UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP DE MERCADO, JULIACA 2022

TESIS

PRESENTADA POR:

Bach. SONIA YESENIA COSME ESCALERA

Bach. DIEGO DEYVIS QUISPE CANSAYA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO – PERÚ

2024

Reporte de similitud

NOMBRE DEL TRABAJO

ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP DE MERCADO, JULIACA 2022

AUTOR

SONIA YESENIA COSME ESCALERA y DIEGO DEYVIS QUISPE CANSAYA

RECUENTO DE PALABRAS

14476 Words

RECUENTO DE CARACTERES

83015 Characters

RECUENTO DE PÁGINAS

97 Pages

TAMAÑO DEL ARCHIVO

5.5MB

FECHA DE ENTREGA

Jan 8, 2024 9:34 AM GMT-5

FECHA DEL INFORME

Jan 8, 2024 9:36 AM GMT-5

● 19% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base d

- 19% Base de datos de Internet
- Base de datos de Crossref
- 9% Base de datos de trabajos entregados
- 1% Base de datos de publicaciones
- Base de datos de contenido publicado de Crossr

● Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Material citado
- Coincidencia baja (menos de 15 palabras)

José P. Tito Lipa
Ing. Estadístico e Informático D.Sc.
CIP. 159645

M.Sc. Elqui Yeye Pari Condori

Resumen

DEDICATORIA

Con mucho cariño a mi madre Robertina, quisiera agradecerle por su apoyo y confianza al permitirme llegar a este momento tan importante en el desarrollo de mi carrera profesional.

A mis hermanos Juana, Nelson, Aida y Elvia por su apoyo constante y con quienes puedo contar para todos mis éxitos.

Sonia Yesenia C.E.

DEDICATORIA

Quiero agradecer a mis queridos padres Armando y Leonarda por todos los esfuerzos y sacrificios que han hecho por mí, por sus valiosos consejos y por permitirme llegar a este importante momento en mi formación profesional. A mis hermanos Jhon, Fredy y Elvis por su constante apoyo y con quienes sé que puedo contar para todos mis logros.

Diego Deyvis Q.C.

AGRADECIMIENTOS

A la Universidad Nacional del Altiplano, Puno y la facultad de ingeniería estadística e informática, por acogernos en sus aulas durante estos cinco años de formación.

También A los profesores de la Facultad de Estadística e Informática que compartieron sus conocimientos y contribuyeron a mi formación profesional, por responder todas nuestras dudas durante las sesiones de aprendizaje, quisiera expresarles mi cariño, Respeto y admiración a todos y cada uno de los docentes.

Nos gustaría expresar nuestro más sincero agradecimiento a nuestros jurados D.Sc. Percy Huata Panca. D.Sc. Juárez Vargas, Juan Carlos, M.Sc. Torres Cruz, Fred y en especial a nuestro director de tesis M.Sc. Elqui Yeyé Parí Condori, que participaron en nuestra formación profesional.

Y a todas las personas y amigos que de una forma u otra han estado a nuestro lado y siempre nos han apoyado.

¡ Mil gracias a todos ellos!

Sonia Yesenia C.E.

Diego Deyvis Q.C

ÍNDICE GENERAL

Pág.

Área : Ingeniería de Software

Tema : Arquitectura de software.

FECHA DE SUSTENTACIÓN: 8 de enero del 2024

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ACRÓMIMOS

UML: UNIFIED MODELING LANGUAGE.

XP: EXTREMA O EXTREME PROGRAMMING.

ISO 9126: Es un estándar internacional para la evaluación de la calidad del software. Fue reemplazado en 2005 por el conjunto de normas SQuaRE, ISO 25000:2014, la cual desarrolla los mismos conceptos

