



EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Tecnológico Nacional de México

**Centro Nacional de Investigación
y Desarrollo Tecnológico**

Tesis de Maestría

**Análisis de métricas en el desarrollo de
microservicios**

presentada por

Ing. Said Misael Venoso Lara

como requisito para la obtención del grado
de

**Maestro en Ciencias de la
Computación**

Director de tesis

Dr. Juan Carlos Rojas Pérez

Cuernavaca, Morelos, México. Marzo de 2026.



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Mor., 04/Febrero/2026

OFICIO No. DCC/022/2026

Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

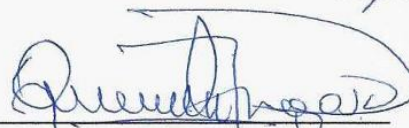
CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial de **Said Misael Venoso Lara** con número de control **M24CE008**, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado "Análisis de métricas en el desarrollo de microservicios" y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

ATENTAMENTE

Excelencia en Educación Tecnológica®
"Conocimiento y Tecnología al Servicio de México"


Dr. Juan Carlos Rojas Pérez
Director de tesis


Dr. Olivia Graciela Fragozo Díaz
Revisor 1


Dr. René Santaolaya Salgado
Revisor 2

C.c.p.

Depto. Servicios Escolares.
Expediente / Estudiante



cenidet
Centro Nacional de Investigación
y Desarrollo Tecnológico



Interior Internado Palmira S/N, Col. Palmira,
C.P. 62490, Cuernavaca, Morelos Tel. 01 (777) 3627770, ext. 3201,
e-mail: dcc_cenidet@tecnm.mx | cenidet.tecnm.mx



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Centro Nacional de Investigación y Desarrollo Tecnológico
Subdirección Académica

Cuernavaca Mor, 04/febrero/2026

Oficio No. SAC/043/2026

Asunto: Autorización de impresión de tesis

SAID MISAEL VENOSO LARA
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS
DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Análisis de métricas en el desarrollo de microservicios"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

ATENTAMENTE

Excelencia en Educación Tecnológica®

"Conocimiento y Tecnología al servicio de México"

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO



c.c.p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz



Interior Internado Palmira S/N, Col. Palmira,
C.P. 62490, Cuernavaca, Morelos Tel. 01 (777) 3627770, ext. 4106,
e-mail: acad_cenidet@tecnm.mx | cenidet.tecnm.mx

Dedicatoria

A mi familia, por siempre darme el amor y apoyo incondicional para poder lograr una meta más en mi vida, principalmente a mi mamá Marcelina Mendoza Alvarado, quien siempre ha sido mi motor y mi razón de ser, quien siempre me apoya en todo y está en todos los momentos.

A mi pareja, Perla Itzul Reza Peralta, por apoyarme todos los días en todo lo que estuvo en tus manos, ofrecer su amor incondicional en todo momento, ser mi soporte emocional y acompañarme tanto en los buenos días, como en los malos.

A mis amigos Alexei, Jesús, Miguel y Sebastián por acompañarme en medida de lo posible en este viaje, las divertidas salidas para el desestrés y el ánimo que me ofrecieron todos los días.

A todos los amigos y compañeros que conocí a lo largo de mi estancia en la maestría, quienes me apoyaron en todo momento, ya fuese desde comentarios constructivos, opiniones, soporte emocional, regaños, etc. Ustedes hicieron más que agradable ser parte de la familia CENIDET.

Agradecimientos

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por el programa Sistema Nacional de Posgrados (SNP) por medio del cual recibí apoyo económico como estudiante a tiempo completo.

Al Tecnológico Nacional de México campus Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), por brindarme la oportunidad de pertenecer a su comunidad estudiantil.

A mi director de tesis, el Dr. Juan Carlos Rojas Pérez, quien fue uno de los pilares para la culminación de este trabajo, desde su seguimiento y apoyo constante, hasta ser la voz de la razón cuando todo parecía difícil, asimismo, ofreciendo todo su conocimiento y abriéndome las puertas para poder culminar este gran paso en mi vida.

A la Dra. Olivia Graciela Fragoso Díaz, por dedicarme gran parte de su tiempo para atender las revisiones de este trabajo, por apoyarme todo el tiempo y por ser parte fundamental de mi desarrollo como profesionista y persona, ya que sin sus oportunas revisiones, consejos y asesorías esto no sería posible.

Al Mtro. Humberto Hernández García, por ofrecerme sus valiosas revisiones, correcciones, consejos y sugerencias, las cuales contribuyeron en gran medida para el desarrollo de este trabajo, así como el apoyo brindado como persona.

Al Dr. René Santaolaya Salgado, por ofrecerme siempre su consejo oportuno, sus revisiones, correcciones y sugerencias, por confiar en mi para la realización de este trabajo, y siempre estar disponible cuando lo fue necesario.

Al Mtro. Daniel Flores Guerrero, quien me motivó e impulsó para superarme como profesionista continuando con mis estudios y dar este gran paso en mi vida personal y profesional, gracias totales.

Resumen

En los últimos años, las arquitecturas basadas en microservicios han ganado una adopción creciente tanto en la industria como en la comunidad académica, debido a ventajas como la flexibilidad, la escalabilidad, y la agilidad en el desarrollo de software. Este enfoque arquitectónico permite descomponer sistemas monolíticos de gran tamaño en componentes independientes, lo que facilita su mantenimiento, evolución y despliegue. Sin embargo, a pesar de su amplia popularidad, se ha identificado que muchas métricas tradicionales, originalmente diseñadas para arquitecturas orientadas a objetos o arquitecturas orientadas a servicios, no resultan plenamente aplicables al contexto de los microservicios. Además, una gran parte de las métricas propuestas en la literatura carece de validación empírica o de un consenso claro respecto a su definición de uso.

Ante esta problemática, el presente trabajo tiene como objetivo presentar un estudio integral de métricas orientadas al diseño e implementación de los microservicios, con la finalidad de desarrollar un modelo de calidad que sirva como guía para arquitectos y desarrolladores de software. Para ello, se llevó a cabo una recopilación y clasificación de métricas reportadas en la literatura especializada, así como la identificación de los atributos de calidad propuestos por la norma ISO/IEC 25010, con el propósito de alinear las características propias de los microservicios con dichos atributos. El análisis llevado a cabo permitió identificar tendencias, enfoques recurrentes y vacíos existentes en la revisión de la literatura en cuanto a métricas para microservicios.

Los resultados del análisis cuantitativo evidencian una cobertura desigual de los atributos de calidad, además de una ausencia generalizada de umbrales claramente definidos, lo cual dificulta la aplicación práctica de las métricas existentes. Con el objetivo de validar el modelo de calidad propuesto, se realizaron pruebas preliminares a tres repositorios de libre acceso desarrollados en Java. Los resultados de estas pruebas mostraron que los proyectos evaluados presentan un bajo nivel de flexibilidad, lo cual se puede atribuir principalmente a deficiencias en los subatributos de escalabilidad y portabilidad. Finalmente, se concluyó que el modelo resulta fundamental para visibilizar las debilidades estructurales en implementaciones reales de arquitecturas de microservicios, especialmente en entornos basados en Java, así como para orientar futuras mejoras en su diseño y evaluación.

Abstract

In recent years, microservice-based architectures have gained increasing adoption in both industry and academia due to advantages such as flexibility, scalability, and agility in software development. This architectural approach allows large monolithic systems to be broken down into independent components, facilitating maintenance, evolution, and deployment. However, despite its widespread popularity, it has been identified that many traditional metrics, originally designed for object-oriented or service-oriented architectures, are not fully applicable to the context of microservices. In addition, a large part of the metrics proposed in the literature lack empirical validation or a clear consensus regarding their definition of use.

Given this problem, the objective of this paper is to present a comprehensive study of metrics oriented toward the design and implementation of microservices, with the aim of developing a quality model that serves as a guide for software architects and developers. To this end, metrics reported in the specialized literature were compiled and classified, and the quality attributes proposed by the ISO/IEC 25010 standard were identified, with the aim of aligning the characteristics of microservices with these attributes. The analysis carried out made it possible to identify trends, recurring approaches, and gaps in the literature review regarding metrics for microservices.

The results of the quantitative analysis show uneven coverage of quality attributes, as well as a general lack of clearly defined thresholds, which hinders the practical application of existing metrics. In order to validate the proposed quality model, preliminary tests were carried out on three open-access repositories developed in Java. The results of these tests showed that the projects evaluated have a low level of flexibility, which can be attributed mainly to deficiencies in the sub-attributes of scalability and portability. Finally, it was concluded that the model is essential for highlighting structural weaknesses in real implementations of microservice architectures, especially in Java-based environments, as well as for guiding future improvements in their design and evaluation.

Contenido

Capítulo 1	Introducción.....	1
1.1.	Antecedentes	2
1.2.	Planteamiento del problema	3
1.3.	Objetivos	4
1.3.1.	Objetivo General.....	4
1.3.2.	Objetivos Específicos	4
1.4.	Alcances y Limitaciones.....	4
1.4.1.	Alcances	4
1.4.2.	Limitaciones.....	4
1.5.	Estructura del documento	4
Capítulo 2	Marco Conceptual	6
2.1.	Arquitectura de Microservicios.....	7
2.2.	Microservicio	7
2.3.	Métrica	7
2.4.	Atributo de calidad de software	7
2.5.	Flexibilidad.....	7
2.5.1.	Adaptabilidad	7
2.5.2.	Escalabilidad.....	8
2.5.3.	Instalabilidad	8
2.5.4.	Reemplazabilidad.....	8
2.6.	Fiabilidad	8
2.7.	Mantenibilidad.....	8
2.8.	Eficiencia de desempeño	8
2.9.	Seguridad	8
2.10.	Compatibilidad	8
2.11.	Portabilidad	9
2.12.	Modelo de calidad	9
2.13.	Umbral	9
2.14.	Endpoint.....	9
2.15.	Arquitectura de monolitos	9
2.16.	API REST.....	9

2.17.	Kubernetes	9
2.18.	Docker	10
2.19.	Calidad de un producto de software	10
2.20.	Arquitectura Orientada a Servicios	10
2.21.	Arquitectura Orientada a Objetos	10
2.22.	Principios IDEALS.....	10
2.22.1.	Segregación de interfaces	10
2.22.2.	Despliegue autónomo.....	10
2.22.3.	Basado en eventos	11
2.22.4.	Disponibilidad sobre consistencia	11
2.22.5.	Bajo acoplamiento	11
2.22.6.	Única responsabilidad	11
2.23.	Medición en ingeniería de software	11
2.24.	Entidad.....	11
2.25.	Contenerización	11
Capítulo 3	Trabajos Relacionados.....	12
3.1.	Trabajos que mencionan características de los microservicios	16
3.2.	Trabajos que utilizan métricas en microservicios.....	17
3.3.	Trabajos respecto migración a microservicios	17
Capítulo 4	Desarrollo de la solución	18
4.1.	Planeación	19
4.2.	Conceptualización de los microservicios	20
4.3.	Información de métricas para microservicios	26
4.3.1.	Relación de métricas para microservicios con atributos de calidad según la ISO 25010	26
4.3.2.	Cuantificación de métricas	29
4.3.3.	Umbrales para métricas en microservicios	38
4.3.4.	Discusión de los resultados obtenidos tras el análisis de las métricas.....	40
4.4.	Modelo de calidad	44
4.4.1.	Adaptabilidad	46
4.4.2.	Escalabilidad.....	46
4.4.3.	Portabilidad.....	47

4.4.4.	Ventajas del modelo propuesto.....	49
Capítulo 5	Pruebas y resultados.....	51
5.1.	Proceso de pruebas.....	52
5.2.	Interpretación de resultados	56
5.2.1.	Repositorio ZULL	56
5.2.2.	Repositorio ACME Air Microservice.....	58
5.2.3.	Repositorio E-Commerce.....	59
Capítulo 6	Conclusiones y Trabajos Futuros	65
Capítulo 7	Publicaciones y Actividades Complementarias.....	69
7.1.	Publicaciones	70
7.2.	Eventos Académicos	70
Referencias.....		71
Anexos		77
Anexo A.	Formas de evaluación de las métricas.....	77
Anexo B.	Métricas encontradas en la literatura	85
Anexo C.	Resúmenes de los trabajos relacionados.....	114
Anexo D.	Implementación del modelo de calidad propuesto para evaluar microservicios.....	120
Anexo E.	Revisión Sistemática de la Literatura	128
1.	Planificación	130
2.	Realización de la revisión.....	135

Índice de Figuras

Figura 1 Diagrama de flujo del proceso de investigación	19
Figura 2 Esquema de la definición de microservicio.	22
Figura 3 Modelo de calidad según la ISO/IEC 25010.....	23
Figura 4 Principios IDEALS para microservicios [28].	25
Figura 5 Relación entre atributos de calidad según la ISO 25010 y microservicios.	26
Figura 6 Métricas relacionadas a flexibilidad.....	29
Figura 7 Menciones de las métricas de flexibilidad en la literatura	30
Figura 8 Métricas relacionadas a fiabilidad.....	31
Figura 9 Menciones de las métricas de fiabilidad en la literatura	31
Figura 10 Métricas relacionadas a mantenibilidad	32
Figura 11 Menciones de las métricas de mantenibilidad en la literatura.....	33
Figura 12 Métricas relacionadas a eficiencia de desempeño.....	33
Figura 13 Menciones de las métricas de eficiencia de desempeño en la literatura	34
Figura 14 Métricas relacionadas a seguridad	34
Figura 15 Menciones de las métricas de seguridad en la literatura	35
Figura 16 Métricas relacionadas a compatibilidad	35
Figura 17 Menciones de las métricas de compatibilidad en la literatura.....	36
Figura 18 Top 10 métricas con más formas de evaluar	38
Figura 19 Umbrales presentes en la literatura	39
Figura 20 Presencia de las 10 métricas principales en los atributos de calidad.	44
Figura 21 Modelo de calidad de flexibilidad de un microservicio	45
Figura 22 Promedio de resultados totales por los tres repositorios [74], [75], [76]	63
Figura 23 Metodología de revisión sistemática de la literatura según Torres-Carrion et. al [77]	128
Figura 24 Procedimiento de la metodología propuesta por Torres-Carrion Et. al [77].	129
Figura 25 Mentefacto Conceptual del concepto de microservicios.....	131
Figura 26 Nube de palabras referentes a métricas de microservicios.....	131

Índice de Tablas

Tabla 1 Clasificación de los trabajos relacionados sobre métricas para microservicios	13
Tabla 2 Características de los microservicios según Fowler y Lewis.	21
Tabla 3 Relación de atributos de un microservicio con los principios IDEALS.	24
Tabla 4 Relación de métricas con los atributos de calidad para microservicios.	27
Tabla 5 Métricas con número de formas de evaluación ≤ 4	36
Tabla 6 Top 10 métricas con más formas de evaluar	37
Tabla 7 Relación de las métricas con más formas de evaluar y los atributos de calidad.	43
Tabla 8 Repositorios seleccionados	52
Tabla 9 Actividades para evaluar los microservicios	54
Tabla 10 Resultados del repositorio ZULL [74] por microservicio	56
Tabla 11 Resultados del repositorio ACME Air Microservice [75] por microservicio	58
Tabla 12 Resultados del repositorio E-Commerce [76] por microservicio	59
Tabla 13 Resultados de flexibilidad del microservicio product-recommendation-service del repositorio E-Commerce [76]	61
Tabla 14 Resultados obtenidos por repositorio y microservicio	62
Tabla 15 Formas de evaluación de las métricas	78
Tabla 16 Cadenas de búsqueda	132
Tabla 17 Cadenas de búsqueda en IEEEExplorer	132
Tabla 18 Cadenas de búsqueda en ACM	133
Tabla 19 Cadenas de búsqueda en Google Academic	133

Capítulo 1 Introducción



Este capítulo presenta una introducción al tema de la tesis, así como un breve resumen de los antecedentes que motivan este estudio, una descripción del problema de investigación con sus respectivos objetivos generales y específicos, los cuales fueron de gran utilidad al orientar la investigación. Asimismo, se presentan los alcances y limitaciones del proyecto.

En los últimos años, la arquitectura de microservicios (MSA) ha ganado gran popularidad para el desarrollo de aplicaciones distribuidas, ofreciendo ventajas como flexibilidad, escalabilidad, agilidad, etc. [1], [2], [3]. Este enfoque permite descomponer grandes sistemas monolíticos en componentes independientes facilitando su mantenimiento y evolución. No obstante, la transición de un sistema monolítico hacia los microservicios ofrece una amplia gama de desafíos en términos de calidad, pues las métricas utilizadas en las Arquitecturas Orientadas a Objetos (OOA) [4], Arquitecturas Orientadas a Servicios (SOA) [5] o la Computación Orientada a Servicios (SOC) [6] no resultan siempre aplicables de forma directa a las MSA. En muchos casos, las métricas propuestas se han quedado en etapas conceptuales, carecen de validación empírica o presentan dificultades de implementación en entornos de microservicios [1], [7], [8].

Por este motivo, dada la gran cantidad de métricas en la literatura, surge la necesidad de realizar un análisis que muestre el estatus actual de todas estas métricas, cómo se han aplicado, cómo miden, cuáles son propuestas y cuáles pueden ser aplicables directamente a las MSA; con este estudio fue posible obtener un compendio que sirviera de base a los desarrolladores y arquitectos de las MSA.

1.1. Antecedentes

En el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) se han llevado a cabo algunos trabajos relacionados con métricas orientadas a Microservicios, de los cuales destacan: “Adaptación de un patrón de software en seguridad a la arquitectura de un Microservicio” [9], “Estudio de métricas y patrones de seguridad en microservicios” [10], “Selección y adaptación de métricas para Microservicios” [11], “Metodología en la determinación de la granularidad de un Microservicio” [12] y “Modelo de calidad para determinar la granularidad de microservicios” [13].

En “Adaptación de un patrón de software en seguridad a la arquitectura de un Microservicio” [9] se investiga una amplia gama de riesgos de seguridad con la adopción de las arquitecturas de microservicios, dándose a la tarea de diseñar patrones de seguridad específicos para los microservicios protegiendo los puntos de acceso a estos y así aumentar la confianza a la hora de implementar estas arquitecturas.

Por otro lado, en “Estudio de métricas y patrones de seguridad en microservicios” [10] se realiza un mapeo sistemático con el fin de resaltar el estado actual de la seguridad orientada a los microservicios. Dicho mapeo demostró que las métricas y los patrones hoy en día, presentan inconvenientes a la hora de garantizar la seguridad del microservicio debido a que éstos aún son tecnologías relativamente nuevas.

Por otra parte, en el trabajo “Selección y adaptación de métricas para Microservicios” [11] se menciona que las arquitecturas de software han evolucionado, haciendo énfasis en las arquitecturas monolíticas orientadas a servicios, las cuales cada vez son más utilizadas para hacer la transición hacia arquitecturas de microservicios con el fin de proporcionar a los desarrolladores y arquitectos indicaciones claras de las deficiencias y errores de los microservicios.

En el trabajo “Metodología en la determinación de la granularidad de un Microservicio” [12] se presentan 4 formas distintas para determinar el tamaño adecuado o granularidad de un microservicio, como lo son: modelo de dominio, casos de uso, diseño de la base de datos y la declaración de objetivos.

Por último, en “Modelo de calidad para determinar la granularidad microservicios” [13] presentan un modelo de calidad para determinar la granularidad de microservicios mediante un balance de las métricas cohesión y complejidad.

1.2. Planteamiento del problema

Las métricas son uno de los elementos esenciales para la evaluación de la calidad de software y su importancia ha ido al alza ante la adopción de nuevos paradigmas de desarrollo. Bajo este contexto, las arquitecturas de microservicios se están consolidando como un paradigma objetivo dentro de la evolución de los sistemas de software, ya sea como resultado de procesos de migración para arquitecturas monolíticas o para el desarrollo de nuevas aplicaciones. Sin embargo, es necesario un estudio que permita comprender el estado actual de las métricas y su aplicación, ya que algunas métricas específicas para ciertos atributos de calidad aún no están completamente definidas. Además, muchas métricas adaptadas de otras arquitecturas carecen de un estudio, por lo tanto, no se encuentran fundamentadas de manera formal o empírica [1], [7], [8], [14]. Esta falta de claridad dificulta la evaluación de la calidad

en sistemas basados en microservicios ya que no siempre se conoce qué métricas son las más adecuadas ni cómo deben medirse los microservicios.

1.3. Objetivos

1.3.1. Objetivo General

Analizar las métricas propuestas en la literatura que definen algunas de las propiedades de los microservicios para su diseño e implementación.

1.3.2. Objetivos Específicos

- Obtener un listado de métricas identificadas en la literatura, las cuales son clasificadas según su propósito.
- Conocer los umbrales de referencia pertenecientes a las métricas con el fin de identificar estándares comunes y áreas de mejora.
- Desarrollar un modelo de calidad para proponer al menos un atributo de calidad para microservicios.

1.4. Alcances y Limitaciones

1.4.1. Alcances

- ✓ El modelo contiene las principales métricas sustentadas y no sustentadas en la literatura.
- ✓ Las métricas analizadas fueron las disponibles en la literatura accesible
- ✓ Se usaron herramientas de libre acceso.

1.4.2. Limitaciones

- No se probaron todas las métricas encontradas.
- Las métricas fueron probadas en tres proyectos de repositorios de microservicios desarrollados en Java.
- No se modificó ningún proyecto evaluado.

1.5. Estructura del documento

El presente documento se encuentra organizado de la siguiente manera, el Capítulo 2 presenta el marco conceptual con los conceptos que se manejarán a partir de este punto en el

documento; el Capítulo 3 presenta la recopilación de los trabajos que tienen una relación con este trabajo de tesis, así como su análisis; en el Capítulo 4 se presenta el desarrollo de la solución propuesta, en este caso, el desarrollo del modelo de calidad propuesto; en el Capítulo 5 se muestran los resultados obtenidos tras aplicar el modelo de calidad; en el Capítulo 6 se presentan las conclusiones obtenidas tras la realización del presente trabajo, así como sugerencias de posibles trabajos futuros, por último, en el Capítulo 7 se presentan las publicaciones y actividades complementarias que fueron realizadas durante el periodo de maestría.

Capítulo 2 Marco Conceptual



Este capítulo presenta los conceptos principales y fundamentos que serán abordados a lo largo de la presente investigación.

2.1. Arquitectura de Microservicios

Establece construir una aplicación como un conjunto de servicios, donde cada uno de estos es independiente del otro y hasta pueden ser escritos en lenguajes diferentes y mantenidos por equipos diferentes. Lo anterior permite que el problema de escalar se solucione incrementando el procesamiento necesario para el servicio específico que lo esté requiriendo y no de otros que no lo necesitan [15].

2.2. Microservicio

“Son un enfoque para desarrollar una sola aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, a menudo una API de recurso HTTP. Estos servicios están contruidos alrededor de requisitos comerciales y son desplegables de manera independiente por maquinaria de despliegue completamente automatizada” [16].

2.3. Métrica

“Es una medida o conjunto de medidas destinadas a evaluar o estimar el tamaño o características de un software o sistema, comúnmente utilizada para la planificación y comparación en proyectos de desarrollo” [17].

2.4. Atributo de calidad de software

Requisito que especifica el grado de un atributo que afecta la calidad que debe poseer el sistema o software [18].

2.5. Flexibilidad

Capacidad del producto para adaptarse a cambios en sus requisitos, contextos de uso o entorno del sistema [19].

2.5.1. Adaptabilidad

Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso [19].

2.5.2. Escalabilidad

Capacidad del producto para gestionar cargas de trabajo crecientes o decrecientes y para adaptar su capacidad a la variabilidad [19].

2.5.3. Instalabilidad

Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno [19].

2.5.4. Reemplazabilidad

Capacidad del producto para ser utilizado en lugar de otro producto de software determinado con el mismo propósito y en el mismo entorno [19].

2.6. Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados sin interrupciones o fallos [19].

2.7. Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas [19].

2.8. Eficiencia de desempeño

Representa el desempeño de un producto en la realización de sus funciones dentro de unos parámetros de tiempo y rendimiento especificados y con un uso eficiente de recursos (CPU, memoria, almacenamiento, energía, etc.) utilizados bajo determinadas condiciones [19].

2.9. Seguridad

Capacidad de protección de la información y los datos de manera que las personas u otros productos tengan el grado de acceso a los datos adecuado a sus tipos y niveles de autorización, y para defenderse de los patrones de ataque de agentes malintencionados [19].

2.10. Compatibilidad

Capacidad de un producto de intercambiar información con otros productos y/o llevar a cabo sus funciones requeridas cuando comparten un mismo entorno y recursos [19].

2.11. Portabilidad

Conjunto de atributos relacionados con la capacidad de un sistema de software para ser transferido y adaptado desde una plataforma a otra [20].

2.12. Modelo de calidad

Es un marco de características y subcaracterísticas que definen las propiedades de un producto software para evaluarlo y describir su calidad en diferentes aspectos, como su funcionalidad, rendimiento y usabilidad, tanto en su uso en contexto como en su calidad intrínseca [19].

2.13. Umbral

“Un valor para una métrica que divide el conjunto de entidades de software en dos grupos; uno con valores de la métrica menores o iguales que el umbral y otro con valores mayores al mismo.”[21].

2.14. Endpoint

Es un punto en el que una interfaz de programación de aplicaciones permite que dos programas de software se comuniquen. [22]

2.15. Arquitectura de monolitos

Aplicación de software cuyos módulos no pueden ejecutarse de forma independiente y dependen de recursos compartidos en una misma máquina. [23]

2.16. API REST

Es una interfaz de programación de aplicaciones que sigue los principios de REST (Representational State Transfer). Es una forma de comunicación entre sistemas de software que se basa en el protocolo HTTP y los métodos de consulta HTTP (GET, POST, PUT, DELETE).[24]

2.17. Kubernetes

Es una plataforma de orquestación de contenedores de código abierto diseñada para programar y automatizar el despliegue, la gestión y el escalamiento de aplicaciones en contenedores. [25]

2.18. Docker

Es una plataforma de software que permite empaquetar software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y versión ejecutable [26].

2.19. Calidad de un producto de software

Se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor [19].

2.20. Arquitectura Orientada a Servicios

Es una colección de muchos servicios en la red; estos servicios se comunican entre sí y también intercambian datos [6].

Se refiere a aplicaciones que exponen y consumen funcionalidades como un servicio usando contratos y mensajes [27].

2.21. Arquitectura Orientada a Objetos

Un sistema orientado a objetos se constituye con objetos que interactúan y mantienen su propio estado local y ofrecen operaciones sobre dicho estado [27].

2.22. Principios IDEALS

Los principios IDEALS constituyen un conjunto de principios de diseño arquitectónico orientados a la construcción de microservicios debidamente estructurados. Este acrónimo agrupa seis principios que buscan garantizar independencia, adecuación funcional y escalabilidad en sistemas distribuidos como los microservicios [28].

2.22.1. Segregación de interfaces

Cada servicio debe exponer interfaces claras y específicas para sus consumidores y asegurar que sus diferentes clientes obtengan solo lo que necesitan [28].

2.22.2. Despliegue autónomo

Un microservicio debe ser capaz de ser desarrollado, construido, probado y desplegado sin depender de otros servicio, de modo que un cambio o actualización en un servicio no requiera volver a desplegar todo el sistema [28].

2.22.3. Basado en eventos

Los microservicio deberán activarse mediante mensajes o eventos asíncronos en lugar de llamadas síncronas directas [28].

2.22.4. Disponibilidad sobre consistencia

La disponibilidad del sistema para responder a las solicitudes de los usuarios suele ser más valiosa que garantizar que todos los datos estén perfectamente consistentes en todo momento [28].

2.22.5. Bajo acoplamiento

Los microservicios deben mantener mínimas dependencias entre sí, de modo que los cambios en uno no obliguen a modificar otros servicios [28]

2.22.6. Única responsabilidad

Cada microservicio debe tener una única responsabilidad o propósito bien definido, centrado en una funcionalidad específica del dominio de negocio [28].

2.23. Medición en ingeniería de software

La medición es el proceso mediante el cual se asignan números o símbolos a los atributos de las entidades del mundo real, de tal manera que se describan según reglas claramente definidas [29]

2.24. Entidad

Es un objeto (como una persona o una habitación) o un evento (como un viaje o la fase de pruebas de un proyecto de software) del mundo real [29].

2.25. Contenerización

La contenerización es el empaquetado de código de software con solo las bibliotecas del sistema operativo y las dependencias necesarias para ejecutar un código y crear un único ejecutable y ligero, llamado contenedor, que se ejecuta de manera constante en cualquier arquitectura [30].

Capítulo 3 Trabajos Relacionados



Este capítulo presenta el análisis de las investigaciones que tienen relación con el estudio de métricas propuestas, adaptadas e implementadas para microservicios.

Los trabajos identificados en esta investigación abordan el estudio de las arquitecturas de microservicios desde enfoques diferentes. Estos trabajos describen características y principios de los microservicios; además, incluyen propuestas de métricas, estudios de migración desde arquitecturas monolíticas y revisiones sistemáticas que contribuyen al estado del arte.

De estos trabajos se analizaron y extrajeron métricas, definiciones y criterios de evaluación que pudieran permitir identificar cómo se puede medir o caracterizar a las arquitecturas de microservicios. Cabe mencionar que varios de estos trabajos pertenecen a más de una categoría, por lo cual la *Tabla 1* tiene como objetivo facilitar el análisis comparativo entre los distintos enfoques.

Tabla 1 Clasificación de los trabajos relacionados sobre métricas para microservicios

Artículo	Enfoque principal	Menciona características	Utiliza métricas	Aborda migración
<i>A method for monitoring the coupling evolution of microservice based architectures</i> [7]	Métricas	Si	Si	No
<i>A Metrics Framework for Evaluating Microservices Architecture Designs</i> [1]	Métricas	Si	Si	Parcialmente
<i>A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures</i> [31]	Métricas	No	Si	No
<i>A metrics suite for object-oriented design</i> [32]	Métricas	No	Si	No
<i>A New Decomposition Method for Designing Microservices</i> [33]	Migración	Si	Si	Si
<i>A quantitative assessment method for microservices granularity to improve maintainability</i> [34]	Métricas	Si	Si	Parcialmente
<i>A Survey on Service Oriented Architecture and Metrics to measure coupling</i> [6]	Métricas	No	Si	No

(Continuación) Tabla 1 Clasificación de los trabajos relacionados sobre métricas para microservicios

Artículo	Enfoque principal	Menciona características	Utiliza métricas	Aborda migración
<i>Assessing Architecture Conformance to Coupling Related Patterns and Practices in Microservices</i> [35]	Métricas	Si	Si	No
<i>Automatically Measuring the Maintainability of Service and Microservice based Systems: a literature review</i> [36]	Métricas	Si	Si	No
<i>Challenges and Solution Directions of Microservice SLR</i> [37]	Revisión Sistemática	Si	Si	Si
<i>Chidamber and Kemerers metrics suite: A Measurement Theory Perspective</i> [4]	Métricas	No	Si	No
<i>Cohesion Driven Decomposition of Service Interfaces Without Access to Source Code</i> [38]	Métricas	No	Si	No
<i>Cohesion Metrics for Predicting Maintainability of service-oriented software</i> [39]	Métricas	No	Si	No
<i>Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review</i> [8]	Revisión Sistemática	Si	Si	Si
<i>Defining and measuring microservice granularity a literature overview</i> [40]	Revisión Sistemática	Si	Si	Si
<i>Evaluation of Microservices Architectures A Metric and Tool Based Approach</i> [41]	Métricas	Si	Si	No
<i>Evaluation of SOA Based Web Services and Microservices Architecture Using Complexity Metrics</i> [14]	Métricas	Si	Si	Si
<i>Extracting Microservices Candidates from Monolithic Applications Interface Analysis and Evaluation Metrics Approach</i> [42]	Migración	Si	Si	Si
<i>From Monolith to Microservice: Measuring Architecture Maintainability</i> [43]	Migración	Si	Si	Si

(Continuación) Tabla 1 Clasificación de los trabajos relacionados sobre métricas para microservicios

Artículo	Enfoque principal	Menciona características	Utiliza métricas	Aborda migración
<i>From Monolithic Systems to Microservices A Decomposition Framework based on Process Mining</i> [44]	Migración	Si	Si	Si
<i>Functionality Oriented Microservice Extraction Based on Execution Trace Clustering</i> [45]	Migración	Si	Si	Si
<i>Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect on Government Data Sharing</i> [46]	Migración	Si	Si	Si
<i>Improving Observability in event sourcing Systems</i> [3]	Métricas	Si	Si	No
<i>Microservice Fault Root Cause Location based on Correlation Analysis</i> [47]	Métricas	Si	Si	No
<i>Microservice Security Metrics for Secure Communication, Identity Management, and Observability</i> [48]	Métricas	Si	Si	No
<i>Microservices Identification Methods and quality metrics</i> [49]	Métricas	Si	Si	Si
<i>Microservices Monitoring with Event Logs and Black Box Execution Tracing</i> [50]	Métricas	Si	Si	No
<i>Microservices yesterday, today, and tomorrow</i> [23]	Revisión Sistemática	Si	No	Si
<i>MicroValid A Validation Framework for auto decomposed Microservices</i> [51]	Métricas	Si	Si	Si
<i>Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios</i> [52]	Métricas	Si	Si	Si
<i>Nonintrusive Monitoring of Microservice based Systems</i> [53]	Métricas	Si	Si	Parcialmente
<i>Osmotic Monitoring of Microservices between the Edge and Cloud</i> [54]	Métricas	Si	Si	No

(Continuación) Tabla 1 Clasificación de los trabajos relacionados sobre métricas para microservicios

<i>Artículo</i>	<i>Enfoque principal</i>	<i>Menciona características</i>	<i>Utiliza métricas</i>	<i>Aborda migración</i>
<i>Securing microservices and microservice architectures: a systematic mapping study</i> [55]	Revisión Sistemática	Si	Si	No
<i>Software Architectural Metrics for the Scania Internet of Things Platform From a Microservice Perspective</i> [56]	Métricas	Si	Si	No
<i>Structural Coupling for Microservices</i> [57]	Métricas	Si	Si	Si
<i>The pains and gains of microservices A Systematic grey literature</i> [58]	Revisión Sistemática	Si	Si	Si
<i>Understanding and Addressing Quality Attributes of Microservices SLR</i> [2]	Revisión Sistemática	Si	Si	Si
<i>Using Cohesion and Coupling for Software Remodularization: is it enough?</i> [59]	Métricas	No	Si	No
<i>Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems</i> [60]	Métricas	No	Si	No

3.1. Trabajos que mencionan características de los microservicios

Existen diversos trabajos que se enfocan en describir y analizar las características fundamentales de las arquitecturas de microservicios, tales como bajo acoplamiento, alta cohesión y granularidad.

Algunos de estos trabajos [23], [41], [52] presentan características desde una perspectiva conceptual y de diseño, destacando sus beneficios frente a arquitecturas monolíticas. Estos artículos están enfocados en analizar dichas características, las cuales pueden mejorar atributos como la flexibilidad y mantenibilidad, aunque no siempre proponen una forma para llevar a cabo dicha evaluación.

Asimismo trabajos como [14] hacen una validación empírica de dichas características mediante el uso de métricas formales, con el objetivo de mostrar que los microservicios pueden reducir la complejidad del sistema en comparación con otras arquitecturas.

3.2. Trabajos que utilizan métricas en microservicios

Para estos trabajos se identificaron dos tipos de enfoques, por un lado se tienen algunos trabajos que proponen métricas específicas y adaptadas de arquitecturas orientadas a servicios, logrando enfocarse principalmente en atributos como granularidad, acoplamiento y mantenibilidad [1], [7], y por otro lado se tienen trabajos que utilizan métricas como base para evaluar el grado de conformidad arquitectónica o el cumplimiento de principios de diseño, por lo que proponer nuevas métricas ha resultado bastante necesario para capturar las nuevas demandas de este estilo arquitectónico [35].

3.3. Trabajos respecto migración a microservicios

Los trabajos relacionados con la migración de arquitecturas orientadas a servicios u objetos, entre otras, hacia las de microservicios principalmente se centran en evaluar la calidad de la arquitectura objetivo antes o durante la migración. Por lo que algunos trabajos como [43] utilizan conjuntos de métricas estructurales con el fin de facilitar la toma de decisiones importantes en una etapa temprana del proceso, mientras que otros proponen marcos de descomposición los cuales están basados en datos y el comportamiento de la aplicación en tiempo de ejecución para poder hacer más fácil identificar microservicios [61].

