UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Comparación del rendimiento de las arquitecturas monolíticas y microservicios en los sistemas web

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTOR:

Toledo Azorza, Miguel Angel Jesus (orcid.org/0000-0002-4888-6571)

ASESOR:

Dr. Alfaro Paredes, Emigdio Antonio (orcid.org/0000-0002-0309-9195)

LÍNEA DE INVESTIGACIÓN:

Infraestructura de Servicio de Redes y Comunicaciones

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo económico, empleo y emprendimiento

LIMA - PERÚ

2017

Dedicatoria

A mis padres por ser los pilares fundamentales, en toda mi educación, tanto académica, como la vida profesional, por su incondicional apoyo en todos mis proyectos, sus consejos para cada etapa de mi vida, ahora presentes en esta nueva etapa de mi vida.

Agradecimiento

Doy Gracias a nuestro señor de los milagros por todas las bendiciones recibidas en mi familia, a mi nuevo hogar con mi esposa, por alcázar mis metas profesionales y personales. Agradecer a mis colegas de trabajo y de estudios por apoyarme incondicionalmente durante toda la carrera.

Índice de contenidos

Pág.

Índice de tablas

Índice de figuras

Índice de abreviaturas

Sigla	Significado	Pág.
CPU	Central Processing Unit (Unidad Central de Procesamiento)	15
API	Application Program Interface (Interfaz de programación de aplicaciones)	8
HTTP	Hyper Text Transfer Protocol (Protocolo de Transferencia de Hipertexto)	14

Resumen

El objetivo de la investigación fue determinar si las arquitecturas mejoran los sistemas web en comparación con microservicios la arquitectura monolítica. En relación a la metodología, el tipo de investigación fue aplicada, el enfoque del estudio fue cuantitativo, el diseño fue preexperimental y el tipo de diseño fue descriptivo. La muestra estuvo conformada por 30 procesos similares de los sistemas, para la definición de la métrica.

La arquitectura de microservicios mejoró la eficiencia de los sistemas web en comparación de arquitecturas monolíticas en 10.63%; además, la arquitectura de microservicios mejoró la fiabilidad de los sistemas web en 27.87% en comparación con las arquitecturas monolíticas. En conclusión, la implementación de una arquitectura de microservicios mejoró la eficiencia y la fiabilidad en los sistemas web en comparación a las arquitecturas monolíticas. Se recomendó investigar los costos de implementación de las arquitecturas de microservicios y evaluar la viabilidad de su implementación, así como la implementación de librerías para su aplicación dentro de los sistemas de información.

Palabras clave: Sistema web, Arquitectura de microservicios, Arquitectura monolítica.

Abstract

The purpose of the research was to determine if the architectures improve web systems compared to microservices and monolithic architecture. Regarding the methodology, the type of research was applied, the focus of the study was quantitative, the design was pre-experimental and the type of design was descriptive. The sample consisted of 30 similar processes of the systems, for the definition of the metrics.

The microservices architecture improved the efficiency of web systems compared to monolithic architectures by 10.63%; furthermore, the microservices architecture improved the reliability of web systems by 27.87% compared to monolithic architectures. In conclusion, the implementation of a microservices architecture improved the efficiency and reliability of web systems compared to monolithic architectures. It was recommended to investigate the implementation costs of microservices architectures and evaluate the feasibility of their implementation, as well as the implementation of libraries for their application within information systems.

Keywords: Web system, Microservices architecture, Monolithic architecture.

I. INTRODUCCIÓN

Los avances en la tecnología han cambiado el ritmo de vida de los seres humanos y estos no se limitan a nuevas formas de compartir, procesar y adquirir conocimientos a través de las redes sociales [1]. Para ello, antes a través de la arquitectura monolítica, que consistía en una estructura de software en la que todas las funciones de una aplicación se combinaban en un solo programa [2]. Y en términos de rendimiento, las arquitecturas monolíticas eventualmente podían sufrir cuellos de botella y tener dificultades para escalar horizontalmente, lo que limitó su capacidad para manejar grandes volúmenes de tráfico de información y de usuarios.

En este sentido, con el paso del tiempo y debido a la evolución de los sistemas, y dado que la idea de esta tecnología es acercar más a las personas y minimizar un número considerable de problemas derivados del hecho de que varios programas de software se desarrollan para ser utilizados e implementados en entornos web [2]. Las opciones de solución deficientes (es decir, errores de software, configuración modular inconsistente, coherencia y acoplamiento, y falta de niveles de seguridad) pueden conducir a mayores costos de recursos a largo plazo. Entre estas, la novedad del diseño de microservicios ha llevado al desarrollo de ideas, métodos y tecnologías nuevas (y mejoradas) tanto en sistemas distribuidos como en ingeniería de software [3].

Con la creación de la arquitectura de microservicios, se pensó y desarrollo una estructura de software en la que una aplicación se divide en servicios más pequeños y autónomos que trabajan juntos para lograr el objetivo final [4]. Bandeira et al. (2020) señalan que los microservicios son un nuevo paradigma arquitectónico que se está expandiendo rápidamente. Su objetivo es crear sistemas de software altamente escalables basados en componentes desplegables y evolutivos de forma autónoma [3]. Como, por ejemplo, Netflix Eureka es el tema del 13,68% de los artículos técnicos sobre microservicios en StackOverflow (SO) [3]. Otras empresas, como Amazon y LinkedIn, también introdujeron cambios en su estructura de software, para desarrollar e implementar aplicaciones en la nube, así lo mencionan Villamizar et al. (2017), Además de mejorar la agilidad, el desarrollo independiente y la escalabilidad, los microservicios afectan los costos de infraestructura, lo cual es un tema crucial para las empresas que adoptan este enfoque y utilizan servicios

diseñados específicamente para implementar y escalar microservicios, como el AWS Lambda, que reduce los costos de infraestructura en un 70% o más [5].

En este sentido, se ha observado que en mercado asegurador actualmente se encuentra en una fase de expansión, y sus indicadores así lo demuestran [6]. En el ámbito global, luego de la crisis sanitaria, el sector asegurador ha experimentado un crecimiento significativo. Solo en Europa se ha invertido un 7,2% más que en 2020, instrumentos financieros, a la par de la formulación de marcos regulatorios que amplíen la protección de los clientes. Además, ha experimentado una transformación impulsada por la tecnología, dado que las compañías de seguros han adoptado nuevas herramientas y aplicaciones para mejorar la experiencia del cliente, agilizar los procesos de suscripción y reclamaciones, y analizar grandes cantidades de datos para evaluar riesgos con mayor precisión [7].

En Latinoamérica y el Caribe este sector mostró un crecimiento económico del 6,5% en 2021, frente a un drástico descenso del -6,9% en 2020. En 2021, las primas totales en este continente alcanzaron los 149.794 millones de dólares, de los cuales el 57,5% correspondieron a seguros no de vida, el resto a seguros no de vida y el 42,5% restante, a seguros de vida. Las primas totales en región aumentaron un 11,5% y un -11,9% (frente al fuerte descenso del -11,9% del año anterior), respectivamente. Esto representa un crecimiento significativo del mercado asegurador, aunque todavía no ha alcanzado el nivel de primas anterior a la pandemia [8].

Asimismo, en el ámbito nacional, pese al descenso registrado en 2020, las primas netas del sector aumentaron por término medio alrededor de un 24,9% en el segundo semestre de 2021. Hecho que se tradujo en una tendencia a la expansión en 2021, en la industria aseguradora peruana. Equivalente a un incremento potencial de esta de 56,023 millones de soles, lo que sugiere que este crecimiento fue cuatro veces mayor que para fines del 2020 [8]. Por ello las aseguradoras están valorando la cobertura que ofrece su línea de productos, y los gobiernos están promulgando nuevas leyes, con énfasis en cinco aspectos clave: el tamaño y contribución económica, la diversidad de productos, las tendencias tecnológicas, el desarrollo de mercados emergentes y, los riesgos y desafíos.

En este orden de ideas, Bolo [9], el principal reto al que se enfrentan las empresas hoy en día es integrar sus procesos de forma "transversal" en varias aplicaciones. Es obvio que las arquitecturas informáticas anteriores eran incapaces de responder plenamente a esta necesidad. Por su parte, Laudon & y Laudon [10] mencionan que el requisito para que un sistema pueda ser escalable, primero es que se diseñe correctamente dado que de no ser así representará un costo alto para sus desarrolladores. Este panorama motivó la investigación que permitió comparar el rendimiento de las arquitecturas monolíticas con la arquitectura de microservicios en los sistemas web de empresas aseguradoras.

A continuación, se planteó el problema general: ¿Cómo las arquitecturas de microservicios mejoran los sistemas web en comparación a las arquitecturas monolíticas? Lo que genera los siguientes interrogantes:

PE1: ¿Cómo la arquitectura de microservicios mejora la eficiencia en los sistemas web en comparación con la arquitectura de monolítica?

PE2: ¿Cómo la arquitectura de microservicios mejora la fiabilidad en los sistemas web en comparación con la arquitectura de monolítica?

De estos interrogantes se desprende el objetivo general: Determinar si las arquitecturas mejoran los sistemas web en comparación con microservicios la arquitectura monolítica. Así como los específicos:

OE1: Determinar si la arquitectura microservicios mejora la eficiencia en los sistemas web en comparación con la arquitectura monolítica.

OE2: Determinar si la arquitectura microservicios mejora la fiabilidad en los sistemas web en comparación con la arquitectura monolítica.

En cuanto a la justificación teórica, la comparación de la funcionalidad de estos modelos producirá conocimientos útiles para la el diseño de los sistemas y para el sector específico de los seguros. Las arquitecturas monolíticas, tradicionalmente utilizadas para la generación de software con máquinas virtuales, han sido una fórmula exitosa y eficiente para proyectos pequeños y grandes desde la llegada de los sistemas de software. Pero, es bien sabido que el rendimiento de una aplicación monolítica sufre en el momento que los datos a procesar aumentan o superan un cierto nivel [2].

Acerca de la justificación metodológica se debe indicar que la contrastación de los elementos presentes en una arquitectura monolítica con respecto a los de una arquitectura de microservicios aporta elementos técnicos y el diseño que permiten implementar los cambios en los servicios de las empresas aseguradoras. Este tipo de sistemas prevé la migración, permitirá la compatibilidad para comunicarse entre sí, es escalable, resistente, seguro y ofrece una implementación sencilla para las nuevas áreas o proyectos que se construyan [5].

En cuanto a la justificación tecnológica se debe indicar que los beneficios de la adopción de sistemas de operaciones digitales alcanzan a la mayoría de las personas de una u otra manera. En el caso de la arquitectura monolítica, esta ha empezado a resultar inadecuada para satisfacer los requisitos de los sistemas y de las aplicaciones que se utilizan de forma habitual. Además de reducir la complejidad del proceso de desarrollo de software, la arquitectura de microservicios proporciona una forma novedosa de actualizar los sistemas heredados, de sistemas escalables con menores costes operativos, y de menor complejidad reducida [9].

Por su parte, la justificación práctica de esta investigación, hace posible la incorporación de la arquitectura de microservicios al ámbito de los seguros que puede ayudar a innovar y mejorar su rendimiento y eficacia formativa mediante el uso de dispositivos accesibles. De tal forma, que, debido a la integración de servicios a este modelo proporciona agilidad al momento de la comercialización y el intercambio entre los prestadores de servicios y los clientes [3].