RESUMEN

La presente investigación denominada “ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP D’MERCADO, JULIACA”, tuvo como propósito mejorar los servicios abarcando las áreas de ventas, marketing e inventario y almacén, para ello ha diseñado una arquitectura de software basada en microservicios que es accedida a través de la web. La metodología adoptada para el desarrollo del software basada en microservicios planteada fue el desarrollo de software ágil conocida como la programación extrema XP (extreme programming). El modelo de composición de microservicio fue realizado con BPMN, además se propuso un diseño arquitectónico de dicha composición de microservicios. La implementación se realizó del lado del servidor en PHP y el microservicio FrontEnd utilizó Vue.js y para la seguridad se utilizó JWT, el despliegue se hizo en un servidor PHP con soporte para API para cada microservicio. La información requerida para el desarrollo del sistema fue proporcionada por la empresa, los resultados de la evaluación del software elaborado nos indican que el diseño y la implementación de la arquitectura basada en microservicios logro mejorar los servicios prestados en el área de ventas, marketing e inventario y almacén., donde se concluye que la arquitectura de microservicios, para la implementación de una aplicación Web de la empresa GROUP D’MERCADO utilizando Kubernetes mejora significativamente las métricas de rendimiento, disponibilidad y tiempo de respuesta en un 100 % en comparación con los modelos monolíticos.

Palabras clave: Arquitectura de software, Microservicios, Aplicaciones Web.

ABSTRACT

The present research called “SOFTWARE ARCHITECTURE BASED ON MICROSERVICES FOR THE DEVELOPMENT OF WEB APPLICATIONS OF THE COMPANY GROUP D'MERCADO, JULIACA”, had the purpose of improving services covering the areas of sales, marketing and inventory and warehouse, for which it has designed a software architecture based on microservices that is accessed through the web. The methodology adopted for software development based on microservices was agile software development known as extreme programming XP (extreme programming). The microservice composition model was made with BPMN, and an architectural design of said microservice composition was also proposed. The implementation was done server-side in PHP and the FrontEnd microservice used Vue.js and for security JWT was used, the deployment was done on a PHP server with API support for each microservice. The information required for the development of the system was provided by the company, the results of the evaluation of the developed software indicate that the design and implementation of the architecture based on microservices managed to improve the services provided in the area of sales, marketing and inventory. and warehouse., where it is concluded that the microservices architecture, for the implementation of a Web application of the company GROUP D'MERCADO using Kubernetes, significantly improves the performance, availability and response time metrics by 100% compared to the models monolithic.

Keywords: Software architecture, Microservices, Web applications

CAPÍTULO I

INTRODUCCIÓN

El mundo cambia constantemente en términos de cómo se hacen negocios, cómo se realizan las operaciones y cómo se ejecutan los procesos. Internet se ha convertido en la fuerza impulsora y dominante detrás de las empresas y organizaciones modernas a escala global, y nuestro entorno local no es inmune a esta realidad. Por tanto, necesitamos cambiar la forma en que diseñamos y construimos los sistemas que soportan cada uno de estos procesos, y en la situación sanitaria actual, esta es una necesidad imprescindible para las organizaciones empresariales. En cualquier proyecto de software, los desarrolladores de software pretenden crear aplicaciones que serán relevantes y funcionales en un futuro próximo. Desafortunadamente, muchas aplicaciones evolucionan con el tiempo y tienden a volverse cada vez más complejas a medida que las organizaciones evolucionan para adaptarse a las necesidades del mercado. Los desarrolladores de software reconocen instintivamente la rigidez y vulnerabilidad de las aplicaciones que desarrollan. Por ejemplo, un pequeño cambio en la estructura de una aplicación puede tener efectos negativos que se extiendan a toda la aplicación y, por extensión, a otras partes de la organización. La complejidad y la necesidad de adaptarse a un mundo en constante cambio representan desafíos de desarrollo que requieren soluciones innovadoras y decisiones reflexivas durante la etapa de diseño arquitectónico. La arquitectura basada en microservicios representa una de las alternativas tecnológicas actuales, un paradigma centrado en dividir un sistema en múltiples servicios más pequeños e independientes relacionados con funciones comerciales específicas. El foco de este trabajo está en el desarrollo de software soportado en una arquitectura de microservicios que minimice la complejidad y facilite el mantenimiento.

Las secciones siguientes muestran los resultados de la investigación se presentan en primer lugar en el Capítulo I la exposición masa relevante de la literatura referente a los microservicios, además también dentro de este capítulo los antecedentes de investigación útiles para lograr los objetivos planteados en este.

Prosigue, en el capítulo II una descripción de la problemática y la formalización del problema de investigación, los objetivos e hipótesis de investigación correspondiente a la presente investigación.

Sigue en el capítulo III una descripción detallada de los métodos manejados para obtener los objetivos trazados.