Partiendo del análisis de los trabajos relacionados es posible observar que aunque existe una amplia variedad de propuestas para evaluar la calidad en arquitecturas de microservicios, todavía no se ha alcanzado un consenso consolidado sobre el enfoque más apropiado, ya que normalmente las métricas no están alineadas con los atributos de calidad que deben contemplarse al diseñar una solución con microservicios. En el **Anexo C** se describe un breve resumen de los trabajos listados en la *Tabla 1*.

.

Capítulo 4 Desarrollo de la solución



En este capítulo se presenta el análisis de los trabajos identificados durante el desarrollo de la investigación. Además, se detalla el concepto de “microservicio” que será utilizado para este trabajo de investigación, junto con sus características y los atributos de calidad que son descritos en la literatura especializada para la presentación del modelo de calidad planteado en los objetivos.

4.1. Planeación

Con el fin de proporcionar una visión estructurada del proceso metodológico que se siguió a lo largo de esta investigación, la *Figura 1* presenta el diagrama de flujo que describe las principales etapas desarrolladas.

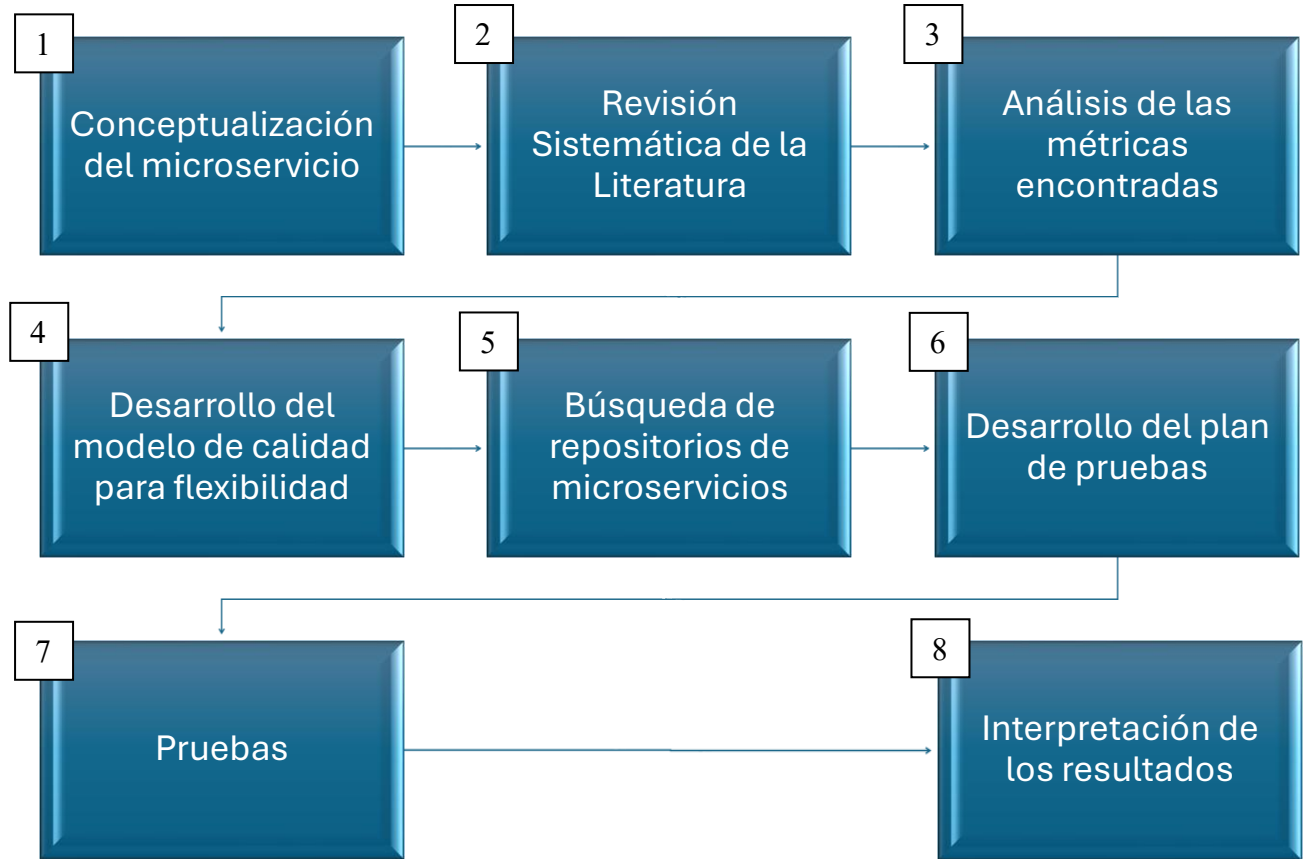


Figura 1 Diagrama de flujo del proceso de investigación

A continuación se detalla cada uno de los pasos presentados en la *Figura 1*:

1. Se llevó a cabo la conceptualización del microservicio como objeto de estudio, mediante el análisis de sus características, su forma de desarrollo y los principios que orientan su diseño arquitectónico.
2. Una vez conceptualizado el microservicio, se realizó una revisión sistemática de la literatura con los pasos que se detallan en el **Anexo E**; la cual fue orientada a la identificación de trabajos que presenten métricas y conceptos asociados a la arquitectura de microservicios.

3. Posteriormente, las métricas encontradas fueron analizadas con el fin de establecer su relación con los atributos de calidad considerados en la ISO/IEC 25010, realizar un proceso de cuantificación y clasificar los umbrales reportados en la literatura y por último se llegó a una discusión de los hallazgos encontrados tras revisar las fuentes.
4. Partiendo del análisis anterior, se desarrolló un modelo de calidad para el atributo de flexibilidad, el cual constituye la base para los próximos pasos.
5. Se llevó a cabo la identificación y selección de repositorios de microservicios con base en criterios de inclusión y exclusión.
6. Se desarrolló un plan de pruebas que fuese capaz de aplicar el modelo y la ejecución de las pruebas de manera sistemática para garantizar el uso adecuado del modelo de calidad propuesto.
7. Finalmente, los resultados obtenidos tras aplicar el modelo de calidad fueron analizados e interpretados con el propósito de identificar patrones de comportamiento, brechas respecto a los atributos analizados y oportunidades de mejora en las implementaciones evaluadas.

4.2. Conceptualización de los microservicios

Para contextualizar el estudio, se retoma la definición propuesta por Martin Fowler, quien describe a los microservicios como “un enfoque para desarrollar una única aplicación como un conjunto de pequeños servicios, cada uno de los cuales se ejecuta en su propio proceso y se comunica mediante mecanismos ligeros, a menudo una API de recursos HTTP. Estos servicios se crean en torno a capacidades empresariales y se pueden implementar de forma independiente mediante un mecanismo totalmente automatizado. La gestión centralizada de estos servicios es mínima, y pueden estar escritos en diferentes lenguajes de programación y utilizar diferentes tecnologías de almacenamiento de datos” [16]. Con base en esta definición, se establece el marco conceptual que apoya el análisis de calidad para arquitecturas basadas en microservicios. Asimismo Fowler identifica un conjunto de características esenciales que deben presentarse en este tipo de arquitectura, así como las razones que justifican su relevancia, las cuales se presentan en la *Tabla 2*.

Tabla 2 Características de los microservicios según Fowler y Lewis.

<i>Característica</i>	<i>¿Qué significa?</i>	<i>¿Por qué es importante?</i>
<i>Componentización mediante servicios</i>	Dividir la aplicación en servicios independientes que corren en sus propios procesos, en vez de módulos internos de un solo programa.	Permite que cada servicio se despliegue, escale, mantenga y evolucione por separado.
<i>Organizado alrededor de capacidades de negocio</i>	En vez de organizarse por capas técnicas, los servicios se organizan por dominios de negocio.	Mejor alineación con lo que el negocio realmente necesita. Permite que los equipos sean más autónomos, reduzcan dependencias entre ellos y reaccionen más rápido al cambio.
<i>Productos, no proyectos</i>	Un equipo de desarrollo “posee” un microservicio como producto vivo, no solo como un proyecto temporal que termina al entregarlo. El producto se mantiene, opera y evoluciona continuamente.	Fomenta responsabilidad sobre la calidad, operación, rendimiento en producción. Mejora la estabilidad, la mejora continua y la satisfacción con el resultado final.
<i>Endpoints inteligentes y canales simples</i>	Las comunicaciones entre servicios se mantienen lo más simple posible (“Canales simples”), mientras que la lógica reside en los endpoints/servicios. Se evitan embudos con lógica centralizada que complican o ralentizan el sistema.	Ayuda a mantener cada servicio independiente, reduciendo complejidad en los canales de comunicación, permitiendo que los servicios evolucionen más libremente y se desacoplen.
<i>Gobernanza descentralizada</i>	No forzar un único stack tecnológico rígido para todos los servicios. Permitir que los servicios usen diferentes lenguajes, frameworks, bases de datos si conviene.	Permite usar la mejor tecnología para cada problema. Mejora la adaptabilidad, innovación, también motivación de los equipos. Pero requiere disciplina para evitar caos.
<i>Gestión descentralizada de datos</i>	Cada servicio gestiona su propio almacenamiento de datos, en vez de tener una base de datos central común para todo el sistema.	Reduce el acoplamiento por datos, permite que cada servicio elija la tecnología de almacenamiento más adecuada, y mejora escalabilidad
<i>Automatización de la infraestructura</i>	Usa integración continua (CI) y entrega continua (CD), automatización de despliegues, pruebas automáticas, entornos automatizados, herramientas que soportan despliegues y operación.	Hace que desplegar frecuentes cambios sea menos riesgoso, más confiable y ágil. Reduce errores manuales, permite reaccionar rápido ante fallas o nuevas necesidades.
<i>Diseñar para el fallo</i>	Anticipar qué servicios individuales fallarán: Diseñar tolerancia a fallos, degradaciones, detectar fallos rápidamente.	Al distribuir el sistema, las fallas de uno pueden afectar a otros. Sin diseño para fallo, el sistema puede volverse frágil. Diseñar para fallo mejora disponibilidad y resiliencia.
<i>Diseño evolutivo</i>	Hay que reconocer que los límites de los servicios pueden no estar bien al inicio y que los requisitos cambiarán. Tener la capacidad de refactorizar, reubicar responsabilidad entre servicios, incluso descartar servicios si dejan de tener sentido.	Permite que la arquitectura crezca con el tiempo y se ajuste conforme cambian los requisitos, en vez de tratar de “predecirlo todo” de antemano y quedar atrapado.

Teniendo en cuenta que las características presentadas en la *Tabla 2* se aplican a la arquitectura de los microservicios, se optó por considerar únicamente las características que

puedan ser aplicables de manera individual a cada uno de los microservicios que componen dicha arquitectura.

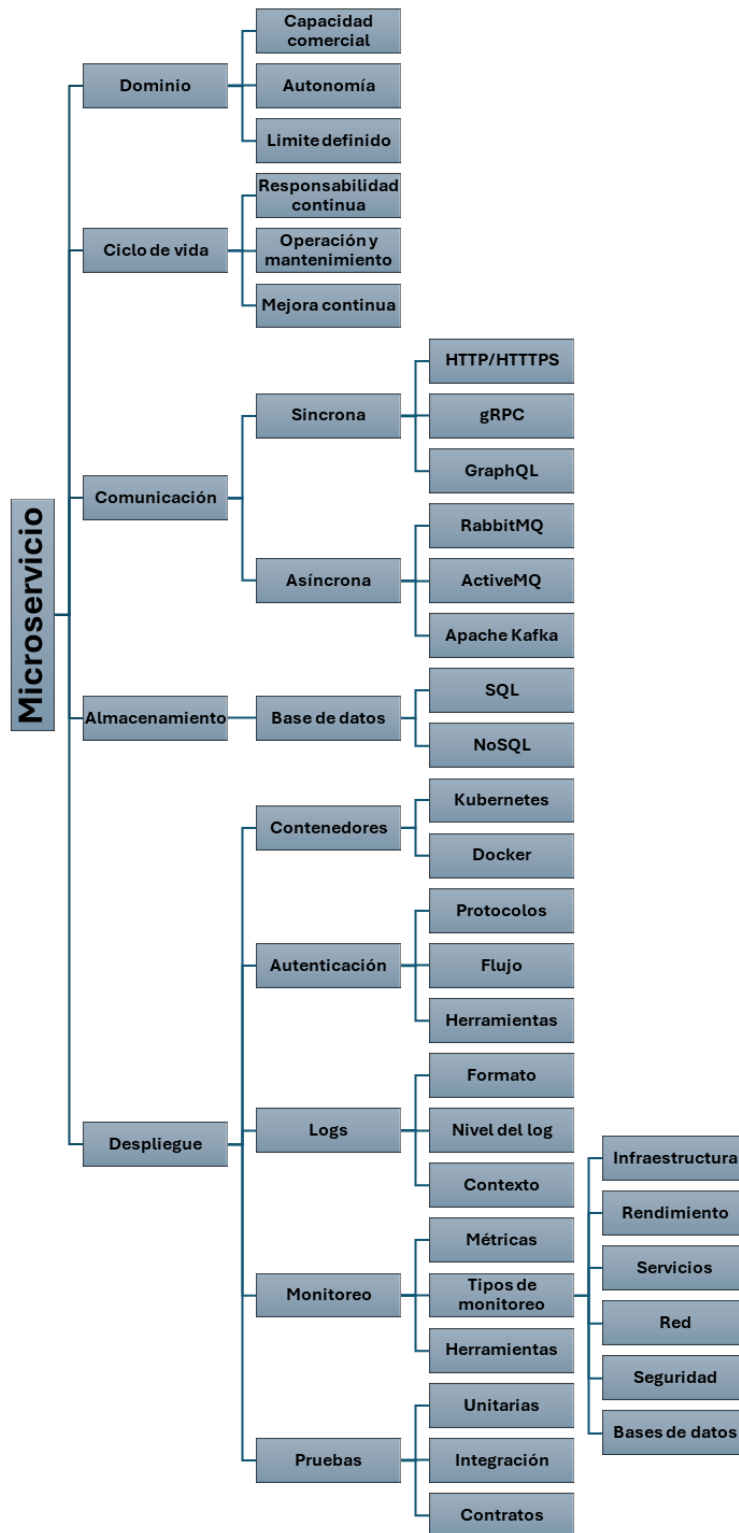


Figura 2 Esquema de la definición de microservicio.

Como resultado se presenta la *Figura 2*, el cual presenta un esquema que contiene algunos de los atributos que pueden ser aplicables a un microservicio. Sin embargo, es importante mencionar que en la literatura se contemplan diferentes definiciones para dicho término, por lo tanto, existe una gran variedad de atributos de calidad para cada uno de ellos, es por eso que para este trabajo se consideraron solo los atributos de calidad que menciona la norma ISO/IEC 25010 [19], la cual, menciona nueve atributos de calidad, los cuales son presentados en la *Figura 3*, sin embargo, con base en el análisis realizado a los trabajos relacionados y la propia definición propuesta por Fowler, únicamente se consideraron los seis atributos enmarcados en color azul en dicha figura, dado que la ISO 25010 se orienta a la evaluación de calidad para productos de software en términos generales, sin embargo, un microservicio consta de características específicas que pueden estar o no presentes en otras arquitecturas de software. Es por esta razón que podemos discernir de algunos de los atributos definidos por la norma, y así darle prioridad a aquellos que se alinean con la definición y principios de un microservicio.

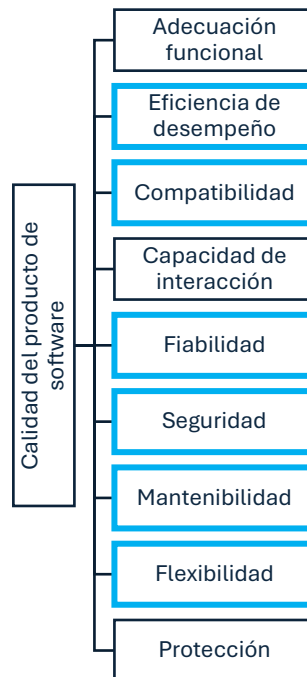


Figura 3 Modelo de calidad según la ISO/IEC 25010.

Asimismo, Paul Merson propone principios de diseño que denomina IDEALS para la arquitectura de microservicios [28], la *Tabla 3* presenta la relación entre dichos principios y la norma ISO 25010.

Tabla 3 Relación de atributos de un microservicio con los principios IDEALS.

<i>Principios IDEALS</i>	Atributo de Calidad según la ISO 25010	¿Cómo se solventa?
<i>Segregación de interfaces</i>	Escalabilidad	Interfaces minimalistas permiten escalar componentes individualmente sin afectar a otros.
	Rendimiento	Reduce sobrecarga al exponer solo operaciones necesarias
	Capacidad de prueba	Interfaces pequeñas y cohesionadas facilitan pruebas unitarias e integración.
<i>Despliegue</i>	Disponibilidad	Despliegues independientes evitan tiempo de caída en otros servicios
	Monitoreabilidad	Facilita la inclusión de métricas y logs específicos por servicio al desplegar.
	Escalabilidad	Permite escalar instancias de microservicios de forma dinámica (ej.: Contenedores)
<i>Basado en eventos</i>	Rendimiento	Procesamiento asíncrono evita bloqueos y mejora eficiencia.
	Escalabilidad	Escala de consumidores de eventos de forma independiente
	Disponibilidad	Colas de mensajes aíslan fallos y garantizan resiliencia.
<i>Disponibilidad sobre consistencia</i>	Escalabilidad	Orquestadores (Kubernetes) distribuyen carga automáticamente en nodos disponibles
	Disponibilidad	Permite reiniciar y reubicar servicios caídos sin intervención manual.
	Monitoreabilidad	Integra con herramientas para observar el estado de los pods.
<i>Acoplamiento débil</i>	Seguridad	Menos dependencias reducen la superficie de ataque entre servicios.
	Capacidad de prueba	Pruebas asiladas con stubs/mocks, sin depender de servicios externos.
	Disponibilidad	Evita fallos en cascada al desacoplar servicios.
<i>Responsabilidad única</i>	Rendimiento	Servicios especializados consumen solo los recursos necesarios.
	Monitoreabilidad	Métricas por función específica ayudan a identificar cuellos de botella fácilmente
	Capacidad de prueba	Menor complejidad por servicio simplifica las pruebas unitarias.

Los principios SOLID, surgidos del paradigma orientado a objetos (POO), pueden ser utilizados en microservicios, pero con matices ya que mientras el POO gira en torno a clases y herencias, los microservicios son servicios distribuidos que priorizan la autonomía y la comunicación remota. Por ello, aplicar SOLID requiere reinterpretar sus ideas para un entorno distribuido [28]. Es por esta razón que Merson propone los principios IDEALS, los cuales se presentan en la *Figura 4*. Estos principios, aunque son una propuesta para microservicios no cubren en su totalidad todo el alcance de las decisiones de diseño, sin embargo, abordan los fundamentos clave y los factores de éxito para crear un microservicio [28].



Figura 4 Principios IDEALS para microservicios [28].

Al analizar el concepto de microservicio, así como los principios que orientan su diseño y desarrollo, se facilita una comprensión más integral de su naturaleza y de los elementos que influyen en su calidad. Este análisis permite establecer de manera más clara la relación entre las características propias de los microservicios y los atributos de calidad definidos por la norma ISO/IEC 25010.

Dicha relación, entre los atributos de ambos enfoques, se ilustra de forma gráfica en la *Figura 5*, donde se representa el mapeo entre los atributos mencionados en la literatura para microservicios y los atributos de calidad de la ISO 25010.

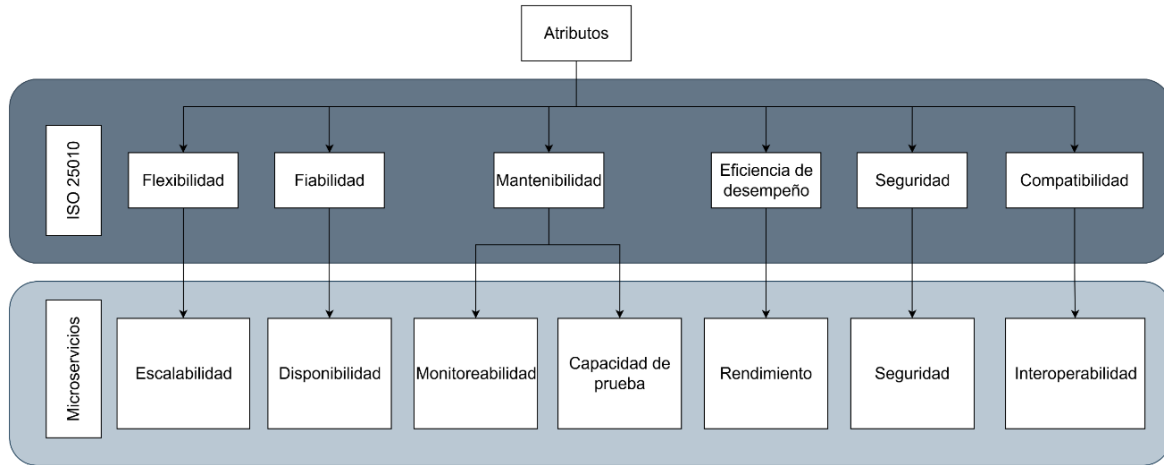


Figura 5 Relación entre atributos de calidad según la ISO 25010 y microservicios.

4.3. Información de métricas para microservicios

Una vez definido el concepto de microservicio y considerando las características que lo definen, se realizó un análisis de los trabajos identificados durante el desarrollo de la investigación, donde se encontraron un total de 258 métricas, algunas del paradigma orientado a objetos, otras de las arquitecturas orientadas a servicios y principalmente aplicables a microservicios, destacando 35 métricas recopiladas. Sin embargo, es importante resaltar que, aunque solo se presentan 35 métricas, muchas de estas presentan diversas formas de evaluar a los microservicios, como en el caso de métricas como Cohesión y Acoplamiento. Asimismo, métricas como tamaño, tiempo y seguridad presentan diversas fórmulas y características que se evalúan desde diferentes perspectivas, por lo que son consideradas métricas diferentes entre sí.

4.3.1. Relación de métricas para microservicios con atributos de calidad según la ISO 25010

La *Tabla 4* presenta una relación entre las métricas encontradas en la literatura y los atributos de calidad según la ISO/IEC 25010. En los **Anexos A y B** se describen con mayor detalle las distintas formas de evaluación que se reportan en la literatura para las métricas, así como las agrupaciones que se consideraron para el presente trabajo.

Tabla 4 Relación de métricas con los atributos de calidad para microservicios.

Métricas para microservicios	Atributos de calidad según la ISO/IEC 25010					
	Flexibilidad	Fiabilidad	Mantenibilidad	Eficiencia de desempeño	Seguridad	Compatibilidad
Acoplamiento [6], [7], [14], [31], [32], [34], [36], [45], [51], [52], [56], [57], [59], [61], [62]	Si	No	Si	No	No	Si
Coefficiente de Silhouette [33]	Si	No	No	No	No	No
Cohesión [1], [32], [34], [36], [38], [39], [42], [45], [46], [49], [52], [56], [59], [60]	Si	No	Si	No	No	No
Complejidad [32], [42], [52], [56]	No	No	Si	No	No	No
Dependencias [7], [36], [41], [45], [46], [51], [52], [56], [62]	Si	Si	Si	No	No	Si
Diseño del servicio [31], [36], [52]	Si	No	Si	No	No	Si
Disponibilidad [2], [33], [47], [50], [54]	No	Si	No	No	No	No
Factor de partición del sistema [6]	Si	No	No	Si	No	Si
Factor de pureza [6]	Si	No	Si	No	No	No
Factor de peso [57]	Si	No	Si	No	No	No
Granularidad [1], [45], [49], [52], [56]	Si	No	Si	No	No	No
Historias de usuario [52]	Si	No	Si	No	No	No
Importancia [7], [31], [36], [46], [47], [52], [56]	No	Si	Si	No	No	Si
Interacciones con bases de datos compartidas [62]	No	No	Si	No	No	Si
Interacción de servicio a través de un componente intermediario [2], [47], [52], [62]	No	Si	No	Si	No	Si
Interdependencia [52], [56]	No	No	Si	No	No	Si
Latencia [2], [41], [50], [53]	No	Si	No	Si	No	No
Media F [33]	No	Si	No	Si	No	No
Modularización [51], [52], [59]	Si	No	Si	No	No	No

(Continuación) Tabla 4 Relación de métricas con los atributos de calidad para microservicios.

Métricas para microservicios	Atributos de calidad según la ISO/IEC 25010					
	Flexibilidad	Fiabilidad	Mantenibilidad	Eficiencia de desempeño	Seguridad	Compatibilidad
Medida de eficacia de MoJo [59]	Si	No	Si	No	No	No
Monitoreabilidad [47]	No	Si	Si	Si	Si	No
Observabilidad [47], [48], [51], [52], [53]	No	Si	Si	Si	Si	No
Precisión [33], [47]	No	Si	No	Si	No	No
Recall [33]	No	Si	No	Si	No	No
Rendimiento [2], [50], [54]	No	Si	No	Si	No	No
Responsabilidad [33]	Si	Si	No	No	No	No
Respuesta [2], [6], [32], [36]	No	Si	No	Si	No	No
Seguridad [48]	No	No	No	Si	Si	No
Servicios compartidos de forma transitiva [62]	No	No	Si	No	No	Si
Similitud [38], [51], [52], [60]	Si	No	Si	No	No	No
Tamaño [14], [31], [34], [41], [46], [50], [51], [56]	No	No	Si	Si	No	No
Tiempo [2], [3], [31], [50], [52], [53]	No	Si	No	Si	No	No
Tipo de base de datos [62]	Si	No	No	Si	Si	Si
Uso de recursos [2], [54]	No	No	No	Si	No	No
Utilización de la comunicación asíncrona [41], [62]	Si	Si	No	Si	No	Si

La Tabla 4 permite visualizar cada métrica identificada en la literatura y su vinculación con los atributos de calidad seleccionados [19], ofreciendo una vista general del nivel de cobertura que existe para cada uno de los atributos dentro del conjunto de métricas recopiladas

4.3.2. Cuantificación de métricas

En esta sección se presenta el análisis cuantitativo donde se detallan las métricas que pueden ser aplicadas a cada uno de los distintos atributos que caracterizan a los microservicios. Dichas métricas fueron agrupadas y fundamentadas a partir de las referencias recopiladas y organizadas en el **Anexo A**.

4.3.2.1. Flexibilidad

Para el atributo de flexibilidad es posible observar que de las 35 métricas presentadas en la *Tabla 4* solo 16 de ellas podrían ser aplicadas para evaluar este atributo, lo cual constituye un 45.7% de las métricas.

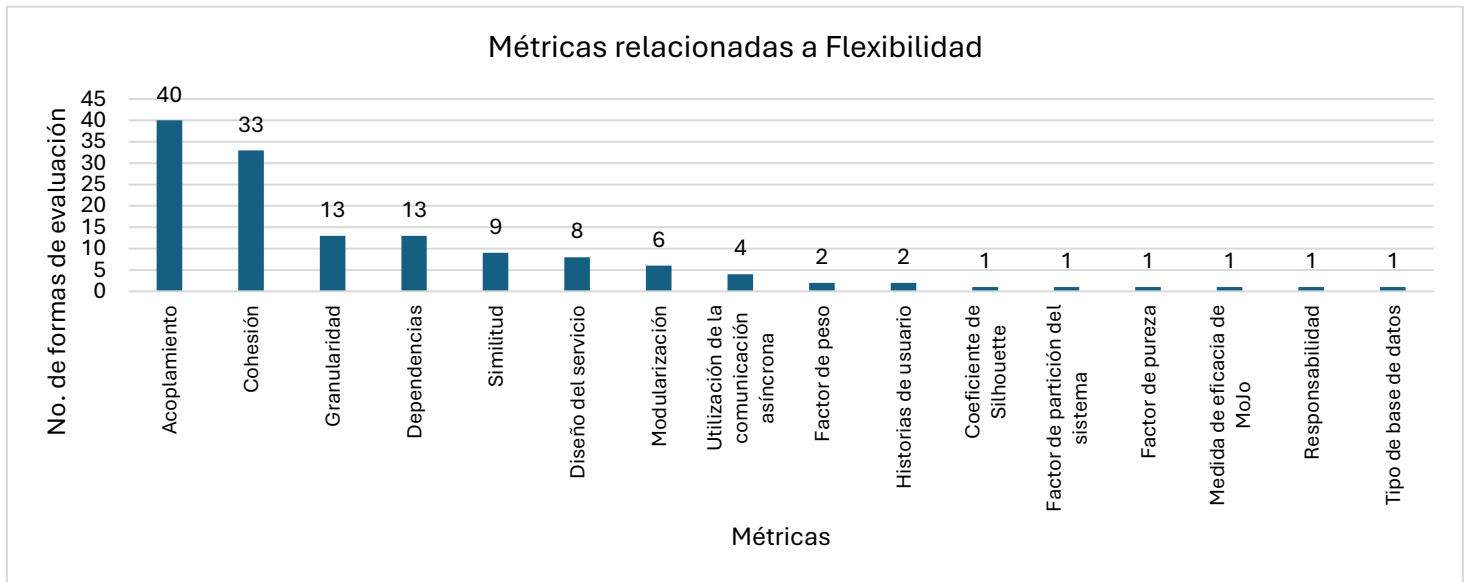


Figura 6 Métricas relacionadas a flexibilidad

La *Figura 6* muestra las métricas que son aplicables al atributo de la flexibilidad, es importante mencionar que éstas, no son iguales entre sí, existen diferentes formas de evaluación relevantes, tanto en su propósito y forma con la que evalúan este atributo.

NOTA: En este trabajo, algunas métricas se denominan con el mismo nombre del atributo de calidad que evalúan (Por ejemplo, acoplamiento o cohesión), debido a que en la literatura no cuentan con una denominación formal estandarizada. Esta situación origina que existan múltiples métricas propuestas para medir un mismo atributo; la nomenclatura es utilizada únicamente con fines de identificación y referencia

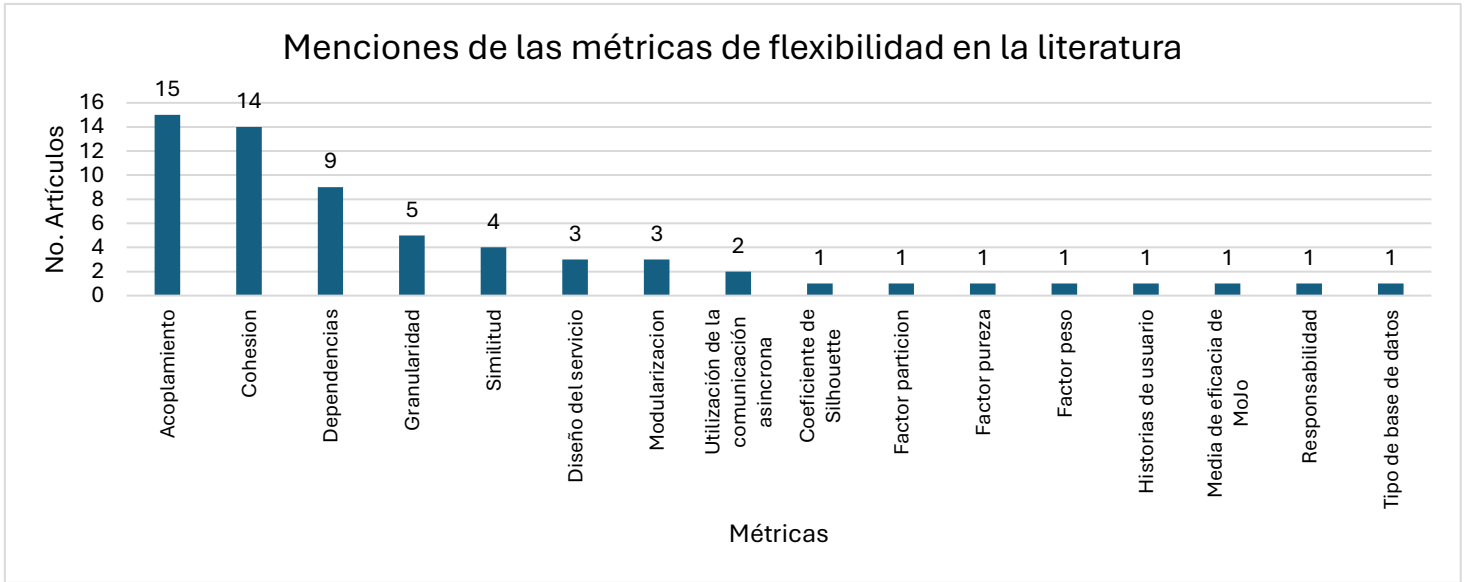


Figura 7 Menciones de las métricas de flexibilidad en la literatura

Como es posible observar en la *Figura 7*, métricas como acoplamiento y cohesión son ampliamente predominantes para el atributo de flexibilidad, lo cual sugiere que la flexibilidad suele ser más abordado desde la perspectiva estructural, ya que la capacidad de modificar o extender un servicio depende fuertemente de su nivel de dependencia con los demás microservicios en la arquitectura.

La *Figura 7* muestra que para el atributo de flexibilidad existen 16 métricas que pueden ser utilizadas para medirla. Sin embargo, la misma literatura en trabajos como [1], [7], [23] evidencia que las MSA ofrecen beneficios como escalabilidad, facilidad de mantenimiento e independencia, los cuales son subatributos contemplados por la ISO 25010 y a menudo pueden atribuirse a una arquitectura flexible, razón por la cual la flexibilidad podría llegar a pasar a segundo plano por los desarrolladores ya que se considera una característica propia de la arquitectura.

4.3.2.2. Fiabilidad

El atributo de fiabilidad cuenta con 15 métricas aplicables, lo que representa el 42.8% del total. Estas métricas presentan entre 1 y 17 formas de evaluar, lo que evidencia una dispersión

significativa en la forma en que la literatura ha definido métodos para cuantificar la fiabilidad del microservicio.

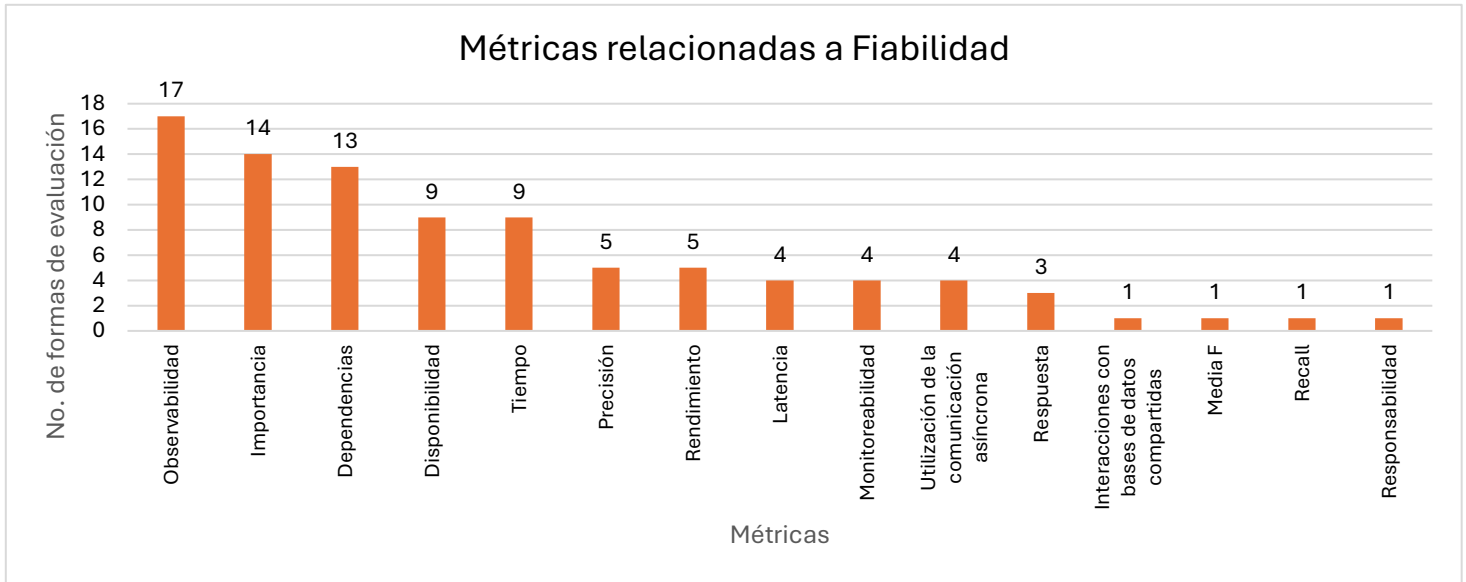


Figura 8 Métricas relacionadas a fiabilidad

Aunque existen 15 grupos de métricas relacionados con la fiabilidad como se presenta en la *Figura 8*, y algunos de ellos presentan un número elevado de formas de evaluar (por ejemplo, Observabilidad con 17 formas distintas de evaluar), la literatura no los menciona de forma uniforme, como lo muestra la *Figura 9*.

