Recibido 2 Nov 2025 *ReCIBE, Año 15 No. 1, mayo 2025*

Aceptado 22 Abri 2026

Evaluación de flexibilidad para microservicios basado en atributos de calidad

Flexibility assessment for microservices based on quality attributes

Said Misael Venoso Lara[1]
m24ce008@cenidet.tecnm.mx
Juan Carlos Rojas Pérez[1]
juan.rp@cenidet.tecnm.mx
Olivia Graciela Fragoso Díaz[1]
olivia.fd@cenidet.tecnm.mx

[1]*Centro Nacional de Investigación y Desarrollo Tecnológico Cuernavaca Morelos, México*

Resumen.
En los últimos años, la Arquitectura de Microservicios (MSA) se ha consolidado como un enfoque clave para el desarrollo de sistemas distribuidos, ofreciendo ventajas como flexibilidad, escalabilidad y agilidad. No obstante, la literatura evidencia una carencia de un esquema de calidad específico que permita evaluar atributos críticos en este contexto, particularmente la flexibilidad, ya que las métricas heredadas de otras arquitecturas no siempre resultan aplicables o carecen de validación empírica. Ante esta limitación, se presenta un esquema de evaluación de la flexibilidad, organizada en tres características principales: adaptabilidad, escalabilidad y portabilidad, cada una con subatributos específicos derivados de la ISO/IEC 25010, la ISO/IEC 9126, la ISO/IEC 2382, entre otras fuentes. Los resultados preliminares, obtenidos a partir del análisis de catorce microservicios en Java asociados a tres repositorios: Zull (47.37%), ACME Air Microservice (43.47%) y E-Commerce (33.7%) sugieren que la adopción de microservicios no garantiza por sí sola una alta flexibilidad, sino que depende del cumplimiento de prácticas de diseño adecuadas. El esquema permitió identificar debilidades recurrentes en documentación, instalabilidad y mecanismos de seguridad, lo que confirma su utilidad como herramienta para visibilizar áreas de mejora y orientar el diseño de sistemas más robustos y adaptables.

Palabras Clave: Microservicios - Atributos de calidad - Métricas de calidad - Desarrollo - Flexibilidad - Medición - Modelo de calidad - Métricas.

Abstract.
In recent years, Microservices Architecture (MSA) has established itself as a key approach to distributed systems development, offering advantages such as flexibility, scalability, and agility. However, the literature shows a lack of a specific quality framework that allows for the evaluation of critical attributes in this context, particularly flexibility, since metrics inherited from other architectures are not always applicable or lack empirical validation. Given this limitation, we present a flexibility assessment framework organized into three main characteristics: adaptability, scalability, and portability, each with specific sub-attributes derived from ISO/IEC 25010, ISO/IEC 9126, ISO/IEC 2382, among other sources. Preliminary results, obtained from the analysis of fourteen microservices in Java associated with three repositories: Zull (47.37%), ACME Air Microservice (43.47%), and E-Commerce (33.7%), suggest that the adoption of microservices alone does not guarantee high flexibility, but rather depends on compliance with appropriate design practices. The scheme allowed us to identify recurring weaknesses in documentation, installability, and security mechanisms, confirming its usefulness as a tool for highlighting areas for improvement and guiding the design of more robust and adaptable systems.

Keywords: Microservices - Quality attributes – Quality metrics – Development – Flexibility – Measurement – Quality model - Metrics.

1. Introducción

En los últimos años, la Arquitectura de Microservicios (MSA) se ha consolidado como una de las principales estrategias para el desarrollo de aplicaciones distribuidas, ofreciendo ventajas como flexibilidad, escalabilidad y agilidad [8,10,11]. Este enfoque permite descomponer sistemas monolíticos en componentes independientes, facilitando su mantenimiento y evolución. Sin embargo, a pesar de sus ventajas evidentes en escalabilidad y despliegue, la evaluación de la calidad en microservicios sigue representando un desafío técnico y metodológico relevante pues las métricas empleadas en Arquitecturas Orientadas a Objetos (OOA), Arquitecturas Orientadas a Servicios (SOA) o Computación Orientada a Servicios (SOC) no siempre pueden aplicarse directamente sobre las MSA [17, 38]. En muchos casos, las métricas propuestas para estas arquitecturas se han quedado en etapas conceptuales, carecen de validación empírica o presentan dificultades de implementación en entornos de microservicios [8, 41].

La calidad en el contexto de los microservicios no puede medirse únicamente con los criterios tradicionales aplicados a arquitecturas monolíticas u orientadas a objetos, pues la MSA introduce dimensiones adicionales relacionadas con la distribución, la comunicación entre servicios, la autonomía y la independencia del despliegue [12,13]. Autores como Dragoni et al. [12] y Soldani et al. [27] destacan que, si bien la MSA facilita la evolución del software, también incrementa la complejidad arquitectónica y operacional, lo que impacta directamente en atributos de calidad como la mantenibilidad, la seguridad y, especialmente, la flexibilidad.

De acuerdo con la ISO/IEC 25010 [2], la flexibilidad forma parte de los atributos vinculados a la capacidad de un sistema para responder a cambios en su entorno o en los requisitos. No obstante, esta definición resulta general y, por tanto, insuficiente para capturar las particularidades de los microservicios, donde la flexibilidad no solo depende del código fuente, sino también de la configuración del entorno, la documentación técnica, el desacoplamiento de servicios y la interoperabilidad. En este sentido, estudios como los de Al-Debagy y Martinek [8] o Vera-Rivera [15,16] coinciden en que el desafío no radica únicamente en definir métricas, sino en adaptarlas a las condiciones dinámicas y distribuidas de la MSA.