El Capítulo IV está dedicado a revelar todos los resultados encontrados se muestra la implementación de la arquitectura y la aplicación desarrollada, así como la estimación correspondiente. Finalmente se presenta las conclusiones y recomendaciones.

1.1. PLANTEAMIENTO DEL PROBLEMA

GRUPO D'MERCADO es una empresa dedicada a la venta de calentadores solares de agua, sistemas solares y sus accesorios, así como todo producto relacionado con tuberías. Está ubicado en Juliaca y cuenta con tres sucursales en Puno, Espinal y otras ciudades. Recuperar información sobre el movimiento del inventario y las ventas se volvió difícil e inconsistente a medida que varios registros de ventas se acumulaban con el tiempo en cuadernos escritos a mano y no había información histórica de ventas. Al realizar una visita técnica, necesita buscar el esquema de su ubicación y la dirección en una libreta, por lo que necesita un sistema para administrar a los clientes para que puedan buscar y actualizar, y realizar ventas y descuentos según los productos que compran. Sus distintas divisiones comercializan calentadores de agua solares y sistemas de energía solar.

Otro aspecto del problema es que el procesamiento de los productos se vuelve muy difícil al estar disponibles en diferentes tamaños y calidades en tuberías, termostatos y flotadores auxiliares de baños termales, algo que los comerciantes y propietarios desconocen. Para recordar en qué almacén se encuentra dicho producto, el transporte a veces requiere tiempo y técnicos para configurarlo, por lo que se necesita un sistema para gestionar los productos y determinar cuántos productos hay en la sede y las sucursales según el tamaño y el color de cada almacén.

Además, se necesita un sistema de ventas para publicitar productos a través de Internet, con fotos, videos y ofertas, por lo que el propósito de este trabajo es mejorar este servicio y brindar una mejor atención al cliente.

1.2. FORMULACIÓN DEL PROBLEMA

Frente a la problemática descrita nos planteamos

¿Ayudará la implementación la arquitectura de software basada en microservicios para el desarrollo de aplicaciones web de la Empresa GROUP D'MERCADO, Juliaca?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. Objetivo general

Implementar una arquitectura de software basada en microservicios para el desarrollo de aplicaciones web de la empresa GROUP D'MERCADO, Juliaca.

1.3.2. Objetivos específicos

- Analizar, los requerimientos arquitectónicos del sistema de software.
- Diseñar el modelo de composición de microservicios, de la empresa GROUP D'MERCADO.

- Desarrollar la arquitectura basado en microservicios de la empresa GROUP D'MERCADO.

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

El sistema de arquitectura de software basada en microservicios para el desarrollo de aplicaciones web de la empresa GROUP D'MERCADO, Juliaca mejora significativamente la administración de sistema de ventas de energía solar.

1.5. JUSTIFICACIÓN DEL ESTUDIO

Mediante la arquitectura de software basada en microservicios, mejora la administración de los productos y bienes, el sistema de ventas del sistema de energía solar mediante la base de datos en la plataforma web basada en Cloud Computing, en la sede central De Juliaca y en sus sucursales, Puno, Espinar - Cusco.

La implementación de la arquitectura de software mediante el método XP asegura la adecuada coordinación de las áreas comerciales y técnicas, así como del almacén responsable de la instalación y mantenimiento de la energía solar en los hogares de los clientes, según el producto y el rubro. Los accesorios de productos y plomería relacionados con sistemas de energía solar y otros sistemas de energía también se administran y actualizan a través de PC y dispositivos móviles.

Donde los beneficiados serán los clientes, el Área Técnica y Administrativa de la empresa GROUP D`MERCADO, mediante la puntualidad en la instalación de energía solar a domicilio y la administración de modelos, tamaños y calidades del sistema de energía solar en almacén central y sucursales, el grupo de Mercado, tiene tiendas de termas solares, tiendas de iluminación, calefacción, bomba de agua solar y tiendas de gasfitería y tinacos para agua.

CAPÍTULO II

REVISIÓN DE LITERATURA

Se desarrolla en este capítulo la exposición el marco teórico referente a la temática de investigación y continua una segunda sección dedicado a los antecedentes de la investigación que expone las investigaciones realizadas referentes a la temática de la presente

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

2.1.1. Antecedentes Internacionales

A continuación, detallaremos los antecedentes que la presente investigación considerados para la elaboración del presente proyecto.