A continuación, la hipótesis general, sostiene que la implementación de las arquitecturas de microservicios mejora los sistemas web de pólizas en comparación a la arquitectura monolítica. Lo que genera las siguientes hipótesis específicas:

HE1.1 La implementación de una arquitectura de microservicios redujo el uso de memoria RAM en los sistemas web en comparación con arquitectura monolítica.

Chulca y Molina [10] señalaron que la migración de una arquitectura monolítica a una de microservicio está en conjunción con los cambios que se sitúan en cambios en aspectos tales como: la estructura de la

organización, si este está en producción, se les hacen nuevas exigencias, y se atienden a problemas específicos de los sistemas. Las características de cada sistema y la técnica a utilizar, debe adaptarse a las necesidades del mismo, determinarán el procedimiento debido a estos aspectos, la metodología y el sistema a utilizar deben modificarse para satisfacer las necesidades de este y las cualidades propias de la organización.

HE1.2 La implementación de una arquitectura de microservicios redujo el uso de CPU en los sistemas web en comparación con arquitectura monolítica.

Mora [11] explicó que la arquitectura de microservicios reduce el uso de la CPU en sistemas web al permitir la escalabilidad granular, el aislamiento y la optimización de recursos, la implementación de cachés eficientes y el uso de tecnologías específicas para cada microservicio. Esto se traduce en una utilización más efectiva de los recursos de la CPU y en una mayor capacidad para administrar y ajustar el rendimiento de la aplicación de manera precisa.

HE2.1 La implementación de una arquitectura de microservicios redujo el tiempo de respuesta ante una prueba de estrés en los sistemas web en comparación con arquitectura monolítica.

Iranzo [12] explicó que mediante la adopción de una arquitectura de microservicios puede proporcionar ventajas significativas en términos de reducción del tiempo de respuesta ante pruebas de estrés en sistemas web en comparación con una arquitectura monolítica. Ello se debe a que con la arquitectura de microservicios se reduce el tiempo de respuesta ante pruebas de estrés en sistemas web al permitir una escalabilidad más precisa, actualizaciones más rápidas, aislamiento de fallos, balanceo de carga dinámico, implementación de cachés eficientes y sistemas de detección y recuperación automatizados. Estas características trabajan en conjunto para mantener un rendimiento óptimo incluso bajo cargas extremas, con un potencial significativo para su incorporación a diferentes entornos organizacionales.

HE2.2 La implementación de una arquitectura de microservicios redujo la cantidad de conexiones simultáneas en los sistemas web en comparación con arquitectura monolítica.

Figuera [13] halló que con la implementación de una arquitectura de microservicios es posible reducir la cantidad de conexiones simultáneas en los sistemas web ya que permite descomposición en componentes independientes los cuales dependerán de las necesidades y características de sistema adoptado. Está asociado a algunas ventajas entre las que se encuentran: el servicio de enrutamiento por versiones; propicia la ejecución de las diferentes funciones, particularidades como seguridad y enrutamiento son implementadas fuera del código del microservicio y son reusables; el monitoreo, que permite la telemetría de los programas; y otorga libertad en la elección del lenguaje de implementación, dado que por sus cualidades puede tomar de las librerías para su construcción de aplicación en red. Estas ventajas contribuyen a una mejor distribución de las conexiones y a un manejo más eficiente de la carga, lo que resulta en un rendimiento más estable y una menor congestión de conexiones simultáneas.

II. MARCO TEÓRICO

A continuación, se presentan los antecedentes del estudio, se consideraron investigaciones internacionales y nacionales. Para la selección se tomaron las variables de estudio, artículos de revistas e investigaciones de repositorios.

Ortiz et al. (2022) mejoraron la eficiencia del comercio electrónico de una empresa peruana de calzado utilizando una aplicación web con una arquitectura de microservicios y aplicación móvil. El tipo de diseño fue cuasi-experimental, con un enfoque cuantitativo y un alcance descriptivo y explicativo. Implementaron el método RUP (Rational Unified Process), que mejoró la eficiencia de los procesos. Concluyeron que la implementación de una arquitectura de microservicios mejora la eficiencia del comercio electrónico de una empresa de calzado [14].

Rojas (2022) proporcionó una herramienta digital para cerrar más ventas durante sus reuniones iniciales con los clientes y evaluar sus necesidades mediante una aplicación web a los asesores comerciales. La metodología implementó la aplicación SCRUM para el desarrollo de sistemas. Los resultados mostraron que con esta se liberan gradualmente nuevos elementos y funcionalidades de la cartera de la empresa. Concluyó que una de las ventajas se asocia a la disminución de los costes operativos para los asesores comerciales, del tiempo para obtener una cotización de seguro de vida, en el papeleo y en todo el proceso [15].

Alamilla et al. (2021) diseñaron un modelo de arquitectura REST para el desarrollo de aplicaciones web empresariales. La metodología incluyó el análisis de las aplicaciones monolíticas en empresas del sector privado mexicano. Los resultados indican que este particular diseño ha ganado espacio en muchas empresas con el aumento en el uso de NodeJS y las bases de datos NoSQL. En conclusión, indican que el diseño arquitectónico REST es una API (Application Program Interface) que se caracteriza por ser más rápida en las respuestas que ofrecen a las empresas independiente del sector y presentan mayor rendimiento que las monolíticas [16].

Canqui (2021) propuso diseñar microservicios utilizando el enfoque hexagonal como base del estudio, creando una arquitectura de microservicios para la facturación electrónica, principalmente para la generación de facturas. Se centró en el aspecto arquitectónico de los microservicios y herramientas utilizadas en el

proceso de prototipo según la metodología hexagonal. En su conclusión, señaló que, en comparación con la facturación electrónica monolítica, la facturación de microservicios es mucho mejor en términos de tiempo y costo [17].

Auer et al. (2021) propusieron un sistema de apoyo a la toma de decisiones basado en evidencia para empresas que necesitan migrar a microservicios. Realizaron una encuesta en forma de entrevistas a profesionales. El resultado les permitió delinear un sistema de evaluación que pueda ayudar a las empresas a decidir migrar. Concluyen señalando que este marco de referencia con base en evidencias ayudará a evitar la migración si no es necesaria, especialmente si la solución está en la reestructuración de su sistema monolítico o renovando su organización interna [18].

Macarlupu y Marín (2020) compararon las arquitecturas microservicios y REST para determinar cuál presenta un mejor rendimiento de acuerdo al tiempo de respuesta, uso de recursos y nivel de seguridad utilizando la metodología METSA. La metodología fue aplicada, de enfoque cuantitativo, no experimental de diseño transversal y enfoque descriptivo. Los resultados evidenciaron que los microservicios se caracterizaron por un menor tiempo de respuesta al momento de la carga de imágenes con respecto a las tecnologías REST. Para concluir indican que en un 10.89%, de tiempo menor con relación al uso de REST [19].

Chicaiza (2020) diseñó e implementó un prototipo de arquitectura basada en microservicios para la integración de aplicaciones web orientadas a entidades financieras altamente transaccionales. La metodología consistió en la creación de sprint 0, tiene relación con los elementos transversales, que consideran entre otros el diseño y la construcción del marco de trabajo. Los resultados establecieron que al momento de planificar el tiempo requerido en las tareas porque, se perfilan como un elemento clave para el diseño. Concluyó explicando que la intermitencia y los cuellos de botella a los que hacen frente los microservicios y los sistemas de TI altamente transaccionales podrían reducirse con la creación de una arquitectura de este tipo bien diseñada e implementada [20].

Macedo (2020) definió el concepto de eventos y una arquitectura de comunicación basado en ellos para microservicios, presentar las dificultades y posibles soluciones que se presentan al implementar esta arquitectura. Se trató de

una investigación de revisión bibliográfica. Concluyó indicando que esta arquitectura es recomendable cuando se desarrollan aplicaciones con una arquitectura distribuida, como es el caso de las arquitecturas de microservicios, porque permite aprovechar la plataforma en la nube y obtener un mejor rendimiento cuando se necesita un sistema con escalabilidad y flexibilidad [21].

Collado (2020) identificó la incidencia de la arquitectura de microservicios en el software del Ministerio de las mujeres. El método utilizado es RUP, que es compatible con los instrumentos del esquema de Designer Power y Rose Rational. Los microservicios se desarrollan utilizando Java versión 8 como lenguaje de programación, y utilizó el framework SpringBoot y Docker como contenedores de microservicios. Concluye señalando que la arquitectura de microservicios incide en la eficacia del software en el Ministerio de la Mujer y la Discapacidad [22].

Bogner (2020) suministró métodos, técnicas y herramientas apropiados para evaluar y mejorar la escalabilidad y promover la sostenibilidad a largo plazo a los desarrolladores de microservicios. En la metodología usaron métricas y las evaluaciones basadas en escenarios, y los contras de modelos basados en servicios se pueden utilizar para identificar puntos críticos. En conclusión, los profesionales de microservicios pueden usar artefactos que se elaboraron para el estudio para analizar y mejorar la escalabilidad de sus sistemas y obtener una comprensión conceptual de las garantías de esta [23].

Tapia et al. (2020) evaluaron algunas obras relacionadas de estas dos arquitecturas (monolítica y de microservicios). Evaluaron la relación entre el rendimiento y varias variables asociadas a aplicación que se ejecuta en una estructura monolítica en comparación con una que emplea la estructura de los microservicios. En su conclusión indicaron que ambas plataformas (monolíticos y microservicios) podrían usarse para una variedad de propósitos y de condiciones tecnológicas [24].

Muzaffar et al. (2020) lograron una comprensión básica de los sistemas basados en arquitectura de microservicios. En este estudio, utilizaron un enfoque de investigación mixta. El método incluyó plataformas, metodologías y herramientas relacionadas con aplicaciones en la nube, Internet de las cosas, DevOps, aplicaciones en tiempo real e IA para adaptarse a este modelo

arquitectónico. Concluyen señalando que las arquitecturas de microservicios son generalmente aplicables a sistemas distribuidos a gran escala [25].

Lanarduzzi et al. (2020) analizaron los cambios en la deuda técnica del código antes y después de la migración a microservicios. La metodología que implementaron fue el estudio de casos. Los resultados arrojaron que la migración a microservicios les permitió a reducir a largo plazo la deuda técnica. En sus conclusiones indicaron que su trabajo es uno de primeros estudios en los que se compara la deuda técnica (TD) de un proyecto de software creado por una pyme italiana antes y después de la migración a microservicios [26].

Navarro y Cabrera (2020) plantearon como objetivo implementar una arquitectura de microservicios con una malla de servicios. La metodología fue Scrum ya que tiene un ciclo de vida incremental e iterativo y funciona con procedimientos ágiles. Los resultados indican que los hallazgos se obtienen con rapidez y se pueden entregar partes del producto con regularidad, se obtienen las mejores soluciones. Concluyen indicando que a pesar de que la implementación de los servicios es bastante simple, es suficiente para lograr los resultados deseados y obtener una comprensión profunda de las configuraciones de Istio [27].

Caballero y Requena (2019) analizaron el proceso de evaluación de propuestas de crédito consumo vehicular y la solución tecnológica que le da soporte. La metodología para el análisis de datos, usaron la prueba t-Student y cálculos de frecuencia para encontrar la satisfacción. En sus conclusiones indican que la aplicación web de los microservicios es una solución técnica moderna para el proceso de evaluación y tiene un efecto positivo en el manejo y gestión del crédito en Caja Rural Prymera [28].