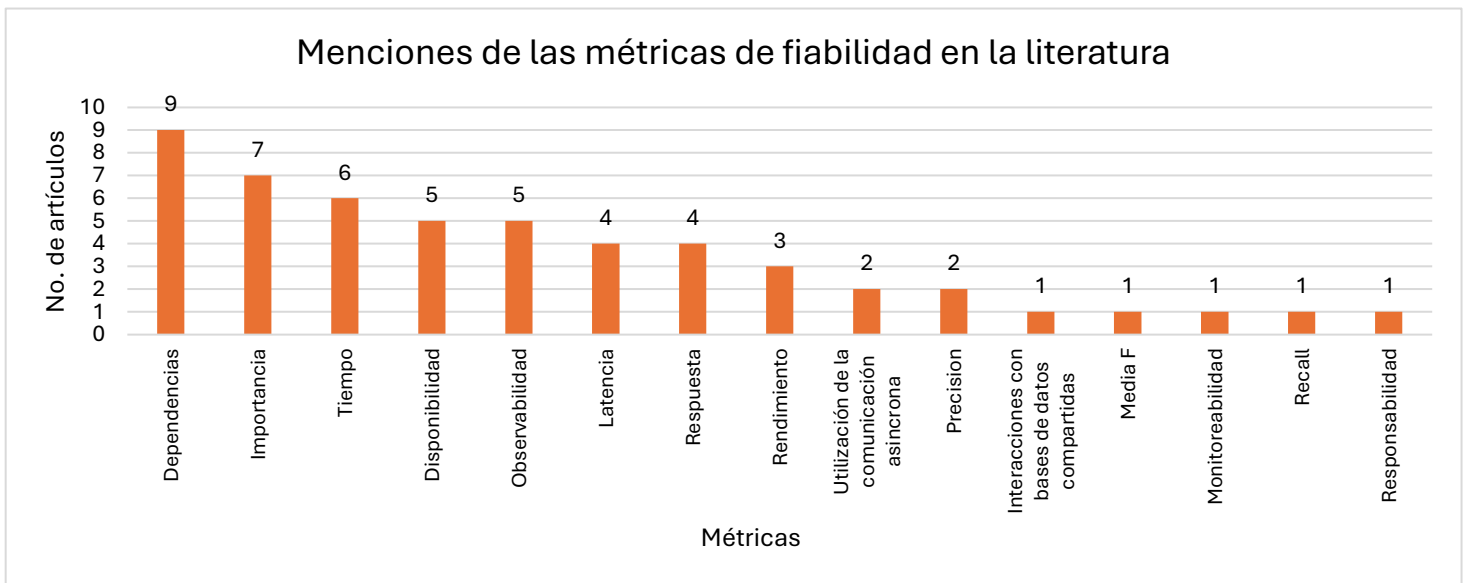


Figura 9 Menciones de las métricas de fiabilidad en la literatura

Los datos muestran que la distribución de menciones es amplia: el rango observado abarca desde 1 hasta 9 artículos para la mayoría de los casos, con algunas excepciones aisladas.

4.3.2.3. Mantenibilidad

El atributo de mantenibilidad es uno de los que presenta mayor cobertura cuantitativa. Un total de 19 métricas, equivalentes al 54.3% del conjunto total, se identificaron como aplicables a este atributo.

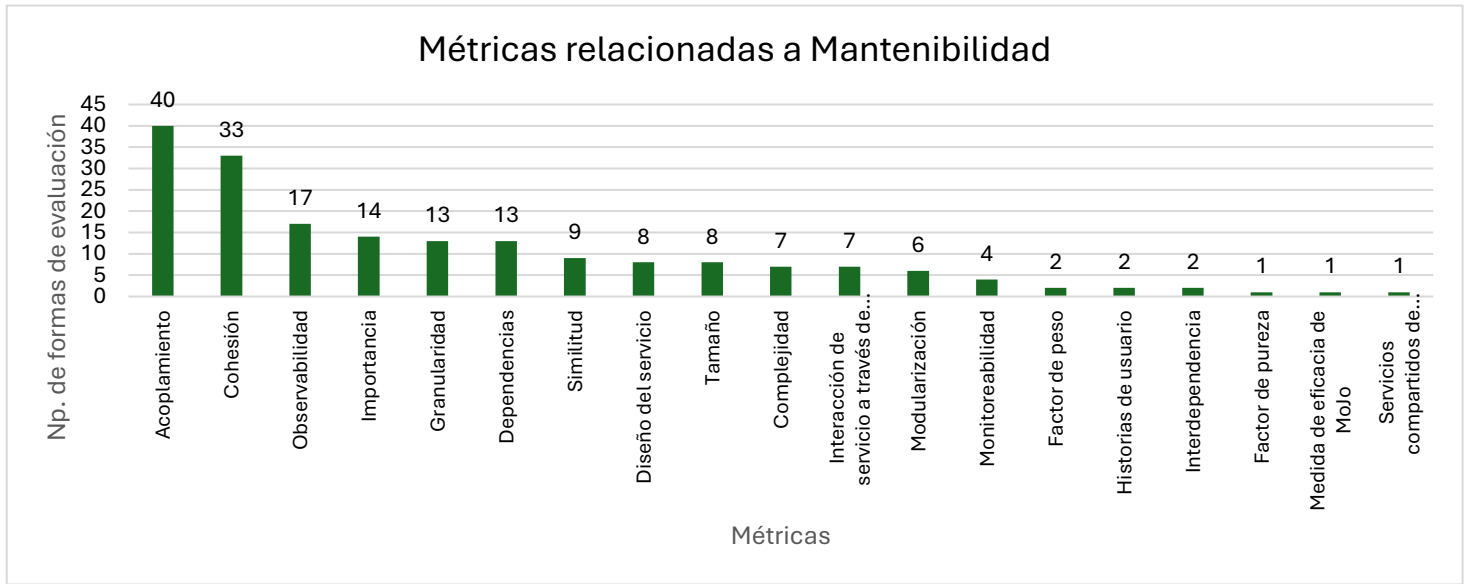


Figura 10 Métricas relacionadas a mantenibilidad

El número de formas de evaluar para cada métrica se distribuye desde 1 hasta 40, como se muestra en la *Tabla 4*, siendo uno de los atributos con mayor diversidad de propuestas. Asimismo, las métricas relacionadas a mantenibilidad suman más de 130 formas, representando una parte significativa del total identificado en la literatura.

Sin embargo, en la literatura no hay muchos trabajos que realmente aborden todas estas métricas, ya que las principales resultan ser acoplamiento, cohesión, etc. Como lo presenta la *Figura 11*.

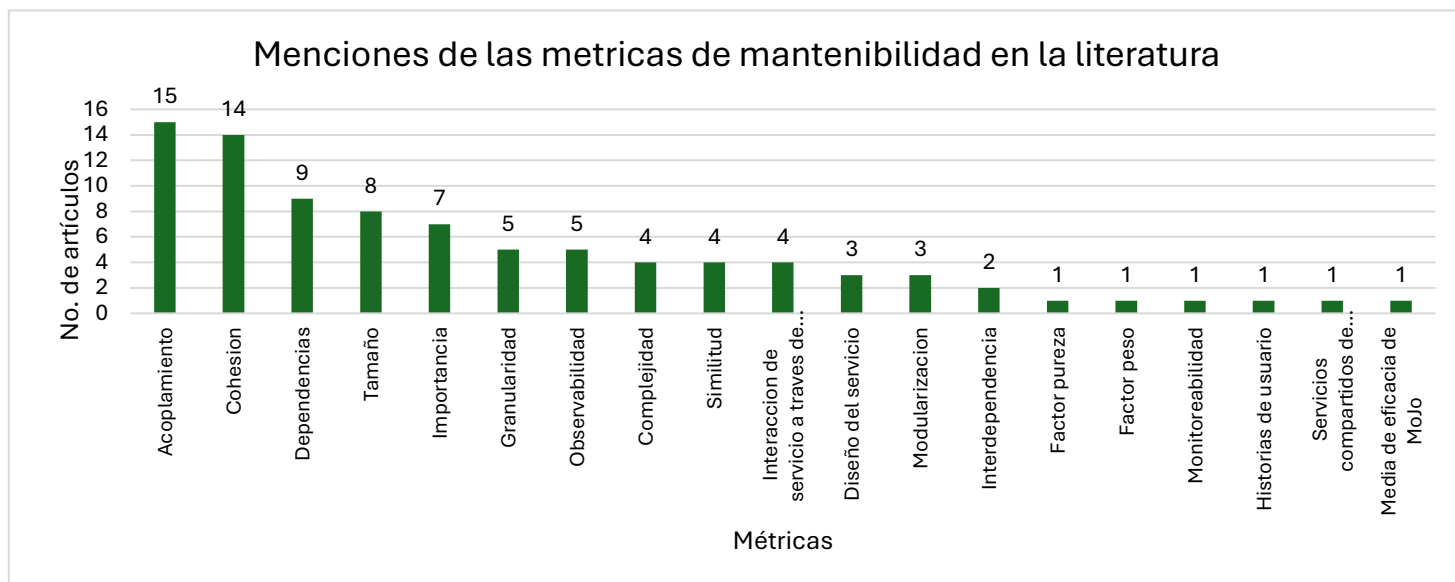


Figura 11 Menciones de las métricas de mantenibilidad en la literatura

4.3.2.4. Eficiencia de desempeño

Para este atributo se identificaron 16 métricas aplicables, lo cual constituye el 45.7% del total analizado. Estas métricas presentan entre 1 y 20 formas de evaluación, abarcando tanto métricas simples como métricas con un mayor número de adaptaciones.



Figura 12 Métricas relacionadas a eficiencia de desempeño

El número de artículos que mencionan estas métricas se distribuye principalmente entre 1 y 8 menciones, siendo menor en comparación con atributos como mantenibilidad.

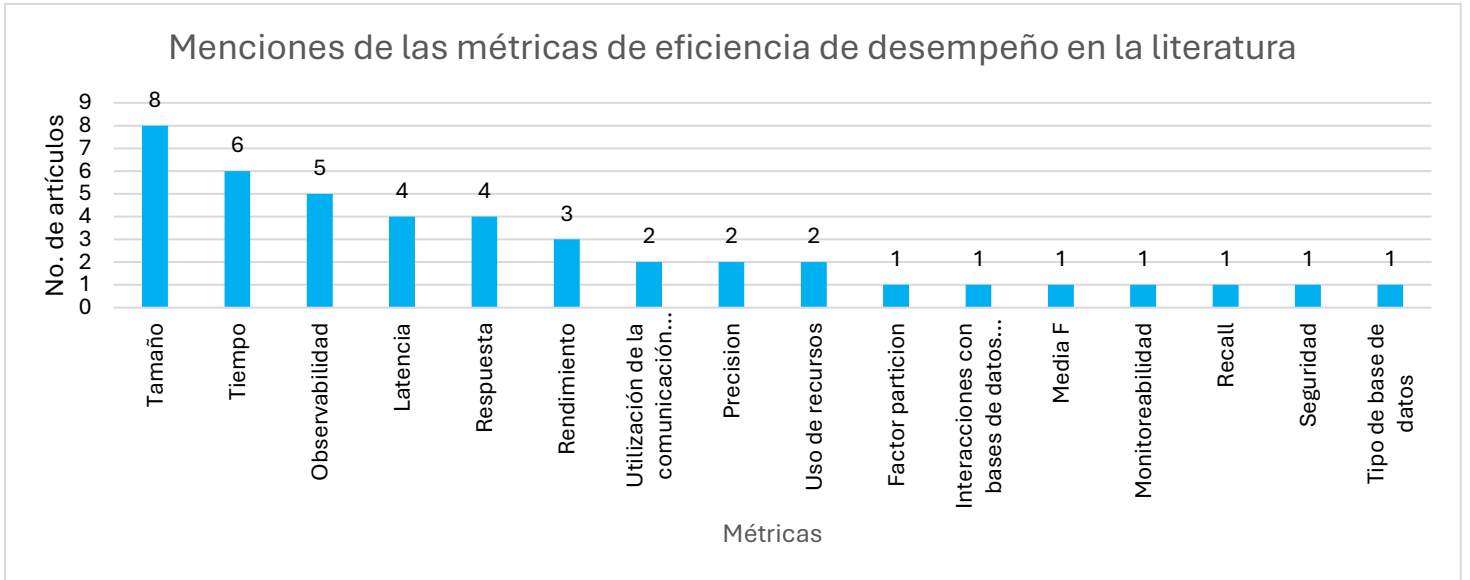


Figura 13 Menciones de las métricas de eficiencia de desempeño en la literatura

4.3.2.5. Seguridad

En el caso de seguridad, solo 4 métricas pueden ser aplicadas para evaluar este atributo, lo cual representa únicamente el 11.4% del total identificado. A pesar de su número limitado, presentan entre 1 y 20 formas de evaluación dependiendo de la métrica.

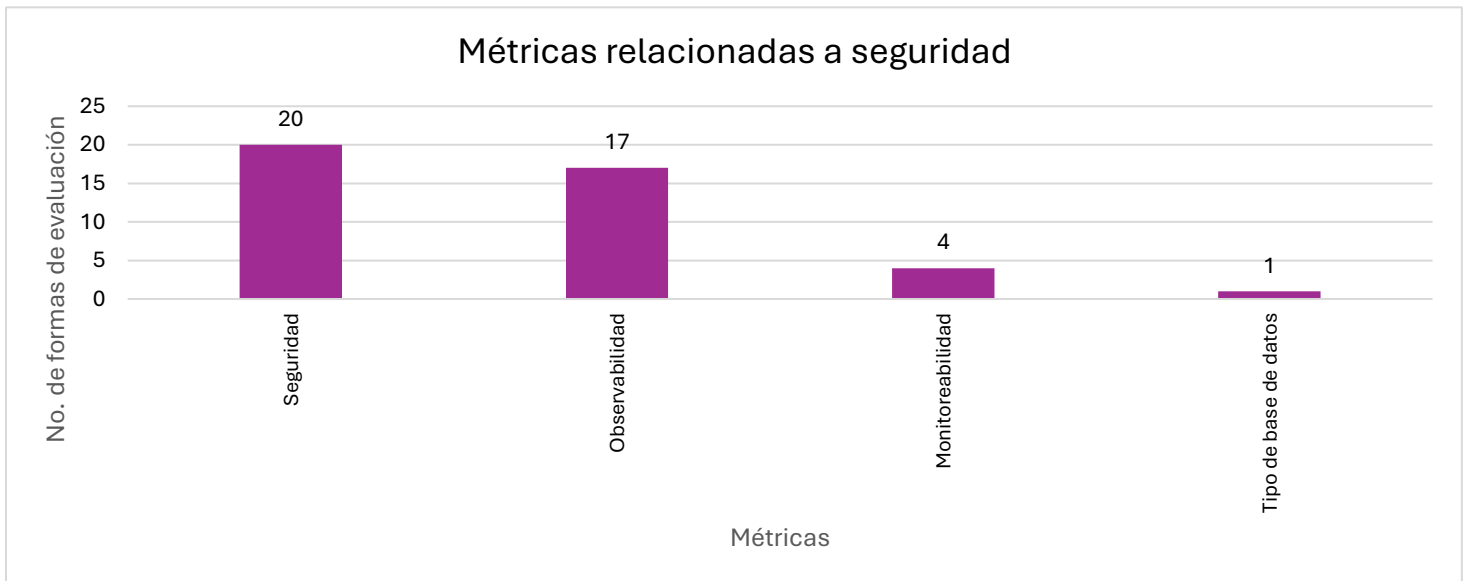


Figura 14 Métricas relacionadas a seguridad

El número de menciones en la literatura es reducido: las métricas relacionadas a seguridad muestran entre 1 y 5 artículos, colocando a este atributo como el de menor cobertura tanto en cantidad de métricas como en menciones.

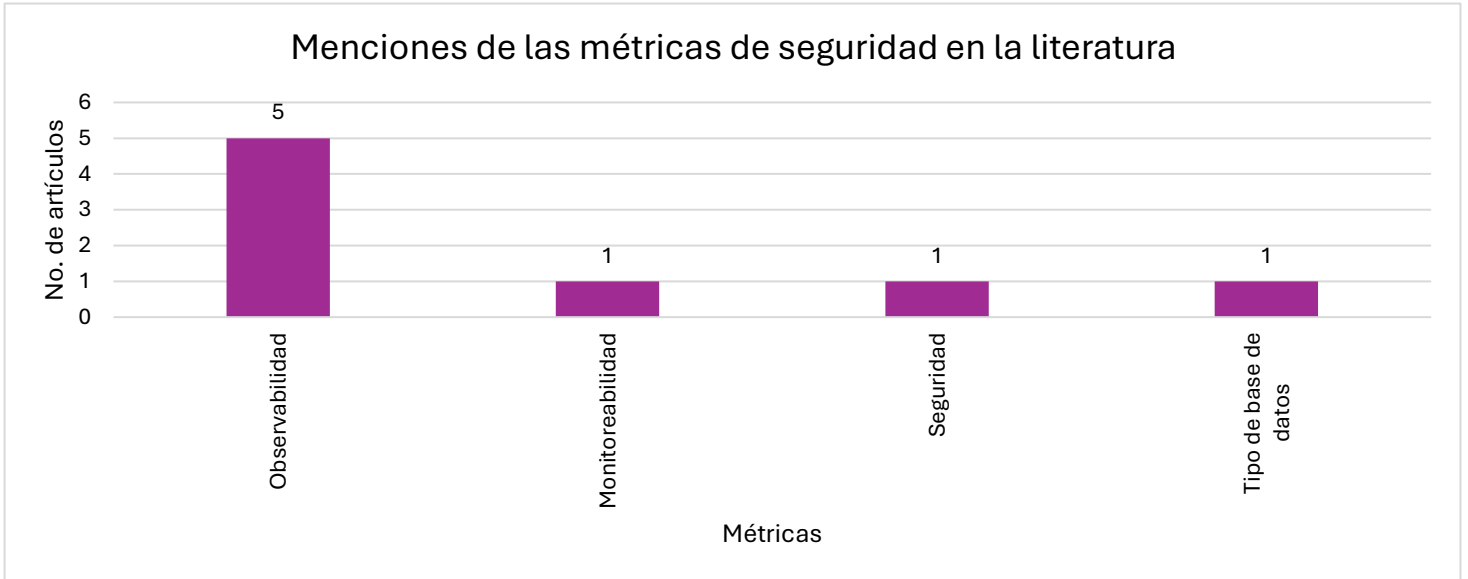


Figura 15 Menciones de las métricas de seguridad en la literatura

4.3.2.6. Compatibilidad

Para el atributo de compatibilidad, se identificaron 11 métricas, que equivalen al 31.4% de todas las métricas recopiladas. Las formas de evaluación registradas para estas métricas oscilan entre 1 y 40, ya que varias métricas de alto número de variaciones (como acoplamiento) están también relacionadas con este atributo.

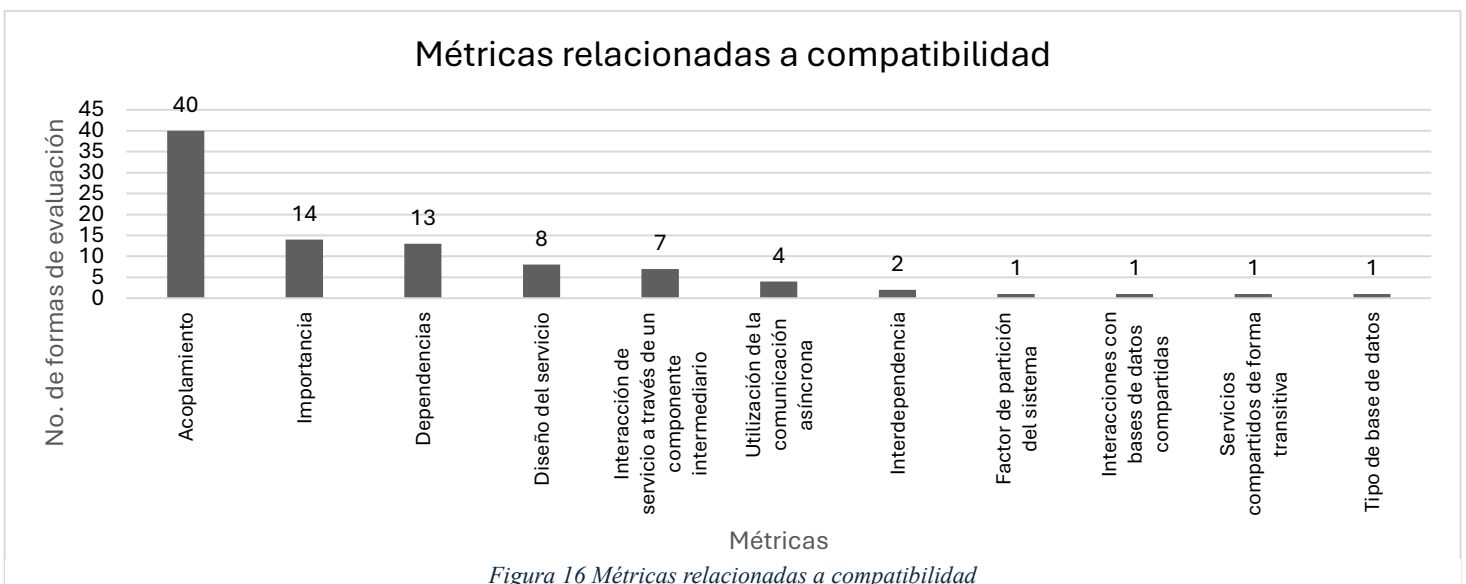


Figura 16 Métricas relacionadas a compatibilidad

El número de menciones en la literatura varía entre 1 y 15 artículos, lo cual evidencia la coexistencia de métricas con múltiples referencias y métricas emergentes, las cuales son mencionadas solo una vez.

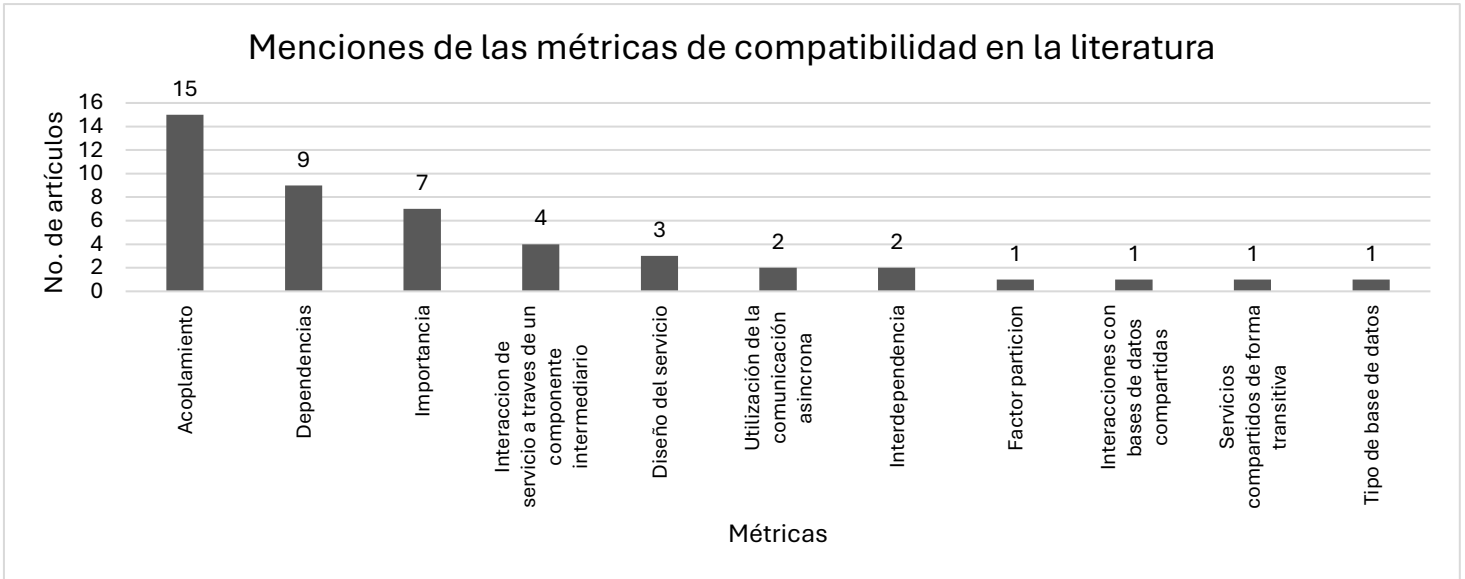


Figura 17 Menciones de las métricas de compatibilidad en la literatura

4.3.2.7. Propuesta de nuevas métricas

Con base en lo presentado para cada atributo de calidad se identificaron 18 métricas, con entre 1 y 4 formas de evaluar, lo cual representa un 51.4% de las 35 métricas principales. Estas métricas son brevemente mencionadas en la literatura lo cual evidencia que aunque actualmente los autores están abordando desde diferentes puntos de vista las métricas, también algunos están optando por proponer métricas relativamente novedosas con el fin de adaptarse al paradigma de los microservicios, lo cual refuerza la premisa de que tras no haber un consenso formal de la definición para microservicios, existe una amplia gama de formas de evaluación para medirlos.

La *Tabla 5* presenta 18 métricas que tienen entre 1 y 4 formas de evaluar presentes en la literatura consultada.

Tabla 5 Métricas con número de formas de evaluación ≤ 4

Métrica	No. De Formas de evaluación
Latencia	4
Monitoreabilidad	4
Uso de recursos	4

(Continuación) Tabla 5 Métricas con número de formas de evaluación ≤ 4

Métrica	No. De Formas de evaluación
Utilización de la comunicación asíncrona	4
Respuesta	3
Factor de peso	2
Historias de usuario	2
Interdependencia	2
Coefficiente de Silhouette	1
Factor de partición del sistema	1
Factor de pureza	1
Interacciones con bases de datos compartidas	1
F-Medida	1
Medida de eficacia de MoJo	1
Recall	1
Responsabilidad	1
Servicios compartidos de forma transitiva	1
Tipo de base de datos	1

4.3.2.8. Métricas con mayor número de formas de evaluación

Sin embargo, también existen métricas con un gran número de formas de evaluar a los microservicios, lo cual pone en evidencia que cada autor presenta un punto de vista diferente sobre como emplear las métricas con base en la definición analizada de microservicio.

Tabla 6 Top 10 métricas con más formas de evaluar

Métrica	No. de Formas de evaluación
Acoplamiento	40
Cohesión	33
Seguridad	20
Observabilidad	17
Importancia	14
Granularidad	13
Dependencias	13
Disponibilidad	9
Similitud	9
Tiempo	9

La Tabla 6 presenta las 10 métricas con mayor número formas para evaluar presentes en la literatura, como es posible observar, existe un gran sesgo en el número de formas para evaluar las métricas presentadas en las Tablas 5 y 6, es por esta razón que se optó por tomar la mediana para saber el punto medio de los datos, dando como resultado cuatro, lo que

fundamenta la premisa de que poco más de la mitad de las métricas son propuestas nuevas en la literatura.

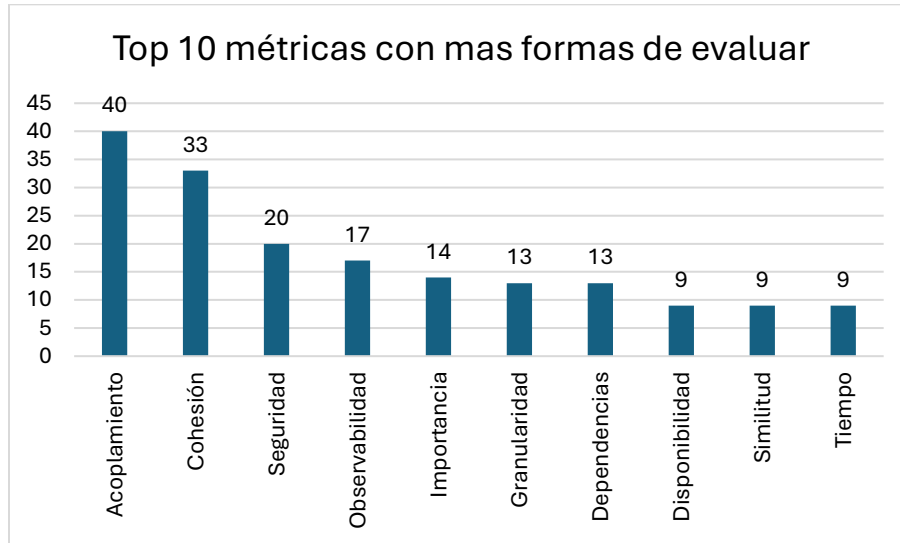


Figura 18 Top 10 métricas con más formas de evaluar

La *Figura 18* presenta el gráfico de las diez métricas con más formas de evaluar en la literatura consultada.

4.3.3. Umbrales para métricas en microservicios

Se realizó un análisis de los umbrales mencionados en la literatura revisada. Los resultados evidencian que más de la mitad de las métricas identificadas carecen de umbrales definidos, lo cual dificulta la interpretación precisa de los resultados obtenidos al aplicar dichas métricas.

La *Figura 19* muestra los porcentajes de presencia de umbrales en la literatura, destacando la notable ausencia de valores de referencia claros. Los umbrales son clasificados de la siguiente manera:

- **Normalizado:** Agrupa todos los umbrales que son probabilidades, coeficientes o índices estándar como 0 y 1. Comúnmente son los más fáciles de procesar matemáticamente.
- **Cualitativo:** Esta categoría contiene los umbrales que ofrecen interpretaciones como “bueno” y “malo”, “mejor” y “peor”. Estos umbrales son valiosos ya que dan retroalimentación al usuario.

- **Descriptivo:** Son aquellos umbrales que no dan propiamente un número, sino una regla lógica como “Cuantos menos mensajes se envíen y más servicios haya en el sistema, mayor cohesión”.
- **Numérico:** Estos umbrales son aquellos que no necesariamente son 0 y 1, sino que son conteos o límites absolutos.
- **Binario:** Donde el resultado solo puede tomar uno de dos valores posibles, generalmente 0 o 1, Verdadero o Falso, para indicar la ausencia o presencia de una característica.
- **No específica:** Esta categoría incluye todos aquellos umbrales que no fueron especificados por los distintos autores.

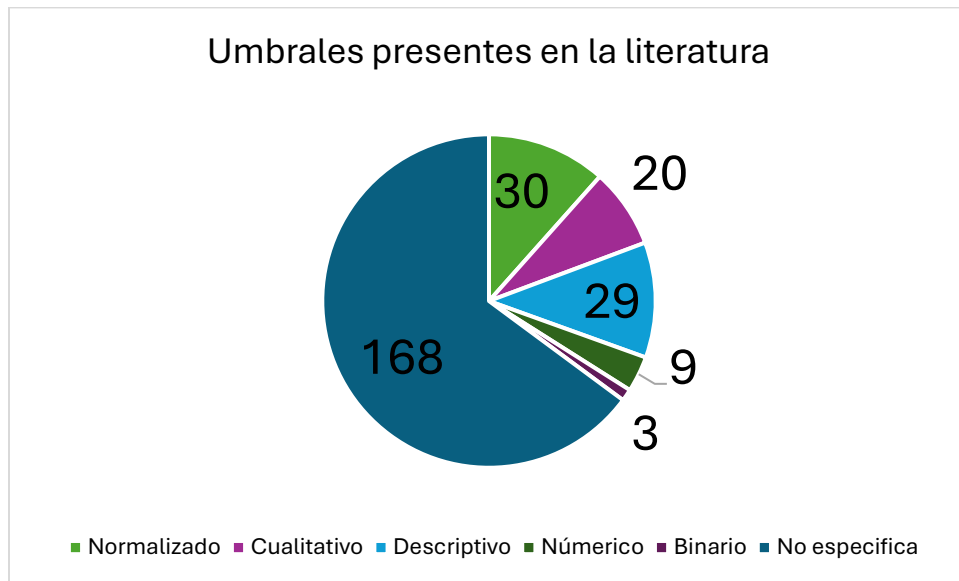


Figura 19 Umbrales presentes en la literatura

Para una descripción más completa de las fórmulas y umbrales asociados a cada métrica, se recomienda consultar el **Anexo B**, donde se presentan todas las métricas identificadas en los artículos analizados.

Asimismo, el análisis de la literatura permitió distinguir dos tipos principales de métricas según su contexto de aplicación: Métricas en tiempo de diseño y métricas en tiempo de ejecución.

Las métricas en tiempo de diseño pueden aplicarse antes del desarrollo de la solución basada en microservicios, brindando soporte para la toma de decisiones tempranas respecto a la

arquitectura y diseño del sistema. En cambio, las métricas en tiempo de ejecución se obtienen a partir del comportamiento del sistema una vez implementado, permitiendo monitorear en tiempo real los aspectos relevantes de su desempeño.

En síntesis, las métricas pueden implementarse en distintos ambientes de análisis con el propósito de obtener una visión general del sistema y facilitar la toma de decisiones informadas. Sin embargo, la mayoría de las fuentes se centran en la definición y uso de métricas, sin profundizar en los aspectos técnicos de su implementación. De esta manera, se infiere que la aplicación práctica de las métricas dependerá de las herramientas, tecnologías y nivel de dominio técnico del equipo de desarrollo, lo que sugiere que su implementación en entornos industriales o de prueba variará según las particularidades de cada proyecto.

4.3.4. Discusión de los resultados obtenidos tras el análisis de las métricas

Tras realizar el análisis cuantitativo sobre las métricas fue posible identificar distintas tendencias relevantes en la literatura, especialmente con relación a los atributos de calidad del modelo ISO/IEC 25010, la diversidad conceptual de las métricas, y la distribución del interés por parte de los autores. A continuación, se discuten los principales hallazgos derivados de la cuantificación.

4.3.4.1. Diversidad entre los atributos de calidad

Uno de los resultados más evidentes en la literatura consultada es la gran cantidad de métricas disponibles para cada uno de los diferentes atributos de calidad. Mientras atributos como mantenibilidad y flexibilidad cuentan con una gran diversidad de métricas, otros atributos como seguridad y compatibilidad muestran una presencia considerablemente menor.

Esta diversidad sugiere que la comunidad académica se ha concentrado principalmente en atributos que forman parte estructural del diseño de microservicios, tales como la modularidad, cohesión, granularidad o acoplamiento. En contraste, los atributos que dependen más del contexto, como seguridad o compatibilidad, reciben mucho menos atención, lo cual podría deberse a que estos dependen de factores externos a la arquitectura

de los microservicios, como tecnologías específicas, infraestructuras de despliegue o políticas de las organizaciones.

4.3.4.2. Falta de consenso en las definiciones

El análisis reveló que existen métricas que cuentan con una gran cantidad de formas de evaluar, lo cual resulta bastante evidente en métricas utilizadas ampliamente en la literatura como acoplamiento, cohesión, o granularidad. Este fenómeno podría indicar que si bien estas métricas son fundamentales para evaluar características estructurales para los microservicios, la falta de consenso sobre su definición o la mejor forma de aplicarlas en el entorno de los microservicios se encuentra sujeto al contexto de la misma arquitectura.

Asimismo, la ausencia de estos consensos provoca que los autores propongan adaptaciones específicas de métricas tradicionales o definan nuevas versiones que pretenden ajustarse a las necesidades de las arquitecturas de microservicios. Este comportamiento nos indica que actualmente las MSA aún se encuentran en consolidación, lo cual obliga a la industria y comunidad académica a explorar distintas alternativas que permitan medir de forma precisa a estos sistemas.

4.3.4.3. Adaptación de las métricas existentes y propuesta de nuevas

Más de la mitad de las métricas identificadas presentan entre 1 y 4 formas de evaluar, lo cual indica que una gran parte de ellas son métricas nuevas o de reciente aparición en la literatura. Estas métricas a menudo son mencionadas por estudios con enfoques específicos y que aún no han sido adoptadas por más autores o validadas en diversos contextos.

Asimismo, se observa que algunas de las métricas clásicas han sido adaptadas al paradigma de los microservicios mediante reformulaciones que añaden elementos como patrones de comunicación asíncrona, interacción entre servicios, etc. Esta combinación de métricas nuevas con reinterpretaciones refleja que tanto la industria y academia se encuentran en un proceso de exploración conceptual en el ámbito de microservicios.