López (2017), La arquitectura genérica propuesta permite al equipo de desarrollo y operaciones tomar las mejores decisiones a la hora de implementar o migrar una funcionalidad para una aplicación web. Si bien, los microservicios son mayormente utilizados en una migración de un sistema monolítico que, en la creación de nuevos, el diseño no limita su aplicación para este último debido a que puede construirse microservicios de “grano grueso” (con más de una funcionalidad) para luego, conforme avanza el desarrollo del proyecto volver a dividirlo hasta llegar a un tamaño adecuado para cada microservicio. Universidad Técnica del Norte - Ecuador.

Xiao y Martina (2016), emprende una investigación crítica de los conceptos clave alrededor de SOA, API y Microservicios, identificando similitudes y diferencias entre ellos; llegando a la conclusión que se debe

aprovechar lo mejor de cada concepto para poseer la capacidad de responder a las necesidades del mundo digital.

Dragoni et al (2017), en su investigación presenta el actual estado de arte de los. Microservicios, describiendo los problemas abiertos y desafíos futuros respecto a los microservicios. En una de sus conclusiones menciona que debido a que la arquitectura de los microservicios es muy reciente, no se ha encontrado una colección suficientemente amplia de literatura sobre el terreno y se tiene el desafío en el aspecto de seguridad en la comunicación entre los microservicios.

2.1.2. Antecedentes nacionales

Herrera et al (2017), donde concluye que el implementar un sistema permite automatizar, reducir y mejorar los tiempos en los procesos sin perder información importante para la empresa, como son en los procesos de compras y ventas. Universidad autónoma del Perú.

Vargas y Quispe (2023), donde concluye que con la realización de un Sistema de Información web para la mejora de la Gestión del Área de Ventas se logra tener un mejor orden de la información del producto y una mejor atención al cliente muy rápido. se proyecta para mejorar a muchas Empresas Comerciales para optimizar sus ventas. Universidad Nacional de Trujillo.

2.1.3. Antecedentes Locales

Ruelas (2017), donde concluye que el modelo de composición de microservicios, basado en la arquitectura de microservicios, para la implementación de una aplicación Web de comercio electrónico utilizando Kubernetes funciona significativamente en comparación con un modelo

monolítico en un 104% con respecto a los indicadores de rendimiento, disponibilidad y tiempo de respuesta.

Chura (2015), Se mejoró el proceso de Ventas en la tienda Minimarket José Carlos observamos el 100% de los encuestados opinó que, si se mejoró el proceso de Ventas, además se puede observar que el tiempo para brindar información es de un minuto alcanzado el 100% de opiniones de los encuestados. De la Post-Test nos muestra el 100% de los encuestados opinaron que el sistema si mejora el control de productos por tanto podemos afirmar que el sistema informático implementado en la tienda Minimarket José Carlos si mejoro.

2.2. MARCO TEÓRICO

2.2.1. Ingeniería de Software

La ingeniería de software es el estudio práctica del conocimiento científico en proyecto y construcción de programas informáticos y en la preparación de legajo relacionados necesarios para su desarrollo, operación y mantenimiento. Además, conocido como desarrollo de software o preparación de software.

La IEEE Raddatz, define a la ingeniería de software como: “La aplicación de un enfoque sistemático, disciplinado y cuantitativo para el desarrollo, operación y mantenimiento de software; Es decir, la aplicación de la ingeniería al software”.

2.2.2. Proceso de Ingeniería de Software

La ingeniería de software define un proceso no rígido (Pressman, 2012) para la construcción de software, consta de una serie de tareas y actividades interrelacionadas. Especificación del software, donde se establecen los

requerimientos. en donde los requerimientos de entrada a través de actividades de transformación generan resultados. En síntesis, establece cuatro actividades fundamentales

- Especificaciones de software que especifican los requisitos.
- Diseño e implementación de software, es decir, la construcción del software en sí.
- Validación del software, comprobando su funcionalidad con énfasis en el cumplimiento de los requisitos especificados.
- Desarrollo de software, añadiendo nuevas funcionalidades. El cambio de software es la piedra angular del desarrollo de software.

