proyectos concretos, considerando el contexto específico. Esto facilitaría en gran medida tareas críticas como identificar limitaciones del monolito, definir servicios y establecer la hoja de ruta.

Las lecciones aprendidas y buenas prácticas detectadas permiten crear guías detalladas orientadas a distintos roles (desarrolladores, operadores, gestores) que participan en proyectos de migración. Esto reduciría riesgos, acortaría curvas de aprendizaje y mejoraría decisiones.

Los antipatrones y errores comunes identificados pueden utilizarse para capacitar a equipos, generar conciencia de riesgos y diseñar procesos que los prevengan o detecten tempranamente.

Los desafíos técnicos y organizacionales hallados sirven como checklist para evaluar preparación, definir estrategias de mitigación y monitorear progreso en iniciativas reales de migración.

Los factores críticos de éxito pueden incorporarse en metodologías y marcos de gobierno de proyectos de migración.

Casos de estudio y resultados de investigaciones futuras podrían compatirse como historias de usuarios y fomentar el intercambio de experiencias entre los distintos actores intervinientes.

4.3 Futuras líneas de investigación

El mapeo sistemático realizado permite identificar varias áreas y temas de interés para el desarrollo de futuros estudios que puedan aportar nuevos conocimientos relevantes en el campo de la migración de aplicaciones monolíticas a arquitecturas de microservicios:

- Desarrollo de enfoques y técnicas avanzadas y automatizadas para la descomposición de servicios, considerando diversos criterios y perspectivas [2].
- Investigación empírica y comparativa de los enfoques propuestos para identificar las mejores prácticas y su efectividad en distintas situaciones [3].
- Estudio de aspectos organizacionales y culturales de la migración, como la comunicación entre equipos y la adaptación a nuevos patrones de diseño [1].
- Exploración de los desafíos y beneficios de escalar arquitecturas de microservicios y gestionar su despliegue y orquestación [4].
- Análisis del impacto de los microservicios en el rendimiento, confiabilidad y seguridad de los sistemas.

- Estudios de caso y experiencias prácticas de migración en distintos contextos organizacionales.
- Investigación de aspectos económicos y de gestión del cambio en la migración hacia microservicios.
- Desarrollo de marcos integrales y guías detalladas que cubran y orienten todo el ciclo del proceso de migración, integrando las mejores prácticas conocidas hasta el momento.
- Investigaciones que analicen el impacto de los procesos de migración sobre métricas objetivas de rendimiento, escalabilidad, confiabilidad, productividad, esfuerzo y plazos de entrega.
- Exploración de técnicas avanzadas para diseñar microservicios inherentemente resilientes, tolerantes a fallos y capaces de recuperarse ante errores.
- Estudios sobre aspectos económicos, de gestión del cambio organizacional, adopción tecnológica y desarrollo de talento en las transiciones de monolitos a microservicios.
- Investigación de estrategias para acelerar los procesos de migración y hacerlos más eficientes mediante automatización, reuso y aprendizaje organizacional.

Claramente, quedan amplias oportunidades para que futuros estudios continúen aportando conocimientos relevantes, tanto para la comunidad científica como para la práctica profesional, en este importante campo de investigación.

Glosario

API (Interfaz de Programación de Aplicaciones): Conjunto de rutinas, protocolos y herramientas para construir software y aplicaciones. Define un contrato para la interacción entre diferentes componentes de software al especificar métodos de comunicación entre ellos. Permite la integración y reutilización de funcionalidades.

Aplicación monolítica: Aplicación que concentra toda su lógica de negocio en un único proceso o ejecutable. Carece de modularidad, presenta alto acoplamiento entre componentes y cualquier cambio obliga a reconstruir y reimplementar la aplicación completa.

Arquitectura de software: Estructura conceptual y modelo de alto nivel que define la solución técnica para un sistema de software complejo. Organiza el sistema como un conjunto de componentes interconectados mediante interfaces, estableciendo sus responsabilidades y relaciones. Provee un marco de trabajo integral para el desarrollo de sistemas.