En esta parte se presenta la fundamentación teórica, se explicaron los conceptos y definiciones que respaldan el estudio, a los fines de darle sustento epistemológico al estudio.

La escalabilidad, de acuerdo a Laudon y Laudon [29], se refiere a la capacidad de un ordenador, dispositivo o sistema para proporcionar un servicio fiable a muchos usuarios. Esta tiene diferentes tipos: Escalabilidad vertical: Benítez [30] menciona que consiste en actualizar el hardware actual a otro más potente y caro, añadiendo procesadores o sustituyendo el hardware existente por

procesadores más rápidos. Sin embargo, la escalabilidad vertical está limitada, ya que a veces los precios aumentan exponencialmente con la potencia y al final ya no puedes permitirte un sistema más potente. En la figura 1 muestra la escalabilidad horizontal.

Figura 1 La escalabilidad horizontal.

Adaptado de Benítez [26]. Recuperado de Sistemas web escalables

Escalabilidad horizontal: Benítez [30] indica que se emplea para aumentar el número de servidores, pero no necesariamente su potencia, la escalabilidad horizontal implica dividir la carga de procesamiento. Se consigue utilizando muchos servidores, el sistema combinado se comporta como si fuera una sola máquina dedicada a una sola tarea, lo que maximiza la tolerancia a fallos, pero también plantea un problema importante para el administrador del sistema.

El diseño de un sistema escalable, considera una red necesaria, dentro del sistema o los atributos de proceso que muestran que puede continuar aumentando o indicando habilidades de preparación de calidad en todos los servicios en todos los servicios. Un ejemplo de cualquier empresa que tenga una red de usuarios en Internet, la empresa quiere crear un sistema que permita trabajar con clientes existentes, pero también con clientes potenciales, lógicamente existe la posibilidad de cambiar la configuración, en caso de que sea útil [31]. Se puede ver en la figura 2 la pirámide de escalabilidad.

Figura 2 La pirámide de escalabilidad.

Adaptado de Benítez [26]. Recuperado de Sistemas web escalables

En cuanto a las dimensiones de la escalabilidad esta la eficiencia, que de acuerdo a Casanova [32] esta se define como la capacidad de los sistemas de cumplir de manera adecuada sus funciones, describir áreas que podrían mejorarse, sobre todo en software y departamentos. Esta puede ser: eficiencia de recursos que se refiere la identificación de los recursos necesarios para mejorar los resultados y justificarlos en esta situación.

Esta la eficiencia de flujos, que Casanova [32] señala que se enfoca en aumentar la velocidad en lugar del tiempo de trabajo. Esto significa que cada miembro del equipo debe ser capaz de ejecutar todas las tareas necesarias para completar un proyecto de principio a fin, sin importar en qué departamento o especialidad trabajen; y la eficacia en un equipo de software, esta se deriva del análisis de las opiniones sobre uso de funciones por parte de los usuarios y recopilar estas a través de formularios de comentarios como una forma de obtener información objetiva y mejorar continuamente.

Estas capacidades se pueden determinar mediante las pruebas de software, las cuales Sommerville [33] distinguió como valoraciones que se hacen antes de su uso. Asimismo, uno de los indicadores es la fiabilidad, para Pressman [34], la investigación sobre esta cualidad del software se esforzó por predecir la fiabilidad del software utilizando los mismos principios matemáticos que subyacen a la fiabilidad del hardware. La disposición del software es la posibilidad de que un programa pueda funcionar según sea necesario en un momento dado, y su formula se expresa de este modo:

$$\text{Disponibilidad} = \frac{TMPF}{TMPF+} x100\%$$

Con relación a las pruebas de estrés, carga y rendimiento estas permiten saber la cantidad de datos o el momento en que la aplicación comienza a bloquearse o no responde a las solicitudes. Existen también las Pruebas de carga y escalabilidad, que son útiles para asegurarse de que la aplicación responde con rapidez en los picos de carga, se realizan pruebas de carga y escalabilidad. Luego,

se puede comparar los tiempos de respuesta medios tras evaluar estos tres niveles de carga para determinar si su sistema es escalable, es decir, si el tiempo de respuesta aumenta linealmente.

Sobre la arquitectura de software Reynoso [35] explica que la estructura fundamental de un sistema está representada por sus componentes, las conexiones entre estos y su entorno, así como los principios rectores que guían su diseño y desarrollo. Montoya [36] agrega que es posible referirse a un proyecto de desarrollo como "desarrollo de arquitectura de software" independientemente del enfoque empleado. Las fases de este desarrollo previo al sistema incluyen los requisitos, el diseño, la documentación y la evaluación.

Newman [37] indica que los sistemas monolíticos son aquellos cuyo centro es un conjunto de estructuras fijas que interactúan. En estos el núcleo tiene grandes módulos interactúan entre sí y las diferentes partes de este que se ensamblan en capas para articular su estructura; de este modo, los microservicios, son pequeños servicios que posee autonomía y que trabajan juntos. En relación a las arquitecturas basadas en microservicios, se refiere a aquellas que buscan dividir una aplicación en servicios independientes que intercambian datos entre sí utilizando el protocolo HTTP (Hiper Text Transfer Protocol); es decir, las respuestas generadas por los procesos del lado del servidor se agregan y se presentan al usuario final para mostrar que la aplicación funciona correctamente [38].

En la figura 3 se muestra la diferencia de una aplicación de arquitectura de

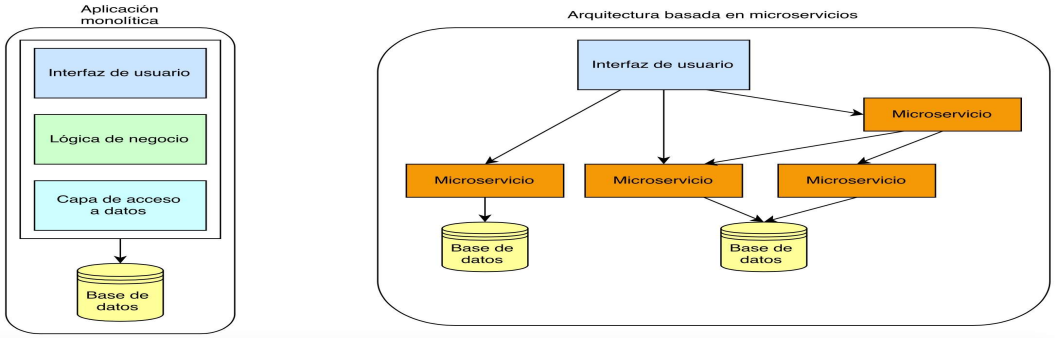


Figura 3 Diferencia en la aplicación monolítica y arquitectura de microservicios.

Adatado de IBM [39]. Recuperado de https://developer.ibm.com

Uno de los conceptos más difundidos es el aportado por Fowler y Lewis, dos de los primeros en explicarla su artículo “Microservicios” [40], y señalaron que se refiere a un estilo arquitectónico en el cual varios servicios funcionan de forma independiente y se implementan de la manera más automatizada posible, comunicándose entre sí a través de un mecanismo ligero (normalmente, un recurso API basado en HTTP). De este modo, el estado actual del desarrollo de microservicios a nivel empresarial cuenta con una amplia cantidad de experiencias exitosas. Un ejemplo es eBay. Esta empresa incorporó cinco cambios significativos en su infraestructura desde su creación (2002). Comenzó usando Perl y C++, luego evolucionó usando Java en 2007 y, por el crecimiento masivo en 2011, incorporó microservicios como la base de su desarrollo [41].

Entre otras JMETER [42] es una la aplicación Apache JMeter dentro del programa Java 100% gratuito y de código abierto hecho para monitorizar el rendimiento y realizar pruebas funcionales de comportamiento. Esta puede utilizarse para examinar el rendimiento de recursos dinámicos y aplicaciones web estáticas y dinámicas. Entre las cualidades que presenta se encuentran: a) capacidad para cargar y probar el rendimiento de diferentes aplicaciones / servidores / tipos de protocolos; b web (HTTP, HTTPS [Java, NodeJS, PHP, ASP.NET], servicios web SOAP / REST, FTP, base de datos a través de JDBC, LDAP, Middleware orientado a mensajes (MOM) a través de JMS; c) Correo (SMTP [S], POP3 [S] e IMAP [S]); d) comandos nativos o scripts de Shell; y TCP. Sobre los objetos de Java, se debe decir que la aplicación no consume muchos recursos en CPU (Central Processing Unit) y en memoria por tal motivo no afecta a las pruebas (ver anexo 3).

III. METODOLOGÍA

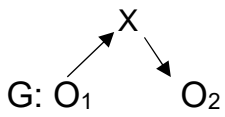
El capítulo III expone de manera amplia la metodología utilizada para lograr los objetivos de la investigación. Este consta de los aspectos teóricos y procedimentales que justifican la elección del tipo y diseño de la investigación, la definición de las variables, la descripción de la población y los criterios establecidos para la selección de la muestra. Además, se presentaron los instrumentos de medición y los programas que ayudaron en análisis de los datos. Por último, las consideraciones éticas implícitas en cada momento de la investigación.

3.1 Tipo y diseño de investigación

Creswell [43] explicó que los experimentos se definen como estudios de intervención, ya que los investigadores quieren que la situación intente explicar cómo influyen en las personas involucradas en comparación con las que no lo hacen. Puede probarse utilizando individuos, entidades y determinados elementos. Los experimentos alteran intervenciones, tratamientos, estímulos, efectos u otras variables controladas (denominadas variables dependientes) para observar sus efectos. Utilizando la definición de referencia, se llevó a cabo una aplicación experimental en este estudio.

El estudio presenta un diseño preexperimental ya que habrá tanto un pretest como un post-test. Un grupo recibió una prueba de pre estímulo o un tratamiento experimental, seguido de un tratamiento y finalmente una prueba de pos estímulo.

Bernal [44] explicó: "Un diseño preexperimental es aquel en el que el investigador no controla variables externas o intervinientes, los sujetos del estudio no se asignan al azar y no hay un grupo de control".



Donde:

G: Es el grupo experimental.

Es la muestra donde se aplicará la medición

X: es el sistema de información que aplicara (el experimento)

La aplicación será el que manipule la variable dependiente para la evaluación si el proceso de ventas genero cambios.

O_1: Pre-prueba

Es la evaluación de proceso de ventas antes de que el sistema empiece la producción. Esta medición de comparar con la post-prueba

O_2: Post-prueba

Es la evaluación de proceso de ventas después de que el sistema se encuentre en producción. Ambas mediciones se comparan para determinar el mejoramiento del proceso de ventas.

3.2 Variables y operacionalización

Definición Conceptual

Variable 1: Rendimiento de las arquitecturas monolíticas

Bencardino [45] indica que una arquitectura de red monolítica se refiere a una arquitectura en la que el software consta de grupos funcionales altamente relacionados (si los hay) que incluyen aspectos relacionados con la presentación, el procesamiento y el almacenamiento de información. Se implementan en un único componente de software. Históricamente, representaron la primera estructura de software, que consiste esencialmente en un solo programa unido por un conjunto de funciones que pueden llamarse entre sí.

Variable 2: Rendimiento de las arquitecturas de microservicios

Para Newman [37] una estrategia conocida como "arquitectura de microservicios" divide una aplicación informática en una serie de pequeños servicios, cada uno de los cuales funciona de forma independiente y se comunica con los demás, por ejemplo, mediante peticiones HTTP a sus API. Para gestionar otras tareas comunes (como el acceso a bases de datos), suele haber pocos servicios, pero cada microservicio es compacto y se ajusta a las necesidades de la aplicación.