2.2.3. Software

El software informático es un producto creado y mantenido en el tiempo por programadores profesionales. Incluye programas que se ejecutan en computadoras de todos los tamaños y arquitecturas, el contenido que muestra el programa de computadora cuando se ejecuta e información descriptiva tanto en formato impreso como virtual que cubre prácticamente todos los medios electrónicos. La ingeniería de software consta de un proceso, un conjunto de métodos (prácticas) y un conjunto de herramientas que permiten a los profesionales producir software informático de alta calidad. ¿Quién lo hizo? Los ingenieros de software crean y mantienen software que casi todas las personas en el mundo industrializado utilizan directa o indirectamente. ¿Porque es importante? El software es importante porque afecta casi todos los aspectos de nuestras vidas y ha invadido nuestros negocios, cultura y actividades diarias. La ingeniería de software es importante porque nos permite construir sistemas complejos con

calidad y en un tiempo razonable. ¿Cuáles son los pasos? El software informático se construye como cualquier producto exitoso, utilizando procesos ágiles y adaptables para lograr resultados de alta calidad que satisfagan las necesidades de los usuarios del producto. En estas etapas se utilizan técnicas de ingeniería de software. ¿Cuál es el producto final? Desde el punto de vista de un ingeniero de software, el producto final es la colección de programas, contenido (datos) y otros productos terminados que componen el software de computadora. Pero desde la perspectiva del usuario, el producto final es el mensaje definitivo que de alguna manera hace de su mundo un lugar mejor.

2.2.4. La naturaleza única de las webapps

Entre 1990 y 1995, los sitios web consistían en una serie de archivos de hipertexto vinculados que mostraban información mediante texto y gráficos limitados. Con el tiempo, el auge del HTML a través de herramientas de desarrollo (XML, Java) permitió a los desarrolladores web proporcionar potencia informática y contenido informativo.

> Nacieron los sistemas y aplicaciones basados en web (conocidos colectivamente como aplicaciones web). Hoy en día, las aplicaciones web se han convertido en herramientas informáticas sofisticadas que no sólo brindan funcionalidad individualizada a los usuarios finales, sino que también se integran con bases de datos corporativas y aplicaciones comerciales. Las aplicaciones web son una de varias categorías diferentes de software. Sin embargo, algunos sostienen que las aplicaciones web son diferentes. Se sugiere que los sistemas y aplicaciones basados en la web "incluyan una combinación de publicación impresa y desarrollo de software, marketing e

informática, comunicaciones internas y relaciones externas, y arte y tecnología". La mayoría de las aplicaciones web tienen los siguientes atributos:

a) **Uso intensivo de redes.** Una webapp reside en una red y debe atender las necesidades de una comunidad diversa de clientes. La red permite acceso y comunicaciones mundiales (por ejemplo, internet) o tiene acceso y comunicación limitados (por ejemplo, una intranet corporativa).
b) **Concurrencia.** Una gran cantidad de usuarios pueden acceder a una aplicación web simultáneamente. En muchos casos, los hábitos de uso de los usuarios finales varían mucho.
c) **Cargas impredecibles.** El número de usuarios de aplicaciones web cambia en varios órdenes de magnitud cada día. Cien personas podrían utilizarlo el lunes y 10.000 personas el jueves.
d) **Rendimiento.** Si un usuario de una aplicación web tiene que esperar demasiado (inicio de sesión, procesamiento del lado del servidor, capacitación e implementación del cliente), es posible que decida ir a otra parte.
e) **Disponibilidad.** Aunque no es razonable esperar una disponibilidad de 100%, es frecuente que los usuarios de webapps populares demanden acceso las 24 horas de los 365 días del año. Los usuarios en Australia o Asia quizá demanden acceso en horas en las que las aplicaciones internas de software tradicionales en Norteamérica no estén en línea por razones de mantenimiento.
f) **Orientadas a los datos.** La función principal de muchas webapp es el uso de hipermedios para presentar al usuario final contenido en forma de texto,

gráficas, audio y video. Además, las webapps se utilizan en forma común para acceder a información que existe en bases de datos que no son parte integral del ambiente basado en web (por ejemplo, comercio electrónico o aplicaciones financieras) (Pressman, 2012).