Autores como Berander [9] señalan que medir la calidad del software es un proceso complejo que implica seleccionar un esquema de calidad adecuado, identificar los atributos relevantes y definir métricas claras y medibles. Actualmente, dentro de la literatura se menciona que, en microservicios, existe una falta de consenso sobre qué métricas utilizar para ciertos atributos de calidad, cómo medirlas y en qué situaciones aplicarlas [15-19]. Esta ausencia de estandarización no solo genera ambigüedad en la evaluación, sino que también aumenta el riesgo de problemas como acoplamiento excesivo, baja cohesión, vulnerabilidades de seguridad y dificultades de mantenimiento. Por ello, surge la necesidad de contar con esquema de calidad específicos que permitan medir de manera objetiva dichos atributos, ofreciendo a desarrolladores y arquitectos una guía clara para garantizar que el diseño e implementación de microservicios cumpla con los estándares de calidad requeridos.

El presente trabajo propone un esquema para evaluar la flexibilidad en microservicios, abordando así la brecha de evaluación existente. El esquema se fundamenta en las normas internacionales (ISO/IEC 25010 [2], ISO/IEC 9126 [3], ISO/IEC 2382[4]) y literatura reciente. Para validar dicho esquema, se aplica un análisis preliminar sobre repositorios de prueba en Java, midiendo la flexibilidad total a través de la evaluación de sus tres subatributos propuestos por el esquema (adaptabilidad, escalabilidad, portabilidad).

Durante el análisis realizado en esta investigación de la arquitectura MSA se determinó que los desarrolladores a menudo suelen pasar a segundo plano el atributo de flexibilidad,

ya que la MSA ofrece beneficios como escalabilidad, facilidad de mantenimiento e independencia [8, 12, 37], que son características contempladas por la ISO/IEC 25010 como subatributos, y que a menudo suelen atribuirse a una arquitectura flexible. En el presente trabajo se llevó a cabo un análisis preliminar de catorce microservicios en tres repositorios para conocer el grado de flexibilidad presente en estos de acuerdo con las normas y estándares citados.

El objetivo central es proporcionar un esquema medible y adaptable que permita valorar la flexibilidad desde un enfoque empírico, brindando una herramienta útil para desarrolladores y arquitectos en la toma de decisiones relacionadas con el diseño, la refactorización y la evaluación de arquitecturas basadas en microservicios.

2. Trabajos relacionados

La evaluación de la calidad del software en arquitecturas de microservicios (MSA) continúa siendo un desafío tanto conceptual como práctico. A pesar de su adopción masiva en la industria, particularmente en sistemas de alta disponibilidad y entornos de despliegue continuo, los marcos teóricos y las métricas específicas para MSA aún están en consolidación. Las ventajas tradicionalmente atribuidas a las MSA, como la flexibilidad, la independencia de despliegue y la capacidad de escalado selectivo, han sido ampliamente documentadas [2, 23, 24]; sin embargo, los métodos cuantitativos para medir estos beneficios no han seguido el mismo ritmo de evolución.

En esta revisión de la literatura se clasifican los trabajos existentes en tres categorías principales, aplicación de métricas de arquitecturas monolíticas y orientadas a servicios (OOA y SOA), enfoques centrados en métricas de granularidad y acoplamiento, y modelos de calidad conceptuales que carecen de validación empírica.

2.1 Métricas Tradicionales (OOA y SOA)

Las métricas empleadas en OAA y SOA fueron pioneras en la cuantificación de la calidad estructural y funcional del software. Sin embargo, no son directamente aplicables a MSA. Los trabajos de la era de SOA, por ejemplo, se centraron extensamente en la cohesión de servicios y la complejidad de la orquestación [24, 37, 38, 39]. Si bien son relevantes, esas métricas fallan en capturar la naturaleza de MSA, que favorece la coreográfica sobre la orquestación y define la independencia de despliegue como un pilar central, un concepto ajeno a la mayoría de las métricas de OOA.

En el contexto de microservicios, la granularidad del despliegue y la autonomía del ciclo de vida introducen nuevas variables que no se contemplaban en las métricas tradicionales, por lo que éstas no capturan adecuadamente dependencias asíncronas o la interacción mediante colas de mensajería [15, 38, 40]. Además, la noción de interoperabilidad de seguridad o de configuración dinámica están ausente en estos modelos [39, 42].

2.2 Granularidad y Acoplamiento

La investigación más reciente sobre calidad en MSA se ha centrado casi exclusivamente en granularidad y la medición de acoplamiento, considerados como indicadores indirectos de flexibilidad y mantenibilidad. En cuanto a la granularidad autores como Vera-Rivera han propuesto dentro de sus trabajos [15, 16] modelos para definir la granularidad, pero el enfoque suele ser como definir el tamaño del microservicio, no cómo el tamaño resultante impacta en un atributo de calidad superior como flexibilidad. Para el acoplamiento [8, 14, 30, 38, 40] proponen métricas para medir el acoplamiento estructural y el acoplamiento sincrónico (llamadas HTTP directas) entre servicios. Estos trabajos son fundamentales y se alinean con nuestra investigación. Sin

embargo, estos enfoques son insuficientes. Miden indicadores de flexibilidad (como el bajo acoplamiento) pero no miden la flexibilidad en sí misma. Un sistema puede tener bajo acoplamiento y aun así ser inflexible si, por ejemplo, su instalabilidad es pobre o si carece de interoperabilidad semántica para adaptarse a nuevos requisitos comerciales.

2.3 Modelos de Calidad Integrales

La principal motivación de este trabajo es abordar la falta de un esquema integral y validado para la flexibilidad. Mientras que la norma ISO/IEC 25010 provee una base taxonómica, su aplicación directa es demasiado genérica. Otros autores han propuesto modelos de calidad para MSA, pero tienden a permanecer a nivel conceptual o se enfocan en otros atributos [35, 36]. Sin embargo, estos modelos suelen quedarse en el plano conceptual y carecen de validación empírica en entornos de producción. Adicionalmente, pocas propuestas han considerado la flexibilidad como un atributo autónomo de calidad, tratándola más bien como una consecuencia indirecta de otros factores.