Definición Operacional

Variable 1: Rendimiento de las arquitecturas monolíticas

Actualmente el sistema web de la organización se encuentra basado en la arquitectura web monolítica, la cual se encontraron fallas que representan una importante debilidad dentro de los procedimientos que se ejecutan para las diferentes áreas de la organización.

Variable 2: Rendimiento de las arquitecturas de microservicios

Se creará un sistema de arquitectura (web) de microservicios para convertir las debilidades de la organización en fortalezas.

3.3 Población (criterios de selección), muestra, muestreo y unidad de análisis

Población

Este trabajo se desarrolló en torno a la campaña de “Auto Seguro Online”.

Para determinar la población se consideró la cantidad de procesos del sistema web, siendo este de 30 para el Métrica “Porcentaje de consumo de CPU a una prueba de estrés”, “Porcentaje de consumo de memoria ante una prueba de estrés”, “Porcentaje de Tiempo de respuesta ante una prueba de estrés” y “Porcentaje de conexiones simultaneas”

Criterio de inclusión

- Auto Seguro Online sobre microservicios.
- Abordar explícitamente el estudio de estilos arquitectónicos orientados/basados en microservicios.
- Una exploración de los microservicios desde una perspectiva arquitectónica (aunque no se menciona el estilo).
- Este material científico analizado para esta investigación se ha publica desde 2018.

Criterio de exclusión

- Artículos que solo hablan de microservicios sin mencionar el aspecto arquitectónico.
- El estudio es un caso de uso/aplicación del estilo de arquitectura de microservicios.
- Artículos sobre las características de calidad de un estilo de arquitectura de microservicios.

Cabe destacar que este fue un proceso dinámico, ya que hubo cierto desacuerdo en la evaluación de los principales estudios, lo que llevó a la redefinición de los criterios de inclusión/exclusión.

Muestra

En la presente investigación para la Métrica cantidad de utilización de recursos se considera 30 procesos similares de los sistemas.

Muestreo

Hernández et al. [46], para este estudio se usó el muestreo probabilístico aleatorio simple o el muestreo aleatorio irrestricto, en el que cada elemento o unidad de la población total posee igual posibilidad de ser seleccionado. La ventaja del muestreo probabilístico aleatorio es que proporciona una base sólida para realizar inferencias estadísticas válidas sobre la población. Al garantizar la igualdad de oportunidades para cada unidad de la población, se minimiza el sesgo y se aumenta la representatividad de la muestra. En el caso de este estudio se empleó debido a que dadas las características de la empresa así lo permitían.

Unidad de análisis

La unidad de análisis está conformada por lo procesos sobre los cuales se realizará las pruebas de rendimiento.

3.4 Técnicas e instrumentos de recolección de datos

La técnica que se empleó en este trabajo fue la observación cuantitativa. Para Rodríguez [47], estas técnicas son los medios de recolección de información, de los cuales los más destacados son la observación, los cuestionarios, las entrevistas y las encuestas. Esta estrategia de recogida de información implica el reconocimiento eficaz y preciso de acciones y circunstancias aparentes.

En cuanto al instrumento que se empleó en el estudio fue la ficha de registro: donde se realizó un estudio de la eficacia del proceso del sistema en la organización antes y después del uso del sistema informático.

3.5 Procedimientos

Se utilizó para el análisis estadístico descriptivo, el uso de los programas Excel y SPSS versión 26, para obtener el resultado consolidado de datos estadísticos de tendencia central, normalidad y regresión logística ordinal. Las herramientas utilizadas para la recopilación de datos fueron desarrolladas por el investigador y validadas por tres ingenieros con experiencia en la materia que observaron las relaciones que existían entre las variables de la matriz de consistencia, las métricas y dimensiones y las preguntas del cuestionario.

Paso 1. Se definieron los objetivos a partir de la formulación de los problemas.

Paso 2. Se determinaron los indicadores o criterios de comparación entre los dos modelos (arquitectura monolítica y de microservicios).

Paso 3. Se procede a analizar los datos obtenidos para evaluar los efectos del tratamiento o estímulo sobre la variable dependiente.

Paso 4. Se procedió a la comparación de los resultados, a fin de elaborar la discusión respectiva.

3.6 Método de análisis de datos

Se usó la prueba de Shapiro-Wilk para evaluar la normalidad, dado que se tuvo muestras de 30 procesos. Se empleó esta prueba debido a que los datos siguen una distribución normal y es menor a 50 elementos. Al utilizar técnicas no

paramétricas, se puede realizar una comparación de medias adecuada y confiable sin asumir la normalidad de los datos. Se aplicó también pruebas de comparación de medias de Wilcoxon.

3.7 Aspectos éticos

Este estudio está de acuerdo con la ética profesional. Se respeta la autenticidad y origen de la información y datos aportados por los beneficiarios de las respectivas compañías además se respetará y mencionará en la bibliografía a los autores que sustentan el proyecto. La información confidencial a la que se puede acceder dentro de la empresa también se mantiene confidencial.

En este sentido, la producción intelectual de los diversos autores cuyas referencias se citaron en el texto, y sus contribuciones reseñadas a lo largo de la obra, respetando así los principios de la producción intelectual. Se trabajo siguiendo los lineamientos establecidos en el código de ética aprobado por resolución N.º 470-2022-VI-UCV, y por último se consideró la “Guía de elaboración de trabajos conducentes a grados y títulos” con resolución de vicerrectorado de investigación N.º 062-2023-VI-UCV. Así como lo establecido en los artículos 8° y 10° del Código de Ética del Colegio de Ingenieros del Perú (1987), en los cuales se establece que la conducta de todo ingeniero debe estar en consonancia con los fines de la institución y que su comportamiento debe ser en cualquier actividad cónsono con las regulaciones establecidas [48].

IV. RESULTADOS

Este capítulo presenta los resultados obtenidos de los análisis estadísticos, referidos a las variables, sus dimensiones e indicadores. De estos últimos, para la primera hipótesis: porcentaje de consumo de CPU a una prueba de estrés y porcentaje de consumo de memoria ante una prueba de estrés y para la segunda hipótesis: porcentaje de tiempo de respuesta ante una prueba de estrés y porcentaje de conexiones simultáneas. Se pudo comprobar que pasar de una arquitectura monolítica a una arquitectura de microservicios mejora la eficiencia y la fiabilidad de los porcentajes de consumo de CPU a una prueba de estrés.

4.1. Datos descriptivos

En esta sección, se representan los cálculos obtenidos en el estudio junto con las Métricas: "% de consumo de CPU en prueba de estrés", "% de consumo de memoria en prueba de estrés", "% de respuesta de prueba de estrés" y "% de Conexiones concurrentes". Las muestras de datos previas y posteriores a la prueba se procesan para cada muestra de cada Métrica.

En el Test de normalidad se aplicó la técnica de Shapiro-Wilk en las métricas “Porcentaje de eventos de emisión de recibos de pago” y “Porcentaje de gastos de funcionamiento” porque n <= 50. Como se logra apreciar, las muestras para las dos Métricas son inferiores a 50 e iguales y el Test de normalidad se realiza ingresando los datos de prueba previa y posterior para cada Métrica en la herramienta de IBM SPSS Statistics. El nivel de confianza fue 95% y cuando el valor de significancia es mayor o igual a 0.05 se tiene una distribución normal, mientras que si el valor de significancia es menor que 0.05 se tiene una distribución no normal, donde el valor de significancia se refiere al nivel de contraste crítico. Tras comprobar la normalidad de las métricas mencionadas, se obtuvo la información de las siguientes secciones.

4.1.1. Porcentaje de consumo de memoria RAM ante una prueba de estrés

PRE TEST

En la tabla 1 se observan los resultados de la estadística descriptiva de la métrica porcentaje de consumo de memoria RAM.

Tabla 1

Resultados estadísticos descriptivos – % de consumo de memoria ante una prueba de estrés

Estadísticos Descriptivo		
		Estadístico
Porcentaje de consumo de memoria ante una prueba de estrés – Microservicios	Media	44.400
	Desviación estándar	20.5805
Porcentaje de consumo de memoria ante una prueba de estrés – Monolíticos	Media	59.00
	Desviación estándar	24.8235

La distribución del dato porcentaje de consumo de memoria ante una prueba de estrés se consideró 30 usuarios conectados simultáneamente. En el caso de los sistemas monolíticos su promedio fue mayor a comparación a los microservicios.

Prueba de normalidad

Para la tabla 2, se representan los cálculos obtenidos de la Test de normalidad que se aplicó la métrica "porcentaje de consumo de memoria RAM ante una prueba de estrés" correspondiente a la prueba previa, percibiendo que el grado de libertad es menor a 50 por lo cual se tomó la prueba de "Shapiro - Wilk".

Tabla 2

Test de normalidad – Porcentaje de consumo de memoria RAM ante una prueba de estrés

Pruebas de normalidad			
	Shapiro-Wilk		
	Estadístico	Gl	Sig
Porcentaje de consumo de memoria ante una prueba de estrés – Microservicios	0.888	30	0.004
Porcentaje de consumo de memoria ante una prueba de estrés – Monolíticos	0.833	30	0.000

4.1.2. Porcentaje de consumo de CPU a una prueba de estrés

Se puede visualizar los resultados de la descripción del porcentaje de utilización de la CPU de la métrica de la prueba de estrés en la tabla 3.

Tabla 3

Resultados estadísticos descriptivos – % de utilización de la CPU de la métrica de la prueba de estrés

Estadísticos Descriptivo		
		Estadístico
Porcentaje de consumo de CPU ante una prueba de estrés – microservicios	Media	44.400
	Desviación estándar	20.5805
Porcentaje de consumo de CPU ante una prueba de estrés - monolítico	Media	59.000
	Desviación estándar	24.8235

La distribución del dato Porcentaje de consumo de CPU ante una prueba de estrés consideró 30 usuarios conectados simultáneamente. En el caso de los sistemas monolíticos su promedio fue mayor en comparación con los microservicios.

Prueba de normalidad

Para la tabla 4, se representan los cálculos obtenidos de la Test de normalidad que se aplicó a la métrica Porcentaje de consumo de CPU ante una prueba de estrés, correspondiente a la prueba previa, percibiendo que el grado de libertad es menor a 50, por lo cual se tomó la prueba de “Shapiro - Wilk”.

Tabla 4

Métrica Porcentaje de consumo de CPU a una prueba de estrés

Pruebas de normalidad			
	Shapiro-Wilk		
	Estadístico	gl	Sig
Métrica Porcentaje de consumo de CPU ante una prueba de estrés – Microservicios	0.888	30	0.004
Métrica Porcentaje de consumo de CPU ante una prueba de estrés – Monolíticos	0.833	30	0.000

4.1.3. Porcentaje de tiempo de respuesta ante una prueba de estrés

Se puede observar los resultados de la descripción del Porcentaje de Tiempo de respuesta ante una prueba de estrés en la tabla 5.

Tabla 5

Resultados estadísticos descriptivos – % de tiempo de respuesta ante una prueba de estrés

Estadístico Descriptivo		
		Estadístico
Porcentaje de Tiempo de respuesta ante una prueba de estrés – microservicios	Media	11.257
	Desviación estándar	6.8106
Porcentaje de Tiempo de respuesta ante una prueba de estrés - monolítico	Media	13.330
	Desviación estándar	8.8778

En la tabla 5 se aprecia la distribución del dato Porcentaje de tiempo de respuesta ante una prueba de estrés. Se consideró 30 usuarios conectados simultáneamente. En el caso de los sistemas monolíticos, su promedio fue mayor en comparación con los microservicios.