2.2.5. Arquitectura de Software

La arquitectura de software es una abstracción de alto nivel de un sistema de software. (Ran, 2001), (Zhang & Goddard, 2005) Por ejemplo, Zhang y Goddard definen la arquitectura de software como una abstracción de alto nivel que representa la estructura, el comportamiento y las propiedades no funcionales de los sistemas de software (Zhang & Goddard, 2005). Ver la arquitectura como una abstracción del sistema de alto nivel plantea una serie de preguntas. Por ejemplo, ¿qué es exactamente alto nivel y dónde está el límite entre la arquitectura y la funcionalidad de la aplicación? Si uno se enfoca en un componente particular de un sistema de software, ¿todavía hay arquitectura? ¿Cómo se sabe? Además, dado que en tal definición la arquitectura necesita abordar tanto los requisitos funcionales como no funcionales de alto nivel, ¿no se necesitarían los mismos elementos de requisitos para la arquitectura y la funcionalidad de la aplicación y dónde está el límite entre estos? ¿Cómo se abordan los requisitos no funcionales en niveles más bajos de granularidad si no hay una arquitectura en estos niveles más bajos de granularidad? Tenga en cuenta también que esta clase de definiciones identifica abstracciones de alto nivel de diferentes aspectos de un sistema de software, abstracciones estructurales y funcionales de alto nivel. Esto lleva a Bara-gry y Reed a preguntarse si existe un concepto único para la arquitectura de software (Baragry & Reed, 1998). Las diferentes abstracciones generalmente se modelan usando diferentes vistas en un solo modelo

arquitectónico como en, por ejemplo, el modelo Kruchten 4 + 1 View de una arquitectura de software (Kruchten, 1995), o usando diferentes modelos arquitectónicos como es compatible con IEEE Recommended Practice para la descripción arquitectónica de sistemas intensivos en software(«IEEE Recommended Practice for Architectural Description for Software-Intensive Systems», 2000).

La arquitectura de software es la infraestructura de software dentro de la cual se pueden especificar, desplegar y ejecutar los componentes de la aplicación que proporcionan funcionalidad al usuario.

2.2.6. Elementos de Arquitectura de Software

Se identificó los elementos de la arquitectura de software como:

- Conceptos básicos y restricciones dentro de los cuales se especifican los componentes de la aplicación que proporcionan la funcionalidad del usuario.
- Un conjunto de componentes arquitectónicos que abordan inquietudes técnicas.
- Los canales de integración con el medio ambiente, así como la infraestructura de integración interna para los componentes arquitectónicos y de aplicación, y
- Un conjunto de estrategias arquitectónicas que se utilizan para abordar de manera concreta los requisitos de calidad para el sistema de software.