A la fecha de realización de este trabajo, no se han encontrado trabajos que propongan un método de evaluación para la flexibilidad utilizando una descomposición jerárquica como la que se propone en este trabajo y que esté soportada en múltiples estándares ISO. El esquema propuesto permite evaluar la flexibilidad en la arquitectura MSA a través de descomponer la flexibilidad en subatributos medibles (adaptabilidad, escalabilidad y portabilidad), y establece un método de ponderación para su evaluación. El esquema aquí presentado fue validado mediante la aplicación a proyectos de código abierto, permitiendo una interpretación cuantitativa y replicable del atributo de flexibilidad en MSA.

3. Descripción del esquema

Para el desarrollo del esquema de calidad es importante definir el objeto a medir, de modo que la definición permita identificar las propiedades a evaluar. La ISO/IEC 25010 define la flexibilidad como “La capacidad del producto para adaptarse a cambios en sus requisitos, contextos de uso o entorno del sistema” [2], este término resulta ser bastante general, por lo que podría aplicar a ciertos modelos de arquitecturas de software, pero no precisamente en todas.

De acuerdo con Martin Fowler [1] el término Microservicio: “Es 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 construidos alrededor de requisitos comerciales y son desplegables de manera independiente por maquinaria de despliegue completamente automatizada”, donde se enfatiza la independencia, el desacoplamiento y la automatización del despliegue. Este contraste entre la definición de microservicio propuesta por Martín Fowler y flexibilidad según la ISO genera una problemática ya que no se refleja lo que la norma describe como flexibilidad y por ende no refleja adecuadamente los desafíos específicos de MSA, como la interoperabilidad entre servicios, la gestión de dependencias o la capacidad de escalar componentes de forma autónoma. A partir de dichos problemas, se seleccionó como propiedad de interés la flexibilidad, la cual en la ISO/IEC 25010 se compone de los subatributos de adaptabilidad, escalabilidad, instalabilidad y reemplazabilidad. No obstante, para su aplicación en entornos de microservicios fue necesario sintetizar ambas visiones, ya que, para este trabajo, la flexibilidad para MSA no es un atributo monolítico, sino una cualidad emergente que resulta del equilibrio de tres pilares arquitectónicos, como lo son: la capacidad de adaptarse a nuevas reglas de negocio (adaptabilidad), la capacidad de crecer bajo demanda (escalabilidad) y la capacidad de moverse a nuevos entornos (portabilidad).

Para el desarrollo del esquema se analizaron diversas normas que pudieran ofrecer un mayor entendimiento del concepto de flexibilidad, donde la fuente principal es la ISO/IEC 25010; sin embargo, se realizó una adaptación de los subatributos de instalabilidad y reemplazabilidad provenientes de dicha norma. En este proceso se incorporó el concepto de portabilidad, retomado de la ISO/IEC 9126, ubicándolo en un nivel jerárquico superior a los mencionados subatributos. Esta inclusión responde a la necesidad de reflejar de manera más amplia la capacidad del sistema para operar en distintos entornos tecnológicos, característica esencial en los sistemas basados en microservicios, donde los componentes deben poder desplegarse, migrarse o reemplazarse de forma ágil entre plataformas, contenedores o infraestructuras heterogéneas [10, 16, 23]. De este modo, la portabilidad actúa como un elemento integrador que abarca tanto la instalabilidad como la reemplazabilidad, ofreciendo una visión más coherente y representativa del comportamiento flexible que se espera en arquitecturas distribuidas y dinámicas.

El resultado del esquema de calidad propuesto para la flexibilidad se muestra en la *Figura 1*, donde la flexibilidad se organiza en tres características principales con sus respectivos subatributos y la ponderación para estos. Estas características constituyen la base del esquema de evaluación propuesto, el cual utiliza métricas seleccionadas asociadas a los microservicios, la claridad de sus definiciones y la viabilidad de su cálculo.