4.3.4.4. Atributos que son poco atendidos por la industria y academia

Atributos como seguridad y compatibilidad destacan por la baja cantidad de métricas asociados a su evaluación, así como por la limitada presencia de estas en la literatura consultada, lo cual genera un foco de alerta, ya que ambos atributos son considerados atributos de suma importancia para los microservicios.

Para el caso de seguridad, los autores suelen abordarla desde diferentes perspectivas arquitectónicas o por medio del uso de herramientas externas, en lugar de proponer métricas formales que puedan ayudar a evaluar de manera más específica a los microservicios. Trabajos como el de Zdun se centran en el diseño y validación de un conjunto de métricas para seguridad de microservicios, sin embargo, éste resulta ser el principal proveedor ya que es el trabajo que contiene 20 formas distintas de evaluar el atributo de seguridad [48].

Por su parte, el atributo de compatibilidad a menudo suele ser tratado de manera indirecta, por ejemplo, con estudios sobre estilos de comunicación [2], [62], formatos de serialización [23], [46] o integración entre servicios [7], [56], [57], lo cual reduce ampliamente la aparición de métricas enfocadas exclusivamente en este atributo.

La ausencia de una base métrica sólida en estos dos ámbitos sugiere áreas con amplio potencial de investigación, especialmente en contextos donde los microservicios se integran con sistemas heredados, se despliegan en entornos híbridos o intercambian información con servicios de terceros.

4.3.4.5. Métricas asociadas a propiedades internas de los microservicios

Al revisar las métricas que presentan un mayor número de formas de evaluación, como acoplamiento, cohesión, granularidad o dependencias es posible evidenciar que la literatura se ha enfocado en gran medida a las propiedades internas de los microservicios y su relación estructural con los demás componentes del sistema, lo cual sugiere que aspectos como:

- Independencia funcional
- Modularización adecuada

- Mínimo acoplamiento
- Diseño de responsabilidades claras
- Consistencia en la distribución de las funciones.

Son fundamentales para los microservicios, pero a la vez, pueden propiciar que los elementos operativos como eficiencia en tiempo real, seguridad y la compatibilidad pasen a segundo plano.

4.3.4.6. Relación entre los atributos de calidad y las métricas con más formas de evaluación

El análisis mostró que las métricas con mayor número de formas de evaluación tienen impacto en los múltiples atributos de calidad propuestos por la ISO/IEC 25010, como se presenta en la *Tabla 7*.

Tabla 7 Relación de las métricas con más formas de evaluar y los atributos de calidad.

Métricas	Flexibilidad	Fiabilidad	Mantenibilidad	Eficiencia de desempeño	Seguridad	Compatibilidad
Acoplamiento	Si	No	Si	No	No	Si
Cohesión	Si	No	Si	No	No	No
Dependencia del servicio	Si	Si	Si	No	No	Si
Disponibilidad	No	Si	No	No	No	No
Granularidad	Si	No	Si	No	No	No
Importancia del servicio	No	Si	Si	No	No	Si
Observabilidad	No	Si	Si	Si	Si	No
Seguridad	No	No	No	Si	Si	No
Similitud	Si	No	Si	No	No	No
Tiempo	No	Si	No	Si	No	No

Esto indica que algunas métricas actúan como métricas transversales, ya que afectan simultáneamente varios atributos de calidad. Aunque esto no implica necesariamente una definición estándar, sí refleja que estas métricas han resultado útiles y relevantes para diferentes enfoques de evaluación.

Aunque dichas métricas pueden ayudar a evaluar los seis atributos de calidad propuestos para los microservicios, también es posible observar que atributos como seguridad,

compatibilidad y eficiencia de desempeño no están siendo tan abordadas por la industria o la academia en comparación de los demás atributos, tal como se muestra en la *Figura 20*.

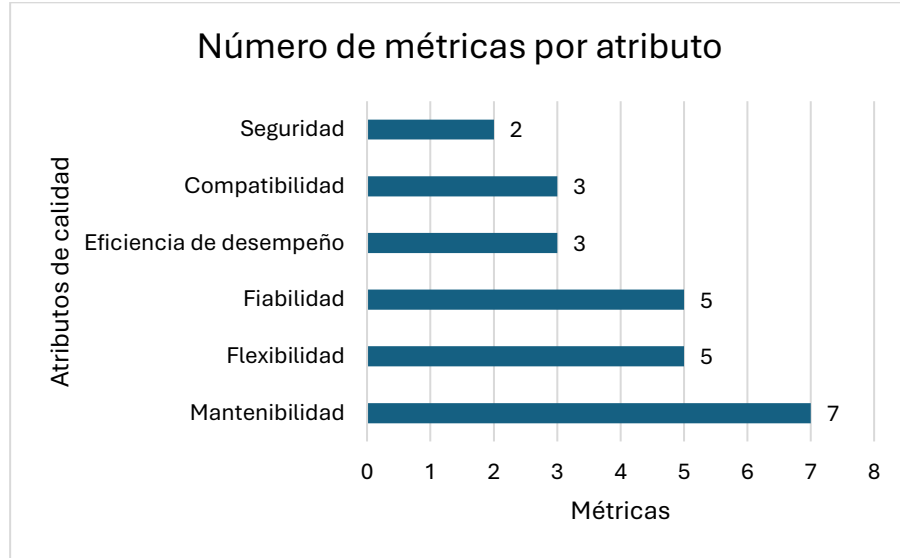


Figura 20 Presencia de las 10 métricas principales en los atributos de calidad.

4.4. Modelo de calidad

Al analizar la forma en que la norma ISO define cada uno de los atributos de calidad e intentar relacionarlo con los principios y características propias de los microservicios, fue posible observar que cada atributo implica una amplia variedad de subatributos que deben ser consideradas para una evaluación adecuada, lo cual dificulta abarcar todos los atributos sin comprometer la profundidad del análisis y del modelo en sí.

Por lo anterior se determinó que sería necesario realizar una delimitación del modelo de calidad a un solo atributo, y con base en lo mencionado por la literatura [6], [7], [23], [41] se encontró que el atributo de flexibilidad es ampliamente mencionado como parte esencial y característica de la arquitectura de microservicios, sin embargo, también fue posible evidenciar que como tal, es un atributo que no ha sido abordado formalmente en el ámbito de dicha arquitectura, es por esto que este modelo se enfoca en el atributo de flexibilidad.

En la *Figura 21* se presenta el modelo de calidad para el atributo de flexibilidad, donde las ponderaciones asignadas a cada subatributo se determinaron mediante un análisis empírico sustentado en la revisión conceptual de la literatura. Si bien no se aplicó un método formal de consenso de expertos, las proporciones se definieron con base en el grado de relevancia

presentada en estudios previos y en la influencia práctica que cada subatributo ejerce sobre la flexibilidad [19], [20], [52], [56], así como el comportamiento del sistema [7], [14], [34], [56]. Esta asignación de ponderaciones permitió establecer una valoración cuantitativa/cualitativa orientada a reflejar la importancia relativa de cada elemento en el modelo propuesto.

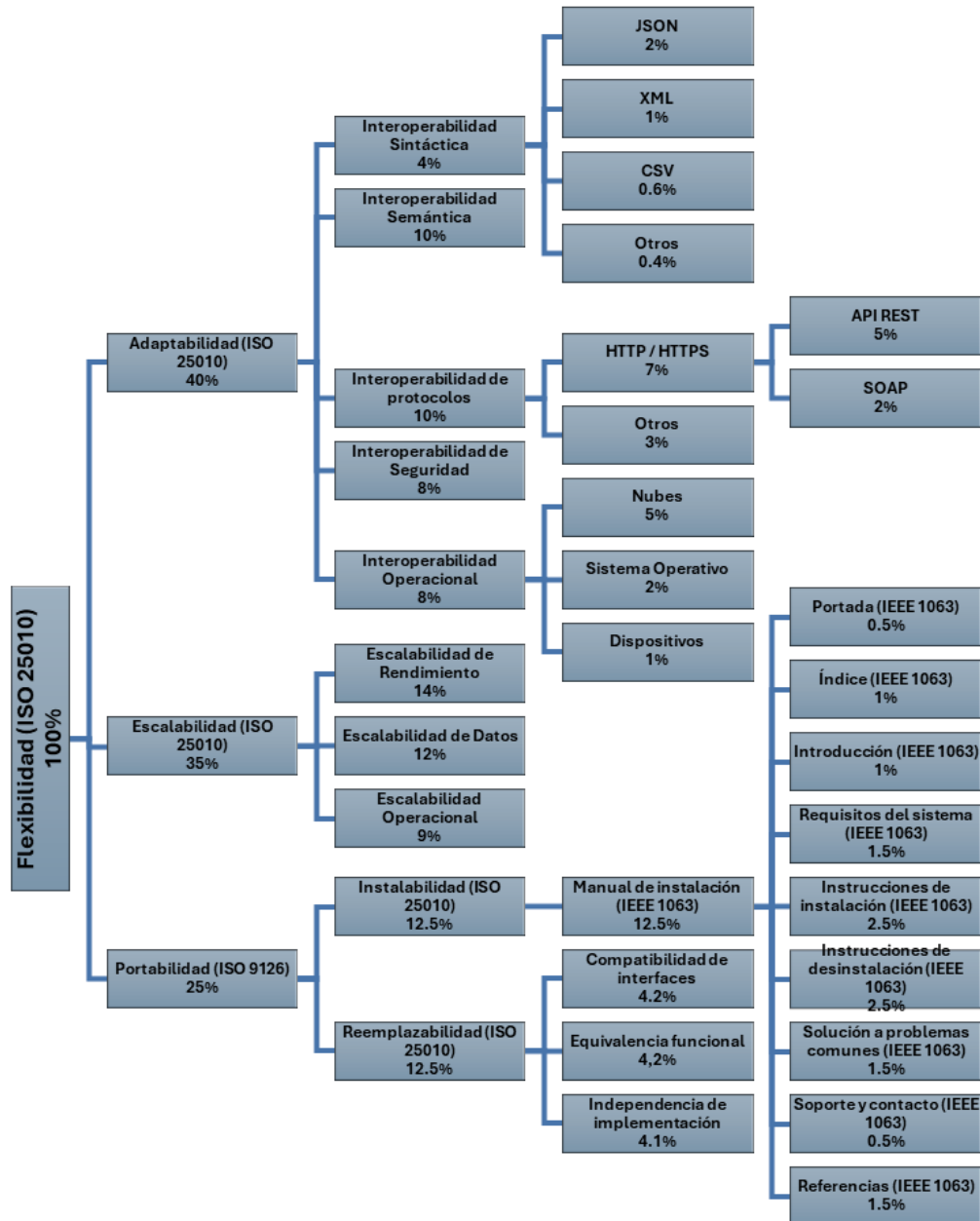


Figura 21 Modelo de calidad de flexibilidad de un microservicio

4.4.1. Adaptabilidad

La adaptabilidad recibe el mayor peso (40%) debido a que representa la agilidad del negocio. En MSA una de las principales características es que los microservicios se diseñan para operar en entornos cambiantes, con distintas tecnologías y requisitos. Su capacidad de interoperar con otros sistemas es crucial.

En MSA la interoperabilidad se convierte en un pilar fundamental de la adaptabilidad, ya que, si bien la norma ISO/IEC 25010 la sitúa como un subatributo de la Compatibilidad, la propia naturaleza de los microservicios obliga a reevaluar esta relación. En MSA un sistema es funcionalmente inadaptable si sus componentes no pueden conectarse, entenderse y cooperar eficazmente [34], [56].

Por lo tanto, la “Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada” [19] no es exclusivamente un asunto de compatibilidad, sino un requisito para que la adaptación sea posible. Por lo que se subdivide la adaptabilidad de la siguiente manera:

- **Interoperabilidad Sintáctica:** Peso de 4% porque se limita a la compatibilidad de formatos de datos.
- **Interoperabilidad Semántica:** Peso de 10%, ya que la correcta interpretación del significado de los datos es crítica para la integración efectiva.
- **Interoperabilidad de protocolos:** Se le otorga el mismo peso que la semántica porque elegir y soportar protocolos adecuados define la capacidad de comunicación.
- **Interoperabilidad de Seguridad:** Peso de 8%, ya que sin un mecanismo de autenticación y autorización la interoperabilidad sería riesgosa.
- **Interoperabilidad Operacional:** Mismo peso que seguridad, ya que asegura la integración en procesos de despliegue y ejecución, garantizando continuidad de negocio.

4.4.2. Escalabilidad

La escalabilidad recibe un peso del 35% porque es la respuesta de la arquitectura a la flexibilidad operativa. Mientras la adaptabilidad se centra en el cambio de la lógica de

negocio, la escalabilidad se centra en la capacidad del sistema para reaccionar a cambios en la demanda (carga de trabajo) sin degradar el servicio.

En MSA la escalabilidad granular es la capacidad de escalar únicamente el servicio que experimenta la alta demanda de forma independiente [52]. Si un servicio no puede escalar de forma aislada, provoca un fallo en cascada o requiere el escalado de componentes no relacionados, tornando la arquitectura rígida e inflexible [47], sin embargo, no siempre es el primer obstáculo en la integración de sistemas y está estrechamente ligada a una capacidad empresarial definida que será distinta para cada usuario.

Esta promesa, sin embargo, es compleja de alcanzar. Un sistema no es escalable simplemente porque está dividido en microservicios. La escalabilidad granular es una cualidad emergente que depende de tres subatributos: escalabilidad de rendimiento para el servicio, escalabilidad de sus datos, y la escalabilidad operacional para la automatización de la infraestructura [1], [40], [41].

Por esta razón, el modelo propuesto subdivide la escalabilidad en los siguientes subatributos:

- **Rendimiento:** Recibe peso de 14% pues la capacidad de responder a más usuarios o procesos es el núcleo de la escalabilidad.
- **Datos:** Se asigna un peso de 12% porque el manejo eficiente de grandes volúmenes de datos es crítico en sistemas distribuidos.
- **Operacional:** Peso de 9% porque está más enfocado en aspectos de gestión y operación, pero dependen en gran medida de la infraestructura.

4.4.3. Portabilidad

La portabilidad se pondera con 25% ya que una de las características clave de MSA, se define como la capacidad de un sistema de software para ser transferido y adaptado de una plataforma a otra [20]. Un sistema es inflexible si está sujeto a un proveedor de nube específico, a un sistema operativo propietario o a una configuración de hardware particular. La portabilidad mide la libertad de la arquitectura para moverse, evolucionar y ser desplegada en diversos entornos, lo cual es vital, ya que resulta fundamental para trasladar el sistema a

diferentes entornos sin grandes esfuerzos para la estrategia de negocio a largo plazo, aunque su impacto es menor que adaptabilidad y escalabilidad.

Según la ISO/IEC 9126, la portabilidad se define como “Conjunto de atributos relacionados con la capacidad de un sistema de software para ser transferido y adaptado desde una plataforma a otra” [20], razón por la cual, se considera que los atributos de instalabilidad y reemplazabilidad quedan en un nivel de jerarquía inferior en el modelo, como se presenta a continuación:

- **Instalabilidad** [19]: Se asigna el peso de 12.5% ya que la facilidad de instalación incide directamente en la aceptación de un sistema.
 - **Manual de instalación** [63]: Se hereda el mismo peso ya que se considera que para que un sistema tenga una “buena instalabilidad”, está sujeto al manual de usuario.
 - **Portada** [63]: Con un peso de 0.5% ya que, aunque el estándar de la IEEE 1063 lo contempla, no se considera de vital importancia.
 - **Índice** [63]: Un peso de 1% ya que el tener un índice aporta a que el usuario pueda encontrar más fácilmente lo que busca.
 - **Introducción** [63]: Igual peso que el índice ya que con él, el usuario podrá entender el contenido del documento.
 - **Requisitos del sistema** [63]: Con un peso mayor de 1.5% ya que resulta de vital importancia conocer los requisitos mínimos para que el sistema opere de la mejor manera posible dentro de nuestro entorno.
 - **Instrucciones de instalación** [63]: Con un peso mayor de 2.5% ya que es parte fundamental saber el paso a paso de la instalación del sistema.
 - **Instrucciones de desinstalación** [63]: Igual peso que el anterior, donde resulta importante conocer la mejor manera para desinstalar el sistema sin tener repercusiones posteriores derivadas de una mala desinstalación.
 - **Solución a problemas comunes** [63]: Con un peso de 1.5% ya que optimiza la búsqueda de soluciones a los principales problemas que puedan surgir antes, durante y después del uso del sistema.

- **Soporte y contacto** [63]: 0.5% de peso ya que el principal motivo del documento es que el usuario pueda encontrar la solución e información necesarias sin la ayuda del fabricante.
- **Referencias** [63]: 1.5% de peso ya que resulta de vital importancia que el usuario pueda encontrar las referencias a los componentes del sistema de los que se refiere en el documento.
- **Reemplazabilidad** [19]: Se asigna el peso de 12.5% porque sustituir un componente sin afectar el resto es clave en los microservicios.
 - **Compatibilidad de interfaces**: Peso de 4.2% porque una interfaz estable es la base de la sustitución.
 - **Equivalencia funcional**: Igual peso que el anterior porque la funcionalidad consistente garantiza que la sustitución no afecte el negocio.
 - **Independencia de implementación**: Un peso menor como lo es 4.1% ya que depende más de las buenas prácticas de desacoplamiento.

4.4.4. Ventajas del modelo propuesto

A continuación, se listan ventajas observadas en el modelo de calidad propuesto:

- El modelo propuesto cuenta con un respaldo normativo, al basarse en los estándares internacionales, ampliamente utilizados como marco de referencia para la evaluación de la calidad del software. [19], [20], [63], [64], [65], [66], [67].
- Diversos estudios señalan que la calidad en sistemas basados en microservicios depende no solo de factores tecnológicos, sino también de elementos organizativos y de documentación [23], [68], por lo que el modelo propuesto integra dichas dimensiones para ofrecer una evaluación más completa.
- La naturaleza modular de los microservicios permite adaptar las métricas o atributos de calidad a diferentes niveles de análisis [58], [69], por lo que el modelo propuesto puede aplicarse de manera parcial o integral según las necesidades de quien lo evalúe.
- La flexibilidad constituye un atributo esencial en los entornos de microservicios, dado que estos requieren adaptarse a entornos cambiantes y despliegues distribuidos [61], [70]. Por ello, el modelo centra su análisis en este atributo como componente clave de la calidad del sistema.

- La posibilidad de ajustar los pesos de los subatributos según el contexto del sistema es consistente con los enfoques modernos de evaluación de calidad que promueven la adaptabilidad del modelo a las prioridades del usuario [71].

Capítulo 5 Pruebas y resultados



En este capítulo se describen las pruebas realizadas para validar el modelo de calidad planteado en el presente trabajo, así como los resultados obtenidos a partir de su aplicación.

5.1. Proceso de pruebas

El proceso de pruebas tuvo como propósito validar la aplicabilidad y consistencia del modelo propuesto para la evaluación del atributo de flexibilidad en arquitecturas de microservicios. Para ello, el modelo fue aplicado sobre repositorios de acceso libre que estuvieran desarrollados en Java, dado que esta tecnología es ampliamente utilizada tanto en contextos académicos como industriales y mantiene una estrecha compatibilidad con los Microservicios, ya que, de acuerdo con el State of Spring Survey 2024 [72], los microservicios constituyen uno de los principales focos dentro del ecosistema Spring, lo cual refleja su relevancia en el desarrollo moderno de software distribuido. Asimismo, el estudio de Sönajalg [73] reporta que el 86% de los desarrolladores Java que fueron encuestados utilizan Spring Boot para la creación de microservicios.

Los repositorios seleccionados para el estudio fueron Zull [74], ACME Air Microservice [75] y E-Commerce [76], ya que, en conjunto, estos proyectos agrupan catorce microservicios, los cuales presentan distintos niveles de complejidad, modularidad y documentación, lo cual permite evaluar el comportamiento del modelo bajo escenarios heterogeneos. La descripción general de cada repositorios, así como el número de microservicios que lo componen y su función principal son presentados en la *Tabla 8*.

Tabla 8 Repositorios seleccionados

<i>Nombre</i>	<i>Autor</i>	<i>No. De Microservicios</i>	<i>Función</i>
ZULL [74]	Lindsey Reynolds	4	Proporciona capacidades de enrutamiento dinámico, monitorización, resistencia, seguridad y mucho más.
Acme AIR-Microservice [75]	Joe McClure	4	Un controlador de carga de trabajo para la aplicación de muestras de Air Acme.
E-Commerce [76]	Rainbow Forest	6	Implementación de un Microservicio REST en un E-Commerce con Spring boot, Cloud y múltiples módulos.

La selección de dichos repositorios se llevó a cabo mediante la aplicación de criterios de inclusión y exclusión, con el fin de garantizar la pertinencia de los casos de estudio y la comparabilidad de los resultados. Los criterios de inclusión considerados fueron los siguientes:

- Repositorios desarrollados completamente en lenguaje de programación Java
- Repositorios que estén implementados en algún framework orientado al desarrollo de microservicios, preferentemente Spring Boot.

Por su parte, se establecieron los siguientes criterios de exclusión:

- Repositorios que no estuvieran desarrollados en Java
- Repositorios que presenten un número reducido de componentes o microservicios (Menor a dos), ya que esto limitaría el análisis de propiedades asociadas a la flexibilidad arquitectónica.

Una vez seleccionados los repositorios, el proceso de evaluación se realizó mediante la revisión sistemática de código estático, cuyo propósito fue identificar evidencias estructurales y configuracionales que permitieran aplicar a los subatributos definidos en el modelo de calidad propuesto. Este enfoque consiste en el análisis directo del código fuente, archivos de configuración y artefactos asociados, sin necesidad de ejecutar el sistema, lo que permite evaluar propiedades internas relacionadas con la adaptabilidad, escalabilidad y portabilidad.

El proceso de pruebas fue estructurado de la siguiente manera:

1. **Identificación de microservicios:** Se delimitó el conjunto de microservicios que componen cada repositorio, considerando su independencia funcional y su rol en la arquitectura.
2. **Revisión de artefactos de configuración:** Se analizaron archivos como *Application.yml*, *Application.properties*, archivos de construcción como *Pom.xml*, *Build.gradle*, y scripts asociados al despliegue, con el fin de identificar mecanismos de parametrización, dependencia y portabilidad.
3. **Análisis estructural del código:** Se revisó la organización de paquetes, el uso de anotaciones, la definición de interfaces y la interacción entre componentes, con el objetivo de evaluar aspectos relacionados con la adaptabilidad y el acoplamiento.
4. **Evaluación de documentación y contrato del servicio:** Se verificó la presencia de documentación técnica, especificaciones de APIs (por ejemplo, OpenAPI/Swagger) y manuales de instalación, los cuales influyen directamente en la portabilidad y mantenibilidad del sistema.
5. **Aplicación del modelo de calidad:** A partir de las evidencias identificadas, se aplicaron los criterios y ponderaciones definidos en el modelo propuesto para calcular

los valores correspondientes a los subatributos de adaptabilidad, escalabilidad y portabilidad, así como la flexibilidad total.

A continuación, la *Tabla 9* presenta de manera estructurada las actividades de evaluación realizadas durante el proceso de pruebas, así como las evidencias consideradas para la asignación de los valores correspondientes a cada subatributo.

Tabla 9 Actividades para evaluar los microservicios

ID	Atributo	Dimensión / Subatributo	Elementos evaluados	Artefactos analizados	Técnicas de análisis
<i>A1</i>	Adaptabilidad	Interoperabilidad Sintáctica	Formatos de intercambio de datos (JSON, XML, CSV, otros)	Controladores REST, modelos de datos, configuración	Inspección estructural
<i>A2</i>	Adaptabilidad	Interoperabilidad Semántica	Documentación de APIs, claridad y consistencia de endpoints y parámetros, uso de estándares	Swagger/OpenAPI, README, código fuente	Análisis documental
<i>A3</i>	Adaptabilidad	Interoperabilidad de Protocolos	Protocolos de comunicación soportados (HTTP, HTTPS, otros)	Anotaciones REST, configuración de red	Revisión arquitectónica
<i>A4</i>	Adaptabilidad	Interoperabilidad de Seguridad	Autenticación, autorización, uso de HTTPS, integración con servicios de identidad	Configuración de seguridad, dependencias, documentación	Análisis estructural
<i>A5</i>	Adaptabilidad	Interoperabilidad Operacional	Capacidad de operar en distintos entornos (SO, contenedores, nube)	Scripts, Dockerfiles, README	Análisis de configuración
<i>E1</i>	Escalabilidad	Escalabilidad de Rendimiento	Manejo de concurrencia, balanceo, caché, pruebas de carga	Código, configuración, documentación	Revisión arquitectónica
<i>E2</i>	Escalabilidad	Escalabilidad de Datos	Gestión de crecimiento de datos, bases de datos distribuidas, replicación	Configuración de persistencia, dependencias	Inspección estructural
<i>E3</i>	Escalabilidad	Escalabilidad Operacional	Despliegue multi-instancia, automatización, observabilidad	CI/CD, Docker, Kubernetes, scripts	Análisis documental
<i>P1</i>	Portabilidad	Instalabilidad – Manual de Instalación	Presencia y calidad del manual (portada, índice, introducción, requisitos, instalación, troubleshooting, soporte, referencias)	README, documentación asociada	Análisis documental
<i>P2</i>	Portabilidad	Reemplazabilidad – Compatibilidad de Interfaces	Estabilidad y versionado de APIs	OpenAPI, controladores REST	Revisión de diseño
<i>P3</i>	Portabilidad	Reemplazabilidad – Equivalencia Funcional	Claridad de responsabilidades y funcionalidades	Código fuente, pruebas	Análisis estructural
<i>P4</i>	Portabilidad	Reemplazabilidad – Independencia de Implementación	Dependencia tecnológica y posibilidad de sustitución	Dependencias, arquitectura	Revisión arquitectónica

Con el fin de implementar el proceso de pruebas descrito anteriormente y garantizar su aplicación sistemática, las actividades de evaluación fueron descompuestas en un conjunto de tareas claramente definidas. Cada actividad se asocia con los artefactos analizados, el subatributo del modelo de flexibilidad evaluado y la técnica de análisis empleada, permitiendo así establecer una relación explícita entre el modelo propuesto de manera teórica y su aplicación práctica. Para facilitar la trazabilidad y referencia de dichas actividades a lo largo del capítulo, cada una se le asignó un identificador único (ID), el cual se utiliza para vincular de manera precisa las evidencias analizadas con los valores obtenidos en la evaluación.

La nomenclatura empleada se definió de la siguiente manera:

- Los IDs que inician con la letra A corresponden a actividades asociadas al atributo de Adaptabilidad, específicamente a las distintas dimensiones de la interoperabilidad.
- Los IDs que inician con la letra E hacen referencia a las actividades relacionadas con el atributo de Escalabilidad.
- Los IDs que inician con la letra P corresponden a las actividades vinculadas al atributo de Portabilidad, la cual incluye tanto instalabilidad como reemplazabilidad.
- Los valores numéricos que acompañan a cada letra indica la secuencia de evaluación en cada atributo, sin embargo, esto no implica un orden de prioridad entre las actividades.

El estructurar las actividades de evaluación como se presenta en la *Tabla 9* permitió realizar el análisis a todos los repositorios y microservicios considerados de manera uniforme, lo cual asegura consistencia en la recolección de evidencias y en la aplicación de los criterios definidos. De esta manera, cada microservicio fue evaluado bajo el mismo conjunto de actividades, independientemente de su dominio o complejidad, lo que contribuye a la validez y comparabilidad de los resultados. A partir de la información obtenida mediante estas actividades, se procedió al cálculo de los valores asociados a los subatributos de adaptabilidad, escalabilidad y portabilidad, así como al valor global de flexibilidad.

5.2. Interpretación de resultados

En la siguiente sección se presentan los resultados obtenidos tras la aplicación del modelo, así como su interpretación individual y comparativa para cada uno de los repositorios analizados, con el fin de ofrecer una visión más profunda del comportamiento observado en cada proyecto y analizar de forma diferenciada las evidencias que contribuyen a la flexibilidad.

5.2.1. Repositorio ZULL

El repositorio Zull desarrollado por Netflix [74], se caracteriza por ofrecer capacidades avanzadas de enrutamiento dinámico, supervisión, resiliencia, seguridad y mucho más para aplicaciones basadas en la nube. Actúa como un servicio periférico que enruta y filtra el tráfico hacia los servicios backend, funcionando como puerta de entrada a todas las solicitudes de dispositivos y sitios web al backend del ecosistema de aplicaciones de streaming de Netflix.

En la *Tabla 9* se presentan los resultados obtenidos por cada uno de los microservicios que componen el repositorio Zull [74], así mismo se presentan las calificaciones obtenidas por cada una de las tres características principales que según el modelo propuesto, caracterizan a la flexibilidad.

Tabla 10 Resultados del repositorio ZULL [74] por microservicio

REPOSITORIO	MICROSERVICIO	ADAPTABILIDAD	ESCALABILIDAD	PORTABILIDAD	FLEXIBILIDAD
<i>ZULL [74]</i>	Zull-Core	30%	24%	8.6%	62.6%
	Zull-Discovery	20%	18%	8.6%	46.6%
	Zull-Processor	10%	9%	13.4%	32.4%
	Zull-Integration-Test	21%	17%	9.9%	47.9%

Los resultados obtenidos para el repositorio Zull evidencian fortalezas en adaptabilidad y escalabilidad (valores promedio $\approx 20.25\%$ y 17% respectivamente), pero estas cifras están significativamente por debajo de los objetivos fijados por el modelo (40% y 35%). Además, la portabilidad, que tiene un peso en la evaluación de 25% , se sitúa en torno al 10% ; esto significa que los microservicios de Zull alcanzan aproximadamente el 51% del objetivo en adaptabilidad ($20.25/40 \approx 50.6\%$), el 48.57% en escalabilidad ($17/35 = 48.57\%$) y solo el 40% en portabilidad ($10/25 \approx 40\%$).

La consecuencia práctica es que, aunque Zull cumple bien con funciones propias de un API Gateway (enrutamiento, filtros, resiliencia), la flexibilidad global queda severamente limitada por la suma de déficits en los subatributos del modelo. Esto evidencia la necesidad de no conformarse con “buenas prácticas aisladas” sino de implementar mejoras transversales (automatización del despliegue, abstracción de dependencias del ecosistema Netflix OSS, documentación operativa y estrategias de migración) para aproximarse a los valores objetivo y, por ende, elevar la flexibilidad total del sistema.

Estos resultados también sugieren que, a pesar de la fortaleza del proyecto en los aspectos de comunicación y gestión de tráfico, su despliegue depende en gran medida de configuraciones propias del ecosistema Spring y de componentes específicos de Netflix OSS, lo cual reduce su independencia tecnológica y limita la posibilidad de trasladar el sistema a otros entornos. Esta limitación en portabilidad se convierte en el factor crítico que modera el nivel total de flexibilidad, confirmando lo señalado por Dragoni et al.[23] y Li et al. [2] respecto a la necesidad de equilibrar la adaptabilidad con la capacidad de despliegue en entornos heterogéneos.

En cuanto a la adaptabilidad, se observó un uso extensivo de configuraciones en formato YAML y anotaciones Java que permiten parametrizar el comportamiento de los servicios, lo cual incrementa la interoperabilidad sintáctica y semántica. No obstante, la ausencia de mecanismos formales de autenticación y autorización limita la interoperabilidad de seguridad y, con ello, la flexibilidad integral del sistema. Este hallazgo es consistente con lo planteado por Taibi y Lenarduzzi [68], quienes destacan que la falta de políticas unificadas de seguridad y documentación puede afectar negativamente la mantenibilidad y la calidad global en sistemas distribuidos.

Desde una perspectiva comparativa, el microservicio Zull-Core obtuvo la puntuación más alta de flexibilidad (62.6 %), mientras que Zull-Processor presentó la más baja (32.4 %). Esta diferencia evidencia que los microservicios con mayor exposición al flujo de comunicación y gestión de peticiones suelen mostrar mayor madurez arquitectónica, mientras que los componentes de soporte tienden a reflejar un menor grado de desacoplamiento y portabilidad. En síntesis, el análisis del repositorio Zull confirma que la carencia de estrategias de

portabilidad y documentación operativa limita la flexibilidad total, aun cuando la arquitectura exhibe una excelente adaptabilidad y una escalabilidad aceptable.

5.2.2. Repositorio ACME Air Microservice

El proyecto [75] muestra un patrón de arquitectura de microservicios con cuatro servicios independientes que gestionan la autenticación, la gestión de clientes, la información de vuelos y las operaciones de reserva lo cual permite estudiar la escalabilidad y adaptabilidad bajo escenarios de alta concurrencia.

Los resultados presentados en la *Tabla 10* evidencian que, aunque existen elementos de diseño modular (adaptabilidad $\approx 22\%$) y una escalabilidad moderada ($\approx 9.75\%$), estos valores están consistentemente por debajo de los objetivos del modelo (adaptabilidad = 40%, escalabilidad = 35%, portabilidad = 25%). Tomando como ejemplo el BookingService (adaptabilidad 26%, escalabilidad 15%, portabilidad 12.6 %), se observa que alcanza aproximadamente el 65% del objetivo en adaptabilidad (26/40), el 42.85% en escalabilidad (15/35) y solo el 50.4% en portabilidad (12.6/25).

Tabla 11 Resultados del repositorio ACME Air Microservice [75] por microservicio

REPOSITORIO	MICROSERVICIO	ADAPTABILIDAD	ESCALABILIDAD	PORTABILIDAD	FLEXIBILIDAD
<i>ACME AIR MICROSERVICE [75]</i>	AuthService	18%	8%	8.7%	34.7%
	CustomerService	22%	8%	12.8%	42.8%
	FlightService	22%	8%	12.8%	42.8%
	BookingService	26%	15%	12.6%	53.6%

Este patrón revela que, aunque algunos servicios (p.ej. BookingService) presentan mayor madurez funcional, ninguno alcanza el conjunto de metas mínimas esperadas; por tanto, la flexibilidad global queda restringida por déficits acumulados en más de un subatributo. La baja portabilidad, además de limitaciones en pruebas de rendimiento y automatización de despliegues, sugiere que la arquitectura necesita intervenciones integradas: estandarización de APIs (OpenAPI), generación de artefactos de despliegue reproducibles (Dockerfiles, Helm charts), pipelines CI/CD y documentación de dependencias para cerrar las brechas hacia los objetivos planteados.

En conjunto, el análisis del repositorio Acme Air Microservice demuestra que, aunque la arquitectura distribuida logra mantener una adaptabilidad aceptable, la baja portabilidad

resulta el principal obstáculo para alcanzar un nivel de flexibilidad óptimo. Esto refuerza la necesidad de considerar la portabilidad como un factor determinante en el modelo de evaluación y de implementar estrategias orientadas a la automatización, documentación y despliegue multiplataforma.

5.2.3. Repositorio E-Commerce

Este repositorio [76] agrupa un conjunto de microservicios desarrollados en Spring Boot y Spring Cloud, los cuales se enfocan en la gestión de usuarios, pedidos y catálogo de productos. En la *Tabla 11* se presentan los resultados obtenidos tras aplicar el modelo de evaluación a cada microservicio.

Tabla 12 Resultados del repositorio E-Commerce [76] por microservicio

REPOSITORIO	MICROSERVICIO	ADAPTABILIDAD	ESCALABILIDAD	PORTABILIDAD	FLEXIBILIDAD
<i>E-COMMERCE</i> [76]	Api-Gateway	14%	9%	9%	32%
	Eureka-Server	17%	9%	11.1%	37.1%
	User-Service	13%	9%	9.9%	31.9%
	Product-Catalog-Service	15%	9%	8.2%	32.2%
	Order-Service	16%	13%	9.7%	38.7%
	Product-Recommendation-Service	12%	9%	9.5%	30.5%

Los resultados pertenecientes al repositorio E-Commerce [76] reflejan un comportamiento generalizado de baja flexibilidad, con valores que oscilan entre 30.5 % y 38.7 %. Este conjunto de microservicios evidencia deficiencias simultáneas en los tres subatributos principales del modelo: adaptabilidad, escalabilidad y portabilidad. En promedio, los valores registrados (≈ 14.5 %, 9.7 % y 9.6 % respectivamente) se sitúan muy por debajo de los objetivos establecidos (40 %, 35 % y 25 %). Esto significa que, de forma aproximada, los servicios del repositorio alcanzan apenas el 36.25% del ideal en adaptabilidad, el 27.71% en escalabilidad y el 38.4% en portabilidad, lo que explica las bajas puntuaciones globales de flexibilidad.