Prueba de normalidad

Para la tabla 6, se representan los cálculos obtenidos de la Test de normalidad que se aplicó a la métrica Porcentaje tiempo de respuesta ante una prueba de estrés correspondiente a la prueba previa, percibiendo que el grado de libertad es menor a 50, por lo cual se tomó la prueba de “Shapiro-Wilk”.

Tabla 6

Prueba de normalidad

Pruebas de normalidad			
	Shapiro-Wilk		
	Estadístico	gl	Sig
Métrica Porcentaje de consumo de CPU ante una prueba de estrés – Microservicios	0.770	30	0.000
Métrica Porcentaje de consumo de CPU ante una prueba de estrés – Monolíticos	0.759	30	0.000

4.1.4. Porcentaje de conexiones simultaneas

Se puede visualizar los resultados de la descripción del Porcentaje de conexiones simultáneas ante una prueba de estrés en esta tabla 7.

Tabla 7

Resultados estadísticos descriptivos – Porcentaje de conexiones simultáneas

Descriptivo		
		Estadístico
Porcentaje de conexiones simultaneas ante una prueba de estrés - microservicios	Media	52.867
	Desviación estándar	11.6463
Porcentaje de conexiones simultaneas ante una prueba de estrés - monolítico	Media	26.000
	Desviación estándar	4.6238

La distribución del dato Porcentaje de conexiones simultáneas ante una prueba de estrés consideró 30 usuarios conectados simultáneamente. En el caso de los sistemas monolíticos, su promedio fue mayor en comparación con los microservicios.

Prueba de normalidad

Para la tabla 8, se representan los cálculos obtenidos del Test de normalidad que se aplicó a la métrica Porcentaje de conexiones simultáneas ante una prueba de estrés correspondiente a la prueba previa, percibiendo que el grado de libertad es menor a 50, por lo cual se tomó la prueba de “Shapiro-Wilk”.

Tabla 8

Prueba de normalidad

Pruebas de normalidad			
	Shapiro-Wilk		
	Estadístico	gl	Sig
Métrica Porcentaje conexiones simultáneas a una prueba de estrés – Microservicios	0.879	30	0.003
Métrica Porcentaje conexiones simultáneas a una prueba de estrés – Monolíticos	0.703	30	0.000

4.2. Prueba de Hipótesis

Se procedió a realizar la prueba de la siguiente manera:

4.2.1. Prueba de Hipótesis 1

La implementación de una arquitectura de microservicios mejoró la fiabilidad en los sistemas web en comparación a las arquitecturas monolíticas web.

Indicador:

Porcentaje de conexiones simultaneas.

Hipótesis Especifica:

$H1_0$: La implementación de una arquitectura de microservicios no mejoró la eficiencia en los sistemas web en comparación a las arquitecturas monolíticas web.

$H1_A$: La implementación de una arquitectura de microservicios mejoró la eficiencia en los sistemas web en comparación a las arquitecturas monolíticas web.

Prueba de Wilcoxon

Para corroborar la validez de los resultados mencionados, se aplicó la prueba no paramétrica, ya que el resultado de la prueba de normalidad es menor a 0.05, por lo tanto, su comportamiento es no normal, es así como se opta por aplicar la prueba de Wilcoxon para la toma de decisiones respecto a las hipótesis planteadas**".**

En la tabla 9 se aprecia la prueba de Wilcoxon sobre el consumo de CPU a una prueba de estrés

Tabla 9

Prueba de Wilcoxon sobre el consumo de CPU a una prueba de estrés

Rangos

		N	Rango promedio	Suma de rangos
Porcentaje de consumo de CPU a una prueba de estrés post - Porcentaje de consumo de CPU ante una prueba de estrés pre	Rangos negativos	0[a]	.00	.00
	Rangos positivos	29[b]	15.00	435.00
	Empates	1[c]		
	Total	30		

a. Porcentaje de consumo de CPU a una prueba de estrés post < Porcentaje de consumo de CPU a una prueba de estrés pre
b. Porcentaje de consumo de CPU a una prueba de estrés post > Porcentaje de consumo de CPU a una prueba de estrés pre
c. Porcentaje de consumo de CPU a una prueba de estrés post = Porcentaje de consumo de CPU a una prueba de estrés pre

Como se puede ver el p valor resultó 0,000, siendo p < 0,05, de lo que se deduce que existen diferencias entre el porcentaje inicial y el final, en otras palabras, un aumento significativo del promedio de porcentaje de consumo de CPU a una prueba de estrés. En la tabla 10 se puede observar que el nivel de significancia en menos a 0.05, esto conduce a rechazar la hipótesis nula, ya que existen diferencias entre el porcentaje de consumo de CPU a una prueba de estrés de la arquitectura de microservicios con relación a los sistemas monolíticos. En conclusión, con un nivel de confianza del 95% se rechaza la hipótesis nula y se acepta la hipótesis alterna, por lo tanto, se afirma que, la implementación de estos sistemas mejora la eficiencia de los sistemas web.

Tabla 10

Porcentaje de consumo de CPU a una prueba de estrés

Estadísticos de Prueba[b]	
	Relación Pretest- Relación Postest. Porcentaje de consumo de CPU a una prueba de estrés
Z	1.096[b]
Sig. asintótica (bilateral)	.000

Hipótesis Específica 2

$H2_0$: La implementación de una arquitectura de monolítica no mejoró la fiabilidad en los sistemas web en comparación a las arquitecturas microservicios.

$H2_A$: La implementación de una arquitectura de monolítica mejoró la fiabilidad en los sistemas web en comparación a las arquitecturas microservicios.

Tabla 11

Prueba de Wilcoxon sobre el porcentaje de tiempo de respuestas ante una prueba de estrés

Rangos

		N	Rango promedio	Suma de rangos
Porcentaje de tiempo de respuesta ante una prueba de estrés post - Porcentaje de conexiones simultaneas pre	Rangos negativos	0[a]	.00	.00
	Rangos positivos	29[b]	15.00	435.00
	Empates	1[c]		
	Total	30		

a. Porcentaje de tiempos de respuesta ante una prueba de estrés post < Porcentaje de conexiones simultaneas pre

b. Porcentaje de conexiones simultaneas post > Porcentaje de conexiones simultaneas pre

c. Porcentaje de conexiones simultaneas post = Porcentaje de conexiones simultaneas pre

Como se puede ver el p valor resultó 0.000, siendo p < 0.05, de lo que se deduce que existen diferencias entre el porcentaje inicial y el final, es decir, que existe un aumento significativo del promedio de porcentaje de tiempo de respuesta ante una prueba de estrés. En la tabla 12 se puede observar que el nivel de significancia en menos a 0.05, esto conduce a rechazar la hipótesis nula, ya que existen diferencias entre el porcentaje de la arquitectura de microservicios con relación a los sistemas monolíticos. Se concluye, con un nivel de confianza del 95% se rechaza la hipótesis nula y se acepta la hipótesis alterna, por lo tanto, se afirma que, la implementación de estos sistemas mejora la eficiencia de los sistemas web.

Tabla 12

Porcentaje de tiempo de respuesta ante una prueba de estrés

Estadísticos de Prueba[b]	
	Relación Pretest- Relación Postest. Porcentaje de respuesta ante una prueba de estrés
Z	1.096[b]
Sig. asintótica (bilateral)	.000

4.3 Resumen

La tabla 13 muestra un resumen de los resultados de las pruebas de hipótesis específicas, realizadas a cada una de estas, se expresa de acuerdo a los resultados de las diferentes pruebas, si fueron aceptadas o rechazadas.

Tabla 13

Tabla resumen de los resultados de las pruebas de hipótesis específicas de la investigación

Cód.	Hipótesis	Condición
HE1.1	La implementación de una arquitectura de microservicios redujo el uso de memoria RAM en los sistemas web en comparación con arquitectura monolítica.	Aceptada
HE1.2	La implementación de una arquitectura de microservicios redujo el uso de CPU en los sistemas web en comparación con arquitectura monolítica.	Aceptada
HE2.1	La implementación de una arquitectura de microservicios redujo el tiempo de respuesta ante una prueba de estrés en los sistemas web en comparación con arquitectura monolítica.	Aceptada
HE2.2	La implementación de una arquitectura de microservicios redujo la cantidad de conexiones simultáneas en los sistemas web en comparación con arquitectura monolítica.	Aceptada

V. DISCUSIÓN

Los resultados derivados del estudio están detallados a continuación analizando y comparando el comportamiento del porcentaje de consumo de CPU en la prueba de estrés, porcentaje de consumo de memoria antes de la prueba de estrés, porcentaje de tiempo de respuesta antes de la prueba de estrés y porcentaje de concurrencia de las conexiones previamente y posteriormente a la implementación la arquitectura de microservicios. Luego de la comprobación de las hipótesis específicas, se obtuvo que la implementación de una arquitectura de microservicios mejoró la eficiencia del sistema web de una entidad corredora de seguros en comparación con la arquitectura monolítica.

La implementación de la arquitectura de microservicios en sustitución de la arquitectura monolítica redujo el uso de la memoria RAM de los sistemas Web. Ello en términos porcentuales se evidenció en una disminución del 14.6%, en la medición inicial el uso fue de un 59.0%, luego de implantar una arquitectura de microservicios su uso se ubicó en un 44.40%. Esto representa una reducción del uso de la memoria. El porcentaje de consumo de memoria antes de la prueba de estrés fue diferente antes y después de implementar la arquitectura de microservicios para una muestra de 30 procesos.

Este estudio tiene resultados similares a los obtenidos por Alamilla et al. (2021), quienes pasaron de un sistema monolítico a una arquitectura de microservicios con el objetivo principal de resolver los problemas de mantenimiento y fiabilidad, sin degradar significativamente el rendimiento del sistema. En su estudio con el diseño REST en una aplicación, que ofrece una respuesta rápida que con inmediatez se pueden satisfacer las demandas de los clientes. Esta coincidencia con los resultados de esta investigación permite comprender, medir y evaluar el rendimiento, la disponibilidad y la confiabilidad de servicios y hosts en la web, dado que se centra en aspectos críticos de la infraestructura y la aplicación, como el tiempo de respuesta, el tiempo de actividad, la latencia y otros indicadores clave. A la vez de ser una demostración de lo fácil de los procesos de migración de una a otra y del ahorro que puede significar para las empresas.

De acuerdo con estos resultados, antes de la implementación de sistema el valor porcentual del uso del CPU era 13.33% y luego de implementar una arquitectura de microservicios se ubicó en 11.26%, lo que demostró que hubo una reducción del 2.7% entre los dos valores porcentuales, dado que cada acción se realiza por separado evitando el desgaste. Estos resultados son semejantes a los obtenidos el trabajo de Ortiz et al. (2022), quienes a través del diseño de una aplicación dentro del modelo de arquitectura de microservicios registraron una reducción en el uso del CPU, ya que ejecutaron de manera independiente a hardware y plataformas específicos, lo que evita que las organizaciones tengan que realizar costosas actualizaciones.