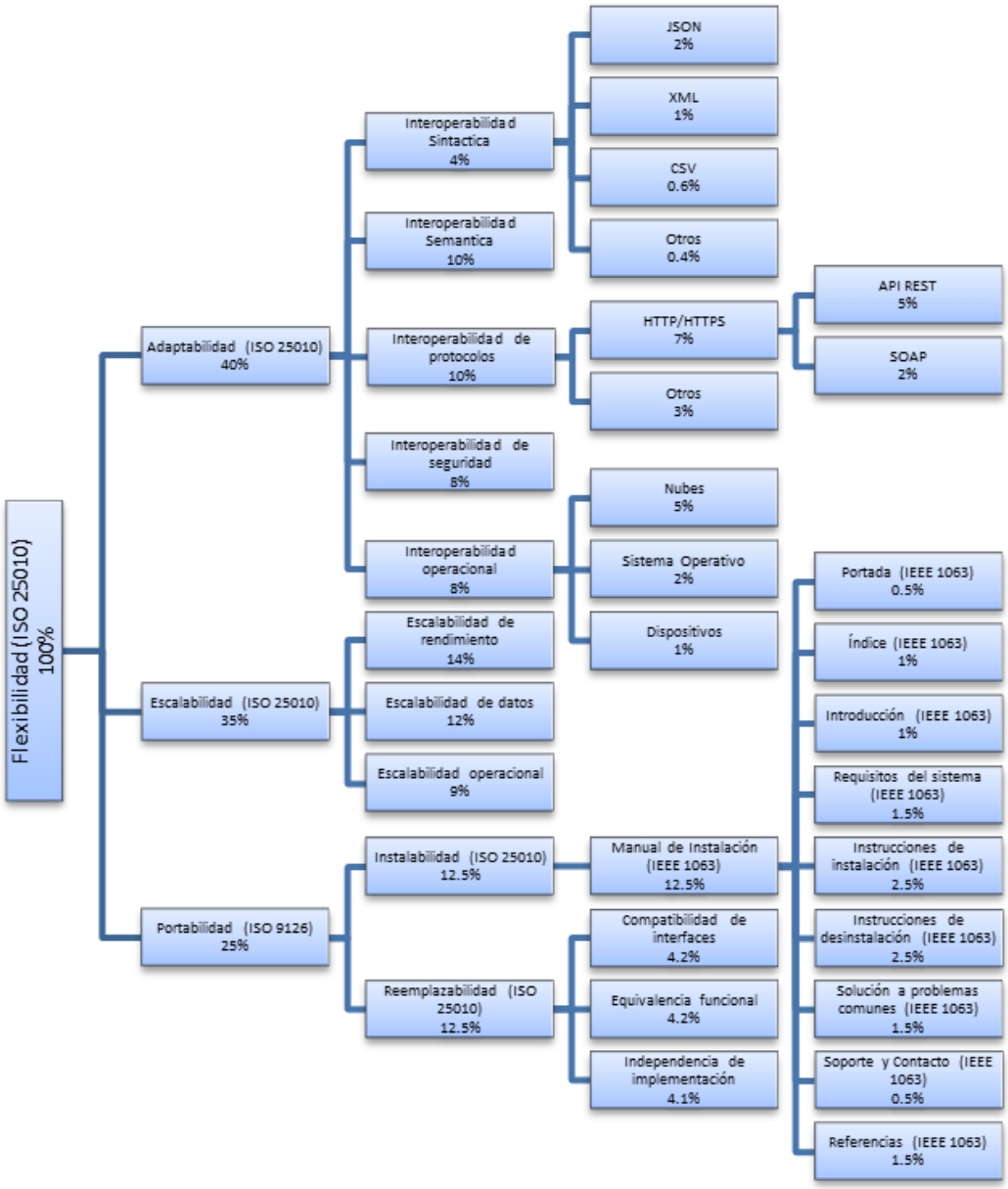

Figura 1 Esquema de calidad para flexibilidad

Es importante mencionar que 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 la relevancia observada en estudios previos y en la influencia práctica que cada subatributo ejerce sobre la flexibilidad [2, 3, 16, 24], así como el comportamiento del sistema [14, 24, 37, 40]. Esta asignación de ponderaciones permitió establecer una valoración cuantitativa/cualitativa orientada a reflejar la importancia relativa de cada elemento dentro del esquema propuesto.

3.1 Adaptabilidad [2].

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 [24, 40].

Por lo tanto, la "Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada" [2] no es exclusivamente un asunto de compatibilidad, sino un requisito para que la adaptación sea posible. Es por ello 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.

3.2 Escalabilidad [2].

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 [16]. 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 [43], 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 verdadera 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 [8, 15, 17].

Por esta razón, nuestro esquema 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.

3.3 Portabilidad [3].

La portabilidad se pondera con 25% ya que una de las características clave de MSA es la capacidad de construir una vez y ejecutar en cualquier lugar. Un sistema es inflexible si está sujeto a un proveedor de nube especifico, 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" [3], razón por la cual, se considera que los atributos de instalabilidad y reemplazabilidad quedan en un nivel de jerarquía inferior dentro del esquema, como se presenta a continuación:

Instalabilidad [2]: Se asigna el peso de 12.5% ya que la facilidad de instalación incide directamente en la aceptación de un sistema**.**

Manual de instalación [7]: 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: Con un peso de 0.5% ya que, aunque el estándar de la IEEE 1063 lo contempla, no se considera de vital importancia

Índice: Un peso de 1% ya que el tener un índice aporta a que el usuario pueda encontrar más fácilmente lo que busca

Introducción: Igual peso que el índice ya que con él, el usuario podrá entender el contenido del documento.

Requisitos del sistema: 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: 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: 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: 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: 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: 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 [2]: 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.

3.4 Ventajas del esquema propuesto.

A continuación, se listan ventajas observadas en el esquema de calidad propuesto.

El esquema propuesto cuenta con un sólido respaldo normativo, al basarse en los estándares internacionales, ampliamente utilizados como marco de referencia para la evaluación de la calidad del software. [2-7, 25]

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 [12, 26], por lo que el esquema 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 [27, 28], por lo que el esquema 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 [30, 31]. Por ello, el esquema 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 esquema a las prioridades del usuario [29].

4. Experimentación y resultados

El proceso experimental tuvo como propósito validar la aplicabilidad del esquema propuesto en repositorios de acceso libre de microservicios desarrollados en Java [20-22], ya que esta tecnología es ampliamente utilizada tanto en entornos empresariales como académicos, y cuenta con una estrecha compatibilidad con las MSA. De acuerdo con el State of Spring Survey 2024 [32], los microservicios constituyen uno de los principales focos dentro del ecosistema Spring, reflejando su relevancia en el desarrollo moderno de software distribuido. Asimismo, el estudio de Sõnajalg [33] reporta que el 86 % de los desarrolladores Java encuestados utilizan Spring Boot para la creación de microservicios, consolidándolo como el framework predominante en este ámbito. No obstante, aunque Spring Boot será considerado como el principal entorno de referencia dentro del análisis, no se establecerá como un criterio de exclusión, permitiendo así la inclusión de microservicios desarrollados con otros frameworks o librerías compatibles con Java. Finalmente, restringir el análisis a una misma plataforma tecnológica permitió garantizar la homogeneidad del entorno de evaluación, reduciendo posibles variaciones derivadas de diferencias en lenguajes o frameworks y facilitando la comparación de los resultados obtenidos.

Para el proceso de experimentación se seleccionaron tres repositorios públicos (Zull [20], ACME Air Microservice [21] y E-Commerce [22]), que en conjunto incluyen catorce microservicios con distintos niveles de complejidad, documentación y modularidad, los cuales se presentan a detalle dentro de la *Tabla 1*.