Como es posible observar, existe una diferencia considerable entre el resultado obtenido tras la evaluación y la ponderación propuesta en el modelo, lo cual sugiere que, aunque el sistema resulta funcional, éste presenta un diseño orientado a la operatividad básica y modular, dejando de lado la calidad arquitectónica y la sostenibilidad. Las causas principales de este

comportamiento se asocian a tres factores: (1) acoplamiento elevado entre servicios por la dependencia del servidor de descubrimiento Eureka; (2) ausencia de documentación técnica y manuales de despliegue, lo cual limita la portabilidad e instalabilidad; y (3) falta de mecanismos automatizados de escalamiento y monitoreo de rendimiento, que restringen la capacidad del sistema para ajustarse a variaciones en la carga de trabajo.

En particular, el microservicio Order-Service obtuvo la puntuación más alta (38.7 %), mostrando una ligera ventaja en escalabilidad (13 %) y adaptabilidad (16 %), lo cual refleja su rol como componente intermedio que coordina transacciones entre clientes y catálogo de productos. Sin embargo, incluso en este caso, los valores alcanzan apenas un valor cercano al 40% del ideal, lo que denota un déficit estructural generalizado. En contraste, Product-Recommendation-Service, con 30.5 %, presentó la menor flexibilidad, evidenciando limitaciones severas en adaptabilidad (12 %) y portabilidad (9.5 %), relacionadas con la escasa documentación de dependencias y la falta de mecanismos de integración semántica con otros módulos del sistema.

El bajo desempeño en portabilidad, particularmente relevante por su ponderación de 35 %, constituye el principal factor que reduce la flexibilidad total del repositorio. Tras el análisis del código estático se denota que la falta de scripts de despliegue, la configuración estática y la inexistencia de contenedores reproducibles hacen que la migración o ejecución del sistema en diferentes entornos requiera intervención manual, afectando directamente su capacidad de adaptación. Así mismo, esta carencia se ve agravada por el hecho de que los valores de adaptabilidad y escalabilidad también están lejos del ideal, de modo que la pérdida de flexibilidad no proviene de un solo atributo, sino de una combinación de brechas acumuladas en los tres.

Tabla 13 Resultados de flexibilidad del microservicio product-recommendation-service del repositorio E-Commerce [76]

Atributos por medir				Elemento	Valor total por subatributo %	Valor total por atributo %	Valor total por atributo global	
Flexibilidad	Adaptabilidad	Interoperabilidad Sintáctica		JSON (Mayor prioridad)	2	2	12	
				XML (Prioridad media)	0			
				CSV (Prioridad baja)	0			
				Otros	0			
		Interoperabilidad Semántica			3	3		
		Interoperabilidad de Protocolos	HTTP/HTTPS	API REST (Mayor prioridad)	3	4		
				SOAP (Prioridad baja)	0			
				Otros	1			
		Interoperabilidad de Seguridad			0	0		
		Interoperabilidad Operacional		Nubes (Mayor prioridad)	2	3		
				Sistema Operativo (Prioridad media)	1			
				Dispositivos (Prioridad baja)	0			
	Escalabilidad			Escalabilidad de Rendimiento	4	9	9	
				Escalabilidad de Datos	3			
				Escalabilidad Operacional	2			
	Portabilidad		Instalabilidad	Manual de instalación	Portada	0.3	3.3	9.5
					Índice	0.4		
					Introducción	0.5		
					Requisitos del sistema	1.2		
					Instrucciones de instalación	0.8		
					Instrucciones de desinstalación	0		
					Solución a problemas comunes	0		
					Soporte y contacto	0		
					Referencias	0.1		
			Reemplazabilidad		Compatibilidad de interfaces	1.8	6.2	
					Equivalencia funcional	2.1		
					Independencia de implementación	2.3		

En la *Tabla 12* se presentan los resultados de manera específica en cuanto al microservicio Product-Recommendation-Service tras aplicar el modelo de calidad propuesto para evaluar la flexibilidad.

En síntesis, los resultados del repositorio E-Commerce confirman la tendencia observada en los demás proyectos: las tres dimensiones de flexibilidad presentan déficits simultáneos respecto a los valores objetivo (40% para adaptabilidad, 35% para escalabilidad y 25% para portabilidad), con la portabilidad como el factor más crítico. La suma de estas brechas explica la baja flexibilidad global e ilustra la necesidad de adoptar un enfoque integral que considere la interacción entre adaptabilidad, escalabilidad y portabilidad como elementos interdependientes en el modelo de evaluación propuesto.

Por otro lado la *Tabla 13* permite observar de manera comparativa el desempeño de cada uno de los microservicios existentes en los tres subatributos considerados por el modelo: adaptabilidad, escalabilidad y portabilidad, así como el valor total de la flexibilidad.

Tabla 14 Resultados obtenidos por repositorio y microservicio

REPOSITORIO	MICROSERVICIO	ADAPTABILIDAD	ESCALABILIDAD	PORTABILIDAD	FLEXIBILIDAD
<i>ZULL [74]</i>	Zull-Core	30%	24%	8.6%	62.6%
	Zull-Discovery	20%	18%	8.6%	46.6%
	Zull-Processor	10%	9%	13.4%	32.4%
	Zull-Integration-Test	21%	17%	9.9%	47.9%
<i>ACME AIR MICROSERVICE [75]</i>	AuthService	18%	8%	8.7%	34.7%
	CustomerService	22%	8%	12.8%	42.8%
	FlightService	22%	8%	12.8%	42.8%
	BookingService	26%	15%	12.6%	53.6%
<i>E-COMMERCE [76]</i>	Api-Gateway	14%	9%	9%	32%
	Eureka-Server	17%	9%	11.1%	37.1%
	User-Service	13%	9%	9.9%	31.9%
	Product-Catalog-Service	15%	9%	8.2%	32.2%
	Order-Service	16%	13%	9.7%	38.7%
	Product-Recommendation-Service	12%	9%	9.5%	30.5%

El análisis general muestra que ninguno de los repositorios alcanza los valores objetivo-establecidos en el modelo (40% en adaptabilidad, 35% en escalabilidad y 25% en portabilidad). Aunque algunos de los microservicios presentan pequeñas fortalezas principalmente en el atributo de adaptabilidad, los niveles de los demás atributos se mantienen considerablemente bajos, lo cual provoca que la calificación global de la flexibilidad se reduzca para cada uno de los repositorios.

En conjunto, los resultados obtenidos de los tres repositorios evaluados (Zull, Acme Air Microservice y E-Commerce) evidencian una tendencia sistemática de baja flexibilidad en los proyectos analizados. Si bien cada repositorio presenta características particulares, se

observa un patrón común: los valores alcanzados para los subatributos de adaptabilidad, escalabilidad y portabilidad se mantienen por debajo de los niveles objetivo, definidos en el modelo (40%, 35% y 25%, respectivamente). Esta situación presenta indicios que las implementaciones de microservicios en entornos Java, incluso aquellas con arquitecturas maduras y un uso extensivo de frameworks consolidados como Spring Boot o Netflix OSS, no logran cubrir integralmente los aspectos de calidad asociados a la flexibilidad.

La *Figura 22* muestra que, en términos relativos, la adaptabilidad presenta en promedio los valores más estables ya que es el subatributo que mejor valuado está, lo cual indica que los proyectos tienden a mantener cierta coherencia modular y semántica entre servicios. Sin embargo, los resultados revelan debilidades más marcadas en escalabilidad y portabilidad, subatributos que dependen en mayor medida de la existencia de mecanismos de automatización, documentación formal y estrategias de despliegue reproducibles. La baja presencia de los subatributos de escalabilidad y portabilidad demuestra un desequilibrio importante en el diseño arquitectónico de los microservicios, donde el desarrollo se centra principalmente a la funcionalidad individual de los servicios, lo cual reduce la capacidad del sistema para adaptarse y afectando de manera directa la flexibilidad de la arquitectura.

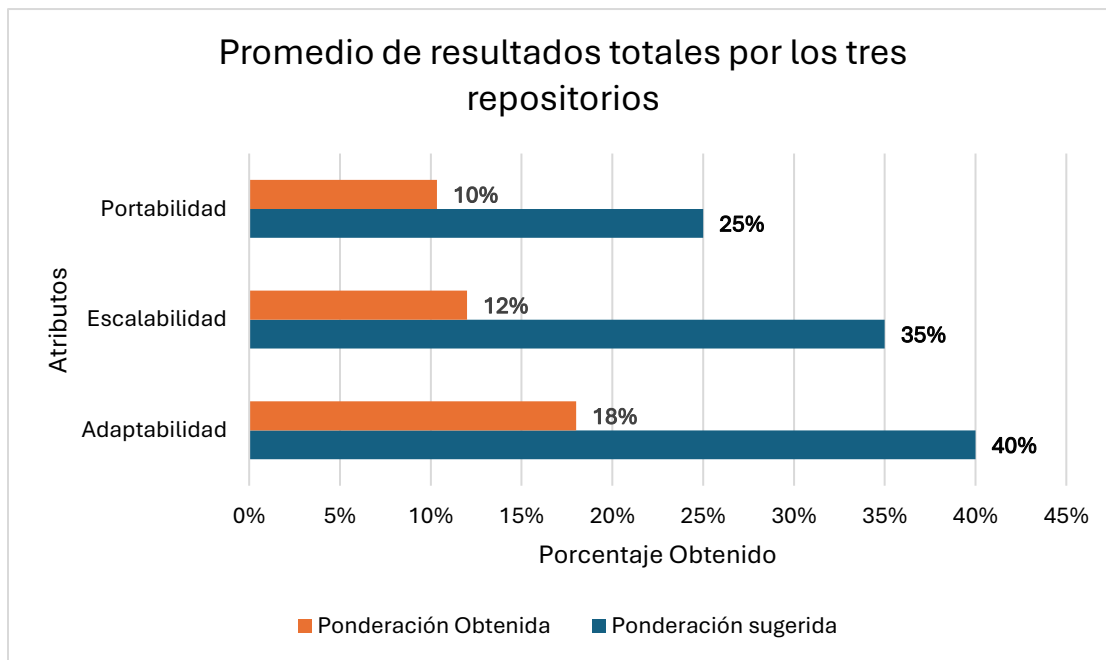


Figura 22 Promedio de resultados totales por los tres repositorios [74], [75], [76]

Por tanto, los resultados del análisis de los tres repositorios refuerzan la validez del modelo propuesto, al permitir identificar y cuantificar las brechas hacia los valores ideales de flexibilidad, lo cual propicia una visión más precisa de las áreas críticas que deben atenderse. Asimismo, se concluye que la flexibilidad no depende de un único subatributo, sino de la interacción equilibrada entre adaptabilidad, escalabilidad y portabilidad, lo que confirma la necesidad de adoptar mejores prácticas para integrar documentación, automatización y diseño desde las etapas iniciales del desarrollo.

Se recomienda la revisión del **Anexo D**, el cual describe más a fondo los resultados obtenidos y recomendaciones identificadas tras la aplicación del modelo para cada uno de los distintos repositorios analizados.

Capítulo 6 Conclusiones y Trabajos Futuros



En este capítulo se presentan las conclusiones derivadas del desarrollo y evaluación del modelo propuesto, destacando los hallazgos obtenidos a lo largo del presente trabajo. Asimismo, se proponen líneas de trabajo futuro con el fin de fortalecer, extender y validar la propuesta de este trabajo desde distintos puntos de vista o contextos.

La presente investigación permitió cumplir de manera satisfactoria con los objetivos específicos que fueron trazados. En primer lugar, se obtuvo un listado exhaustivo de 258 métricas aplicables para los microservicios, las cuales fueron clasificadas según su propósito, evidenciando que si bien hay avances significativos en el entendimiento estructural de las MSA, también existen una limitada madurez en la medición empírica de sus atributos de calidad, ya que, algunas de las métricas tradicionales resultan insuficientes, existen distintas definiciones y por ende distintos enfoques, los cuales complican la estandarización del concepto y su evaluación, lo cual justifica una necesidad de un modelo integrador, como el propuesto que pueda combinar formalismo normativo, operacionalización métrica y aplicación empírica en contextos reales de desarrollo de microservicios. Asimismo, se identificaron y clasificaron los umbrales de referencia reportados en la literatura, con el propósito de homogenizar su análisis y facilitar su comparación. Esta agrupación permitió evidenciar que la mayoría de los trabajos analizados no definen umbrales explícitos para la evaluación de los atributos analizados, lo cual confirma la ausencia de criterios cuantitativos comunes, principalmente en atributos de portabilidad e instalabilidad.

Tras analizar las métricas encontradas y los distintos umbrales para cada una de ellas, se desarrolló un modelo de evaluación el cual constituye una contribución inicial para el estudio de la flexibilidad en arquitecturas de microservicios, al integrar criterios normativos [19], [20], [65], [66] con un enfoque empírico. Su aplicación permitió comprobar que la flexibilidad depende de un conjunto de factores interrelacionados como la documentación, el desacoplamiento, la interoperabilidad y la utilización del despliegue, que deben ser gestionados de forma conjunta para garantizar la calidad del sistema.

Los resultados obtenidos confirman que las implementaciones actuales presentan brechas importantes en portabilidad y más puntualmente en instalabilidad, reflejando la necesidad de incorporar prácticas de ingeniería estandarizadas y mecanismos de documentación más completos. Además, se demuestra que la flexibilidad no es un atributo estático, sino un indicador dinámico de madurez arquitectónica, que puede evolucionar conforme se adapten mejores prácticas de diseño y despliegue. El modelo propuesto aporta valor en este sentido, pues permite visibilizar de manera estructurada las debilidades de los microservicios. La asignación de pesos para cada atributo facilita identificar con mayor precisión qué

componentes tienen mayor impacto sobre la flexibilidad de la arquitectura. No obstante, la determinación de dichos pesos exige un análisis riguroso, ya que implica valorar tanto la importancia relativa de cada subatributo como el grado de relación que guardan entre sí.

Es importante señalar que los resultados obtenidos son preliminares, al estar basados en un conjunto limitado de repositorios seleccionados bajo criterios específicos: proyectos diseñados en lenguaje Java, con el framework Spring Boot y de libre acceso. Asimismo, cabe mencionar que, si bien existen herramientas como MAAT (Microservice Architecture Analysis Tool) [41], que ayudan a analizar aspectos de semántica, refactorización, calidad y arquitectura de microservicios, estas se encuentran fundamentadas en marcos metodológicos de medición, estándares técnicos de rastreo y principios arquitectónicos de microservicios respaldados por la literatura, sin embargo, no se encuentran explícitamente alineados con un estándar de calidad reconocido como el que se presenta en este modelo, ni se enfocan de manera específica en el atributo de flexibilidad, el cual con frecuencia se asume erróneamente como garantizado por el simple hecho de ser un atributo de calidad asociado a la arquitectura de microservicios.

Como hallazgos se encontró una gran variedad de formas para evaluar una misma métrica, es decir, métricas como Acoplamiento, Cohesión, etc. son abordadas desde diferentes puntos de vista y con diferentes definiciones para microservicios, razón por la cual podemos encontrar una gran variedad de formas para implementar dichas métricas en la literatura.

Por otro lado, la gran variedad de formas de implementar a las métricas también se refleja en una gran variedad de umbrales de aceptación para los microservicios, es decir, existen muchas formas de evaluar y por ende, muchos umbrales diferentes. Sin embargo, aunque existen dichos umbrales en la literatura, estos no siempre son presentados por los autores en sus trabajos, lo cual es una de las razones que propician que los resultados obtenidos tras aplicar las métricas no se puedan interpretar, esto propicia que implementar métricas sea una actividad mucho más difícil de lo que realmente es y aquellos autores que sí presentan los umbrales implementados para sus métricas son una pequeña minoría en la academia.

Una de las razones que se presentan a menudo por el cual muy pocas métricas cuentan con umbrales claramente definidos tiene que ver con que a menudo son provenientes de arquitecturas anteriores como las de Orientadas a Servicios, Orientadas a Objetos, etc.

También es debido a que estas métricas por lo general suelen ser propuestas nuevas por parte de la academia, las cuales están pensadas y orientadas directamente a las arquitecturas de microservicios.

➤ **Trabajos futuros**

Como trabajos futuros se propone lo siguiente:

- Realizar una herramienta que pueda automatizar la evaluación de microservicios con base en el modelo propuesto.
- Realizar modelos similares para cada uno de los atributos de calidad propuestos por la ISO/IEC 25010 que puedan ser aplicables a microservicios.
- Realizar un estudio sobre la cantidad de métricas que son adaptaciones y las que son diseñadas para microservicios con el fin de verificar cual está más presente en la literatura y por qué.
- Implementar un benchmark que sirva como punto de referencia tomando en cuenta las diez métricas con diferentes formas de evaluación en la literatura.
- Analizar más repositorios con el fin de generar una estadística que permita verificar si existe la tendencia de no documentar la instalabilidad y reemplazabilidad de los microservicios por parte de los desarrolladores.
- Extender la validación del modelo mediante la aplicación de métodos formales de ponderación.
- Aplicar normas nuevas al modelo propuesto como la ISO 5055 que puedan facilitar y orientar de mejor manera la evaluación de la arquitectura interna de los microservicios.

Capítulo 7 Publicaciones y Actividades Complementarias



En este capítulo se presentan las publicaciones científicas derivadas de esta investigación, las cuales han contribuido a la difusión de los resultados y al fortalecimiento de la formación académica, asimismo, se presentan las actividades complementarias que se realizaron a lo largo del mismo.

7.1. Publicaciones

7.1.1. Metrics Applicable to Microservices: A Systematic Literature Review

Publicado en la revista Tecnología y Ciencia Aplicada, Vol. 8, No. 1, p. 592-598 Enero-Junio 2025, ISSN: 2992-8338.

El artículo presenta una revisión sistemática de la literatura enfocada en las métricas propuestas para el desarrollo e implementación de arquitecturas de microservicios.

7.1.2. Esquema de evaluación de flexibilidad en microservicios basado en atributos de calidad ISO-IEC análisis preliminar de repositorios java

Artículo presentado en las Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC 2025).

El trabajo propone un modelo de calidad para evaluar la flexibilidad en microservicios con base en las normas ISO/IEC, así como los resultados obtenidos tras su aplicación a tres repositorios de microservicios implementados en Java

7.2. Eventos Académicos

7.2.1. Ponencia “Análisis de métricas en el desarrollo de microservicios”

Presentada de manera virtual en la 1 Jornadas de Transferencia de Resultados de Investigación, Ciencia de Datos e Inteligencia Artificial, Colegio Universitario Científico de Datos, 18 de enero de 2026

7.2.2. Ponencia: “Métricas para microservicios”

Presentada en las Jornadas Académicas de Innovación Tecnología, Liderazgo y Sostenibilidad 2025, Instituto Tecnológico de Zacatepec, 14 de octubre de 2025.

7.2.3. Proyecto MONARCA

Participación en el evento InnovaTecNM 2024.

Se obtuvo el primer lugar en la etapa local, celebrada en el TecNM campus CENIDET; primer lugar en la etapa regional, realizada en modalidad virtual; y participación en la etapa nacional, llevada a cabo en el Instituto Tecnológico de Colima.

Referencias

- [1] O. Al-Debagy y P. Martinek, “A metrics framework for evaluating microservices architecture designs”, *Journal of Web Engineering*, vol. 19, núm. 3–4, pp. 341–369, jun. 2020, doi: 10.13052/jwe1540-9589.19341.
- [2] S. Li *et al.*, “Understanding and addressing quality attributes of microservices architecture: A Systematic literature review”, el 1 de marzo de 2021, *Elsevier B.V.* doi: 10.1016/j.infsof.2020.106449.
- [3] S. Lima, J. Correia, F. Araujo, y J. Cardoso, “Improving observability in Event Sourcing systems”, *Journal of Systems and Software*, vol. 181, nov. 2021, doi: 10.1016/j.jss.2021.111015.
- [4] M. Hitz y B. Montazeri, “Chidamber and Kemerer’s Metrics Suite: A Measurement Theory Perspective”, 1996.
- [5] M. Hirzalla, Cleland-Huang Jane, y Arsanjani Ali, “A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures”, dic. 2008. doi: https://doi.org/10.1007/978-3-642-01247-1_5.
- [6] D. Vinay Babu y M. P. Darsi, “A Survey on Service Oriented Architecture and Metrics to Measure Coupling”, ago. 2013.
- [7] D. R. F. Apolinário y B. B. N. de França, “A method for monitoring the coupling evolution of microservice-based architectures”, *Journal of the Brazilian Computer Society*, vol. 27, núm. 1, dic. 2021, doi: 10.1186/s13173-021-00120-y.
- [8] Y. Abgaz *et al.*, “Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review”, *IEEE Transactions on Software Engineering*, vol. 49, núm. 8, pp. 4213–4242, ago. 2023, doi: 10.1109/TSE.2023.3287297.
- [9] K. M. Hernández Guzmán, “Adaptación de un patrón de software en seguridad a la arquitectura de un Microservicio”, oct. 2023.
- [10] J. V Penagos Sánchez, “Estudio de Métricas y Patrones de Seguridad en Microservicios”, ene. 2023.
- [11] I. D. Joya de la Cruz, “Selección y adaptación de métricas para Microservicios”, 2022.
- [12] C. Merino Ibañez, “Metodología en la determinación de la granularidad de un Microservicio”, ene. 2023.

- [13] A. Roque Martínez, “Modelo de calidad para determinar la granularidad de microservicios”, 2024, Consultado: el 2 de octubre de 2025. [En línea]. Disponible en: <https://www.researchgate.net/publication/389960795>
- [14] V. Raj y R. Sadam, “Evaluation of SOA-Based Web Services and Microservices Architecture Using Complexity Metrics”, *SN Comput. Sci.*, vol. 2, núm. 5, sep. 2021, doi: 10.1007/s42979-021-00767-6.
- [15] A. D. Barrios Contreras, “Arquitectura de microservicios Microservice architecture”, 2018. [En línea]. Disponible en: <http://revistas.udistrital.edu.co/ojs/index.php/tia/issue/archive>
- [16] M. Fowler y J. Lewis, “Microservicios”. Consultado: el 30 de septiembre de 2025. [En línea]. Disponible en: <https://martinfowler.com/articles/microservices.html>
- [17] “IEEE SA - IEEE 1061-1998”. Consultado: el 30 de septiembre de 2025. [En línea]. Disponible en: <https://standards.ieee.org/ieee/1061/1549/>
- [18] S. Engineering Standards Committee of the IEEE Computer Society, “ISO/IEC/IEEE 24765-2010(E), Systems and software engineering — Vocabulary”, 2010. [En línea]. Disponible en: www.iso.org
- [19] “ISO 25010”. Consultado: el 30 de septiembre de 2025. [En línea]. Disponible en: <https://iso25000.com/index.php/normas-iso-25000/iso-25010>
- [20] “ISO/IEC 9126 - Wikipedia, la enciclopedia libre”. Consultado: el 18 de noviembre de 2025. [En línea]. Disponible en: https://es.wikipedia.org/wiki/ISO/IEC_9126
- [21] M. Foucault, M. Paylart, J.-R. Falleri, y X. Blanc, *Computing Contextual Metric Thresholds*. ACM, 2014. doi: <https://doi.org/10.1145/2554850.2554997>.
- [22] “What is an API Endpoint? | Definition from TechTarget”. Consultado: el 4 de febrero de 2026. [En línea]. Disponible en: https://www.techtarget.com/searchapparchitecture/definition/API-endpoint?utm_source=chatgpt.com
- [23] N. Dragoni *et al.*, “Microservices: Yesterday, Today, and Tomorrow”, *Present and Ulterior Software Engineering*, pp. 195–216, nov. 2017, doi: 10.1007/978-3-319-67425-4_12.
- [24] Matthias Biehl, *API Architecture: The Big Picture for Building APIs*, Kindle Edition. 2015. Consultado: el 3 de junio de 2025. [En línea]. Disponible en: https://books.google.com.mx/books?hl=es&lr=&id=6D64DwAAQBAJ&oi=fnd&pg=PA15&dq=what+is+API%3F&ots=zb6O4uKd7r&sig=2zP_GAaRqmM9TojJQUHIMQDlykA&redir_esc=y#v=onepage&q=what%20is%20API%3F&f=false

- [25] “¿Qué es Kubernetes? | IBM”. Consultado: el 4 de febrero de 2026. [En línea]. Disponible en: https://www.ibm.com/mx-es/think/topics/kubernetes?utm_source=chatgpt.com
- [26] “Contenedores de Docker | ¿Qué es Docker? | AWS”. Consultado: el 4 de febrero de 2026. [En línea]. Disponible en: <https://aws.amazon.com/es/docker/>
- [27] A. A. Segura, “Arquitectura de Software de Referencia para Objetos Inteligentes en Internet de las Cosas”, 2016.
- [28] P. Merson y Bryant Daniel, “Principles for Microservice Design: Think IDEALS, Rather than SOLID - InfoQ”. Consultado: el 5 de octubre de 2025. [En línea]. Disponible en: <https://www.infoq.com/articles/microservices-design-ideals/>
- [29] N. Fenton y J. Bieman, “Software Metrics A Rigorous and Practical Approach”, 2014.
- [30] “¿Qué es la contenerización? | IBM”. Consultado: el 15 de febrero de 2026. [En línea]. Disponible en: <https://www.ibm.com/mx-es/think/topics/containerization>
- [31] M. Hirzalla, J. Cleland-Huang, y A. Arsanjani, “A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5472 LNCS, pp. 41–52, 2009, doi: 10.1007/978-3-642-01247-1_5.
- [32] S. R. Chidamber y C. F. Kemerer, “A Metrics Suite for Object Oriented Design”, 1994.
- [33] O. Al-Debagy y P. Martinek, “A new decomposition method for designing microservices”, *Periodica polytechnica Electrical engineering and computer science*, vol. 63, núm. 4, pp. 274–281, 2019, doi: 10.3311/PPee.13925.
- [34] F. Driessen Supervisors, L. Ferreira Pires, J. Luiz Rebelo Moreira, A. Sperotto, S. van den Bosch, y P. Verhoeven June, “A quantitative assessment method for microservices granularity to improve maintainability”, jun. 2023.
- [35] E. Ntentos, U. Zdun, K. Plakidas, S. Meixner, y S. Geiger, “Assessing architecture conformance to coupling-related patterns and practices in microservices”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12292 LNCS, pp. 3–20, 2020, doi: 10.1007/978-3-030-58923-3_1;TAXONOMY:TAXONOMY:ACM-PUBTYPE;PAGEGROUP:STRING:PUBLICATION.

- [36] J. Bogner, S. Wagner, y A. Zimmermann, “Automatically measuring the maintainability of service- and microservice-based systems - a literature review”, en *ACM International Conference Proceeding Series*, Association for Computing Machinery, oct. 2017, pp. 107–115. doi: 10.1145/3143434.3143443.
- [37] M. Söylemez, B. Tekinerdogan, y A. K. Tarhan, “Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review”, el 1 de junio de 2022, *MDPI*. doi: 10.3390/app12115507.
- [38] D. Athanasopoulos, A. V Zarras, G. Miskos, V. Issarny, y P. Vassiliadis, “Cohesion-Driven Decomposition of Service Interfaces Without Access to Source Code”, sep. 2013. doi: 10.1109/TSC.2014.2310195.
- [39] M. Perepletchikov, C. Ryan, y K. Frampton, “Cohesion Metrics for Predicting Maintainability of Service-Oriented Software”, 2007.
- [40] F. H. Vera Rivera, C. Gaona, y H. Astudillo, “Defining and measuring microservice granularity—a literature overview”, *PeerJ Comput. Sci.*, vol. 7, p. e695, sep. 2021, doi: 10.7717/peerj-cs.695.
- [41] T. Engel, M. Langermeier, B. Bauer, y A. Hofmann, “Evaluation of microservice architectures: A metric and tool-based approach”, en *Lecture Notes in Business Information Processing*, Springer Verlag, 2018, pp. 74–89. doi: 10.1007/978-3-319-92901-9_8.
- [42] O. Al-Debagy y P. Martinek, “Extracting Microservices’ Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach”, en *SOSE 2020 - IEEE 15th International Conference of System of Systems Engineering, Proceedings*, Institute of Electrical and Electronics Engineers Inc., jun. 2020, pp. 289–293. doi: 10.1109/SoSE50414.2020.9130466.
- [43] M. H. Hasan, M. H. Osman, N. I. Admodisastro, y M. S. Muhammad, “From Monolith to Microservice: Measuring Architecture Maintainability”, *International Journal of Advanced Computer Science and Applications*, vol. 14, núm. 5, pp. 857–866, 2023, doi: 10.14569/IJACSA.2023.0140591.
- [44] D. Taibi y K. Systä, “From monolithic systems to microservices: A decomposition framework based on process mining”, en *CLOSER 2019 - Proceedings of the 9th International Conference on Cloud Computing and Services Science*, SciTePress, 2019, pp. 153–164. doi: 10.5220/0007755901530164.
- [45] W. Jin, T. Liu, Q. Zheng, D. Cui, y Y. Cai, “Functionality-Oriented Microservice Extraction Based on Execution Trace Clustering”, en *Proceedings - 2018 IEEE International Conference on Web Services, ICWS 2018 - Part of the 2018 IEEE*

World Congress on Services, Institute of Electrical and Electronics Engineers Inc., sep. 2018, pp. 211–218. doi: 10.1109/ICWS.2018.00034.

- [46] Y. Li, C. Z. Wang, Y. C. Li, J. Su, y C. H. Chen, “Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing”, *Discrete Dyn. Nat. Soc.*, vol. 2020, 2020, doi: 10.1155/2020/1057902.
- [47] Y. H. Chen, N. J. Chen, W. X. Xu, L. M. Lian, y H. Tu, “MFRL-CA: Microservice Fault Root Cause Location based on Correlation Analysis”, en *Proceedings - 2021 8th International Conference on Dependable Systems and Their Applications, DSA 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 90–101. doi: 10.1109/DSA52907.2021.00018.
- [48] U. Zdun *et al.*, “Microservice Security Metrics for Secure Communication, Identity Management, and Observability”, *ACM Transactions on Software Engineering and Methodology*, vol. 32, núm. 1, feb. 2023, doi: 10.1145/3532183.
- [49] D. Booklet, O. Al-Debagy, y P. Martinek, “Microservices Identification Methods and Quality Metrics”, 2021.
- [50] M. Cinque, R. Della Corte, y A. Pecchia, “Microservices Monitoring with Event Logs and Black Box Execution Tracing”, *IEEE Trans. Serv. Comput.*, vol. 15, núm. 1, pp. 294–307, 2022, doi: 10.1109/TSC.2019.2940009.
- [51] M. Cojocaru, A. Uta, y A. Oprescu, “MicroValid: A Validation Framework for Automatically Decomposed Microservices”, dic. 2019. doi: 10.1109/CloudCom.2019.00023.
- [52] F. H. Vera Rivera, “Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios”, p. 152, 2021.
- [53] F. Pina, J. Correia, R. Filipe, F. Araujo, y J. Cardoso, *Nonintrusive Monitoring of Microservice-based Systems*. IEEE, 2018. doi: 10.1109/NCA.2018.8548311.
- [54] A. Souza, N. Cacho, A. Noor, P. P. Jayaraman, A. Romanovsky, y R. Ranjan, “Osmotic Monitoring of Microservices between the Edge and Cloud”, en *Proceedings - 20th International Conference on High Performance Computing and Communications, 16th International Conference on Smart City and 4th International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2018*, Institute of Electrical and Electronics Engineers Inc., ene. 2019, pp. 758–765. doi: 10.1109/HPCC/SmartCity/DSS.2018.00129.
- [55] A. Hannousse y S. Yahiouche, “Securing Microservices and Microservice Architectures: A Systematic Mapping Study”, jun. 2020, doi: 10.1016/j.cosrev.2021.100415.

- [56] D. Ulander, “Software Architectural Metrics for the Scania Internet of Things Platform From a Microservice Perspective”, jun. 2017. [En línea]. Disponible en: <http://www.teknat.uu.se/student>
- [57] S. Panichella, M. I. Rahman, y D. Taibi, “Structural Coupling for Microservices”, abr. 2021. doi: 10.5220/0010481902800287.
- [58] J. Soldani, D. A. Tamburri, y W. J. Van Den Heuvel, “The pains and gains of microservices: A Systematic grey literature review”, *Journal of Systems and Software*, vol. 146, pp. 215–232, dic. 2018, doi: 10.1016/j.jss.2018.09.082.
- [59] I. Candela, G. Bavota, B. Russo, y R. Oliveto, “Using cohesion and coupling for software remodularization: Is it enough?”, *ACM Transactions on Software Engineering and Methodology*, vol. 25, núm. 3, may 2016, doi: 10.1145/2928268.
- [60] A. Marcus, D. Poshyvanyk, y R. Ferenc, “Using the conceptual cohesion of classes for fault prediction in object-oriented systems”, *IEEE Transactions on Software Engineering*, vol. 34, núm. 2, pp. 287–300, mar. 2008, doi: 10.1109/TSE.2007.70768.
- [61] A. Bucchiarone *et al.*, “From Monolithic to Microservices An Experience Report from the Banking Domain”, 2018. doi: 10.1109/MS.2018.2141026.
- [62] E. Ntontos, U. Zdun, K. Plakidas, S. Meixner, y S. Geiger, “Assessing Architecture Conformance to Coupling-Related Patterns and Practices in Microservices”, sep. 2020. doi: 10.1007/978-3-030-58923-3_1.
- [63] “IEEE SA - IEEE 1063-2001”. Consultado: el 21 de agosto de 2025. [En línea]. Disponible en: <https://standards.ieee.org/ieee/1063/1554/>
- [64] B. Kitchenham y S. Lawrence Pfleeger, “Software quality: The elusive target”, ene. 1996, doi: 10.1109/52.476281.
- [65] “ISO/IEC 11179-1:2015 - Information technology — Metadata registries (MDR) — Part 1: Framework”. Consultado: el 3 de junio de 2025. [En línea]. Disponible en: <https://www.iso.org/standard/61932.html>
- [66] “ISO/IEC 27001:2022 - Information security management systems”. Consultado: el 3 de junio de 2025. [En línea]. Disponible en: <https://www.iso.org/standard/27001>
- [67] “ISO/IEC 2382:2015 - Tecnologías de la información — Vocabulario”. Consultado: el 26 de noviembre de 2025. [En línea]. Disponible en: <https://www.iso.org/standard/63598.html>
- [68] D. Taibi y V. Lenarduzzi, “On the Definition of Microservice Bad Smells”, 2018. doi: 10.1109/MS.2018.2141031.

- [69] S. Newman, “Building Microservices”, 2015. [En línea]. Disponible en: <http://safaribooksonline.com>
- [70] I. Nadareishvili, R. Mitter, M. McLarty, y M. Amundsen, “Microservice Architecture: Aligning Principles, Practices and Culture”, p. 146, ago. 2016.
- [71] R. E. Al-Qutaish y A. Ain, “Quality Models in Software Engineering Literature: An Analytical and Comparative Study”, 2010. [En línea]. Disponible en: <http://www.americanscience.org/editor@americanscience.org166>
- [72] M. Minella, “State of Spring Survey 2024 Results”. Consultado: el 28 de octubre de 2025. [En línea]. Disponible en: https://spring.io/blog/2024/06/03/state-of-spring-survey-2024-results?utm_source=chatgpt.com
- [73] S. Sõnajalg, “Microservices in Java: Survey Highlights | JRebel”. Consultado: el 28 de octubre de 2025. [En línea]. Disponible en: https://www.jrebel.com/blog/microservices-in-java-survey?utm_source=chatgpt.com
- [74] “Netflix/zuul”. Consultado: el 25 de noviembre de 2025. [En línea]. Disponible en: <https://github.com/Netflix/zuul>
- [75] “AcmeAir_Microservice/AcmeAir-Microservices”. Consultado: el 25 de noviembre de 2025. [En línea]. Disponible en: https://github.com/anitagoel/AcmeAir_Microservice/tree/main/AcmeAir-Microservices/AcmeAir-master
- [76] “RainbowForest/e-commerce-microservices”. Consultado: el 25 de noviembre de 2025. [En línea]. Disponible en: <https://github.com/RainbowForest/e-commerce-microservices/>
- [77] P. V. Torres-Carrion, C. S. González-González, S. Aciar, y G. Rodríguez-Morales, *Methodology for Systematic Literature Review applied to Engineering and Education*. IEEE, 2018. doi: 10.1109/EDUCON.2018.8363388.

Anexos

Anexo A. Formas de evaluación de las métricas

La tabla 10 presenta el listado de 35 métricas y para cada una de ellas las variaciones o complementos que corresponden a cada una, así mismo se listan las referencias de las cuales han sido recopiladas.