Por otro lado, el porcentaje de conexiones concurrentes antes de implementar una arquitectura de microservicios fue 53.87% para una muestra de 30 procesos y después de implementar la arquitectura fue 26%. Según estos resultados, se produjo una reducción del 27.87% en el porcentaje de conexiones concurrentes, puesto que la implementación de la arquitectura de microservicios mejoró el sistema de benchmarking web y Web Architecture Group. Al hacer una comparación con los datos obtenidos de este estudio con el de Chicaiza (2020), se pudo apreciar que los atascos y cuellos de botella disminuyen con respecto a factores como la planificación del tiempo, la arquitectura de microservicios y sistemas TI y les permiten a las empresas incorporar estas estrategias para favorecer los procesos internos de la empresa y la atención a sus consumidores o clientes.

La implementación de una arquitectura de microservicios redujo la cantidad de conexiones simultáneas en los sistemas web en comparación con arquitectura monolítica, ya que su implementación en una arquitectura de microservicios puede reducir la cantidad de conexiones simultáneas en los sistemas web en comparación con una arquitectura monolítica debido a la modularidad y a la mayor eficiencia en el uso de recursos. Al igual que en este estudio, la eficiencia se evidencia en la reducción del estrés de los sistemas para responder a las necesidades de los clientes. Asimismo, los resultados son similares a los de Rojas (2022), quien halló que la implementación de un sistema con la estructura de microservicios está

relacionada con la baja en los costos operativos para los empleados y con relación al tiempo de cotización de seguros, dado que se gasta menos en papeleo y trámites.

En este sentido, se debe indicar que entre los aspectos asociados a la eficiencia se encuentra la reducción de costos, tiempos y de los modelos de gestión de los procesos. Collado (2020) coincidió en cuanto a la dimensión fiabilidad, ya que sus resultados demuestran la capacidad de ser implementados en cualquier institución y que el alcance de su calidad se compagina con su buena estructura y organización. En el contexto de la web, REST permite que se comuniquen y se enfoca en las respuestas para mejorar el rendimiento de las aplicaciones web.

VI. CONCLUSIONES

Las conclusiones de esta investigación son las siguientes:

1. La implementación de la arquitectura de microservicios ha demostrado mejorar significativamente la eficiencia de los sistemas web en comparación con las arquitecturas monolíticas. Esto se refleja en un aumento promedio del 14.06% en la eficiencia de los procesos de este sistema, como se evidencia en los resultados observados, con un porcentaje de consumo menor del sistema ante la prueba de estrés y por ende menos carga para la memoria RAM y la CPU.
2. La implementación de la arquitectura de microservicios ha demostrado ser una mejora significativa en la fiabilidad de los sistemas web en comparación con las arquitecturas monolíticas. Esto se evidencia claramente en un aumento del 11.26% en la fiabilidad general y un significativo aumento del 27.87% en la gestión de conexiones concurrentes, como se puede apreciar en los resultados obtenidos. Ello permitió incrementar la velocidad del tiempo de respuesta ante la prueba de estrés del sistema.
3. La implementación de la arquitectura de microservicios ha llevado a mejoras notables tanto en la eficiencia como en la fiabilidad de los sistemas web en comparación con las arquitecturas monolíticas, lo que fue confirmado por los puntos de encuentro entre los resultados de este estudio y los de otras investigaciones. Al comparar las cualidades del sistema monolítico con respecto a los microsistemas, las conexiones son más eficientes y los problemas se resuelven mediante cambios técnicos a todo el sistema, permitiendo escoger diferentes lenguajes y herramientas que pueden resolver mejor un determinado problema

VII. RECOMENDACIONES

Las recomendaciones para las próximas investigaciones son:

1. Las arquitecturas de microservicios tienen muchas ventajas sobre las arquitecturas monolíticas, aun así, este tipo de construcción de software es joven; de tal manera, se recomienda una investigación orientada a una perspectiva de costo de implementación dentro de la organización con el fin de comprobar la viabilidad de una migración o construcción de los sistemas actuales con esta nueva propuesta.
2. Investigar sobre las herramientas y/o librerías para la mejor implementación de las arquitecturas de microservicios, ya que por limitación de tiempo en esta investigación no se pudo aplicar los mecanismos para su mejor aplicación dentro los sistemas de información.
3. Ampliar la evaluación de la forma en que cada enfoque afecta al uso de CPU en situaciones normales y bajo carga; es decir, la eficiencia del CPU [10].
4. Otro aspecto que podría ser abordado en investigaciones futuras se relaciona con los efectos de escalabilidad granular en una arquitectura de microservicios y su impacto en el uso del CPU, ya que si bien existen investigaciones sobre este aspecto, en el ámbito de los seguros no se ha explorado mucho [11].
5. Analizar la manera en que la arquitectura de microservicios responde en términos de uso del CPU a partir de las pruebas de estrés y carga. Esto puede incluir simulaciones de picos de tráfico y la observación de los microservicios en el proceso de escalar y ajustar su consumo de recursos [13].
6. Abarcar estudios de casos en aplicaciones reales que han migrado de arquitecturas monolíticas a microservicios en los cuales se contraste el tiempo de respuesta a las pruebas de estrés, antes y después de la migración, tomando en cuenta diferentes cargas de trabajo y patrones de tráfico [12].
7. Indagar sobre la optimización de microservicios, en particular sobre estrategias y técnicas específicas que pueden aplicarse para optimizar el uso del CPU en microservicios individuales. Esto podría incluir el uso de cachés, la elección de tecnologías de implementación eficientes, la gestión de recursos compartidos y

la consideración de patrones de acceso a bases de datos y servicios externos [12].

8. Hacer estudios amplios sobre cómo el tamaño y la complejidad de los microservicios pueden influir en el uso del CPU, lo que implica comparar microservicios más pequeños y especializados con aquellos que realizan múltiples tareas para determinar cómo el diseño del microservicio afecta la utilización de recursos [12].

REFERENCIAS

[1] M. Occelli, L. García, N. Valeras y M. Quintanilla, Las tecnologías de la información y la comunicación como herramientas mediadoras en los procesos educativos., Santiago de Chile: Bellaterra Ltda., 2018.

[2] J. Salmerón, Desarrollo de una aplicación web basada en FaaS con .Net Core. Evolución desde una aplicación monolítica., Madrid: (Tesis de Master, Universidad Politécnica de Madrid), 2020.

[3] A. Bandeira, C. Madeiros, M. Paixao y P. Mendez, «We need to talk about microservices: an analysis from the discussions on StackOverflow» de Conference: Mining Software Repositories 2019, Montreal- Canadá, 2019.

[4] Hitachi Vantara , ¿Cuál es la diferencia entre la arquitectura monolítica y la de Microservicios?, Hitachi , 2021. [En línea]. Available: https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/. [Último accesso: 20 junio 2023].

[5] M. Villamizar, O. Gracés, L. Ochoa, H. Castro, L. Salamanca, M. Verano, R. Casallas y S. Gil, «Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures» SOCA, vol. 11, pp. 233-247, 2017.

[6] J. Segarra-Cobos y D. Murillo-Párraga, Desarrollo y gestión de riesgos del Sector Asegurador en el Ecuador, Revista Arbitrada Interdisciplinaria KOINONIA, vol. 6, nº 12, pp. 273-303, 2021.

[7] España Gobierno, Seguros y fondos de pensiones. Informe 2021. Ministerio de Asuntos Económicos y Transformación Digital, Madrid, 2021.

[8] MAPFRE Fundación, MAPFRE Economics, 2022. [En línea]. Disphttps://documentacion.fundacionmapfre.org/documentacion/publico/es/media/group/1116648.do.

[9] Bolo, M. Arquitectura de integración orientada a servicios. *Interfases*, 2006, no 001, p. 19-46.

[10] Chulca C. y Molina R. «Migración hacia una arquitectura basada en microservicios del sistema de gestión centralizada en laboratorios de la DGIP,» (Tesis de grado, Escuela Politécnica Nacional), 2020.

[11] Mora M., N. «Arquitectura basada en microservicios para sistemas E-Commerce en la empresa LCC Opentech, C.A.,» (Tesis de grado, Universidad Católica Andrés Bello), 2019.

[12] Iranzo J., V. «Desarrollo de software basado en microservicios: un caso de estudio para evaluar sus ventajas e inconvenientes» (Tesis de grado, Universitat Politécnica de Valencia), 2018.

[13] Figuera P., A. «Arquitectura de microservicios-Servicie Mesh,» (Tesis de grado, Universitat Politécnica de Catalunya), 2019.

[14] Ortiz, et al. Aplicación web con arquitectura de microservicios y el incremento de la eficiencia en el comercio electrónico de una empresa peruana de calzado.2022.

[15] J. Rojas T., Implementación de un sistema de información para la la automatización de un proceso de venta de seguros con AWS en una empresa aseguradora, 2022.

[16] L. Alamilla, V. Pérez, S. Sosa y J. Valentín, Arquitectura REST para el desarrollo de aplicaciones web empresariales. Revista Electrónica sobre Ciencia, Tecnología y Sociedad., vol. 8, nº 15, pp. 1-14, 2021.

[17] Canqui, J. Microservicios como estrategia en la facturación electrónica. Tesis Doctoral. 2021.

[18] Auer, et al. De los sistemas monolíticos a los microservicios: Un marco de evaluación. Tecnologías de la información y el software, 2021, vol. 137, pág. 106600.

[19] A. Macarlupu y E. Marín I., Estudio comparativo cuantitativo de las tecnologías Microservicios y REST, 2020.

[20] D. Chicaiza, Diseño de un prototipo de una arquitectura basada en microservicios para la integración de aplicaciones web altamente transaccionales. Caso: entidades financieras., 2020.

[21] Macedo, A. Uso de una arquitectura basada en eventos como capa de comunicación para microservicios.2020.

[22] Collado, O. Arquitectura de microservicios para el aseguramiento de la calidad del software en el Ministerio de la Mujer y Poblaciones Vulnerables. 2020.

[23] Bogner, J. Sobre la garantía de la capacidad de evolución de los microservicios: métricas, escenarios y patrones. 2020.

[24] Tapia, et al. De sistemas monolíticos a microservicios: un estudio comparativo de rendimiento. Ciencias aplicadas, 2020, vol. 10, nº 17, pág. 5797

[25] Muzaffar, A., & Sabahat, A. Architecture for microservice based system. A report. 2020.

[26] V. Lanarduzzi, F. Lomio, N. Sasrimäki y D. Taibi, «Does migrating a monolithic system to microservices decrease the technical debt?,» Journal of Systems and Software, vol. 169, 2020.

[27] R. Navarro y R. Cabrera, Aplicación basada en arquitectura de microservicios, Madrid, 2020.

[28] Caballero, J. & Requena, E. Aplicación web basada en una arquitectura de microservicios para la mejora en la evaluación de créditos de consumo en Caja Rural Prymera 2019. 2020.

[29] Laudon, K. y Laudon, J. Sistemas de información gerencial. Naucalpan de Juárez. 2012.

[30] Benítez, C. Sistemas web escalables. 2013, Revista USER. Buenos Aires, Argentina: Fox Andina en coedición con DÁLAGA S.A.

[31] Portal de arquitectura Arqhys.com. Equipo de redacción profesional. (2012, 12). La escalabilidad. Escrito por: Arqhys Construcciones. [Obtenido en fecha 09, 2022], desde el sitio web: https://www.arqhys.com/construcciones/escalabilidad.html

[32] Casanova, S. Efectividad en equipos de desarrollo. [Internet] Eficiencia de recursos y eficiencia de flujo en el software. [Citado 15 de septiembre del 2022]. https://samuelcasanova.com/2015/11/eficiencia-de-recursos-y-eficiencia-de-flujo/.