El análisis se centró en observar la presencia de prácticas y artefactos que evidencien la existencia de los subatributos definidos (adaptabilidad, escalabilidad y portabilidad). La revisión se realizó de manera iterativa y manual, complementada con herramientas de apoyo como DeepWiki [44], con el fin de localizar evidencias de interoperabilidad, rendimiento, portabilidad o documentación técnica.

Nombre	Autor	No. De Microservicios	Función
ZULL [20]	Lindsey Reynolds	4	Proporciona capacidades de enrutamiento dinámico, monitorización, resistencia, seguridad y mucho más.
Acme AIR-Microservice [21]	Joe McClure	4	Un controlador de carga de trabajo para la aplicación de muestras de Air Acme.
E-Commerce [22]	Rainbow Forest	6	Implementación de un Microservicio REST en un ECommerce con Spring boot, Cloud y múltiples módulos.

Tabla 1. Repositorios seleccionados

Con el motivo de garantizar validez de los resultados entre los proyectos analizados, se establecieron criterios de exclusión específicos. En primer lugar, se descartaron los repositorios cuyo código fuente no estuviera implementado íntegramente en Java, dado que esta tecnología representa uno de los entornos más maduros y ampliamente adoptados para la implementación de arquitecturas de microservicios [32, 33].

Aunque Spring Boot fue el framework predominante en los proyectos seleccionados, no se limitó la inclusión exclusivamente a este; se admitieron otras herramientas o librerías siempre que mantuvieran compatibilidad con el ecosistema de Java y con principios arquitectónicos equivalentes. Esta decisión se alinea con lo señalado en [12, 27] donde se subraya la necesidad de conservar consistencia tecnológica al evaluar sistemas distribuidos, sin restringir la diversidad de implementaciones.

Así mismo, se excluyeron los repositorios que no evidenciaban una estructura claramente orientada a microservicios, por ejemplo, aplicaciones monolíticas o hibridas, siguiendo con las recomendaciones propuestas en [34] ya que la mezcla de estilos arquitectónicos impacta directamente en la solidez de cualquier evaluación de calidad.

Por último, se excluyeron los repositorios con un número reducido de componentes o microservicios (menor a dos), ya que no permitían una evaluación representativa de los atributos de calidad definidos en el esquema propuesto.

Para realizar las pruebas se realizó una revisión directa al código fuente, con el fin de identificar evidencias que permitieran aplicar el esquema de calidad propuesto. Este procedimiento no se basó en un protocolo formal documentado, la revisión se realizó de forma iterativa sobre los repositorios seleccionados, verificando la presencia de prácticas o patrones asociados a los indicadores definidos en el esquema. Esta aproximación permitió mantener coherencia metodológica, priorizando la interpretación contextual del código y la arquitectura frente a la simple medición automática.

El servició de DeepWiki [44] se utilizó como recurso complementario con el propósito de agilizar la localización de segmentos de código, documentación técnica y descripciones arquitectónicas relevantes. Dicha herramienta, basada en inteligencia artificial y entrenada sobre proyectos disponibles en GitHub, facilitó la obtención de información preliminar sobre la estructura y funcionamiento de los microservicios. No obstante, toda la información recuperada a través de DeepWiki [44] fue contrastada manualmente con el contenido real de los repositorios, asegurando la verificación y validez de los hallazgos.

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, a continuación, se presenta un análisis individual para cada repositorio.

4.1 Repositorio Zull

El repositorio Zull desarrollado por Netflix [20], 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 eficazmente como puerta de entrada a todas las solicitudes de dispositivos y sitios web al backend del ecosistema de aplicaciones de streamming de Netflix.

En la *Tabla 2* se presentan los resultados obtenidos por cada uno de los microservicios que componen el repositorio Zull [20], así mismo se presentan las calificaciones obtenidas por cada una de las tres características principales que según el esquema propuesto, caracterizan a la flexibilidad.

REPOSITORIO	**MICROSERVICIO**	**ADAPTABILIDAD**	**ESCALABILIDAD**	**PORTABILIDAD**	**FLEXIBILIDAD**
ZULL [20]	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%

Tabla 2. Resultados del repositorio Zull [20] por microservicio

Los resultados obtenidos para el repositorio Zull evidencian fortalezas en adaptabilidad y escalabilidad (valores promedio ≈ 20.25% y 17% respectivamente), pero estas cifras están significativamente por debajo de los objetivos fijados por el esquema (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 ≈ 50.6%), el 48.57% en escalabilidad (17/35 = 48.57 %) y solo el 40 % en portabilidad (10/25 ≈ 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 esquema. 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. [12]

y Li et al. [10] 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 [26], 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.

4.2 Repositorio ACME Air Microservice

El proyecto [21] 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 3* evidencian que, aunque existen elementos de diseño modular (adaptabilidad ≈ 22%) y una escalabilidad moderada (≈ 9.75%), estos valores están consistentemente por debajo de los objetivos del esquema (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).

REPOSITORIO	**MICROSERVICIO**	**ADAPTABILIDAD**	**ESCALABILIDAD**	**PORTABILIDAD**	**FLEXIBILIDAD**
ACME AIR MICROSERVICE [21]	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%

Tabla 3. Resultados del repositorio ACME Air Microservice [21] por microservicio

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 se erige como el principal obstáculo para alcanzar un nivel de flexibilidad óptimo. Esto refuerza la necesidad de considerar la portabilidad como un factor determinante dentro del esquema de evaluación y de implementar estrategias orientadas a la automatización, documentación y despliegue multiplataforma.

4.3 Repositorio E-Commerce

Este repositorio [22] 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. Dentro de la *Tabla 4* se presentan los resultados obtenidos tras aplicar el esquema de evaluación a cada microservicio.