Tabla 15 Formas de evaluación de las métricas

Métrica	Formas de evaluación
Acoplamiento [6], [7], [14], [31], [32], [34], [36], [45], [51], [52], [56], [57], [59], [61], [62]	Acoplamiento entre clases de objetos
	Acoplamiento entrante extra-servicio ponderado de un elemento
	Acoplamiento saliente extra-servicio ponderado de un elemento
	Acoplamiento entrante de interfaz de elemento a servicio adicional
	Acoplamiento saliente de interfaz de elemento a servicio adicional
	Interfaz de servicio para acoplamiento intraelemento
	Factor de acoplamiento del servicio
	Acoplamiento del servicio
	Acoplamiento relativo del servicio
	Acoplamiento de un sistema
	Acoplamiento del sistema
	Acoplamiento indirecto del servicio
	Acoplamiento indirecto relativo de un servicio
	Promedio de acoplamiento del servicio
	Acoplamiento entre microservicios
	Acoplamiento de cambio bidireccional
	Acoplamiento de cambio direccional
	Acoplamiento estructural
	Acoplamiento
	Acoplamiento Relativo de Servicios
	Factor de Acoplamiento de Servicios
	Vector que contiene las métricas de acoplamiento calculadas para MSBA
	Acoplamiento estructural
	Acoplamiento conceptual entre métodos
	Acoplamiento conceptual entre un método y una clase
	Acoplamiento conceptual entre clases
	Acoplamiento conceptual
	Acoplamiento estructural
	Acoplamiento de los servicios
	Acoplamiento relativo de los servicios
	Factor de acoplamiento del servicio
	Número promedio de servicios conectados directamente
	Inestabilidad
	Compartir servicios directamente
	Número ponderado de interfaces de servicio
	Recuento ponderado de la interfaz de servicios
	Recuento de interfaz de servicio ponderado
	Recuento de interfaces de servicio ponderado
	Componentes fuertemente conectados
	Número de llamadas a métodos entre dos servicios.

<i>Coeficiente de Silhouette</i> <i>[33]</i>	Coeficiente de Silhouette
<i>Cohesión</i> <i>[1], [32], [34], [36], [38], [39], [42],</i> <i>[45], [46], [49], [52], [56], [59],</i> <i>[60]</i>	Cohesión
	Falta de cohesión
	Promedio de la cohesión del servicio
	Promedio de cohesión de las operaciones
	Falta de cohesión a nivel de mensajes
	Falta de cohesión a nivel de conversación
	Falta de cohesión a nivel de dominio
	Cohesión de datos en la interfaz de servicio
	Falta de cohesión
	Grado de falta de cohesión
	Vector de cohesión
	Cohesión total para MSBA
	Falta de cohesión estructural
	Falta de cohesión conceptual
	Cohesión de datos de la interfaz de servicio
	Cohesión del uso de la interfaz de servicio
	Cohesión total de la interfaz de servicio
	Cohesión de datos de la interfaz de servicio
	Cohesión del uso de la interfaz de servicio
	Cohesión del uso secuencial del servicio
	Cohesión estricta en la implementación del servicio
	Cohesión de implementación de servicios flexibles
	Cohesión total de la interfaz de un servicio
	Falta de cohesión
	Media de la falta de cohesión
	Falta de cohesión
	Falta de cohesión
	Cohesión media de las interfaces de servicio a nivel de mensajes.
	Cohesión media de las interfaces de servicio a nivel de dominio
	Cohesión de datos de interfaz
	Cohesión de uso de interfaz
	Cohesión total de interfaz
	Frecuencia de cambios
<i>Complejidad</i> <i>[32], [42], [52], [56]</i>	Complejidad del sistema
	Complejidad del servicio
	Puntos de complejidad cognitiva
	Complejidad
	Métodos ponderados por clase
	Número de hijos
	Profundidad del árbol de herencia
	Dependencia absoluta del servicio

<i>Dependencias</i> [7], [36], [41], [45], [46], [51], [52], [56], [62]	Distribución de dependencia de servicios
	Interdependencia de servicios
	Detección de dependencias cíclicas
	Norma del vector de la dependencia absoluta del microservicio
	Valor total de dependencia absoluta del microservicio
	Valor total de interdependencia de los microservicios
	Interdependencia de servicios en el sistema
	Dependencia absoluta del servicio
	Composición de dependencias
	Número de dependencias asíncronas
	Dependencia entre la clase compartida y los clústeres.
	Dependencia del servicio
<i>Diseño del servicio</i> [31], [36], [52]	Aplicación basada en microservicios
	Servicios sin estado
	Patrón de realización del servicio
	Soporte de servicio para transacciones
	Patrón de composición del servicio
	Soporte de servicio para transacciones
	Método de acceso al servicio
	Selección dinámica VS selección estática de servicios
<i>Disponibilidad</i> [2], [33], [47], [50], [54]	Memoria disponible
	Disponibilidad del sistema
	Disponibilidad del dispositivo
	Disponibilidad de mensajes
	Disponibilidad propia
	Disponibilidad
	Tasa de éxito de la solicitud
	Tasa de error
	Confianza
<i>Factor de partición del sistema</i> [6]	Factor de partición del sistema
<i>Factor de pureza</i> [6]	Factor de pureza
<i>Factor de peso</i> [57]	Factor de peso local
	Factor de peso global
<i>Granularidad</i> [1], [45], [49], [52], [56]	Granularidad del servicio
	Granularidad
	Número de operaciones por servicio
	Granularidad de datos de un servicio
	Granularidad funcional de un servicio
	Promedio de granularidad
	Granularidad de datos de un servicio
	Granularidad funcional de un servicio

	Granularidad
	Número de operaciones por servicio
	Número de operaciones
	Número medio de interfaces que un servicio extraído proporciona a otros servicios.
	Número de operaciones proporcionadas por los microservicios extraídos
<i>Historias de usuario</i> [52]	Número máximo de historias de usuario asociadas a un microservicio
<i>Importancia</i> [7], [31], [36], [46], [47], [52], [56]	Puntos de historia del usuario del microservicio
	Importancia absoluta del servicio
	Distribución de la importancia del servicio
	Importancia del servicio
	Importancia relativa del servicio
	Número de versiones por servicio
	Importancia absoluta para cada microservicio
	Valor total de importancia absoluta del microservicio
	Importancia absoluta del servicio
	Importancia del servicio
	Criticidad absoluta de un servicio
	Criticidad absoluta del servicio
	Criticidad del servicio
	Soporte
	Número de versiones por servicio
<i>Interacciones con bases de datos compartidas</i> [62]	Interacciones con bases de datos compartidas
<i>Interacción de servicio a través de un componente intermediario</i> [2], [47], [52], [62]	Interacción de servicio a través de un componente intermediario
	Promedio de llamadas
	Frecuencia de llamadas al servicio
	Probabilidad de acceso del nodo
	Probabilidad de acceso hacia adelante
	Probabilidad de acceso hacia atrás
	Probabilidad de permanencia
<i>Interdependencia</i> [52], [56]	Centralización del sistema
<i>Latencia</i> [2], [41], [50], [53]	Número de pares de microservicios
	Latencia media de solicitud
	Latencia HTTP
	Duration
<i>Media F</i> [33]	Traza de llamada síncrona más larga
<i>Modularización</i> [51], [52], [59]	Media F
	Calidad de la modularización
	I-ésimo microservicio
	Composición de relación

	Composición de nanoentidades
	Composición de las entidades
	Composición de relación
<i>Medida de eficacia de MoJo</i> [59]	Medida de eficacia de MoJo
<i>Monitoreabilidad</i> [47]	Algoritmo de puntuación de excepciones
	Puntuación de anomalía c-SBD
	Puntuación de excepción
	Puntuación de anomalía del nodo
<i>Observabilidad</i> [47], [48], [51], [52], [53]	Servicios del sistema observados
	Componentes de la fachada observados
	Servicios y fachadas del sistema observado
	Origin IP
	Origin Port
	Destination Service
	Destination Instance
	Destination IP
	Destination Function
	Vector que contiene de las métricas calculadas para MSBA
	Coeficiente de variación
	Causalidad entre x e y
	Análisis de causalidad de Granger bidimensional.
	Causalidad de Granger
	Causalidad de Granger entre los datos de series temporales X e Y
	Correlación cruzada con desplazamiento w
	Coeficiente de correlación cruzada normalizada
<i>Precisión</i> [33], [47]	Precisión individual del objeto
	Precisión en los K primeros
	Precisión media promedio
	Número de nodos falsos revisados antes del verdadero
	Falsos positivos promedio
<i>Recall</i> [33]	Recall
<i>Rendimiento</i> [2], [50], [54]	Gastos generales de rendimiento
	Rendimiento HTTP
	Cantidad de bytes recibidos
	Cantidad de bytes transmitidos
	Gastos generales del tamaño del registro
<i>Responsabilidad</i> [33]	Responsabilidad
<i>Respuesta</i> [2], [6], [32], [36]	Respuesta para una clase
	Respuesta para la Operación
	Respuesta total del servicio

<i>Seguridad</i> [48]	Función común para conectores backend distribuidos
	Función común para conectores backend distribuidos que requieren autenticación.
	Conectores de backend autenticados
	Conectores de backend autenticados con claves API
	Conectores de backend autenticados con credenciales de texto sin formato
	Función común para calcular los conectores en la ruta del cliente/interfaz de usuario al servicio que requiere autenticación.
	Conectores de ruta de servicio IU/cliente autenticados
	Conectores de ruta de cliente/IU-servicio autenticados con claves API
	Conectores de ruta de cliente/IU-servicio autenticados con credenciales de texto sin formato
	Conectores distribuidos seguros
	Conectores de cliente seguros
	Conectores de interfaz de usuario seguros
	Conectores externos seguros de cliente/IU
	Conectores distribuidos internos seguros
	Conectores de backend autenticados de forma segura
	Conectores de backend autenticados a través de una conexión segura
	Conectores de backend autenticados que utilizan un método o una conexión seguros
	Conectores de ruta de servicio IU/cliente autenticados de forma segura
	Conectores de ruta de cliente/IU-servicio a través de una conexión segura
	Conectores de ruta de cliente/IU-servicio que utilizan un método o conexión seguros
<i>Servicios compartidos de forma transitiva</i> [62]	Servicios compartidos de forma transitiva
<i>Similitud</i> [38], [51], [52], [60]	Similitud conceptual entre metodos
	Similitud conceptual promedio de los métodos en una clase
	Similitud a nivel de mensaje
	Similitud a nivel de conversación
	Similitud a nivel de dominio
	Diccionario de similitud semántica entre las historias de usuario
	Similitud Semántica del i-ésimo microservicio
	Similitud Semántica Total
	Similaridad semántica
<i>Tamaño</i> [14], [31], [34], [41], [46], [50], [51], [56]	Número de Servicios
	Tamaño promedio de mensajes asíncronos
	Tamaño del servicio
	Número de servicios
	Líneas de código
	Líneas de código
	Número de servicios

	Tamaño del sistema de microservicios
<i>Tiempo</i> <i>[2], [3], [31], [50], [52], [53]</i>	Tiempo de desarrollo de cada microservicio
	Tiempo de desarrollo de la aplicación
	Tiempo de finalización
	Tiempo de respuesta del servicio
	Tiempo de ejecución
	Start Time
	End Time
	Número de tareas humanas
	Duración del conjunto de pruebas
<i>Tipo de base de datos</i> <i>[62]</i>	Utilización del tipo de base de datos
<i>Uso de recursos</i> <i>[2], [54]</i>	Uso de recursos
	Uso del CPU
	Uso de memoria
	Utilización del CPU
<i>Utilización de la comunicación</i> <i>asíncrona</i> <i>[41], [62]</i>	Utilización de la comunicación asíncrona
	Número de interfaces asíncronas
	Distribución de llamadas síncronas
	Número de ciclos síncronos

Anexo B. Métricas encontradas en la literatura

La *Tabla 11* presenta las métricas encontradas en la literatura consultada, así como sus umbrales y el artículo al cual pertenecen.

Ref. Art.	Métrica	Formula	Umbrales
[60]	Similitud conceptual entre métodos	$CSM(m_k, m_j) = \frac{V_{mk}^T V_{mj}}{ V_{mk} \times V_{mj} }$ V_{mk} y V_{mj}: Son los vectores LSI de los métodos	Entre -1 y 1
	Similitud conceptual promedio de los métodos en una clase	$ACSM(c) = \frac{1}{N} \sum_{i=1}^N CSM(m_i, m_j)$ N: Es el número de pares de métodos de clase	No especifica
	Cohesión conceptual de las clases	$C3(c) = \begin{cases} ACSM(c) & \text{si } ACSM(c) > 0 \\ 0, & \text{En otro caso} \end{cases}$	Entre 0 y 1
[54]	Uso del CPU	Porcentaje del uso del CPU por un proceso o sistema	No especifica
	Uso de memoria	Cantidad de memoria usada en MB por un proceso o por el sistema	No especifica
	Memoria disponible	Cantidad de memoria disponible en MB en el sistema	No especifica
	Disponibilidad del sistema	Indicador binario que indica si el sistema (Microservicio o contenedor) está disponible	No especifica, solo devuelve verdadero o falso
	Cantidad de bytes recibidos	Cantidad de bytes recibidos en KB por un sistema o interfaz de red	No especifica
	Cantidad de bytes transmitidos	Cantidad de bytes subidos en KB por un sistema o interfaz de red	No especifica
	Disponibilidad del dispositivo	Indicador binario que indica disponibilidad de un dispositivo IoT	No especifica
[32]	Métodos ponderados por clase	$MC = \sum_{i=1}^n C_i$ n: Número de métodos en la clase Ci: Complejidad del método	No se especifican umbrales exactos, pero valores altos indican mayor complejidad y menor cohesión.

Ref. Art.	Métrica	Formula	Umbrales
	Número de hijos	NOC = Número de subclases directas de la clase	Bajo: 0-2 (poca reutilización). Alto: >10 (Riesgo de mala abstracción)
	Acoplamiento entre clases de objetos	CBO = Número de clases acopladas a la clase evaluada	Bajo: 0-5 (buena modularidad). Alto: > 10 (Dificultad para reutilizar y mantener)
	Respuesta para una clase	$RFC = RS $ RS: Conjunto de métodos que pueden ser invocados en respuesta a un mensaje recibido por un objeto de la clase (incluye métodos locales y externos llamados por estos).	Bajo: <20 (fácil de probar). Alto: >50 (complejidad elevada, requiere más pruebas).
	Falta de cohesión	$LCOM = P - Q $ (Si $ P > Q $) o 0 en caso contrario P: Pares de métodos que no comparten variables de instancia. Q: Pares de métodos que comparten variables de instancia	Bajo: 0 (alta cohesión). Alto: >1 (baja cohesión, posible necesidad de dividir la clase).
	Profundidad del árbol de herencia	DIT = Profundidad de la clase en el árbol de herencia	Bajo: 0-2 (jerarquía plana). Alto: >5 (jerarquía profunda, puede ser difícil de mantener).
[6]	Acoplamiento intraservicio	$WISCE(e) = \{ \langle e, e_i \rangle \times Weight Factor \mid e, e_i \in (C_s \cup I_s \cup P \cup H \cup BPS) \wedge (\langle e, e_i \rangle \vee \langle e_i, e \rangle) \in ISR(s) \} $ e, e_i : Elementos de implementación (Clases, interfaces, etc.) dentro del servicio s	No especifica, pero valores

Ref. Art.	Métrica	Formula	Umbrales
	ponderado entre elementos	ISR(s): Relaciones internas de servicio s C_s, I_s, P, H, BPS : Componentes del servicio (Clases, interfaces, etc).	bajos son deseables para un diseño modular
	Acoplamiento entrante extra-servicio ponderado de un elemento	$WESICE(e) = \{ \langle e, e_i \rangle \times Weight Factor \mid e \in (C - C_s \cup I - I_s \cup \dots) \wedge e_i \in (C_s \cup I_s \cup \dots) \wedge \langle e, e_i \rangle \in IR(s) \} $ IR(s): Relaciones entrantes desde elementos externos al servicio s	No especifica, pero valores bajos indican menor fragilidad ante cambios externos.
	Acoplamiento saliente extra-servicio ponderado de un elemento	$WESICE(e) = \{ \langle e, e_i \rangle \times Weight Factor \mid e \in (C_s \cup I_s \cup \dots) \wedge e_i \in (C - C_s \cup I - I_s \cup \dots) \wedge \langle e, e_i \rangle \in OR(s) \} $ OR(s): Relaciones salientes del servicio s	Similar a WESICE, valores bajos son preferibles.
	Acoplamiento entrante de interfaz de servicio extra-servicio	$ESICSI(s) = SIR(s) $ SIR(s): Relaciones entrantes a la interfaz del servicio s	Interfaces con ESICSI >10 pueden ser problemáticas.
	Acoplamiento saliente de interfaz de elemento a servicio adicional	$EESIOC(e) = \{ \langle e, s_i \rangle \mid e \in (C_s \cup I_s \cup \dots) \wedge s_i \in SI \wedge \langle e, s_i \rangle \in SOR(s) \} $ SI: Conjunto de interfaces de servicios SOR(s): Relaciones salientes a interfaces externas	Valores bajos favorecen la independencia.
	Interfaz de servicio para acoplamiento intraelemento	$SIIEC(s) = IIR(s) $ IIR(s): Relaciones entre la interfaz y elementos internos del servicio	<5
	Factor de partición del sistema	$SPARF(SOS) = \frac{ BPSSER \cup CSER \cup \dots }{ C \cup I \cup \dots }$ SER: Conjunto de servicios en el sistema	0.7 – 1
	Factor de pureza del sistema	$SPURF(SOS) = 1 - \frac{ IS(SOS) }{ SER }$ IS(SOS): Servicios con elementos compartidos	>0.8
	Respuesta para la Operación	$RFO(o) = CS(o) $ CS(o): Secuencia de colaboración para la operación o	<5
[7]	Importancia absoluta del servicio	AIS = Número de consumidores que invocan al menos una operación del servicio	Cuanto mayor sea el

Ref. Art.	Métrica	Formula	Umbrales
			resultado, más importante es el servicio
	Dependencia absoluta del servicio	ADS = Número de servicios de los que depende el servicio	Cuanto mayor sea el ADS, más dependerá este servicio de otros servicios
	Factor de acoplamiento del servicio	$SCF = \frac{SC}{N^2 - N}$ SC: Suma de todas las dependencias entre servicios N: Número total de servicios	No especifica
	Número promedio de servicios conectados directamente	$ADCS = AVG(ADS)$ ADS: Dependencia absoluta del servicio	No especifica
	Distribución de la importancia del servicio	$G = \frac{\sum_{i=1}^n (2i - n - 1)x_i}{n \sum_{i=1}^n x_i}$ $G = \frac{\sum_{i=1}^n (2i - n - 1)x_i}{n \sum_{i=1}^n x_i}$ x_i : Valores de AIS ordenados N: Número de servicios I: Rango del valor x_i	0 - 1
	Distribución de dependencia de servicios	$G = \frac{\sum_{i=1}^n (2i - n - 1)x_i}{n \sum_{i=1}^n x_i}$ x_i : Valores de ADS ordenados N: Número de servicios I: Rango del valor x_i	0-1
[56]	Número de Servicios	$NS = \sum_{s \in \Omega} s$ Ω : Conjunto de servicios del sistema S: un servicio individual	No especifica
	Granularidad del servicio	$SG(s) = \sum_{o \in S} o$ O: Una operación dentro del servicio S: Conjunto de operaciones en el servicio	No especifica
	Complejidad del sistema	$CXSYS1 = NSO + 1.5 * NAO$ NAO: Número de operaciones asíncronas NSO: Número de operaciones síncronas	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Número de operaciones	$NO = \sum_{o \in \Omega} o$ Ω : Conjunto de servicios del sistema o: Una operación dentro del servicio	No especifica
	Importancia del servicio	$IS(s)$ $IS(s)$: Número de servicios que dependen del servicio s	Cuanto mas alto el valor, más complejo el servicio
	Importancia relativa del servicio	$RIS(s) = \frac{IS(s)}{NS}$ $IS(s)$ = Importancia del servicio s NS: Número total de servicios en el sistema	No especifica
	Acoplamiento del servicio	$CS(s)$ $CS(s)$: Número de servicios de los que depende el servicio	Un valor alto aumenta la complejidad del sistema
	Criticidad absoluta de un servicio	$ACS(s) = IS(s) * CS(s)$ $IS(s)$: Número de servicios que dependen del servicio s $CS(s)$: Número de servicios de los que depende el servicio	Un valor alto aumenta la complejidad del sistema
	Inestabilidad	$Ins = \frac{CS(s)}{IS(s) + CS(s)}$ $CS(s)$: Número de servicios de los que depende el servicio $IS(s)$: Número de servicios que dependen del servicio s	0-1 Los valores bajos indican una alta resistencia al cambio.
	Acoplamiento relativo del servicio	$RCS(s) = \frac{CS(s)}{NS}$ $CS(s)$: Acoplamiento del servicio s NS: Número total de servicios en el sistema	0-1 Valores cercanos a 1 indica que necesita rediseño del sistema
	Acoplamiento de un sistema	$CSY S1 = \frac{\sum_{s \in \Omega} CS(s)}{NS^2 - NS}$ s:Un servicio	0-1 Valores bajos indican un

Ref. Art.	Métrica	Formula	Umbral
		Ω : Conjunto de servicios del sistema $CS(s)$: Acoplamiento del servicio s NS : Número total de servicios en el sistema	sistema débilmente acoplado Valores altos indican un sistema dependiente con alta complejidad
	Acoplamiento del sistema	$CSY S2 = \frac{\sum_{s \in \Omega} CS(s)}{SC * SP}$ s: Un servicio Ω: Conjunto de servicios del sistema $CS(s)$: Acoplamiento del servicio s SC: Número de consumidores de servicios en el sistema SP: Número de proveedores de servicios en el sistema	0-1 Un valor de 0 no es razonable porque los servicios deben comunicarse por definición Un valor cercano a 1 implica que el sistema es complejo y difícil de modificar
	Acoplamiento indirecto del servicio	$ICS(s) = CS(s) + \sum_{p(s) \in \Omega} ICS(p(s))$ s: Un servicio Ω: Conjunto de servicios del sistema $p(s)$: Proveedores del servicio s $CS(s)$: Acoplamiento directo del servicio s	No especifica
	Acoplamiento indirecto relativo de un servicio	$RICS(s) = \frac{ICS(s)}{f_1^2 - f_1}$ $ICS(s)$: Acoplamiento indirecto del servicio f_1: Número de servicios (NS) + Número de operaciones (NO)	No especifica
	Promedio de acoplamiento del servicio	$AvgCS = \frac{SP + SC}{NS}$ SP: Número de proveedores de servicios en el sistema	No especifica

Ref. Art.	Métrica	Formula	Umbrales
		SC: Número de consumidores de servicios en el sistema NS: Número de servicios	
	Centralización del sistema	$SCZ = 1 - \frac{\sum_{c \in \Omega.C} CS(c) - \sum_{a \in \Omega.A} CS(a) - k}{\sum_{c \in \Omega.C} CS(c) + (SA - 1)^2}$ $k = 0.9 * (SC-SA)-(SA - 1)^2$ $\Omega.C$: Conjunto de servicios consumidores $\Omega.A$: Conjunto de servicios agregadores SA: Número de servicios agregadores SC: Número de consumidores de servicios en el sistema	0-1 Valores cercanos a 1 indican un sistema centralizado (flexible) Valores cercanos a 0 indican un sistema descentralizado (menos flexible)
	Promedio de la cohesión del servicio	$AvgCohS = \frac{NS}{TM}$ NS: Número de servicios TM: Total de mensajes	Cuanto menos mensajes se envíen y más servicios haya en el sistema, mayor cohesión
	Promedio de cohesión de las operaciones	$AvgCohO = \frac{(\frac{NO}{NS})^2}{(\frac{TM}{NS})^2}$ NO: Número de operaciones NS: Número de servicios TM: Total de mensajes	No especifica
	Complejidad del servicio	$CXS(s) = \frac{ICS(s) + NS(s) + SG(s)}{Coh(s)}$ ICS(s): Acoplamiento indirecto del servicio NS: Número de servicios SG: Granularidad del servicio AvgCohO: Promedio de cohesión de las operaciones	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Interdependencia de servicios	$SI = \sum_{a,b \in \Omega} S_a S_b$ S_a: Servicio que llama a S_b S_b: Servicio que llama a S_a	No especifica
[44]	Acoplamiento entre microservicios	$CBS_{MS_j} = \frac{\text{Número de enlaces externos}}{\text{Número de clases en el microservicio}}$ Número de enlaces externos: Cantidad de llamadas a servicios externos utilizadas en cada clase del microservicio. Número de clases en el microservicio: Cantidad de clases que componen el microservicio MS_j	No especifica
[38]	Falta de cohesión a nivel de mensajes	$LoC_{msg}(si) = 1 - \frac{2 \times \sum \forall (op_i, op_j) \in E_{si} OpS_{msg}(op_i, op_j)}{ V_{si} \times (V_{si} - 1)}$	No especifica
	Similitud a nivel de mensaje	$OpS_{msg}(op_i, op_j) = \frac{MsgS(op_i \times in, op_j \times in) + MsgS(op_i \times out, op_j \times out)}{2}$	No especifica
	Falta de cohesión a nivel de conversación	$LoC_{conv}(si) = 1 - \frac{2 \times \sum \forall (op_i, op_j) \in E_{si} OpS_{conv}(op_i, op_j)}{ V_{si} \times (V_{si} - 1)}$	No especifica
	Similitud a nivel de conversación	$OpS_{conv}(op_i, op_j) = \frac{MsgS(op_i \times in, op_j \times out) + MsgS(op_i \times out, op_j \times in)}{2}$	No especifica
	Falta de cohesión a nivel de dominio	$LoC_{dom}(si) = 1 - \frac{2 \times \sum \forall (op_i, op_j) \in E_{si} OpS_{dom}(op_i, op_j)}{ V_{si} \times (V_{si} - 1)}$	No especifica
	Similitud a nivel de dominio	$OpS_{dom}(op_i, op_j) = \frac{ T_{op_i} \cap T_{op_j} }{ T_{op_i} \cup T_{op_j} }$	No especifica
[62]	Utilización del tipo de base de datos	$DTU = \frac{DSL}{TSDL}$ DSL: Enlaces de base de datos por servicio TSDL: Total de enlaces de servicio a base de datos	No especifica
	Interacciones con bases de datos compartidas	$SDBI = \frac{SISD}{TSI}$ SISD: Interconexiones de servicios con base de datos compartida TSI: Interconexiones totales de servicios	No especifica
	Interacción de servicio a través de un componente intermediario	$SIC = \frac{SI[MB Pub/Sub Stream]}{TSI}$ SI: Interconexiones de servicios MB: Message Brokers Pub/Sub: Publish/Subscribe	No especifica

Ref. Art.	Métrica	Formula	Umbrales
		TSI: Interconexiones de servicios totales	
	Utilización de la comunicación asíncrona	$ACU = \frac{ASI[API Polling Direct Calls Shared BD]}{TSI}$ ASI: Interconexiones de servicios asíncronas TSI: Interconexiones de servicios totales	No especifica
	Compartir servicios directamente	$DSS: \frac{\frac{SS}{TS} + \frac{SSC}{TSI}}{2}$ SS: Servicios compartidos TS: Total de servicios SSC: Conectores de servicios compartidos TSI: Interconexiones de servicios totales	No especifica
	Servicios compartidos de forma transitiva	$TSS = \frac{\frac{TSS}{TS} + \frac{TSSC}{TSI}}{2}$ TSS: Servicios compartidos transitoriamente TS: Total de servicios TSSC: Servicios compartidos transitoriamente TSI: Interconexiones de servicios totales	No especifica
	Detección de dependencias cíclicas	$CDD = \begin{cases} 1: Si Ciclos de servicio(SG) = 0 \\ 0: Cualquier otro \end{cases}$ SG = (S,C) S: Conjunto de nodos del servicio C: Conjunto de aristas de conexión de un microservicio	1: Verdadero 0: Falso
[53]	Start Time	Timestamp de inicio de la solicitud	No especifica
	End Time	Timestamp de fin de la solicitud	No especifica
	Duration	$D = End Time - Start Time$	No especifica
	Origin IP	IP de origen de la solicitud	No especifica
	Origin Port	Puerto de Origen	No especifica
	Destination Service	Nombre del microservicio destino	No especifica
	Destination Instance	Instancia específica del microservicio	No especifica
	Destination IP	IP del microservicio destino	No especifica
	Destination Function	Función o endpoint invocado	No especifica
[34]	Acoplamiento de cambio bidireccional	$CC(i,j) = \frac{shared_revisions(i,j)}{(\frac{total_revisions(i) + total_revisions(j)}{2})}$	Bajo: 0-0.33 Regular: 0.33-0.66 Fuerte: 0.66-1

Ref. Art.	Métrica	Formula	Umbrales
	Acoplamiento de cambio direccional	$CC_{A \rightarrow B} = \frac{Shared_revisions(A, B)}{revisions(A)}$	Bajo: 0-0.33 Regular: 0.33-0.66 Fuerte: 0.66-1
	Acoplamiento estructural	$SC(s1, s2) = 1 - \frac{1}{degree(s1, s2)} \times LWF \times GWF$	Número lo más bajo posible
	Líneas de código	Número de líneas de código excluyendo comentarios y líneas en blanco	No especifica
	Número ponderado de interfaces de servicio	$WSIC(S) = \sum_{i=1}^n W_i$ N: Número de operaciones en la interfaz del servicio S W_i : Peso de la operación	Mejor: [0,5] Peor: [>25]
	Cohesión de datos en la interfaz de servicio	$SIDC(S) = \frac{OC(IS)}{OD(IS)}$ OC: Número de operaciones con al menos un parámetro común OD: Número total de operaciones distintas en la interfaz	Mejor: [1,1] Peor: [0.55,0]
	Frecuencia de cambios	$CF_s = \frac{Número\ de\ commits\ en\ el\ periodo}{Número\ de\ meses\ del\ periodo}$	No especifica
[14]	Número de servicios	$NoS = n$ N: Número de nodos en el grafo de servicios	No especifica
	Acoplamiento	$CS_i = deg(S_i)$ Grado de cada nodo	No especifica
	Acoplamiento Relativo de Servicios	$RCS[s] = \frac{CS[s]}{NoS}$ CS[s]: Grado de acoplamiento NoS: Número de estados	No especifica
	Factor de Acoplamiento de Servicios	$SCF = \frac{\sum_{s \in S} CS[s]}{NoS^2 - NoS}$ CS[s]: Grado de acoplamiento NoS: Número de estados	0-1
[52]	Aplicación basada en microservicios	$MSBA = (MS, \overrightarrow{MT})$ MS: conjunto de microservicios	No especifica

Ref. Art.	Métrica	Formula	Umbrales
		MT: es un vector de 5 dimensiones (1. Acoplamiento (CpT), 2. Falta de cohesión (CohT), 3. Granularidad (WsicT), 4. Complejidad (CxT), 5. Similitud semántica (SsT))	
	Vector que contiene de las métricas calculadas para MSBA	$\overrightarrow{MT} = [CpT, CohT, WsicT, CxT, (100 - SsT)]$ CpT: Acoplamiento CohT: Cohesión WsicT: Número de historias asociadas al microservicio CxT: Puntos de complejidad cognitiva SsT: Similitud semántica	0-1
	Granularidad	$Gm = \overrightarrow{MT} = \sqrt[2]{(CpT^2 + CohT^2 + CxT^2 + Wsic^2 + (100 - Sst)^2)}$ CpT: Acoplamiento CohT: Cohesión WsicT: Número de historias asociadas al microservicio CxT: Puntos de complejidad cognitiva SsT: Similitud semántica	Mientras más bajo, mejor
	I-ésimo microservicio	$ms_i = (HU_i, MTS_i)$ HU_i: Conjunto de historias de usuario asociadas al i-ésimo microservicio MTS_i: Conjunto de métricas calculadas para el i-ésimo microservicio	No especifica
	Importancia absoluta para cada microservicio	$\overrightarrow{AIS} = [AIS_1, AIS_2, \dots, AIS_n]$	No especifica
	Valor total de importancia absoluta del microservicio	$AisT = \overrightarrow{AIS} = \sqrt[2]{AIS_1^2 + AIS_2^2 + \dots + AIS_n^2}$	No especifica
	Norma del vector de la dependencia absoluta del microservicio	$\overrightarrow{ADS} = [ADS_1, ADS_2, \dots, ADS_n]$	No especifica
	Valor total de dependencia absoluta del microservicio	$AdsT = \overrightarrow{ADS} = \sqrt[2]{ADS_1^2 + ADS_2^2 + \dots + ADS_n^2}$	No especifica
	Número de pares de microservicios	$\overrightarrow{SIY} = [SIY_1, SIY_2, \dots, SIY_n]$	No especifica
	Valor total de interdependencia de los microservicios	$SiyT = \overrightarrow{SIY} = \sqrt[2]{SIY_1^2 + SIY_2^2 + \dots + SIY_n^2}$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Vector que contiene las métricas de acoplamiento calculadas para MSBA	$\vec{cp} = [AisT, AdsT, SiyT]$	No especifica
	Acoplamiento	$\overline{CpT} = \vec{Cp} = \sqrt[2]{AisT^2 + AdsT^2 + SiyT^2}$ <i>AisT</i> : Valor total de importancia absoluta del microservicio <i>AdsT</i> : Valor total de dependencia absoluta del microservicio <i>SiyT</i> : Valor total de interdependencia de los microservicios	No especifica
	Falta de cohesión	Número de pares de microservicios que no tienen ninguna dependencia entre ellos	No especifica
	Grado de falta de cohesión	$Coh_i = \frac{LC_i}{n}$ <i>LC_i</i> : proporción de la métrica de falta de cohesión <i>N</i> : número total de microservicios que forman parte de la aplicación	0-1
	Vector de cohesión	$\vec{Coh} = [Coh_1, Coh_2, \dots, Coh_n]$	0-1
	Cohesión total para MSBA	$CohT = \vec{Coh} = \sqrt[2]{Coh_1^2 + Coh_2^2 + \dots + Coh_n^2}$	0-1
	Diccionario de similitud semántica entre las historias de usuario	$DSS = \{ \langle "hu_1 - hu_2", a_{1-2} \rangle, \langle "hu_1 - hu_3", a_{1-3} \rangle, \dots, \langle "hu_j - hu_k", a_{j-k} \rangle \}$ <i>hu_j - hu_k</i> : Llave del diccionario, la cual se forma concatenando los identificadores de las historias de usuario que se comparan <i>a_{j-k}</i> : Valores de similitud semántica entre las historias de usuario	0-1 Entre más cercano a 1, más similares son las historias de usuario
	Similitud Semántica del i-ésimo microservicio	$SS_i = \frac{1}{c} \sum_{j=1}^m k = j + 1 a_{j-k}$ <i>a_{j-k}</i> : Valores de similitud semántica de la comparación realizada entre las historias de usuario <i>C</i> : Número de comparaciones realizadas para calcular SS <i>N</i> : Número de microservicios de MSBA	No especifica
	Similitud Semántica Total	$SsT = \frac{100}{n} \sum_{i=1}^n SS_i$ <i>SS_i</i> : Similitud Semántica del i-ésimo microservicio	No especifica
	Recuento ponderado de la interfaz de servicios	$WSIC = \frac{\text{Número de operaciones}}{\text{Número de parametros o microservicios}}$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Número máximo de historias de usuario asociadas a un microservicio	$WsicT = Max(WSIC_1, WSIC_2, \dots, WSIC_n)$	No especifica
	Promedio de llamadas	$Avg. Calls = \frac{1}{n} \sum_{i=1}^n Calls_i$ N: Número de microservicios Calls: Número de llamadas	No especifica
	Puntos de historia del usuario del microservicio	$P_i = \sum_{j=1}^m PH_j$ PH_j : Puntos de historia estimados por el equipo de desarrollo M: Número de historias de usuario asociadas al microservicio	No especifica
	Puntos de complejidad cognitiva	$Cx = \left(\left(\sum_{i=1}^n Cg_i \right) + Max(P_1, \dots, P_n) + (n * WsicT) + \left(\sum_{i=1}^n Pf_i \right) + \left(\sum_{i=1}^n LC_i \right) \right)$ $CxT = \frac{Cx}{Cx_0}$ I = i-ésimo microservicio Cg_i : $P_i * (Calls + Request)$ P_i : Número total de historias de usuario del i-ésimo microservicio N: Número de microservicios Pf_i : Número de nodos usados secuencialmente desde una llamada que realiza un microservicio a otro LC: Interdependencia de un microservicio Cx_0 : El caso base en el que la aplicación tiene un microservicio	No especifica
	Tiempo de desarrollo de cada microservicio	$t_i = \sum_{j=1}^m TH_j$ M: Número de historias de usuario del microservicio TH_j : Tiempo estimado de desarrollo de la j-ésima historia de usuario	No especifica
	Tiempo de desarrollo de la aplicación	$T = Max(t_1, t_2, \dots, t_n)$	No especifica
[59]	Calidad de la modularización	$MQ = \begin{cases} \left(\frac{1}{k} \sum_{i=1}^k A_i \right) - \left(\frac{1}{\frac{k(k-1)}{2}} \sum_{i,j=1}^k E_{i,j} \right) & Si \ k > 1 \\ A_1 & Si \ k = 1 \end{cases}$ A_i : Intraconectividad	No especifica