[33] Sommerville, I. Ingeniería de software novena edición. I. Somerville, Ingeniería de software Novena Edición, 2011.

[34] Pressman, R. Ingeniería del Software, Un enfoque práctico. España: Editorial Mac Graw Hill. 2010.

[35] Reynoso, C. Introducción a la Arquitectura de Software. Universidad de Buenos Aires, 2004, vol. 33.

[36] Montoya, E. La ingeniería de sistemas y su evolución hacia la arquitectura de sistemas. Lámpsakos, 2009, no 2, p. 96-105.

[37] Newman, S. Creación de microservicios. "O'Reilly Media, Inc.", 2021.

[38] A. Soto, «Diferencias entre una arquitectura monolítica y una de microservicios,» 2019. [En línea]. Available: https://openwebinars.net/blog/diferencia-entre-arquitectura-monolitica-y-microservicios/.

[39] IBM Developer [Internet]. IBM Developer; [consultado el 4 de octubre de 2022]. Disponible en: https://developer.ibm.com/.

[40] Lewis, J. & Fowler, M. Microservicios: una definición de este nuevo término arquitectónico. *MartinFowler.com*, 2014, vol. 25, pág. 14-26.

[41] Share and Discover Knowledge on SlideShare [Internet]. EBay Architecture; [consultado el 4 de octubre de 2022]. Disponible en: https://www.slideshare.net/tcng3716/ebay-architecture

[42] JMETER. Tutorials. 2017. Recuperada de http://jmeter.apache.org/

[43] Creswell, John W.; Tashakkori, Abbas. Journal of Mixed Methods. Sage, 2007, vol. 1, no 4, p. 303-308.

[44] Bernal, C. Metodología de la Investigación (Segunda Edición ed.). Naucalpan. México: Pearson, 2006.

[45] Bencardino, C. Sistemas web escalables. 2013, Revista USER. Bogotá, Colombia: ECOE ediciones Ltda.

[46] Hernández, R.; Fernández, C.; & Baptista, P. Metodología de la investigación. 6ta Edición Sampieri. Soriano, RR (1991). Guía para realizar investigaciones sociales. Plaza y Valdés, 2016.

[47] Rodríguez, Elmina Matilde Rivadeneira. Lineamientos teóricos y metodológicos de la investigación cuantitativa en ciencias sociales. In Crescendo, 2017, vol. 8, no 1, p. 115-121.

[48] Colegio de Ingenieros del Perú. Código de ética del colegio de ingenieros del Perú. Código de Ética del CIP, 26. 1999. http://www.cip.org.pe/publicaciones/reglamentosCNCD2018/codigo_de_etica_del_cip.pdf.

ANEXOS

Anexo 1: Matriz de consistencia

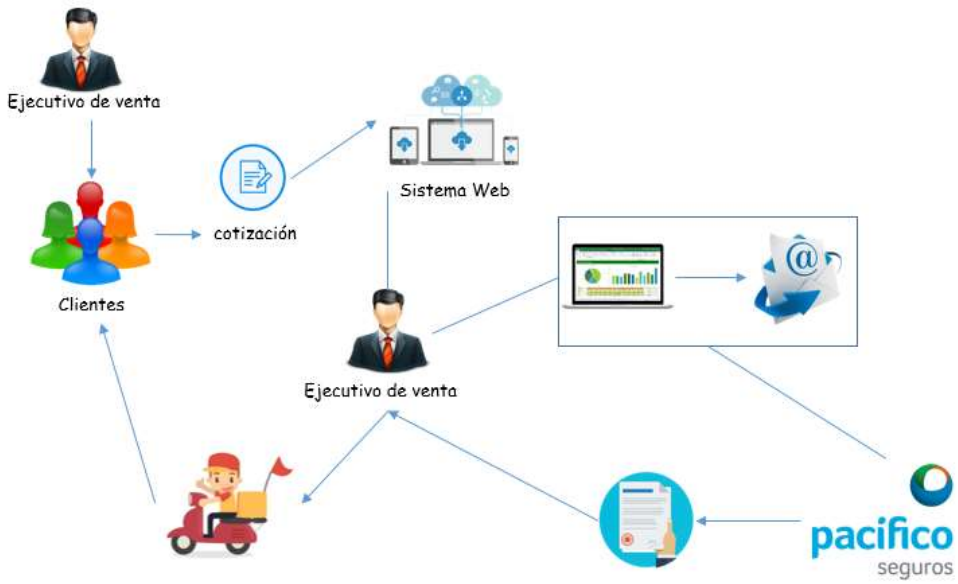
TITULO: COMPARACIÓN DEL RENDIMIENTO DE LAS ARQUITECTURAS MONOLÍTICA Y MICROSERVICIOS EN LOS SISTEMAS WEB DEL GRUPO MARSH

Problema	Objetivos	Hipótesis	Variables	Dimensiones	Indicadores
General	**General**	**General**	**-**	-	
¿Cómo las arquitecturas monolíticas mejoran los sistemas web en comparación a las arquitecturas de microservicio?	Determinar si la arquitectura monolítica mejora los sistemas web en comparación a las arquitecturas de microservicio	La implementación de una arquitectura monolítica mejoró los sistemas web de pólizas en comparación a las arquitecturas de microservicio Ortiz, et al [8]	Rendimiento de la arquitectura web monolíticas. Cabello [27].	-Eficiencia Casanova [15] Fiabilidad. Pressman [17].	Porcentaje de consumo de CPU a una prueba de estrés [12] Porcentaje de consumo de memoria ante una prueba de estrés [12]
Secundario	**Específicos**	**Específicas**			
P1: ¿Cómo la arquitectura monolítica mejora la eficiencia en los sistemas web en comparación a las arquitecturas de microservicios? **P2:** ¿Cómo la arquitectura monolítica mejora la fiabilidad en los sistemas web en comparación a las arquitecturas de microservicios?	O1: Determinar si la arquitectura de monolítica mejora la eficiencia en los sistemas web en comparación a las arquitecturas de microservicios. O2: Determinar si la arquitectura monolítica mejora la fiabilidad en los sistemas web en comparación a las arquitecturas de microservicios.	**H_1:** La implementación de una arquitectura monolítica mejoró la eficiencia en los sistemas web en comparación a las arquitecturas de microservicios. Tapia, et al. [5]. **H_2:** La implementación de una arquitectura monolítica mejoró la fiabilidad en los sistemas web en comparación a las arquitecturas de microservicios. Macedo [11]	Rendimiento de las arquitecturas de microservicios. Newman [21]		Porcentaje de Tiempo de respuesta ante una prueba de estrés [17]
					Porcentaje de conexiones simultaneas [17]

Anexo 2: Matriz de operacionalización de variables

Variable	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Escala de medición
Variable 1 Rendimiento de la arquitectura web monolíticas	Bencardino [41] indicó que una arquitectura de red monolítica se refiere a una arquitectura en la que el software consta de grupos funcionales altamente relacionados (si los hay) que incluyen aspectos relacionados con la presentación, el procesamiento y el almacenamiento de información. Se implementan en un único componente de software. Históricamente, representaron la primera estructura de software, que consiste esencialmente en un solo programa unido por un conjunto de funciones que pueden llamarse entre sí.	Una base de código única para toda la aplicación.	Eficiencia Según Casanova [15] explica que la eficiencia ha sido tradicionalmente un término utilizado para describir áreas que podrían mejorarse, sobre todo en software y departamentos.	Porcentaje de consumo de CPU a una prueba de estrés [12]. Porcentaje de consumo de memoria ante una prueba de estrés [12].	Valor Valor
Variable 2 Rendimiento de las arquitecturas de microservicios	Para Newman [33] una estrategia conocida como "arquitectura de microservicios" divide una aplicación informática en una serie de pequeños servicios, cada uno de los cuales funciona de forma independiente y se comunica con los demás, por ejemplo, mediante peticiones HTTP a sus API. Para gestionar otras tareas comunes (como el acceso a bases de datos), suele haber pocos servicios, pero cada microservicio es compacto y se ajusta a las necesidades de la aplicación.	Múltiples bases de código. Cada microservicio tiene su propia base de código	Fiabilidad: de los sistemas tiene que ver con la posibilidad de que tanto el sistema como el dispositivo cumpla con las funciones para las cuales se le diseñó en determinadas condiciones durante un tiempo establecido [44].	Porcentaje de Tiempo de respuesta ante una prueba de estrés [17] Porcentaje de conexiones simultaneas [17]	Valor Valor

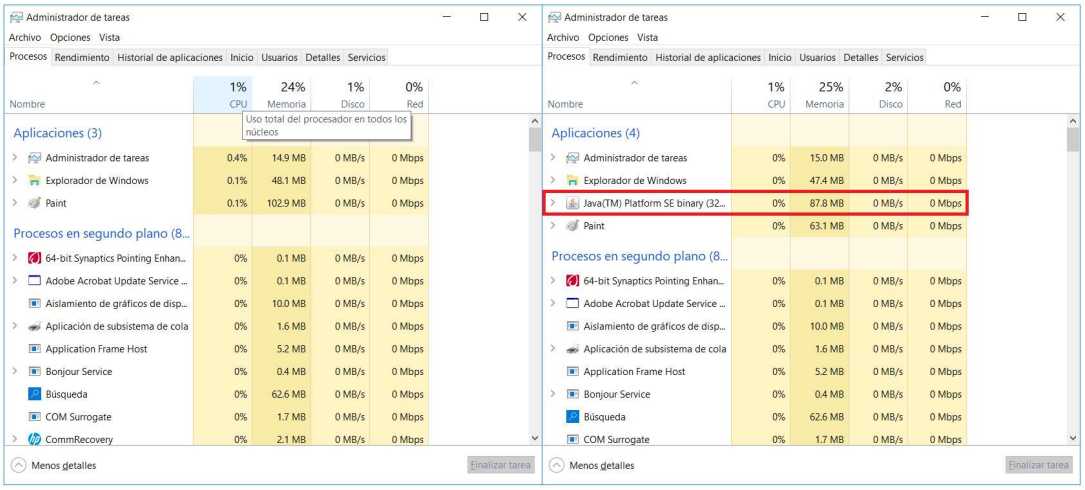
Anexo 3: Flujo actual de la entidad corredora de seguros



Representación del flujo actual de la corredora de seguros

1. El representante de ventas se reúne con los clientes habituales o a los nuevos las opciones de seguro que ofrece la compañía.
2. Se elabora la cotización de acuerdo a la elección realizada por los clientes, que son personalizadas en el sistema web.
3. Los clientes la revisan e indican si responde a sus necesidades.
4. El ejecutivo de ventas la recibe firmada y la consigna a su supervisor de la empresa.
5. Se registra en el sistema y se procesa.

Anexo 4: Cantidad de recursos del JMETER



Anexo 2: La solución

La solución que representa el uso de la arquitectura de microservicio

1. Solicitud de la póliza mediante un correo.
2. Se genera la cotización.
3. Se valida la información.
4. Se procesa y validan los datos, las formas de pago y los términos.
5. Se le manda al cliente la cotización final.
6. Una vez confirmada la elección por el cliente es devuelta de manera digital y el sistema registra el plan seleccionado.
7. Valia la información y responde de manera inmediata al cliente.