REPOSITORIO	**MICROSERVICIO**	**ADAPTABILIDAD**	**ESCALABILIDAD**	**PORTABILIDAD**	**FLEXIBILIDAD**
E-COMMERCE [22]	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%

Tabla 4. Resultados del repositorio E-Commerce [22] por microservicio

Los resultados pertenecientes al repositorio E-Commerce [22] 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 esquema: 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 denotar, existe una diferencia considerable entre el resultado obtenido tras la evaluación y la ponderación propuesta en el esquema, 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.

En la *Tabla 5* se presentan los resultados de manera específica en cuanto al microservicio *Product-Recommendation-Service* tras aplicar el esquema de calidad propuesto para evaluar la flexibilidad.

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	Capacidad Empresarial		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		

Tabla 5. Resultados de flexibilidad del microservicio product-recommendation-service del repositorio E-Commerce [22]

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 dentro del modelo de evaluación propuesto.

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 esquema (40%, 35% y 25%, respectivamente). Esta situación prueba 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 2* 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.

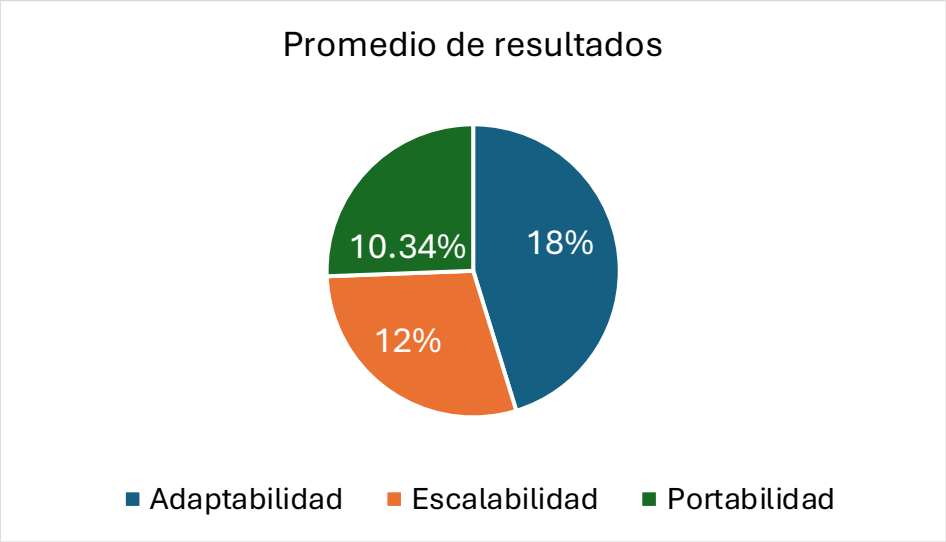

Figura 2 Promedio de resultados por subatributo

Por tanto, los resultados del análisis de los tres repositorios refuerzan la validez del esquema 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.

5. Discusión

Los resultados obtenidos reflejan un patrón consistente con lo reportado en estudios previos, tal como lo señalaron [10, 12], la adopción de microservicios no garantiza por sí misma la consecución de altos niveles de calidad, ya que los problemas de acoplamiento, seguridad y documentación persisten incluso en sistemas bien diseñados. En el contexto del presente estudio, estas limitaciones se manifestaron especialmente en los subatributos de instalabilidad y reemplazabilidad, confirmado que la falta de manuales técnicos y mecanismos de despliegue automatizados reduce la flexibilidad operativa.

Asimismo, los hallazgos coinciden con lo observado en [26], quienes destacan que los denominados *microservice bad smells* como la duplicación de código, el acoplamiento implícito y la carencia de documentación afectan de forma directa la mantenibilidad y, en consecuencia, la flexibilidad del sistema. De modo similar, los trabajos de Vera-Rivera [15, 16] sobre granularidad evidencian que una incorrecta segmentación de servicios puede obstaculizar la adaptabilidad, al aumentar la complejidad de dependencias.

Estos resultados evidencian que la flexibilidad no debería considerarse un atributo garantizado por la MSA, sino como un resultado derivado del equilibrio entre sus subatributos. La interacción entre adaptabilidad, escalabilidad y portabilidad se configura

como un triángulo de calidad donde el debilitamiento de uno de ellos compromete a los demás.

Finalmente, el esquema propuesto demuestra su utilidad para identificar áreas de mejora y orientar estrategias de refactorización o rediseño ya que el análisis de los repositorios evidenció carencias recurrentes en aspectos clave de la flexibilidad, particularmente en lo referente a la documentación, manuales de instalación, gestión de dependencias y mecanismos de seguridad. La ausencia de dichos aspectos afecta directamente a atributos esenciales de la flexibilidad como la portabilidad, la cual a su vez se subdivide en instalabilidad y reemplazabilidad, comprometiendo su adaptabilidad en entornos reales.

De este modo, se demuestra que el simple hecho de adoptar una arquitectura basada en microservicios no garantiza, por sí sola, la flexibilidad. Para alcanzar este atributo se requiere un análisis más profundo que considere no solo la existencia de los subatributos, sino también la manera en que estos se relacionan entre sí. Esta relación resulta crítica, pues la flexibilidad emerge precisamente de la interacción y equilibrio entre sus componentes, y no de manera aislada. Sin embargo, lograr esta articulación no es tarea sencilla, ya que demanda un proceso de evaluación exhaustivo que considere los requerimientos específicos del sistema y la complejidad propia de cada entorno.

6. Conclusiones

La literatura evidencia avances significativos en el entendimiento estructural de las MSA, pero una limitada madurez en la medición empírica de sus atributos de calidad. Las métricas tradicionales resultan insuficientes, los enfoques centrados en granularidad y acoplamiento se mantienen parciales, lo cual justifica la necesidad de un modelo integrador, como el propuesto en este trabajo, que combine formalismo normativo, operacionalización métrica y aplicación empírica en contextos reales de desarrollo de microservicios.