Ref. Art.	Métrica	Formula	Umbrales
		E_i : Interconectividad	
	Falta de cohesión estructural	$LStrCoh(M_{r_i}) = \frac{\sum_{j=1}^m LCOC_{P_j}}{m}$ m: Número de paquetes en r_i LCOC: Falta de cohesión de las clases	No especifica
	Acoplamiento estructural	$StrCoup(M_{r_i}) = \frac{\sum_{j=1}^m FanOut_{P_j}}{m}$ FanOutput: Número de clases fuera de P_j	Cuanto mayor el número, mayor el acoplamiento general
	Acoplamiento conceptual entre métodos	$CCM(m_k, m_j) = \frac{\vec{m}_k \times \vec{m}_j}{\ \vec{m}_k\ \times \ \vec{m}_j\ }$ $\vec{m}_k$: Vector correspondiente a m_k $\vec{m}_j$: Vector correspondiente a m_j	0-1
	Acoplamiento conceptual entre un método y una clase	$CCMC(m_k, m_j) = \frac{\sum_{q=1}^t CCM(m_k, m_j)}{t}$ T: número de objetos en c_j	0-1
	Acoplamiento conceptual entre clases	$CCBC(c_k, c_j) = \frac{\sum_{l=1}^r CCMC(m_{kl}, c_j)}{r}$	0-1
	Falta de cohesión conceptual	$LconCoh(M_{r_i}) = \frac{\sum_{j=1}^m 1 - ConCoh_{P_j}}{m}$ ConCoh: Promedio de CCBC de todos los pares de clases.	0-1
	Acoplamiento conceptual	$ConCoup(M_{r_i}) = \frac{\sum_{j=1}^m CCBC_{out_{P_j}}}{m}$ CCBCout: Suma de CCBC entre clases en P_j	No especifica
	Medida de eficacia de MoJo	$MoJoFM(A, B) = 100 - \left(\frac{mno(A, B)}{\max(mno(\forall E_A, B))} \times 100 \right)$ Mno(A,B): Número mínimo de operaciones Move o Join Max($\forall E_A, B$): Distancia máxima posible de A con respecto a B	0: Si es la partición mas alejada 100: Si son iguales
[36]	Recuento de interfaz de servicio ponderado	$WSIC_{AVG}(Y) = \frac{\sum S \in Y WSIC(S)}{ Y }$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Respuesta total del servicio	$TRS(S) = \sum_{O \in S_{IS}} RFO(O)$	Valores bajos son favorables
	Número de versiones por servicio	$NVS(S, Y)$ Número de versiones usadas concurrentemente del servicio S en el sistema Y	No especifica
	Soporte de servicio para transacciones	$SST(Y) = \frac{TAS(Y)}{ Y }$ TAS: Servicios transaccionales	Valores altos indican alta complejidad
	Interdependencia de servicios en el sistema	Número de pares de servicios bidireccionalmente dependientes	Valor ideal 0
	Importancia absoluta del servicio	$AIS(S)$ = Número de consumidores que dependen del servicio S	No especifica
	Dependencia absoluta del servicio	$ADS(S)$ = Número de otros servicios de los que S depende	No especifica
	Criticidad absoluta del servicio	$ACS(S) = AIS(S) \times ADS(S)$	Valores bajos son favorables
	Cohesión de datos de la interfaz de servicio	$SIDC(S) = \frac{N_{cd}(S)}{N_{td}(S)}$ $N_{cd}(S)$: número de operaciones con tipos de datos comunes en la interfaz del servicio S_{IS} . $N_{td}(S)$: número total de tipos de datos distintos usados por las operaciones en la interfaz S_{IS} .	0-1 (1 es óptimo)
	Cohesión del uso de la interfaz de servicio	$SIUC(S) = \frac{\sum_{i=1}^{N_c} N_{uo}(C_i, S)}{N_c \times N_o(S_{IS})}$ N_c : Número total de clientes que invocan el servicio N_{uo} : Número de operaciones de la interfaz que son utilizadas por el cliente N_o : Número total de operaciones definidas en la interfaz del servicio	0-1 (1 es óptimo) 0-1 (1 es óptimo)
	Cohesión total de la interfaz de servicio	$TSIC = \frac{SIDC(S) + SIUC(S)}{2}$	0-1 (1 es óptimo)
[5]	Recuento de interfaces de servicio ponderado	Número ponderado de interfaces expuestas por servicio	No especifica
	Servicios sin estado	$SS = \frac{SLS}{(SLS + SFS)}$ SLS: Servicios Stateless SFS: Servicios Stateful	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Soporte de servicio para transacciones	$SST = \frac{NTAS}{(NTAS + TAS)}$ NTAS: Servicios No Transaccionales TAS: Servicios transaccionales	No especifica
	Patrón de realización del servicio	$SRP = \frac{IE}{(IE + DE)}$ IE: Exposición indirecta DE: Exposición directa	No especifica
	Número de tareas humanas	$NHT = \frac{HT}{(HT + AT)}$ HT: Tareas Humanas AT: Tareas Automatizadas	No especifica
	Número de servicios	Número total de servicios en la solución SOA	No especifica
	Patrón de composición del servicio	$SCP = \frac{CS}{(AS + CSs + CSwf)}$ CS: Servicios Compuestos AS: Servicios Atómicos CSwf: Servicios Compuestos creados a través de un flujo de trabajo CSs: Servicios Compuestos Estructurales	No especifica
	Método de acceso al servicio	$SAM = \frac{VAS}{(VAS + PPS)}$ VAS: Acceso virtualizado PPS: Punto a punto	No especifica
	Selección dinámica VS selección estática de servicios	$DSSS = \frac{DS}{(DS + SS)}$ DS: Selección Dinámica SS: Selección Estática	No especifica
	Número de versiones por servicio	$NOVS = VERSIONS / SERVICES$	No especifica
[50]	Duración del conjunto de pruebas	Tiempo total de ejecución del conjunto de pruebas	No especifica
	Latencia media de solicitud	Tiempo promedio entre request y response	No especifica
	Tiempo de finalización	Diferencia entre tiempo actual y tiempo de llegada del request	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Tasa de éxito de la solicitud	Porcentaje de requests exitosos	No especifica
	Gastos generales de rendimiento	$O_{BL} = \frac{TSD_{BL} - TSD_{NL}}{TSD_{NL}} \times 100$	No especifica
	Gastos generales del tamaño del registro	$O_{MFB L} = \frac{LS_{MF} - LS_{BF}}{LS_{BF}} \times 100$	No especifica
[51]	Coefficiente de variación	$C_v = \frac{\sigma}{\mu}$	No especifica
	Composición de nanoentidades	Número de nanoentidades por microservicio	No especifica
	Líneas de código	LOC = Número de líneas de código por microservicio	No especifica
	Composición de dependencias	Número de dependencias salientes por microservicio	No especifica
	Componentes fuertemente conectados	$SCC = \frac{\text{Numero de SCC}}{\text{Numero total de microservicios}}$	Ideal: 1 SCC por microservicio
	Composición de las entidades	Número de entidades por microservicio	No especifica
	Composición de relación 1	Variación cuantitativa del "lenguaje publicado" por relación	No especifica
	Composición de relación 2	Número de responsabilidades de caso de uso por microservicio	No especifica
	Similitud semántica	Promedio de algoritmos de similitud léxica	No especifica
[39]	Cohesión de datos de la interfaz de servicio	$SIDC(s) = \frac{ Common(Param(so \in SO(si_s))) }{totalParamTypes}$ Common(Param(so ∈ SO(si))): Conjunto de operaciones del servicio que comparten tipos de parámetros comunes totalParamTypes: Número total de tipos de parámetros distintos en el servicio	No especifica
	Cohesión del uso de la interfaz de servicio	$SIUC(s) = \frac{INV(clients, SO(si_s))}{(num_clients \times SO(si_s))}$ INV(clients, SO(si)): Suma de operaciones utilizadas por cada cliente num_clients: Número total de clientes SO(si) : Número total de operaciones del servicio	Si tiene más de un elemento el valor es 1
	Cohesión del uso secuencial del servicio	$SSUC(s) = \frac{INVSEQ(clients, SO(si_s))}{(num_clients \times SO(si_s))}$ INVSEQ(clients, SO(si)): Suma de operaciones utilizadas secuencialmente por cada cliente	No especifica

Ref. Art.	Métrica	Formula	Umbrales
		num_clients: Número total de clientes SO(si) : Número total de operaciones del servicio	
	Cohesión estricta en la implementación del servicio	$SSIC(s) = \frac{ IC(s) }{(BPS_s \cup C_s \cup I_s \cup P_s \cup H_s \times SO(si_s))}$ IC(s) : Número de elementos de implementación comunes a todas las operaciones BPS ∪ C ∪ I ∪ P ∪ H : Número total de elementos de implementación SO(si) : Número total de operaciones del servicio	No especifica
	Cohesión de implementación de servicios flexibles	$LSIC(s) = \frac{ CC(s) }{(BPS_s \cup C_s \cup I_s \cup P_s \cup H_s \times SO(si_s))}$ CC(s) : Número de elementos de implementación directa o indirectamente conectados BPS ∪ C ∪ I ∪ P ∪ H : Número total de elementos de implementación SO(si) : Número total de operaciones del servicio	No especifica
	Cohesión total de la interfaz de un servicio	$TICS(s) = \frac{(SIDC(s) + SIUC(s) + SSUC(s) + IMPC(s))}{4}$ SIDC(s): Cohesión de datos SIUC(s): Cohesión de uso SSUC(s): Cohesión de uso secuencial IMPC(s): Cohesión de implementación (SSIC o LSIC)	No especifica
[1]	Falta de cohesión	$LCOM = 1 - \frac{\sum_{i=1}^n MF}{M \times F}$ MF: Aparición de un parámetro en un microservicio específico M: Número de operaciones en un microservicio específico F: Número de parámetros únicos de un microservicio	0 y 1, donde 1 representa más falta de cohesión de las aplicaciones, lo que puede dar lugar a más errores en la aplicación
	Media de la falta de cohesión	$ALCOM = \frac{\sum_{i=1}^n LCOM}{NS}$ LCOM: Falta de cohesión NS: Número de servicios N: Número de operaciones	No especifica
	Granularidad	$SGM = \sum_{i=1}^n DGS_i \times FGS_i$ DGS: Granularidad de datos de un servicio FGS: Granularidad funcional de un servicio	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Granularidad de datos de un servicio	$DGS = \frac{IPR}{\sum_{i=1}^n FP_i} + \frac{OPR}{\sum_{i=1}^n CP_i}$ IPR: Número de parámetros de entrada en una operación FP: Número total de parámetros de entrada en un microservicio OPR: Número de parámetros de salida en un microservicio CP: Número total de parámetros de salida en un microservicio	0-1 Mas cerca de 1: Grano grueso Mas cerca de 0: Grano fino
	Granularidad funcional de un servicio	$FGS = \frac{OT}{\sum_{i=1}^n O_i}$ OT: Pero de una operación específica de un microservicio O: La suma de todos los pesos en un microservicio específico Pesos: Operaciones de creación tienen peso de 4, de actualización 3, de borrado 2 y de lectura 1.	0-1 Mas cerca de 1: Grano grueso Mas cerca de 0: Grano fino
	Promedio de granularidad	$ASGM = \frac{\sum_{i=1}^n SGM}{NS}$ SGM: Granularidad NS: Número de servicios	0-1 Mas cerca de 1: Grano grueso Mas cerca de 0: Grano fino
	Número de operaciones por servicio	$NOO = \sum_{i=1}^n M_i$ M: Número de operaciones por servicio	Cuanto mayor sea el número de esta métrica, más errores puede producir la aplicación.
[33]	Coeficiente de Silhouette	$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$ a(i): Disimilitud promedio intra-cluster b(i): Distancia mínima inter-cluster	-1 a 1 (1 es óptimo)
	Precisión individual del objeto	$P(O_o) = \begin{cases} \frac{ S_{t(O_o)} \cap C(O_o) - 1}{ C(O_o) - 1} & , \quad C(O_o) > 1 \\ 1 & , \quad C(O_o) = 1 \end{cases}$ O_o: Es el objeto $S_{t(O_o)}$: Es el conjunto de objetos que pertenecen a un mismo cluster ideal $C(O_o)$: Cluster ideal del objeto $t(O_o)$: Función de pertenencia al cluster computado	0-1 (1 es óptimo)

Ref. Art.	Métrica	Formula	Umbrales
	Recall	$R(O_o) = \begin{cases} \frac{ S_{t(O_o)} \cap C(O_o) - 1}{ C(O_o) - 1} & , \quad t(O_o) > 1 \\ 1 & , \quad t(O_o) = 1 \end{cases}$ O_o: Es el objeto $C(O_o)$: Cluster ideal del objeto $S_{t(O_o)}$: Es el conjunto de objetos que pertenecen a un mismo cluster ideal $t(O_o)$: Función de pertenencia al cluster computado	0-1 (1 es óptimo)
	Media F	$F1 = 2 \times \frac{P \times R}{P + R}$ P: Precisión R: Recall	0-1 (1 es óptimo)
	Responsabilidad	$r(i, k) \leftarrow s(i, k) - \max\{a(i, k') + s(i, k')\}$	No especifica
	Disponibilidad de mensajes	$a(i, k) \leftarrow \min\{0, r(k, k) + \sum_{i' \in S \times t \mid i' \notin \{i, k\}} \max\{0, r(i', k)\}\}$	No especifica
	Disponibilidad propia	$a(k, k) \leftarrow \sum_{i' \in S \times t \mid i' \neq k} \max\{0, r(i', k)\}$	No especifica
[42]	Falta de cohesión	$LCOM = 1 - \frac{\sum_{i=1}^n MF}{M \times F}$ MF: Aparición de un parámetro en un microservicio específico M: Número de operaciones en un microservicio específico F: Número de parámetros únicos de un microservicio	0 - 0.8
	Complejidad	$NOO = \sum_{i=1}^n M$ M: Número de operaciones por microservicio	<=10
[49]	Falta de cohesión	$LCOM = 1 - \frac{\sum_{i=1}^n OP_i}{M \times F}$ OP: Aparición de un parámetro de microservicio específico M: Número de operaciones en un microservicio específico F: Número de parámetros únicos de un microservicio	No especifica
	Granularidad de datos de un servicio	$DGS = \frac{IPR}{\sum_{i=1}^n FP_i} + \frac{OPR}{\sum_{i=1}^n CP_i}$	0-1

Ref. Art.	Métrica	Formula	Umbrales
		IPR: Número de parámetros de entrada en una operación FP: Número total de parámetros de entrada en un microservicio OPR: Número de parámetros de salida en un microservicio CP: Número total de parámetros de salida en un microservicio	Mas cerca de 1: Grano grueso Mas cerca de 0: Grano fino
	Granularidad funcional de un servicio	$FGS = \frac{OT}{\sum_{i=1}^n O_i}$ OT: Pero de una operación específica de un microservicio O: La suma de todos los pesos en un microservicio específico Pesos: Operaciones de creación tienen peso de 4, de actualización 3, de borrado 2 y de lectura 1.	0-1 Mas cerca de 1: Grano grueso Mas cerca de 0: Grano fino
	Granularidad	$SGM = \sum_{i=1}^n DGS_i \times FGS_i$ DGS: Granularidad de datos de un servicio FGS: Granularidad funcional de un servicio	No especifica
	Número de operaciones por servicio	$NOO = \sum_{i=1}^n M_i$ M: Número de operaciones por servicio	Cuanto mayor sea el número de esta métrica, más errores puede producir la aplicación.
[57]	Acoplamiento estructural	$SC(S1,S2) = 1 - \frac{1}{(Degree(s1,s2))} \times LWF \times GWF$ degree(s1,s2): Número total de dependencias estructurales entre s1 y s2 LWF: Factor de ponderación local GWF: Factor de ponderación global	0-1
	Factor de peso local	$LWF(s1,s2) = \frac{1 + outdegree(s1,s2)}{1 + degree(s1,s2)}$ outdegree(s1,s2): Número de dependencias desde s1 hacia s2 degree(s1,s2): Número total de dependencias entre s1 y s2	0-1
	Factor de peso global	$GWF(s1,s2) = \frac{degree(s1,s2)}{\max(degree(all_services))}$ degree(s1,s2): Número total de dependencias entre s1 y s2 max(degree(all_services)): Máximo número de dependencias entre cualquier par de servicios del sistema	0-1
[2]	Utilización del CPU	Tiempo del CPU consumido por el microservicio	No especifica
	Latencia HTTP	Tiempo en que se recibe la respuesta a partir del envío de la petición	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Rendimiento HTTP	$Throughput = \frac{N.Requests}{Time\ period}$ N. Requests: Número de peticiones HTTP exitosas Time Period: Intervalo de tiempo medido	No especifica
	Tiempo de respuesta del servicio	$Response\ Time = T_{process} + T_{network} + T_{queue}$ T_process: Tiempo de procesamiento interno del servicio T_network: Tiempo de latencia de red T_queue: Tiempo en cola esperando ser atendido	No especifica
	Tasa de error	$Error_Rate = \frac{N_Failed_Requests}{Total_requests} \times 100$ N_Failed_Requests: Número de peticiones erróneas Total_requests: Número de peticiones realizadas	No especifica
	Uso de recursos	Uso de memoria (RAM), ancho de banda de red, etc.	Limite basado en la capacidad de la infraestructura
	Disponibilidad	$Availability = \frac{uptime}{(uptime + downtime)} \times 100$ Uptime: Tiempo total que el sistema estuvo operativo Downtime: Tiempo total que el sistema no estuvo disponible	No especifica
	Frecuencia de llamadas al servicio	$Call_Frequency = \frac{Total_service_calls}{Time\ period}$ Total_Service_Calls: Número total de invocaciones del servicio Time Period: Ventana de tiempo de la medición	No especifica
[3]	Tiempo de ejecución	Tiempo de inicio y final de la tarea	No especifica
[41]	Número de interfaces asíncronas	$M1 = IS + IA$ IS: Número de endpoints que permiten comunicación síncrona IA: Número de endpoints que permiten comunicación asíncrona	Mientras el número sea menor es mejor
	Distribución de llamadas síncronas	$M2 = \frac{min_llamadas}{max_llamadas}$ Min_llamadas: Número mínimo de llamas a una interfaz del servicio Max_llamadas: Número máximo de llamadas a una interfaz de servicio	Mientras más cercano a 1 es mejor
	Número de dependencias asíncronas	$M3 = DS + DA$ DS: Número de dependencias de comunicación síncrona DA: Número de dependencias de comunicación asíncrona	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Traza de llamada sincrónica más larga	$M4 = \max(\text{longitud_traza_sincrona})$ Longitud traza sincrónica: Número de servicios en una cadena de llamadas asíncronas	No especifica
	Tamaño promedio de mensajes asíncronos	$M5 = \frac{TM}{M}$ TM: Tamaño individual de cada mensaje asíncrono M: Número total de mensajes	No especifica
	Número de ciclos síncronos	$M6 = CFC$ CFM: Número de grupos de servicios donde cada uno puede alcanzar a todos los demás mediante llamadas asíncronas.	Idealmente 0
[48]	Conectores distribuidos seguros	$SCO = \frac{ \text{secure_connectors}(\text{distributed_connectors}(cn)).\text{succesful} }{ \text{distributed_connectors}(cn) }$	No especifica
	Conectores de cliente seguros	$SCC = \frac{ \text{Secure_connectors}(\text{ui_connectors}(cn)).\text{succesful} }{ \text{client_connectors}(cn) }$	No especifica
	Conectores de interfaz de usuario seguros	$SUC = \frac{ \text{secure_connectors}(\text{ui_connectors}(cn)).\text{succesful} }{ \text{ui_connectors}(cn) }$	No especifica
	Conectores externos seguros de cliente/IU	$SEC = \frac{ \text{secure_connectors}(\text{client_connectors}(cn) \cup \text{ui_connectors}(cm)).\text{succesful} }{ \text{client_connector}(cn) \cup \text{ui_connectors}(cn) }$	No especifica
	Función común para conectores backend distribuidos	$\text{distributed_backend_connectors} = \frac{\text{distributed_connectors}(cn)}{\text{client_connectors}(cn) \cup \text{ui_connectors}(cn)}$	No especifica
	Conectores distribuidos internos seguros	$SIC(cn) = \frac{ \text{secure_connectors}(\text{distributed_backend_connectors}(cn)).\text{succesful} }{ \text{distributed_backend_connectors}(cn) }$	No especifica
	Función común para conectores backend distribuidos que requieren autenticación.	$\text{Connectors_that_require_authentication}(\text{distributed_backend_connectors}(cn))$	No especifica
	Conectores de backend autenticados	$AEI(cn) = \frac{ \text{authenticated_connectors}(\text{distributed_backend_connectors_requiring_authentication}(cn)).\text{succesful} }{ \text{distributed_backend_connectors_requiring_authentication}(cn) }$	No especifica
	Conectores de backend autenticados de forma segura	$AEI_S(cn) = \frac{ \text{securely_authenticated_connectors}(\text{distributed_backend_connectors_requiring_authentication}(cn)).\text{succesful} }{ \text{distributed_backend_connectors_requiring_authentication}(cn) }$	No especifica
	Conectores de backend autenticados con claves API	$AEI_K(cn) = \frac{ \text{connectors_authenticated_whit_api_keys}(\text{distributed_backend_connectors_requiring_authentication}(cn)).\text{succesful} }{ \text{distributed_backend_connectors_requiring_authentication}(cn) }$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Conectores de backend autenticados con credenciales de texto sin formato	$AEI_P(cn) = \frac{ connectors_authenticated_with_plaintext_credentials(distributed_backend_connectors_requiring_authentication(cn)).successful }{ distributed_backend_connectors_requiring_authentication(cn) }$	No especifica
	Conectores de backend autenticados a través de una conexión segura	$AEI_C(cn) = \frac{ authenticated_connectors(secure_connectors(distributed_backend_connectors_requiring_authentication(cn)).successful).successful }{ distributed_backend_connectors_requiring_authentication(cn) }$	No especifica
	Conectores de backend autenticados que utilizan un método o una conexión seguros	$AEI_A(cn) = \frac{ authenticated_backend_connectors_using_secure_method_or_secure_communication(cn).successful }{ distributed_backend_connectors_requiring_authentication(cn) }$	No especifica
	Función común para calcular los conectores en la ruta del cliente/interfaz de usuario al servicio que requiere autenticación.	$connectors_that_require_authentication(connectors_on_client_service_paths(all_paths_from_clients_or_uis_to_system_services(cp)))$	No especifica
	Conectores de ruta de servicio IU/cliente autenticados	$AEC(cp) = \frac{ authenticated_connectors(client_service_path_connectors_requiring_authentication(cp)).successful }{ client_service_path_connectors_requiring_authentication(cp) }$	No especifica
	Conectores de ruta de servicio IU/cliente autenticados de forma segura	$AEC_S(cp) = \frac{ securely_authenticated_connectors(client_service_path_connectors_requiring_authentication(cp)).successful }{ client_service_path_connectors_requiring_authentication(cp) }$	No especifica
	Conectores de ruta de cliente/IU-servicio autenticados con claves API	$AEC_K(cp) = \frac{ connectors_authenticated_with_api_keys(client_service_path_connectors_requiring_authentication(cp)).successful }{ client_service_path_connectors_requiring_authentication(cp) }$	No especifica
	Conectores de ruta de cliente/IU-servicio autenticados con credenciales de texto sin formato	$AEC_P(cp) = \frac{ connectors_authenticated_with_plaintext_credentials(client_service_path_connectors_requiring_authentication(cp)).successful }{ client_service_path_connectors_requiring_authentication(cp) }$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Conectores de ruta de cliente/IU-servicio a través de una conexión segura	$AEC_C(cp) = \frac{ authenticated_connectors(secure_connectors(client_service_path_connectors_requiring_authentication(cp)).succesful).succesful }{ client_service_path_connectors_requiring_authentication(cp) }$	No especifica
	Conectores de ruta de cliente/IU-servicio que utilizan un método o conexión seguros	$AEC_A(cp) = \frac{ authenticated_connectors_on_client_service_path_using_secure_method_api_keys_or_secure_communication(cp).succesful }{ client_service_path_connectors_requiring_authentication(cp) }$	No especifica
	Servicios del sistema observados	$OSS(cp) = \frac{ observed_by_dedicated_component(system_services(cp)).succesful }{ system_services(cp) }$	No especifica
	Componentes de la fachada observados	$OFA(cp) = \frac{ observed_by_dedicated_component(facades(cp)).succesful }{ facades(cp) }$	No especifica
	Servicios y fachadas del sistema observado	$OSF(cp) = \frac{ observed_by_dedicated_component(system_services(cp) \cup facades(cp)).succesful }{ system_services(cp) \cup facades(cp) }$	No especifica
[45]	Dependencia entre la clase compartida y los clústeres.	$D_SCdep(cluster_i, c_k) = \frac{\sum_{j=1}^{size(cluster)} d_tcdep(d_ctrace_j, c_k)}{size(cluster_i)}$	No especifica
	Cohesión media de las interfaces de servicio a nivel de mensajes.	$CHM_j = \frac{\sum_{i=1}^K n_i CHM_j}{\sum_{i=1}^K n_i}$	0-1
	Cohesión media de las interfaces de servicio a nivel de dominio	$CHD = \frac{\sum_{j=1}^K n_i CHD_j}{\sum_{i=1}^K n_i}$	0-1
	Número medio de interfaces que un servicio extraído proporciona a otros servicios.	$IFN = \frac{1}{K} \left \bigcup_{i=1,2,\dots,K} \right $	Menor valor = mejor
	Número de operaciones proporcionadas por los microservicios extraídos	$OPN = \left \bigcup_{i=1,2,\dots} I_i \right \sum_{i=1}^{i=K} n_i$	Menor valor = mejor

Ref. Art.	Métrica	Formula	Umbrales
	Número de llamadas a métodos entre dos servicios.	$IRN = \sum_{(Op_j, Op_k)} W_{j,k}$	Menor valor = mejor
[46]	Tamaño del servicio	$S_{ws} = \sum S \in S \sum O \in S \frac{W_i}{mk}$ W_i: Valor ponderado de las operaciones de una interfaz M: Número de interfaces K: Número total de operaciones	No especifica
	Tamaño del sistema de microservicios	$Y_{ws} = \frac{\sum S \in Y S_{ws}}{ Y }$ S_{ws}: Tamaño de cada servicio Y: Número total de servicios en el sistema	No especifica
	Importancia del servicio	$S_{IOS} =$ Número de consumidores que dependen del servicio	No especifica
	Dependencia del servicio	$S_{DOS} =$ Número de servicios de los que depende el servicio	No especifica
	Criticidad del servicio	$S_{COS} = S_{IOS} \times S_{DOS}$ S_{IOS}: Importancia del servicio S_{DOS}: Dependencia del servicio	No especifica
	Cohesión de datos de interfaz	$S_{IDC} = \frac{p_{type}}{m}$ p_{type}: Número de tipo de datos de parámetros de interfaz M: Número total de interfaces	1 indica una alta cohesión
	Cohesión de uso de interfaz	$S_{IUC} = \frac{ SI_{invok} }{m}$ SI_{invok}: Número de interfaces internas llamadas por el cliente M: Número total de interfaces	1 indica una alta cohesión
	Cohesión total de interfaz	$S_{TIC} = \frac{S_{IDC} + S_{IUC}}{2}$ S_{IDC}: Cohesión de datos de interfaz S_{IUC}: Cohesión de uso de interfaz	1 indica una alta cohesión
[47]	Soporte	$s(f_i) = \frac{\sigma(f_i)}{M^{fault}}$ $\sigma(f_i)$: frecuencia del evento de fallo M^{fault}: Número total de eventos de fallo en el historial	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Confianza	$c(f_i, f_i) = \frac{\sigma(f_i \cup f_i)}{\nabla(f_i)}$ $\sigma(f_i \cup f_i): \text{ frecuencia de ocurrencia conjunta de } f_i \text{ y } f_i$ $\nabla(f_i): \text{ conjunto de eventos de fallo que contienen } f_i$	No especifica
	Causalidad entre x e y	$X_t = \beta_0 + \sum_{k=1}^p \beta_k X_{t-k} + \epsilon_x$	No especifica
	Análisis de causalidad de Granger bidimensional.	$X_t = \theta_0 + \sum_{k=1}^p \beta_k X_{t-k} + \sum_{n=1}^q \theta_n Y_{t-n}$	No especifica
	Causalidad de Granger	$F_{y \rightarrow x} = \ln \frac{\epsilon_{y,x}}{\epsilon_x}$ $\epsilon_x: \text{ Error de predicción usando solo serie temporal}$ $\epsilon_{y,x}: \text{ Error de predicción usando series X y Y}$	No especifica
	Causalidad de Granger entre los datos de series temporales X e Y	$F_{y \rightarrow x} = \begin{cases} 0, \text{ no Granger} \\ F_{y \rightarrow x} > 0 \end{cases}$	No especifica
	Algoritmo de puntuación de excepciones	$X = \begin{cases} (0, \dots, 0, X_1, X_2, \dots, X_{m-s}), s \geq 0 \\ (x_{1-s}, \dots, x_{m-1}, x_m, 0, \dots, 0), s < 0 \end{cases}$	No especifica
	Correlación cruzada con desplazamiento w	$CC_w(X, Y) = \begin{cases} \sum_{i=1}^{m-s} X_i \times Y_{s+i}, s \geq 0 \\ \sum_{i=1}^{m+s} X_{i-s} \times Y_i, s < 0 \end{cases}$	No especifica
	Coefficiente de correlación cruzada normalizada	$NCC(X, Y) = \max \left(\frac{CC_w(X, Y)}{\ X\ _2 \times \ Y\ _2} \right)$ $CC_w(X, Y): \text{ correlación cruzada con desplazamiento w}$ $\ X\ _2, \ Y\ _2: \text{ normas L2 de las series X e Y}$	No especifica
	Puntuación de anomalía c-SBD	$c - SBD(X, Y) = 1 - NCC(X, Y)$ $NCC: \text{ Coeficiente de correlación cruzada normalizada}$	0-2 Cuanto más grande peor
	puntuación de excepción	$S(R(v_i), R(avg)) = 1 - \max \left(\frac{CC_w(R(avg), R(v_i))}{\ R(avg)\ _2 \times \ R(v_i)\ _2} \right)$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
		$S(R(v_i), R(avg)) = \max \left(1 - \max \left(\frac{CC_w(R(avg), M_i(x))}{\ R(avg)\ _2 \times \ M_i(x)\ _2} \right) \right)$	
	puntuación de anomalía del nodo	$S(R(v_i), R(avg)) = \begin{cases} 1 - \max \left(\frac{CC_w(R(avg), R(v_i))}{\ R(avg)\ _2 \times \ R(v_i)\ _2} \right), v_i \in DEG \\ \max \left(1 - \max \left(\frac{CC_w(R(avg), M_i(x))}{\ R(avg)\ _2 \times \ M_i(x)\ _2} \right) \right), v_i \in FCG \end{cases}$	No especifica
	Probabilidad de acceso del nodo	$P(i) = \frac{S(R(v_i), R(avg))}{\sum_{i=1}^n S(R(v_i), R(avg))}$ $S(R(v_i), R(avg))$: puntuación de anomalía del nodo v_i N: Número total de nodos en MFPG	No especifica
	Probabilidad de acceso hacia adelante	$p_{a,i}^{forward} = p(i)$ P(l): Probabilidad de acceso del nodo hijo	No especifica
	Probabilidad de acceso hacia atrás	$p_{j,i}^{back} = \rho P(i)$ ρ: parámetro de acceso hacia atrás personalizado P(i): probabilidad de acceso del nodo padre	No especifica
	Probabilidad de permanencia	$P_i^{self} = \max \left(0, \frac{S(R(v_i), R(avg)) - \max_{j: e_{j,k} \in E} (S(R(v_i), R(avg)))}{\sum_{i=1}^n S(R(v_i), R(avg))} \right)$ Numerador: Diferencia entre la puntuación del nodo actual y la máxima de sus vecinos Denominador: Suma de todas las puntuaciones de anomalías	No especifica
	Precisión en los K primeros	$AP@K = \frac{1}{ A } \sum_{a \in A} \frac{\sum_i i: \varphi a(i) \leq k I_a(i)}{\min(K, \sum_i I_a(i))}$ $I_a(i)$: indicador si el nodo i es la causa real (1) o no (0) $\varphi a(i)$: posición del nodo i en el ranking K: Número de elementos top considerados	No especifica
	Precisión media promedio	$MAP = \frac{1}{ V } \sum_{1 \leq k \leq V } AP@K$ AP@K: Precisión en los K primeros V : Número de nodos en el grafo	No especifica
	Número de nodos falsos revisados antes del verdadero	$F_a(i) = \begin{cases} \sum_{j=1}^{\varphi_a(i)} (1 - I_a(i)), I_a(i) = 1 \\ 0, I_a(i) = 0 \end{cases}$	No especifica

Ref. Art.	Métrica	Formula	Umbrales
	Falsos positivos promedio	$AFP = \frac{1}{ A } \sum_{1 \leq K \leq V } \frac{\sum_{1 \leq i \leq V } F_a(i)}{\sum_{1 \leq i \leq V } I_a(i)}$ $F_a(i)$: Número de nodos falsos revisados antes del verdadero $I_a(i)$: Indicador de causa real	No especifica

Anexo C. Resúmenes de los trabajos relacionados.

"A Metrics Framework for Evaluating Microservices Architecture Designs" [1]: Propone y valida tres métricas clave (SGM, LCOM, NOO) adaptadas de la orientación a objetos para evaluar la calidad del diseño de la MSA, aplicando el marco a cinco aplicaciones de diferentes tamaños, incluyendo Kanban Boards

"A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures" [31]: Propone un marco de métricas para la Arquitectura Orientada a Servicios (SOA) que mide la complejidad, la agilidad y la flexibilidad de las soluciones SOA. El marco utiliza métricas tanto a nivel de servicio como a nivel global, clasificadas en tiempo de diseño y tiempo de ejecución, para predecir el estado "salud" de la arquitectura.

"A New Decomposition Method for Designing Microservices" [33]: Propone un algoritmo de descomposición que aplica Affinity Propagation a las especificaciones Open API para generar candidatos a microservicios, midiendo su rendimiento con precisión y recall en APIs grandes como AWS y PayPal.

"A Survey on Service Oriented Architecture"[6]: Describe métricas de acoplamiento y agregación para SOA (como TRS) y cómo utilizarlas para predecir la mantenibilidad durante la fase de diseño, lo cual permite construir el software de manera flexible.

"A method for monitoring the coupling evolution of microservice-based architectures" [7]: Presenta el método SYMBIOTE para monitorear la evolución del acoplamiento en MSA, utilizando métricas por servicio como la Importancia Absoluta del Servicio (AIS) y la Dependencia Absoluta del Servicio (ADS).

"A metrics suite for object oriented design" [32]: Propone la suite de métricas C&K (WMC, DIT, NOC, CBO, RFC, LCOM) para el diseño orientado a objetos, analizando el compromiso (trade-off) entre la reusabilidad mediante herencia y la simplicidad de comprensión.

"A quantitative assessment method for microservices granularity to improve maintainability" [34]: Tesis que desarrolla un método de evaluación cuantitativa para la granularidad de MSA con foco en la mantenibilidad, identificando y probando seis métricas clave (CC, SC, LOC, WSIC, SIDC, CF) en proyectos que experimentaron refactorizaciones.

"Assessing Architecture Conformance to Coupling-Related Patterns and Practices in Microservices" [35]: Define un metamodelo y propone métricas continuas (rango 0 a 1) basadas en un modelo de descomposición para evaluar la conformidad de los sistemas de microservicios con patrones y prácticas de acoplamiento específicos.