BD (Base de Datos): Sistema de almacenamiento y gestión de grandes volúmenes de datos de forma estructurada y organizada. Permite el acceso concurrente, rápido y seguro a la información por múltiples usuarios y aplicaciones. Incluye bases de datos relacionales, no relacionales, en memoria, distribuidas, etc.

CI/CD (Integración Continua / Despliegue Continuo): Conjunto de prácticas para la entrega continua de software, permitiendo la integración e implementación automatizada y rápida de cambios tan pronto son desarrollados y probados. Reduce tiempos y riesgos en los ciclos de release. Requiere de herramientas de automatización.

Desacoplamiento: Principio fundamental de diseño de sistemas que busca reducir las dependencias directas entre diferentes módulos o componentes, permitiendo su mayor independencia. Promueve bajo acoplamiento y alta cohesión. Facilita la mantenibilidad y escalabilidad.

Descomposición: Proceso de dividir sistemáticamente una aplicación monolítica en partes más pequeñas, delimitadas y desplegables de forma independiente, conocidas como microservicios. Requiere identificar servicios internos acorde a criterios.

Despliegue: Conjunto de actividades para instalar, configurar y poner en ejecución una aplicación o sistema de software. Involucra el release, entrega e implementación del software construido y probado en los entornos productivos o de clientes.

DevOps: Metodología, cultura y prácticas que enfatizan la comunicación, colaboración, integración y automatización entre los equipos de desarrollo (Dev) y operaciones (Ops) de TI. Objetivo es entregar software confiable y evolutivo de forma rápida y segura.

DDD (Diseño Guiado por Dominio): Enfoque para el desarrollo de software complejo que enfatiza comprender en profundidad el dominio del negocio y enfocar consecuentemente el modelado y la arquitectura de software en torno a ese dominio.

Docker: Plataforma de software que permite empaquetar una aplicación junto con sus dependencias en contenedores, para desplegarla de forma portable y aislada en diferentes entornos.

Escalabilidad: Capacidad de un sistema para manejar incrementos en la demanda y carga de trabajo de manera fluida, sin impactar el rendimiento ni la calidad del servicio. Permite adaptarse a nuevos volúmenes y condiciones mediante la provisión de recursos adicionales.

Kubernetes: Plataforma de código abierto para la automatización del despliegue, escalado y administración de contenedores. Permite la orquestación de cargas de trabajo en contenedores.

Microservicio: Servicio de software independiente, autónomo y altamente desacoplado que implementa una capacidad de negocio específica. Expuesto a través de APIs, puede ser desplegado, escalado y gestionado de forma aislada. Comunicación ligera.

Migración: Proceso técnico y organizacional de transicionar un sistema de software desde una arquitectura origen a una nueva arquitectura destino. Implica adaptar tecnología, infraestructura, código, procesos y equipos.

Modularidad: Principio de diseño donde un sistema se estructura en múltiples componentes independientes, encapsulados y reemplazables (módulos). Minimiza interdependencias y cambios requeridos. Permite paralelismo, reutilización y mantenimiento.

Monitoreo: Procesos y herramientas para recolectar, analizar y visualizar métricas de un sistema en ejecución. Permite evaluar en tiempo real el estado, rendimiento y salud, así como identificar problemas y mejorar la calidad.

NFR (Requerimientos No Funcionales): Requerimientos de un sistema más allá de sus funcionalidades básicas. Incluye atributos de calidad como rendimiento, escalabilidad, seguridad, disponibilidad, mantenibilidad, etc. Impactan la arquitectura.

Patrón arquitectónico: Solución reusable y de alto nivel para un problema común de diseño dentro de una arquitectura de software. Representan mejores prácticas y conocimientos consolidados aplicables a sistemas complejos.