El esquema de evaluación propuesto constituye una contribución relevante para el estudio de la flexibilidad en arquitecturas de microservicios, al integrar criterios normativos (ISO/IEC 25010 [2], ISO/IEC 9126 [3], ISO/IEC 2382 [4]) 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 automatización 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 más 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 esquema 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) [17], que ayudan a analizar aspectos de semántica, refactorización, calidad y arquitectura de microservicios, estas no ofrecen un sustento basado en normas y estándares de calidad como el que se presenta en este esquema, ni se enfocan de manera específica en el atributo de flexibilidad, el cual con frecuencia se asume erróneamente como garantizado por el simple uso de la arquitectura de microservicios.

Como trabajo futuro, se propone extender la validación del esquema mediante la aplicación de métodos formales de ponderación, incorporar métricas cuantitativas automáticas y evaluar su efectividad en entornos productivos o industriales. Con ello, se espera avanzar hacia la construcción de un marco de referencia integral para la medición de la calidad de microservicios, capaz de contribuir a la mejora continua del software y a la consolidación de estándares específicos para este tipo de arquitectura.

Así mismo se propone como trabajo futuro realizar una herramienta que pueda automatizar la evaluación de sistemas de microservicios, partiendo del esquema propuesto como base de evaluación, ya que el análisis realizado en este trabajo ha sido llevado a cabo de manera manual lo cual implica emplear una cantidad sustancial de tiempo con el que muchas veces, las empresas principalmente no cuentan.

7. Referencias

[1] Fowler, J. L. (2014). Microservices. Obtenido de MartinFawler.com: https://martinfowler.com/articles/microservices.html

[2] ISO/IEC 25000. (n.d.). ISO/IEC 25010: Características de calidad del producto. Retrieved October 18, 2024, from https://iso25000.com.

[3] International Organization for Standardization (ISO). (2001). *ISO/IEC 9126: Software engineering — Product quality* (Partes 1-4). ISO.

[4] International Organization for Standardization (ISO). (2015). *ISO/IEC 2382: Information technology — Vocabulary* (Partes 1-36). ISO. (Greg Boss et al. 2007)

[5] ISO/IEC 11179-1:2015 - Information technology — Metadata registries (MDR) — Part 1: Framework. n.d. https://www.iso.org/standard/61932.html.

[6] ISO/IEC 27001:2022 - Information security management systems. n.d. https://www.iso.org/standard/27001. (Tanenbaum and Bos 2014)

[7] IEEE SA - IEEE 1063-2001. n.d. Retrieved August 21, 2025. https://standards.ieee.org/ieee/1063/1554/.

[8] Al-Debagy, O. and Martinek, P. (2020a). A metrics framework for evaluating microservices architecture designs. Journal of Web Engineering, 19, 341–370. https://doi.org/10.13,052/jwe1540-9589.19341.

[9] Berander, P. et al. (2005). Software quality attributes and trade-offs.

[10] Li, S. et al. (2021). Understanding and addressing quality attributes of microservices architecture: A systematic literature review. Information and Software Technology, 131, 106449. https://doi.org/10.1016/j.infsof.2020.106449.

[11] Lima, S. et al. (2021). Improving observability in event sourcing systems. Journal of Systems and Software, 181, 111015. https://doi.org/10.1016/j.jss.2021.111015.

[12] Dragoni, N. et al. (2017). Microservices: Yesterday, today, and tomorrow

[13] Newman, Sam. 2015. "Building Microservices - Chapter 1, 4 and 11." Building Microservices 102.

[14] Raj, Vinay & Sadam, Ravichandra. (2021). Evaluation of SOA-Based Web Services and Microservices Architecture Using Complexity Metrics. SN Computer Science. 2. 10.1007/s42979-021-00767-6.

[15] Vera-Rivera, F. H. et al. (2021). Defining and measuring microservice granularity. PeerJ Computer Science, 7, e695. https://doi.org/10.7717/peerj-cs.695.

[16] Vera-Rivera, F. H. (2021). Modelo inteligente de especificación de la granularidad de microservicios (unpublished doctoral thesis, Univ. del Valle).

[17] Engel, T. et al. (2018). Evaluation microservice architectures: A metric-based approach. CAiSE 2018. https://doi.org/10.1007/978-3-319-92901-9_8.

[18] Li, Y. et al. (2020). Granularity decision in microservice splitting. Multi-Goal Decision Making, 1057902. https://doi.org/10.1155/2020/1057902.

[19] Perepletchikov, M., Ryan, C. and Frampton, K. (2007). Cohesion metrics for predicting maintainability of SOA. QSIC 2007, 328–335. https://doi.org/10.1109/QSIC.2007.4385516.

[20] Netflix. (2025). *Zuul* [Repositorio de código]. GitHub. https://github.com/Netflix/zuul