Anexo 3: La solución tecnológica

Proceso de software

Pasos previos

- Definir los requisitos mediante el análisis del contexto del sistema, los motores empresariales y los elementos que las partes interesadas del sistema consideran esenciales para el éxito.
- Crear estructuras arquitectónicas y planes de coordinación que cumplan los requisitos para definir una arquitectura.
- Evaluar la arquitectura decidiendo cuándo y cómo utilizar determinadas técnicas de evaluación arquitectónica, llevar a cabo las evaluaciones y utilizar los resultados de las evaluaciones para mejorar el desarrollo de la arquitectura.
- Documentación adecuada y fácilmente disponible de la arquitectura para los desarrolladores y otras partes interesadas.
- Análisis de la arquitectura para comprobar la seguridad y el rendimiento del sistema.
- Creación de un prototipo para implantar la arquitectura requerida.

Creación de la aplicación

1. El Spring Boot se empleó para el desarrollo de la aplicación de la oferta y los servicios de la aseguradora, en la cotización de pólizas, y trámites posteriores.
2. Para el front-end el desarrollador debió programar el navegador web, para lo cual creó un diseño a partir de la imagen y los colores de la empresa.
3. Esta aplicación basada en el uso del lenguaje Java de código abierto, para el back-end, se caracteriza por ser accesible a usuarios de diferentes edades, y por no representar para la empresa algún tipo de costo o la necesidad de emplear mucho tiempo en ello.
4. Es esencial disponer de una documentación clara del sistema para enfocar la migración hacia un enfoque correcto, para poder priorizar los servicios más importantes para el modelo de la empresa, de modo que la migración cumpla su objetivo principal de modelo, que es la eficacia del sistema para los usuarios. Debido a la información que hay que enviar a cada microservicio, la comunicación entre ellos puede resultar difícil. Se utilizaron llamadas a servicios REST para procesar los datos suministrados y entregar información a cada uno de ellos.

Hardware

Para su uso, otro aspecto favorable de su uso por el desarrollador, en cuanto al hardware, se utilizaron:

Hardware de entrada: son los periféricos, que se encargan de introducir nueva información en la unidad de procesamiento (teclado y ratón)

Hardware de salida: se trata de un dispositivo de salida, cuya función es permitir que el usuario interactúe con el ordenador (monitor).

Hardware de almacenamiento: que incluyó los componentes destinados al almacenamiento de datos, como es el caso de los discos duros.

Hardware de procesamiento: componentes cuya función consiste en procesar la información que los usuarios introducen en el CPU y permiten el funcionamiento del sistema.

Se diseñó una interfaz adaptable tanto en la PC, como al móvil, considerando los colores y la imagen de la empresa.

Anexo 4: Ficha de observación para el indicador Porcentaje de consumo de CPU a una prueba de estrés pre-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	1
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**	**Porcentaje de consumo de CPU a una prueba**
Validar cotización	20	20
Enviar correo	20	20
Buscar cotización	20	30
Calcula prima	20	30
Listar persona	20	30
Listar planes	20	30
Listar pólizas	20	30
Calcular cambio de moneda	20	30
Aprobar planes	20	30
Rechazar planes	20	30
Calcula valor comercial	20	40
Aprobar póliza	20	50
Listar cotización	20	60
Asignar tareas	20	60
Anular cotización	20	70
Editar cotización	20	70
Editar tareas	20	70
Ingresar cotización	20	80
Editar cotización	20	80
Obtener planes	20	80
Buscar persona	20	80
Buscar planes	20	80
Buscar póliza	20	80
Editar persona	20	80
Cargar persona	20	80
Inserta planes	20	80
Generar una renovación	20	80
Generar póliza	20	90
Insertar persona	20	90
Crear tareas	20	90

Anexo 5: Ficha de observación para el indicador Porcentaje de consumo de CPU a una prueba de estrés post-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	2
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**	**Porcentaje de consumo de CPU a una prueba**
Validar cotización	20	10
Enviar correo	20	20
Buscar cotización	20	20
Calcula prima	20	20
Listar persona	20	20
Listar planes	20	20
Listar pólizas	20	20
Calcular cambio de moneda	20	20
Aprobar planes	20	20
Rechazar planes	20	30
Calcula valor comercial	20	30
Aprobar póliza	20	30
Listar cotización	20	40
Asignar tareas	20	50
Anular cotización	20	50
Editar cotización	20	50
Editar tareas	20	50
Ingresar cotización	20	60
Editar cotización	20	60
Obtener planes	20	60
Buscar persona	20	60
Buscar planes	20	60
Buscar póliza	20	60
Editar persona	20	60
Cargar persona	20	60
Inserta planes	20	60
Generar una renovación	20	70
Generar póliza	20	70
Insertar persona	20	70
Crear tareas	20	82

Anexo 6: Ficha de observación para el indicador Porcentaje de consumo de CPU a una prueba de estrés pre-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	3
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**	**Porcentaje de consumo de memoria ante una prueba de estrés**
Validar cotización	20	60
Enviar correo	20	80
Buscar cotización	20	30
Calcula prima	20	20
Listar persona	20	70
Listar planes	20	80
Listar pólizas	20	80
Calcular cambio de moneda	20	80
Aprobar planes	20	80
Rechazar planes	20	80
Calcula valor comercial	20	90
Aprobar póliza	20	30
Listar cotización	20	40
Asignar tareas	20	80
Anular cotización	20	80
Editar cotización	20	90
Editar tareas	20	30
Ingresar cotización	20	30
Editar cotización	20	30
Obtener planes	20	80
Buscar persona	20	30
Buscar planes	20	90
Buscar póliza	20	60
Editar persona	20	20
Cargar persona	20	70
Inserta planes	20	50
Generar una renovación	20	80
Generar póliza	20	70
Insertar persona	20	30
Crear tareas	20	30

Anexo 7: Ficha de observación para el indicador Porcentaje de consumo de memoria ante una prueba de estrés post-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	4
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**	**Porcentaje de consumo de memoria ante una prueba de estrés**
Validar cotización	20	40
Enviar correo	20	50
Buscar cotización	20	10
Calcula prima	20	20
Listar persona	20	50
Listar planes	20	60
Listar pólizas	20	60
Calcular cambio de moneda	20	60
Aprobar planes	20	60
Rechazar planes	20	60
Calcula valor comercial	20	82
Aprobar póliza	20	20
Listar cotización	20	20
Asignar tareas	20	60
Anular cotización	20	50
Editar cotización	20	70
Editar tareas	20	20
Ingresar cotización	20	20
Editar cotización	20	20
Obtener planes	20	70
Buscar persona	20	20
Buscar planes	20	70
Buscar póliza	20	50
Editar persona	20	20
Cargar persona	20	60
Inserta planes	20	30
Generar una renovación	20	60
Generar póliza	20	60
Insertar persona	20	30
Crear tareas	20	30

Anexo 8: Ficha para observar el indicador Porcentaje de tiempo de respuesta ante una prueba de estrés pre-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	5
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**	**Porcentaje de Tiempo de respuesta ante una prueba de estrés**
Validar cotización	20	6.5
Enviar correo	20	25.4
Buscar cotización	20	6.5
Calcula prima	20	8.3
Listar persona	20	10.2
Listar planes	20	20.4
Listar pólizas	20	10.5
Calcular cambio de moneda	20	6.5
Aprobar planes	20	6.5
Rechazar planes	20	6.5
Calcula valor comercial	20	6.5
Aprobar póliza	20	9.6
Listar cotización	20	15.2
Asignar tareas	20	25.4
Anular cotización	20	9.4
Editar cotización	20	25.4
Editar tareas	20	6.5
Ingresar cotización	20	6.5
Editar cotización	20	6.5
Obtener planes	20	25.4
Buscar persona	20	10.2
Buscar planes	20	9.4
Buscar póliza	20	9.4
Editar persona	20	5.4
Cargar persona	20	25.4
Inserta planes	20	40.4
Generar una renovación	20	10.2
Generar póliza	20	25.4
Insertar persona	20	10.2
Crear tareas	20	10.2

Anexo 9: Ficha de observación para el indicador Porcentaje de conexiones simultaneas pre-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	6
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**	**Porcentaje de Tiempo de respuesta ante una prueba de estrés**
Validar cotización	20	5.5
Enviar correo	20	20.4
Buscar cotización	20	5.5
Calcula prima	20	7.3
Listar persona	20	9.4
Listar planes	20	20.4
Listar pólizas	20	10.5
Calcular cambio de moneda	20	5.5
Aprobar planes	20	5.5
Rechazar planes	20	5.5
Calcula valor comercial	20	5.5
Aprobar póliza	20	9.2
Listar cotización	20	9.4
Asignar tareas	20	20.4
Anular cotización	20	9.4
Editar cotización	20	20.4
Editar tareas	20	5.5
Ingresar cotización	20	5.5
Editar cotización	20	5.5
Obtener planes	20	20.4
Buscar persona	20	9.4
Buscar planes	20	8.4
Buscar póliza	20	8.4
Editar persona	20	5.4
Cargar persona	20	20.4
Inserta planes	20	30.4
Generar una renovación	20	9.4
Generar póliza	20	20.4
Insertar persona	20	9.4
Crear tareas	20	9.4

Anexo 10: Ficha de observación para el indicador Cantidad de usuarios simultáneos pre-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	7
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**
Validar cotización	30
Enviar correo	20
Buscar cotización	30
Calcula prima	30
Listar persona	20
Listar planes	30
Listar pólizas	20
Calcular cambio de moneda	30
Aprobar planes	30
Rechazar planes	30
Calcula valor comercial	30
Aprobar póliza	20
Listar cotización	25
Asignar tareas	20
Anular cotización	25
Editar cotización	20
Editar tareas	30
Ingresar cotización	30
Editar cotización	30
Obtener planes	20
Buscar persona	30
Buscar planes	20
Buscar póliza	30
Editar persona	30
Cargar persona	20
Inserta planes	30
Generar una renovación	30
Generar póliza	20
Insertar persona	25
Crear tareas	25

Anexo 11: Ficha de observación para el indicador Cantidad de usuarios simultáneos post-test

FICHA DE OBSERVACIÓN

Nº DE FICHA DE OBSERVACION:	8
Herramienta:	JMETER
Observador:	Miguel Toledo Azorza

Procesos del sistema	**Cantidad de usuarios simultáneos**
Validar cotización	90
Enviar correo	60
Buscar cotización	60
Calcula prima	60
Listar persona	40
Listar planes	30
Listar pólizas	52
Calcular cambio de moneda	60
Aprobar planes	60
Rechazar planes	60
Calcula valor comercial	49
Aprobar póliza	50
Listar cotización	35
Asignar tareas	50
Anular cotización	60
Editar cotización	30
Editar tareas	70
Ingresar cotización	70
Editar cotización	80
Obtener planes	60
Buscar persona	70
Buscar planes	40
Buscar póliza	60
Editar persona	50
Cargar persona	35
Inserta planes	70
Generar una renovación	70
Generar póliza	35
Insertar persona	50
Crear tareas	50

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Autenticidad del Asesor

Yo, ALFARO PAREDES EMIGDIO ANTONIO, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA ESTE, asesor de la tesis completa titulada: "Comparación del Rendimiento de las Arquitecturas Monolíticas y Microservicios en los Sistemas Web", cuyo autor es TOLEDO AZORZA, MIGUEL ÁNGEL JESÚS, constato que la investigación tiene un índice de similitud de 17.00%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis Completa cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

_____

Dr. Emigdio Antonio Alfaro Paredes

DNI 10288238

INVESTIGA UCV