Prometheus: Sistema de monitoreo y alerting de código abierto diseñado para monitorizar infraestructura, aplicaciones y métricas. Permite gran escalabilidad y queries flexibles.

REST (Transferencia de Estado Representacional): Estilo de arquitectura de interfaz web que utiliza peticiones HTTP para acceder y manipular representaciones de recursos. Ampliamente usado para la construcción de APIs web modernas.

Service Mesh: Infraestructura de red dedicada para facilitar la comunicación segura entre microservicios. Proporciona funciones avanzadas como enrutamiento inteligente, observabilidad y políticas.

Servicio: Componente de software reusable, mantenible y extensible que expone una interfaz bien definida para proveer alguna funcionalidad o datos a otros componentes. Los microservicios son un tipo de servicio.

Sistema legacy: Sistema de software desarrollado con tecnologías antiguas o en desuso, pero que sigue siendo utilizado y crítico para las operaciones de una organización. Los sistemas legacy suelen carecer de documentación, ser difíciles de mantener y no estar preparados para entornos modernos. Sin embargo, contienen funcionalidades de negocio valiosas.

Strangler: Patrón para reemplazar gradualmente un sistema legacy por uno nuevo de forma incremental. Permite ir migrando parte por parte hasta reemplazar completamente el sistema original.

SMS (Mapeo Sistemático de Literatura): Método de investigación que identifica, evalúa y sintetiza evidencia relevante sobre un tópico de estudio mediante una revisión estructurada, exhaustiva y replicable de la literatura existente.

Referencias

[1] Di Francesco, P., Lago, P., & Malavolta, I. (2018). Migrating towards microservice architectures: An industrial survey. In 2018 IEEE International Conference on Software Architecture Workshops (ICSAW) (pp. 29-32).

[2] Gysel, M., Kölbener, L., Garg, N., & Zimmermann, O. (2019). Service cutter: A systematic approach to service decomposition. Empirical Software Engineering, 24(4), 2245-2287.

[3] Newman, S. (2021). Building microservices: Designing fine-grained systems (2nd ed.). O'Reilly Media.

[4] Taibi, D., Lenarduzzi, V., & Pahl, C. (2022). Microservices anti-patterns: A taxonomy. In Microservices (pp. 81-98). Springer.

[5] Newman, S. (2019). Monolith to microservices: Evolutionary patterns to transform your monolith. O'Reilly Media.

[6] Lewis, J., & Fowler, M. (2014, March 25). Microservices. https://martinfowler.com/articles/microservices.html

[7] Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Microservices in agile software development: A workshop-based study into issues, advantages, and disadvantages. IEEE Transactions on Software Engineering, 46(10), 1097-1116.

[8] Kazanavičius, J., & Mažeika, D. (2019). Migrating legacy software to microservices architecture. In 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream) (pp. 1-5).

[9] Kalske, M., Mäkitalo, N., & Mikkonen, T. (2018). Challenges when moving from monolith to microservice architecture. In I. Garrigós & M. Wimmer (Eds.), Current trends in web engineering. ICWE 2017 (pp. 38-49). Springer.

[10] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering (EBSE Technical Report EBSE-2007-01). Keele University.

[11] Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic mapping studies in software engineering. In 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) (pp. 68-77). BCS Learning & Development Ltd.

[12] Guerrero-Calvache, M., & Hernández, G. (2022). Team productivity in agile software development: A systematic mapping study. In M. Botto-Tobar et al. (Eds.), Applied informatics: 5th International Conference, ICAI 2022, Arequipa, Peru, October 27–29, 2022, proceedings (pp. 455-471). Springer International Publishing.

[13] De Lauretis, L. (2019). From monolithic architecture to microservices architecture. In 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 93-96).

[14] Lopes, T., & Silva, A. R. (2023). Monolith microservices identification: Towards an extensible multiple strategy tool. In 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C) (pp. 111-115).