[21] anitagoel. (2022). AcmeAir_Microservice [Repositorio de código]. GitHub. https://github.com/anitagoel/AcmeAir_Microservice
[22] RainbowForest. (2025). E-commerce Microservices [Repositorio de código]. GitHub. https://github.com/RainbowForest/e-commerce-microservices/
[23] Booklet, D., Al-Debagy, O., & Martinek, P. (2021). Microservices Identification Methods and Quality Metrics.
[24] Ulander, D. (2017). Software Architectural Metrics for the Scania Internet of Things Platform From a Microservice Perspective. http://www.teknat.uu.se/student
[25] Kitchenham, B., & Lawrence Pfleeger, S. (1996). Software Quality: The Elusive Target. https://doi.org/10.1109/52.476281
[26] Taibi, D., & Lenarduzzi, V. (2018). On the Definition of Microservice Bad Smells. https://doi.org/10.1109/MS.2018.2141031
[27] Soldani, J., Tamburri, D. A., & van den Heuvel, W. J. (2018). The pains and gains of microservices: A Systematic grey literature review. *Journal of Systems and Software*, *146*, 215–232. https://doi.org/10.1016/j.jss.2018.09.082
[28] Newman, S. (2015). Building Microservices. http://safaribooksonline.com
[29] Al-Qutaish, R. E., & Ain, A. (2010). uality Models in Software Engineering Literature: An Analytical and Comparative Study. In *Journal of American Science* (Vol. 6, Issue 3). http://www.americanscience.orgeditor@americanscience.org166
[30] Bucchiarone, A., Kessler, B., Dragoni, N., Dustdar, S., Wien, T. U., Larsen, S. T., & Bank, D. (2018). From Monolithic to Microservices An Experience Report from the Banking Domain. https://doi.org/10.1109/MS.2018.2141026
[31] Nadareishvili, I., Miitra, R., McLarty, M., & Amundsen, M. (2016). Microservice Architecture: Aligning Principles, Practices and Culture. 146.
[32] Minella, M. (2024, June 3). State of Spring Survey 2024 Results. https://spring.io/blog/2024/06/03/state-of-spring-survey-2024-results?utm_source=chatgpt.com
[33] Sõnajalg, S. (2019, August 8). Microservices in Java: Survey Highlights | *JRebel*. https://www.jrebel.com/blog/microservices-in-java-survey?utm_source=chatgpt.com
[34] Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. https://doi.org/10.1109/MCC.2017.4250931
[35] Pulnil, S., & Senivongse, T. (2022). A Microservices Quality Model Based on Microservices Anti-patterns. 2022 19th International Joint Conference on Computer Science and Software Engineering, JCSSE 2022. https://doi.org/10.1109/JCSSE54890.2022.9836297
[36] Yu, J., Ge, J., & Sun, J. (2021). Research on Quality Model and Measurement for Microservices.
[37] Apolinário, D. R. F., & de França, B. B. N. (2021). A method for monitoring the coupling evolution of microservice-based architectures. Journal of the Brazilian Computer Society, 27(1). https://doi.org/10.1186/s13173-021-00120-y
[38] Bogner, J., Wagner, S., & Zimmermann, A. (2017). Automatically measuring the maintainability of service- and microservice-based systems - a literature review. ACM International Conference Proceeding Series, Part F131936, 107–115. https://doi.org/10.1145/3143434.3143443
[39] Vinay Babu, D., & Darsi, M. P. (2013). A Survey on Service Oriented Architecture and Metrics to Measure Coupling.3
[40] Driessen Supervisors, F., Ferreira Pires, L., Luiz Rebelo Moreira, J., Sperotto, A., van den Bosch, S., & Verhoeven June, P. (2023). A quantitative assessment method for microservices granularity to improve maintainability.
[41] Panichella, S., Rahman, M. I., & Taibi, D. (2021). Structural Coupling for Microservices. https://doi.org/10.5220/0010481902800287
[42] Chidamber, S. R., & Kemerer, C. F. (1994). A Metrics Suite for Object Oriented Design. In *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING* (Vol. 20, Issue 6).
[43] Chen, Y. H., Chen, N. J., Xu, W. X., Lian, L. M., & Tu, H. (2021). MFRL-CA: Microservice Fault Root Cause Location based on Correlation Analysis. Proceedings - 2021 8th International Conference on Dependable Systems and Their Applications, DSA 2021, 90–101. https://doi.org/10.1109/DSA52907.2021.00018
[44] DeepWiki | AI documentation you can talk to, for every repo. (n.d.). Retrieved November 3, 2025, from https://deepwiki.com/

NOTAS BIOGRAFICAS

Said Misael Venoso Lara, es Ingeniero en Sistemas Computacionales, egresado del Tecnológico Nacional de México campus Zacatepec, actualmente es estudiante de maestría en el Tecnológico Nacional de México campus CENIDET, donde desarrolla su tesis relacionada a las métricas aplicables para microservicios. Tiene como principales gustos temas como calidad de software, métricas, principios y patrones de diseño de software, arquitecturas basadas en microservicios, arquitecturas basadas en servicios web y modelos de calidad.

Juan Carlos Rojas Pérez, doctor en Ciencias en Ciencias de la Computación egresado del TecNM/CENIDET. Actualmente profesor en el área de ingeniería de software del Departamento de Ciencias Computacionales. Áreas de interés: arquitecturas orientadas a servicios, servicios web, microservicios, lenguajes web, bases de datos, Big data, temas generales de ingeniería de software. Miembro del SNII nivel "c", perfil deseable. Evaluador de artículos para las revistas IEEE Latin America Transactions, Dyna, JCyTA, CYTCA, CIM. Fue miembro del Consejo Técnico del Examen General para el Egreso de la Licenciatura (EGEL Plus) en Ingeniería Computacional EGEL+D-ICOMPU (CENEVAL). Trabajo alrededor de 15 años en empresas de desarrollo de software:

Softtek monterrey: desarrollo de sistemas de software para General Electric Power Systems, General Electric Nuclear.

E-siglo: desarrollo de sistemas de software para Inbursa, Gobierno de México, IMSS.

Instituto de Electricidad y Energías Limpias: desarrollo de sistemas de software para el Centro Nacional de Control de energía (CENACE), Subdirección de Energéticos (SDE).

Olivia Graciela Fragoso Diaz. es actualmente profesora de doctorado de tiempo completo en ciencias de la computación en el Centro Nacional de Investigación y Desarrollo Tecnológico (TecNM/Cenidet), México. También lidera proyectos de ingeniería de software que buscan elementos de calidad utilizando tecnologías abiertas para generar mejoras en los objetivos de aprendizaje electrónico en universidades, instituciones educativas y entornos laborales. Sus intereses de investigación incluyen arquitecturas de microservicios, arquitecturas orientadas a servicios, métricas y modelos de calidad.