"Automatically Measuring the Maintainability of Service-and Microservice-based Systems - A Literature Review [36]: Revisión de literatura que cataloga 46 métricas de mantenibilidad para Sistemas Basados en Servicios (SBS) y analiza su aplicabilidad a MSA, asociándolas a cuatro propiedades: tamaño, complejidad, acoplamiento y cohesión.

"Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review"[37]: Revisión Sistemática que sintetiza los desafíos y las soluciones en MSA, incluyendo un proceso de evaluación de la calidad de los estudios primarios basado en la explicitud de las medidas y variables utilizadas.

Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective" [4]: Evalúa la suite de métricas de diseño orientado a objetos de Chidamber y Kemerer (C&K) (WMC, CBO, LCOM, etc.) aplicando principios de la teoría de la medición. Critica la definición de métricas como el Acoplamiento entre Clases de Objetos (CBO) y propone una formulación alternativa basada en la teoría de grafos para la Falta de Cohesión en Métodos (LCOM), con el fin de mejorar su precisión y validación empírica.

"Cohesion Metrics for Predicting Maintainability of Service-Oriented Software" [39]: Discute cómo la cohesión comunicacional (donde los elementos operan en el mismo conjunto de datos) puede utilizarse para predecir la mantenibilidad en diseños de servicios.

"Cohesion-Driven Decomposition of Service Interfaces Without Access to Source Code" [38]: Propone un enfoque de descomposición de interfaces de servicio impulsado por la cohesión, utilizando adaptaciones de métricas de falta de cohesión (LoC*) para mejorar la calidad y reducir la complejidad.

"Decomposition of Monolith Applications Into Microservices Architecture: A Systematic Review" [8]: Revisión Sistemática (SLR) que organiza las técnicas de descomposición de monolitos en MSA (MD, SA, DA), así como las métricas y los factores que influyen en esta migración.

"Defining and measuring microservice granularity—a literature overview" [40]: Revisión Sistemática (SLR) centrada en identificar los enfoques, métricas (acoplamiento, cohesión, granularidad) y atributos de calidad utilizados para definir y evaluar la granularidad de los microservicios.

"Evaluation of Microservice Architectures A Metric and Tool-Based Approach" [41]: Propone un método de evaluación basado en herramientas para MSA que utiliza datos de comunicación dinámicos para recuperar un modelo de dependencia estático y derivar métricas automáticas ejecutables.

"Evaluation of SOA-Based Web Services and Microservices Architecture Using Complexity Metrics" [14]: Compara empíricamente la Arquitectura de Microservicios (MSA) y los Servicios Web basados en SOA utilizando el bajo acoplamiento como perspectiva de evaluación. Propone un grafo de servicios para modelar las aplicaciones y extraer métricas como el Factor de Acoplamiento de Servicio (SCF). Concluye, basándose en estudios de caso, que la MSA logra un acoplamiento menor (SCF más bajo) y, por lo tanto, es menos compleja que los Servicios Web basados en SOA.

"Extracting Microservices Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach" [42]: Este trabajo propone un método de descomposición de monolitos a microservicios que se basa en el análisis de la interfaz de programación de aplicaciones (API) y evalúa estos candidatos utilizando métricas adaptadas de Falta de Cohesión (LCOM) y Número de Operaciones (NOO) para determinar si el microservicio necesita una descomposición adicional.

"From Monolith to Microservice Measuring Architecture Maintainability" [43]: Propone un modelo de calidad basado en métricas estructurales para asegurar que la migración de monolitos a MSA cumpla el objetivo de mejorar la mantenibilidad, lo cual es crucial para la gestión post-migración.

"From Monolithic Systems to Microservices: A Decomposition Framework Based On Process Mining"[44]: Propone un marco de descomposición de monolitos de 6 pasos que se basa en la minería de procesos (process mining) para identificar alternativas de división (slicing) evaluadas mediante métricas de tamaño y acoplamiento.

"Functionality-Oriented Microservice Extraction Based On Execution Trace Clustering" [45]: Presenta el método FoME, que utiliza trazas de ejecución para guiar el agrupamiento (clustering) de entidades de código monolítico, buscando la independencia funcional y la generación de APIs para microservicios candidatos.

"Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing" [46]: Desarrolla métricas de mantenibilidad y granularidad (como Sws, Recuento Ponderado de Interfaz) para guiar las decisiones de división de microservicios, enfocándose en un sistema regulatorio.

"Improving Observability in event sourcing Systems" [3]: Discute las necesidades de observabilidad en sistemas de event sourcing (que desacoplan aún más los componentes), enfatizando la necesidad de la trazabilidad distribuida (distributed tracing) y la gestión de metadatos para reconstruir las relaciones causales de los eventos.

"MicroValid: A Validation Framework for Automatically Decomposed Microservices" [51]: Marco de validación que evalúa la calidad de las descomposiciones automáticas mediante métricas como Nanoentities Composition y Semantic Similarity, utilizando el Coeficiente de Variación para identificar alta dispersión de tamaño.

"MFRL-CA: Microservice Fault Root Cause Location based on Correlation Analysis" [47]: Propone un método para la localización rápida y precisa de la causa raíz de fallos en MSA utilizando análisis de correlación, con el fin de mejorar la Calidad de Servicio (QoS) y la eficiencia operacional.

"Microservice Security Metrics for Secure Communication, Identity Management, and Observability" [48]: Define métricas formales, continuas (0 a 1), para evaluar automáticamente la conformidad arquitectónica con tácticas de seguridad clave como la comunicación segura y la gestión de identidad.

"Microservices Identification Methods and quality metrics" [49]: Disertación que propone un algoritmo de clustering jerárquico basado en la similitud semántica de los nombres de las operaciones de API para generar diseños de microservicios con alta escalabilidad y mantenibilidad.

"Microservices Monitoring with Event Logs and Black Box Execution Tracing" [50]: Presenta una solución de trazabilidad de caja negra y logs de eventos (MetroFunnel) para monitorear MSA y soportar decisiones informadas, logrando una reducción en el tamaño de los logs y bajo overhead de rendimiento.

"Microservices: yesterday, today, and tomorrow" [23]: Es una revisión y survey académica que presenta la evolución de la arquitectura de software, desde los monolitos hasta la Arquitectura Orientada a Servicios (SOA) y finalmente los microservicio. El trabajo detalla cómo los microservicios, al enfatizar el bajo acoplamiento y la alta cohesión, resuelven los problemas de los monolitos (como el mantenimiento difícil, el dependency hell, y el escalamiento subóptimo). También aborda los desafíos futuros, como la necesidad de interfaces formales, tipos de comportamiento y coreografías para garantizar la compatibilidad y la corrección de los sistemas distribuidos

"Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios" [52]: Propone el Microservices Backlog, un modelo inteligente y una práctica ágil para especificar y evaluar la granularidad de los microservicios en tiempo de diseño, utilizando algoritmos genéticos y de clustering basados en un vector de métricas multifactorial (CpT, CohT, CxT, SsT, WsicT).

"Nonintrusive Monitoring of Microservice-based Systems" [53]: Se enfoca en la recolección de métricas por solicitud (origen y destino) en un módulo de almacenamiento para extraer información de alto nivel, como la topología, el tiempo de respuesta y la carga de cada microservicio, esencial para el dimensionamiento y la QoS.

"Osmotic Monitoring of Microservices between the Edge and Cloud" [54]: Este trabajo presenta un sistema de monitoreo integrado (Osmotic Monitoring) diseñado para aplicaciones IoT descompuestas en microservicios y ejecutadas en un entorno de computación osmótica (es decir, a través de la Nube y el Borde de la red, Edge).

"Securing Microservices and Microservice Architectures: A Systematic Mapping Study" [55]: Realiza un Mapeo Sistemático (SMS) para identificar y clasificar las amenazas de seguridad y los mecanismos de protección en arquitecturas de microservicios (MSA). El estudio revela un foco desequilibrado en la investigación hacia los ataques externos (63%) y

la auditoría (43.48%). Propone una ontología ligera para estructurar las amenazas, los mecanismos de seguridad y sus niveles de aplicación (e.g., Soft-Infrastructure, API).

"Software Architectural Metrics for the Scania Internet of Things Platform From a Microservice Perspective" [56]: Evalúa la arquitectura de microservicios para una Plataforma IoT, concluyendo que para mejorar la flexibilidad se deben dividir los microservicios grandes y utilizar agregadores en el sistema.

"Structural Coupling for Microservices" [57]: Analiza el acoplamiento estructural (SC) y otras métricas de acoplamiento (Degree, CBM) en 17 proyectos de código abierto para evaluar la calidad arquitectónica de la MSA.

"The pains and gains of microservices A Systematic grey literature review" [58]: Revisión de literatura gris que identifica los desafíos (pains) y beneficios (gains) de MSA desde la perspectiva de la industria, destacando que la lógica de negocio pequeña facilita un desarrollo rápido y la comprensión para el nuevo personal.

"Understanding and Addressing Quality Attributes of Microservices Architecture: A Systematic Literature Review" [2]: Revisión Sistemática que identifica seis Atributos de Calidad clave (Scalability, Performance, Availability, Monitorability, Security, Testability) y codifica 19 tácticas arquitectónicas para abordarlos.

"Using Cohesion and Coupling for Software Remodularization: Is It Enough?" [59]: Propone un enfoque de optimización multi-objetivo utilizando métricas estructurales y conceptuales para la remodelación de software, buscando maximizar la cohesión y minimizar el acoplamiento a nivel de paquetes.

"Using the Conceptual Cohesion of Classes for Fault Prediction In Object-Oriented. Systems" [60]: Propone y evalúa métricas de cohesión conceptual basadas en el Análisis Semántico Latente (LSI) para predecir fallos en sistemas orientados a objetos, utilizando frases como la unidad básica de texto.

Anexo D. Implementación del modelo de calidad propuesto para evaluar microservicios.

Las siguientes expresiones no representan métricas matemáticas tradicionales, sino expresiones de agregación definidas como parte del proceso de operacionalización del modelo de calidad propuesto, es decir, dichas sumatorias permiten cuantificar el grado de cumplimiento de cada subatributo a partir de la presencia de determinados elementos del código fuente y sus artefactos asociados, para finalmente obtener una calificación global del atributo de flexibilidad.

1. Identificación de los artefactos de análisis

El proceso inicia con la selección de los artefactos relevantes del microservicio, entre los cuales se incluyen:

- Código fuente del microservicio (clases, interfaces, módulos).
- Archivos de configuración (YAML, JSON, XML, CSV).
- Definiciones de interfaces y contratos (APIs REST)
- Declaración de dependencias (pom.xml, build.gradle, package.json)
- Scripts de despliegue y dependencias (Dockerfiles, Manifest, Configuraciones de infraestructuras).
- Documentación técnica asociada cuando aplique (manuales de instalación, configuraciones soportadas, README).

2. Evaluación de Adaptabilidad

La adaptabilidad se evalúa analizando el grado en que el código facilita la interoperabilidad del microservicio a partir del diseño estructural y de los estándares empleados.

➤ Interoperabilidad Sintáctica

La revisión del código permite identificar principalmente los formatos de intercambio de datos que soporta el microservicio. Dentro de los cuales se encuentran el JSON, XML, CSV, entre otros.

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

Una vez identificados los formatos soportados por el microservicio, se lleva a cabo una sumatoria de los porcentajes obtenidos para obtener el porcentaje total asignado a la interoperabilidad sintáctica.

Formalmente:

$$I.Sintáctica = JSON + XML + CSV + Otros$$

Formula 1 Interoperabilidad Sintáctica

➤ Interoperabilidad Semántica

Para este punto, se analiza la calidad semántica del microservicio, considerando la claridad conceptual de los datos que intercambia y la coherencia entre la implementación y su documentación mediante la presencia de documentación de APIs (Swagger/OpenAPI, README) y la consistencia de los nombres de los endpoints, parámetros y modelos, apoyándonos en la coherencia observable dentro del código y la documentación.

Formalmente:

$$ISM = MS + RV + ED$$

Formula 2 Interoperabilidad Semántica

MS: Modelo semántico

RV: Reglas de validación

ED: Estandarización de datos

➤ Interoperabilidad de Protocolos

La revisión estática del código permitió identificar los protocolos de comunicación utilizados, como HTTP o HTTPS, gRPC, entre otros. Asimismo las configuraciones de clientes y servidores. La calificación refleja si el microservicio se limita solo a un protocolo o si evidencia soporte para múltiples protocolos a la vez.

Formalmente:

$$I.de Protocolos = HTTP/HTTPS + Otros$$

Formula 3 Interoperabilidad de protocolos

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

➤ Interoperabilidad de Seguridad

Se realiza una inspección del código fuente y de los archivos de configuración del microservicio con el objetivo de verificar el uso de mecanismos de seguridad estandarizados como TLS/SSL, OAuth2, OpenID Connect, JWT, entre otros. Con base en dicha inspección se evalúa si el microservicio implementa estándares que permiten intercambiar datos de forma segura.

Formalmente:

$$I.Seguridad = AE + CI + GT$$

Formula 4 Interoperabilidad de Seguridad

Donde:

AE: Autenticación estándar

CI: Cifrado para intercambio

GT: Gestión de tokens

➤ Interoperabilidad Operacional

Para evaluar la interoperabilidad operacional se considera la capacidad del microservicio para operar en distintos entornos, ya sea mediante el uso de contenedores Docker, variables de entorno o evidencias de soporte para orquestadores y/o pipelines de CI/CD.

Al haber obtenido los porcentajes de cada dimensión asignada a interoperabilidad operacional, se realiza una sumatoria para poder asignar un porcentaje total de dicha interoperabilidad.

Formalmente:

$$I.Operacional = Nubes + Sistema Operativo + Dispositivos$$

Formula 5 Interoperabilidad Operacional

Una vez evaluada cada dimensión de interoperabilidad de forma independiente, los valores obtenidos se suman para obtener una calificación total del atributo de adaptabilidad.

Formalmente:

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

$$\text{Adaptabilidad} = I.\text{Sintactica} + I.\text{Semantica} + I.\text{de Protocolos} + I.\text{de Seguridad} \\ + I.\text{Operacional}$$

Formula 6 Adaptabilidad

3. Evaluación de Escalabilidad

La escalabilidad se evalúa analizando decisiones estructurales que afectan la capacidad empresarial del microservicio, es decir, si el diseño interno permite atender un incremento en el número de peticiones realizadas por los usuarios, volumen de datos y número de instancias, manteniendo tiempos de respuesta aceptables.

➤ Operacional

Se revisó si el código está diseñado para una ejecución distribuida, donde se denote la ausencia de un estado compartido, así como la independencia entre las diferentes instancias y por último, si existen configuraciones para el autoescalado y la replicación.

A partir de esta revisión, se evalúa la preparación del repositorio para ser desplegado y gestionado en entornos de nube, mediante la siguiente expresión.

$$E.\text{Operacional} = C + O + CE$$

Formula 7 Interoperabilidad Operacional

Donde:

C: Contenerización

O: Orquestación

CE: Configuración Externa

➤ Datos

En este atributo, se analiza si el código presenta soporte para bases de datos escalables o distribuidas, así como el uso de repositorios y abstracciones que ayuden al acoplamiento entre la lógica del negocio y las bases de datos. Este enfoque permite evaluar la capacidad del microservicio para soportar incrementos en el volumen de información sin afectar la mantenibilidad o el desempeño.

Formalmente:

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

$$ED = PG + CH + MD$$

Formula 8 Escalabilidad de Datos

Donde:

PG: Paginación

CH: Presencia de implementación de estrategias de manejo de caché

MD: Manejo de datos distribuidos

➤ Rendimiento

Se busca dentro del código si existen operaciones que puedan bloquear el uso de mecanismos que faciliten el procesamiento en paralelo o asíncrono, según sea el caso, así como el manejo de las concurrencias. Asimismo se analiza la forma en que el código gestiona la concurrencia, evitando secciones innecesarias o dependencias que limiten el aprovechamiento de los recursos cuando exista una gran carga de datos.

Formalmente:

$$E.Rendimiento = AP + RL + OC + ES$$

Formula 9 Interoperabilidad de Rendimiento

Donde:

AP: Procesamiento asincrono

RL: Limite de recursos

OC: Optimización de consultas

ES: Entradas y Salidas

4. Evaluación de Portabilidad

Bajo el contexto del modelo propuesto, la portabilidad es definida como “la capacidad de un sistema para ser instalado, desplegado, sustituido y ejecutado en diferentes entornos” [20], por lo cual la revisión del código estático permite detectar limitaciones estructurales desde etapas tempranas del desarrollo.

➤ Instalabilidad

La instalabilidad se evalúa mediante la inspección de las dependencias (que sean declaradas de manera clara y reproducible), la automatización del proceso de instalación y la calidad y completitud del manual de instalación según el estándar de la IEEE 1063. Ya que su función

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

es determinar si el microservicio puede instalarse y funcionar de manera reproducible en diferentes entornos.

Una vez obtenidas las dimensiones recomendadas por el estándar de la IEEE 1063 para el manual de instalación, se realiza una sumatoria con el fin de poder asignarle un porcentaje, mismo que será asignado a instalabilidad.

Formalmente:

$$\text{Instalabilidad} = \text{Manual de instalación}$$

Formula 10 Instalabilidad

$$\text{Manual de instalación} = P + In + I + RS + II + ID + SPC + SC + R$$

Formula 11 Manual de instalación

Donde:

P: Portada

In: Índice

I: Introducción

RS: Requisitos del sistema

II: Instrucciones de instalación

ID: Instrucciones de desinstalación

SPC: Solución a problemas comunes

SC: Soporte y Contacto

R: Referencias

➤ Reemplazabilidad

La reemplazabilidad se evalúa a partir de la facilidad con la que el microservicio puede ser sustituido por otro equivalente, por lo que se consideraron los siguientes aspectos:

- Compatibilidad de interfaces

Se realizó una revisión del código fuente y de los artefactos asociados con la definición de las interfaces del microservicio para verificar la estabilidad, así como, a la vez, si cumplen con un versionado de las APIs y contratos de interfaz formales y documentados.

Formalmente:

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

$$C.Interfaces = DC + VE + EC$$

Formula 12 Compatibilidad de Interfaces

Donde:

DC: Documentación del contrato

VE: Versiones estáticas

EC: Esquema y Cumplimiento

- Equivalencia funcional

Se busca dentro del código fuente con el objetivo de verificar que las funciones principales del microservicio estén claramente definidas, exista una adecuada separación de las responsabilidades y cuente con pruebas automatizadas que validen que se obtenga el resultado esperado.

Formalmente:

$$E.Funcional = PP + CD + MR$$

Formula 13 Equivalencia Funcional

PP: Presencia de pruebas

CD: Claridad del diseño

MR: Modularidad y responsabilidades

- Independencia de implementación

Se realizó una inspección del código fuente y de los archivos de configuración con el objetivo de identificar el grado de dependencia del microservicio con respecto a tecnologías y frameworks, así como el uso de algún estándar que facilite su reimplementación en otros lenguajes o plataformas.

Formalmente:

$$I.Implementación = IA + DV + PE$$

Formula 14 Independencia de Implementación

IA: Infraestructura Abstracta

DV: Dependencias del vendor

PE: Protocolos Estandar

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

Por último, los resultados obtenidos son sumados para asignar propiamente un valor a el atributo de reemplazabilidad.

Formalmente:

$$\textit{Reemplazabilidad} = \textit{C.Interfaces} + \textit{E.Funcional} + \textit{I.Implementación}$$

Formula 15 Reemplazabilidad

Por último se hace la sumatoria de los resultados de Instalabilidad y Reemplazabilidad para poder asignar una calificación al atributo de portabilidad.

Formalmente:

$$\textit{Portabilidad} = \textit{Instalabilidad} + \textit{Reemplazabilidad}$$

Formula 16 Portabilidad

5. Asignación de métricas y ponderación

Una vez identificadas las subcaracterísticas que conforman el modelo de calidad, se extraen las siguientes formulas con las cuales podemos evaluar el grado de flexibilidad presente en un solo microservicio:

$$\textit{Flexibilidad} = \textit{Adaptabilidad} + \textit{Escalabilidad} + \textit{Portabilidad}$$

Formula 17 Flexibilidad

Nota metodológica: Todas las expresiones presentadas en esta sección corresponden a sumatorias de indicadores definidos por el autor para la aplicación del modelo de calidad propuesto. Estas operaciones tienen como objetivo operacionalizar los atributos de calidad establecidos en la ISO/IEC 25010 en el contexto de microservicios.

Anexo E. Revisión Sistemática de la Literatura

Para esta investigación se realizó una revisión sistemática con el fin de identificar las métricas existentes en la literatura, las cuales, a su vez, solventan los requerimientos de calidad aplicables a microservicios.

Dicha revisión fue llevada a cabo con base en la adaptación realizada por *Torres-Carrion et.al [77]* al modelo de Kitchenham y Baca, el cual se divide en tres etapas: “Planificación, Desarrollo de la revisión e Informar sobre la revisión”. La *Figura 23* muestra los pasos que fueron abordados a lo largo de la revisión.



Figura 23 Metodología de revisión sistemática de la literatura según Torres-Carrion et. al [77]

Torres-Carrion [77] sugiere que partir del estado temprano del conocimiento del problema, realizando las preguntas de investigación pertinentes y del mentefacto conceptual, es posible investigar acerca de RSL relacionadas con nuestro tema, posteriormente plantea la siguiente pregunta “¿Existen RSL relacionadas?” y dependiendo de la respuesta, sugiere realizar la revisión sistemática con el fin de llegar al estado actual de la investigación del problema, tal cual lo muestra la *Figura 24*.

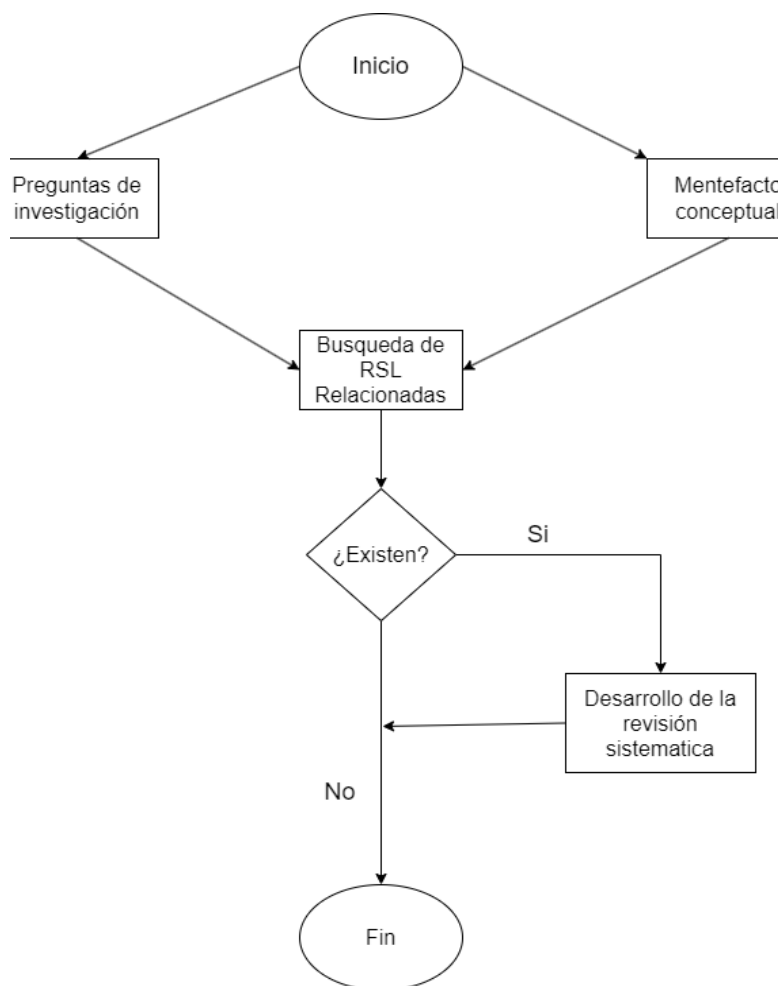


Figura 24 Procedimiento de la metodología propuesta por Torres-Carrion Et. al [77].

Partiendo de esta premisa, a continuación, se describe el desarrollo de las etapas de la revisión sistemática.

1. Planificación

1.1. Identificación de la necesidad de la revisión.

La creciente adopción de arquitecturas de microservicios ha motivado el desarrollo de múltiples métricas orientadas a evaluar sus propiedades estructurales y de calidad. Sin embargo, dichas métricas se encuentran dispersas en la literatura, presentan distintos enfoques y, en muchos casos, no se encuentran explícitamente alineadas con modelos formales de calidad de software como la norma ISO/IEC 25010.

Es por esta razón que, determinar las métricas que solventan los atributos de calidad definidos para un microservicio se vuelve una tarea indispensable debido a la basta cantidad de métricas que existen en la literatura y la nula o poca experiencia que puede tener el usuario final.

Ante este escenario, se consideró necesario realizar una revisión sistemática de la literatura que permitiera identificar, analizar y clasificar las métricas propuestas para microservicios, así como examinar su relación con atributos y subatributos de calidad.

1.1.1. Estado actual del problema de investigación.

Hoy en día existe una basta cantidad de artículos relacionados con microservicios y sus atributos en cuanto a calidad, lo cual da como resultado que la información disponible con relación a las métricas aplicadas para cuantificar los umbrales para dichos atributos se encuentre bastante dispersa.

Con la revisión sistemática orientada a identificar las métricas existentes para cuantificar los atributos de los microservicios se busca conocer las métricas que existen, sus umbrales, como aplicarlas, así como su modelo matemático.

1.1.2. Preguntas de investigación.

1. RQ1. ¿Qué métricas existen en diseño e implementación de Microservicios?
2. RQ2. ¿Cuáles son los ambientes en los que son implementadas dichas métricas?
3. RQ3. ¿Cuáles son las métricas que atienden los atributos de calidad que caracterizan a un microservicio?
4. RQ4. ¿Las métricas propuestas cuentan con un sustento o validación formal?

1.1.3. Mentefacto Conceptual

La *Figura 25* presenta una representación de los conceptos asociados a los microservicios, los cuales, a su vez, fueron tomados en cuenta para realizar la revisión sistemática de la literatura con respecto a las métricas que evalúan los atributos de calidad que los caracterizan.

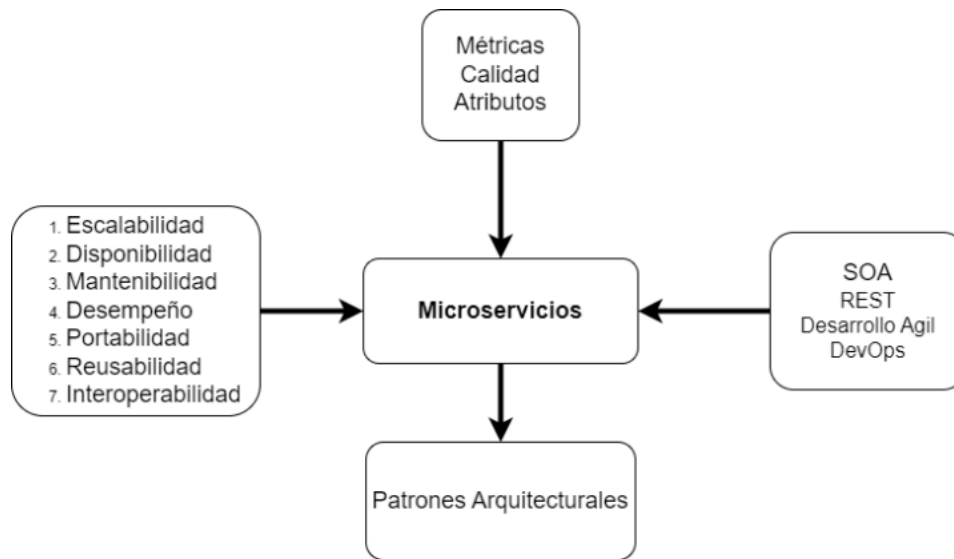


Figura 25 Mentefacto Conceptual del concepto de microservicios.

1.1.4. Revisiones sistemáticas relacionadas.

Se realizó una revisión exploratoria con el fin de identificar términos que hagan referencia a métricas de microservicios, los cuales son representados en la *Figura 26*.



Figura 26 Nube de palabras referentes a métricas de microservicios

La *Figura 26* fue generada mediante la herramienta en línea disponible en el sitio web *Nube de palabras* (<https://www.nubedepalabras.es/>).

Este conjunto de términos es de gran utilidad para poder definir las cadenas de búsqueda apropiadas para la revisión sistemática de la literatura. Dichas cadenas se aplicaron a todo documento existente en las bases de datos electrónicas consideradas (IEEEExplorer, ACM, Google Academic).

Tabla 16 Cadenas de búsqueda

<i>Cadenas de búsqueda</i>	
<i>1</i>	<i>Flexibility metrics for microservices</i>
<i>2</i>	<i>Reliability metrics for microservices</i>
<i>3</i>	<i>Maintainability metrics for microservices</i>
<i>4</i>	<i>Security metrics for microservices</i>
<i>5</i>	<i>Compatibility metrics for microservices</i>
<i>6</i>	<i>Performance efficiency metrics for microservices</i>

Es importante considerar que, aunque la *Tabla 16* presenta las cadenas de búsqueda utilizadas para realizar la revisión sistemática, estas deben ser empleadas con filtros que en dependencia de la base de datos consultada, podrían variar. Es por esta razón que las tablas 17, 18 y 19 presentan los artículos obtenidos por búsqueda, así como los filtros empleados para cada una de las bases de datos.

Tabla 17 Cadenas de búsqueda en IEEEExplorer

<i>Cadenas de búsqueda en IEEEExplorer</i>		
<i>No.</i>	<i>Cadena de búsqueda</i>	<i>No. Artículos encontrados</i>
<i>1</i>	<i>Flexibility metrics for microservices</i>	<i>40</i>
<i>2</i>	<i>Reliability metrics for microservices</i>	<i>96</i>
<i>3</i>	<i>Maintainability metrics for microservices</i>	<i>22</i>
<i>4</i>	<i>Security metrics for microservices</i>	<i>81</i>
<i>5</i>	<i>Compatibility metrics for microservices</i>	<i>3</i>
<i>6</i>	<i>Performance efficiency metrics for microservices</i>	<i>59</i>
<i>Filtros seleccionados: Year Range: 2014-2024</i>		

NOTA: Google Académico (Google Scholar) funciona como un motor de búsqueda académico y no como una base de datos científica estructurada. No obstante, en el presente trabajo se empleó como herramienta complementaria para identificar publicaciones adicionales y rastrear trabajos relevantes que posteriormente fueron verificados y consultados en sus respectivas bases de datos de origen.

Tabla 18 Cadenas de búsqueda en ACM

Cadenas de búsqueda en ACM		
No.	Cadena de búsqueda	No. Artículos encontrados
1	<i>Flexibility metrics for microservices</i>	50
2	<i>Reliability metrics for microservices</i>	3107
3	<i>Maintainability metrics for microservices</i>	3101
4	<i>Security metrics for microservices</i>	3913
5	<i>Compatibility metrics for microservices</i>	3189
6	<i>Performance efficiency metrics for microservices</i>	1573
Filtros seleccionados: Year Range: 2014-2024		

Tabla 19 Cadenas de búsqueda en Google Academic

Cadenas de búsqueda en Google Academic		
No.	Cadenas de búsqueda	No. Artículos Encontrados
1	<i>Flexibility metrics for microservices</i>	1490
2	<i>Reliability metrics for microservices</i>	1590
3	<i>Maintainability metrics for microservices</i>	374
4	<i>Security metrics for microservices</i>	1800
5	<i>Compatibility metrics for microservices</i>	574
6	<i>Performance efficiency metrics for microservices</i>	1730
Filtros seleccionados: 1. Tipo: Artículos de revisión 2. Year Range: 2014-2024		

1.2.Desarrollo de un protocolo de revisión

Una vez obtenidos los resultados de las cadenas de búsqueda, se filtraron y seleccionaron los artículos que se apegan a los siguientes criterios de inclusión y exclusión.

1.2.1. Definición de criterios de Inclusión y Exclusión

El objetivo fue identificar trabajos que propusieran, utilizaran o validaran métricas aplicables a arquitecturas de microservicios, incluyendo estudios teóricos y empíricos, para ello se implementaron los siguientes criterios de inclusión y exclusión.

NOTA: Google Académico (Google Scholar) funciona como un motor de búsqueda académico y no como una base de datos científica estructurada. No obstante, en el presente trabajo se empleó como herramienta complementaria para identificar publicaciones adicionales y rastrear trabajos relevantes que posteriormente fueron verificados y consultados en sus respectivas bases de datos de origen.

1.2.1.1. Criterios de Inclusión

- Artículos que contengan o hablen de métricas para microservicios.
- Artículos que contengan o hablen de adaptaciones de métricas para microservicios.
- Artículos que contengan o hablen de las características de los microservicios.
- Métricas que se acoplen al modelo de la norma ISO/IEC 25010.
- Métricas que analicen los atributos que caracterizan a un microservicio.
- Métricas con o sin umbrales.

1.2.1.2. Criterios de exclusión

- Estudios con abstract o título no relacionado.
- Estudios incompletos.
- Duplicados.
- Estudios que no hablan explícitamente de métricas o calidad para MSA.
- Estudios que no reflejan los conceptos de las MSA
- Estudios que no se relacionan con métricas, migración o características de microservicios.

1.2.2. Preparación de un formulario de extracción de datos

Una vez seleccionados los artículos, se realizó una extracción estructurada de la información recopilada, quedando de la siguiente manera:

- Nombre del artículo
- Nombre de la métrica
- Fórmula matemática (Cuando estuviera disponible)
- Umbrales (Cuando el autor los declarara explícitamente)

La gestión de toda la información recopilada en la investigación se realizó en un “Fichero Bibliográfico”. Dicho fichero contiene los metadatos de los artículos consultados, estos a su vez fueron registrados en un Excel.

1.2.3. Selección de revistas

La presente investigación se realizó siguiendo los pasos propuestos en la *Figura 23*, la cual condujo la búsqueda al uso de las siguientes bases de datos:

- IEEExplorer
- ACM
- Google Academic

2. Realización de la revisión

Se llevó a cabo la revisión sistemática de la literatura con la finalidad de conocer las métricas existentes para microservicios, su relación con los atributos de calidad que definen a un microservicio, así como el grado de madurez de dichas métricas.

Derivado de la búsqueda se seleccionaron los artículos que fueron considerados “más relevantes” y fueron registrados en un fichero analítico en Excel con la finalidad de llevar un control adecuado de los mismos, así como su estatus.

Se llevó a cabo una primera revisión de los títulos y resumen de cada uno de ellos para validar que dentro de ellos se manejaban métricas orientadas a microservicios.

Los trabajos seleccionados fueron organizados en 2 categorías: “Artículos de métricas”, “Artículos teóricos” con el fin de revisar más a detalle aquellos trabajos que han superado los criterios de inclusión y exclusión.

- **Artículos de métricas:** Son aquellos que contienen métricas específicas.
- **Artículos teóricos:** Para aquellos que contienen definiciones, explicaciones o teoría útil.

La revisión fue llevada a cabo sobre las bases de datos de IEEExplorer, ACM y Google Academic con base en las cadenas mostradas en las tablas 17, 18 y 19, con las cuales se identificaron los siguientes artículos:

- Artículos de métricas: 33
- Artículos teóricos: 5

2.1. Clasificación y análisis de métricas

Tras analizar los trabajos encontrados, se identificaron 258 formas de evaluar a los microservicios, estas formas fueron sometidas a un proceso de análisis cualitativo y comparativo con el propósito de:

1. Identificar similitudes conceptuales
2. Detectar redundancias o variaciones en los términos

Posteriormente, se realizó un proceso de mapeo conceptual entre las métricas encontradas y los atributos de calidad definidos en la norma ISO 25010. Esta asociación se llevó a cabo considerando lo siguiente:

- Correspondencia conceptual entre las definiciones propuestas en la norma ISO 25010 y de la métrica declarada por el autor.
- Fundamentación teórica derivada del modelo de calidad.
- Declaraciones de los autores.

Este análisis permitió identificar qué atributos cuentan con mayor respaldo y cuales presentan menor cobertura en la literatura.

2.2. Análisis del contexto de aplicación y validación

Se examinó el tipo de validación reportada en los estudios seleccionados, haciendo una distinción entre:

- Validaciones experimentales bajo ambientes controlados.
- Aplicaciones en casos de estudio.
- Implementaciones en entornos industriales.
- Propuestas conceptuales sin validación empírica.

Este análisis permitió evaluar el nivel de consolidación y evidencia empírica asociada a las métricas identificadas, así como identificar tendencias en los contextos de aplicación.