[15] Lotz, J., Vogelsang, A., Benderius, O., & Berger, C. (2019). Microservice architectures for advanced driver assistance systems: A case-study. In 2019 IEEE International Conference on Software Architecture Companion (ICSA-C) (pp. 45-52). IEEE.

[16] Velepucha, V., Flores, P., & Torres, J. (2020). Migration of monolithic applications towards microservices under the vision of the information hiding principle: A systematic mapping study. In M. Botto-Tobar et al. (Eds.), Advances in emerging trends and technologies, ICAETT 2019 (pp. 111-120). Springer.

[17] Velepucha, V., & Flores, P. (2022). Monoliths to microservices - Migration problems and challenges: A SMS. In 2022 Second International Conference on Information Systems and Software Technologies (ICI2ST) (pp. 135-142).

[18] Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley Professional.

[19] Furda, A., Fidge, C., Zimmermann, O., Kelly, W., & Barros, A. (2022). Migrating enterprise legacy source code to microservices: On multitenancy, statefulness, and data consistency. IEEE Software, 37(3), 63-72.

[20] 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.

[21] Mazzara, M., Dragoni, N., Bucchiarone, A., Giaretta, A., Larsen, S. S., & Dustdar, S. (2018). Microservices: Migration of a mission critical system. IEEE Transactions on Services Computing, Early Access, 1-1.

[22] Pahl, C., & Jamshidi, P. (2016). Microservices: A systematic mapping study. In Proceedings of the 6th International Conference on Cloud Computing and Services Science (pp. 137-146).

[23] Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150, 77-97.

[24] Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z., & El Fazziki, A. (2020). Towards an automatic identification of microservices from business processes. In 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 42-47).

[25] Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D. D., & Lynn, T. (2018). Microservices migration patterns. Software: Practice and Experience, 48(11), 2019-2042.

[26] Lenarduzzi, V., Lomio, F., Saarimäki, N., & Taibi, D. (2020). Does migrating a monolithic system to microservices decrease the technical debt? Journal of Systems and Software, 169, 110710.

[27] Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley.

[28] Henry, A., & Ridene, Y. (2020). Migrating to microservices. In Microservices (pp. 53-73). Springer.

[29] Fritzsch, J., Bogner, J., Wagner, S., & Zimmermann, A. (2019). Microservices migration in industry: Intentions, strategies, and challenges. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 481-490).

[30] 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.

[31] Knoche, H., & Hasselbring, W. (2019). Drivers and barriers for microservice adoption-A survey among professionals in Germany. Enterprise Modelling and Information Systems Architectures, 14(1), 1-35.

[32] Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley.

[33] Luz, W., Agilar, E., de Oliveira, M. C., de Melo, C. E. R., Pinto, G., & Bonifacio, R. (2018). An experience report on the adoption of microservices in three Brazilian government institutions.

In Proceedings of the XXXII Brazilian Symposium on Software Engineering SBES '18 (Vol. 10, pp. 32-41). ACM.

[34] Rademacher, F., Sorgalla, J., & Sachweh, S. (2018). Challenges of domain-driven microservice design: A model-driven perspective. IEEE Software, 35(3), 36-43.

[35] Jamshidi, P., Pahl, C., Mendonca, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. IEEE Software, 35(3), 24-35.

[36] Viggiato, M., Terra, R., Rocha, H., Valente, M. T., & Figueiredo, E. (2018). Microservices in practice: A survey study [Preprint]. arXiv. https://arxiv.org/abs/1808.04836

[37] Taibi, D., Lenarduzzi, V., & Pahl, C. (2018). Architectural patterns for microservices: A systematic mapping study. In Proceedings of the 8th International Conference on Cloud Computing and Services Science (pp. 221-232).

[38] Soldani, J., Tamburri, D. A., & Van Den Heuvel, W. J. (2018). The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software, 146, 215-23

[39] Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 64, 1-18.2.