2.2.7. Límite entre arquitectura y diseño de aplicación

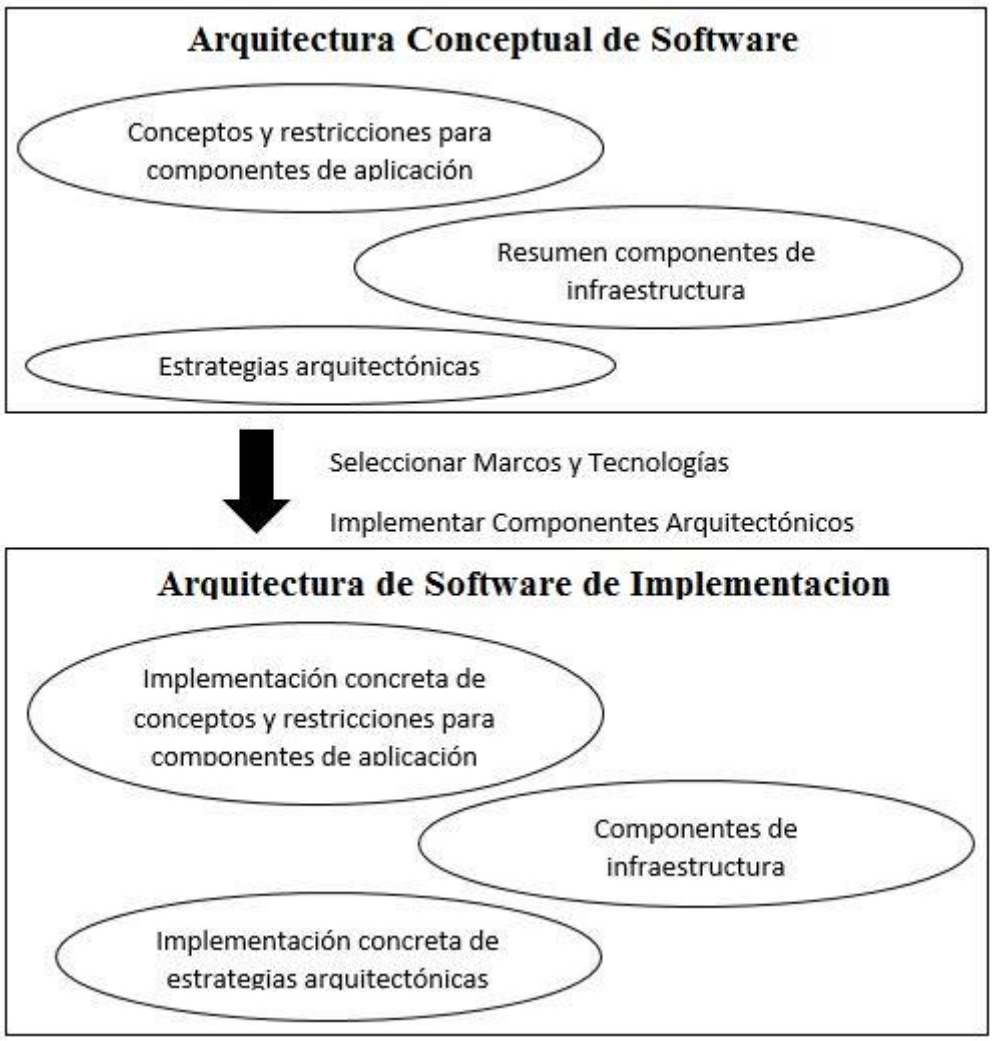
La arquitectura a un entorno dentro del cual se implementa y ejecuta la funcionalidad del usuario. Un elemento del sistema es parte de la arquitectura o parte de una aplicación, dependiendo de si contiene lógica que aborda los requisitos funcionales del problema / dominio de la aplicación o la lógica que implementa la funcionalidad arquitectónica y aborda las preocupaciones técnicas.

2.2.8. Arquitectura y componentes de aplicación en todos los niveles de granularidad

La arquitectura de software no se refiere a una infraestructura de alto nivel. Una arquitectura de software es una infraestructura de software completa en cualquier nivel de granularidad en la que se implementa la funcionalidad del usuario. Por lo tanto, existen diseños tanto funcionales como arquitectónicos que abarcan todos los niveles de granularidad. Desde el punto de vista de la definición de arquitectura de software propuesta en este artículo, estas arquitecturas y componentes de aplicación existen en diferentes niveles de granularidad, correspondiendo las primeras a aspectos de infraestructura y calidad y las segundas a la funcionalidad del software. solicitud. El análisis de la arquitectura de referencia revela que todos los elementos arquitectónicos pueden existir en diferentes niveles de granularidad. Esto incluye patrones arquitectónicos, conceptos, componentes, relaciones y estrategias (Solms, 2012).

Figura 1

Arquitectura conceptual versus implementación



Fuente. *Obtenido de (Solms, 2012).*

2.2.9. Arquitectura conceptual versus arquitectura de implementación

Generalmente se hace una distinción entre arquitecturas conceptuales y de implementación. La arquitectura conceptual introduce conceptos y restricciones para el desarrollo de aplicaciones, componentes arquitectónicos abstractos que abordan problemas de infraestructura y estrategias arquitectónicas que pueden usarse para abordar requisitos no funcionales. Para los componentes de la aplicación, impone algunas limitaciones de los componentes de la aplicación y proporciona implementaciones concretas de componentes de arquitectura abstracta de arquitecturas conceptuales e implementaciones concretas de estrategias arquitectónica **(Solms, 2012).**

2.2.10. Arquitectura de Plataforma de Software Tradicional

La arquitectura de la plataforma de software tradicional, como el paradigma arquitectónico de varios niveles, fue uno de los patrones de diseño más conocidos y dominantes que duró varios años. Incluso ahora, el patrón se utiliza activamente y es popular en varias empresas. Sin embargo, las empresas se están moviendo hacia la adopción de nuevos paradigmas para su nueva plataforma digital o la ampliación de la plataforma existente para minimizar la limitación de la arquitectura de plataforma de software monolítica tradicional.

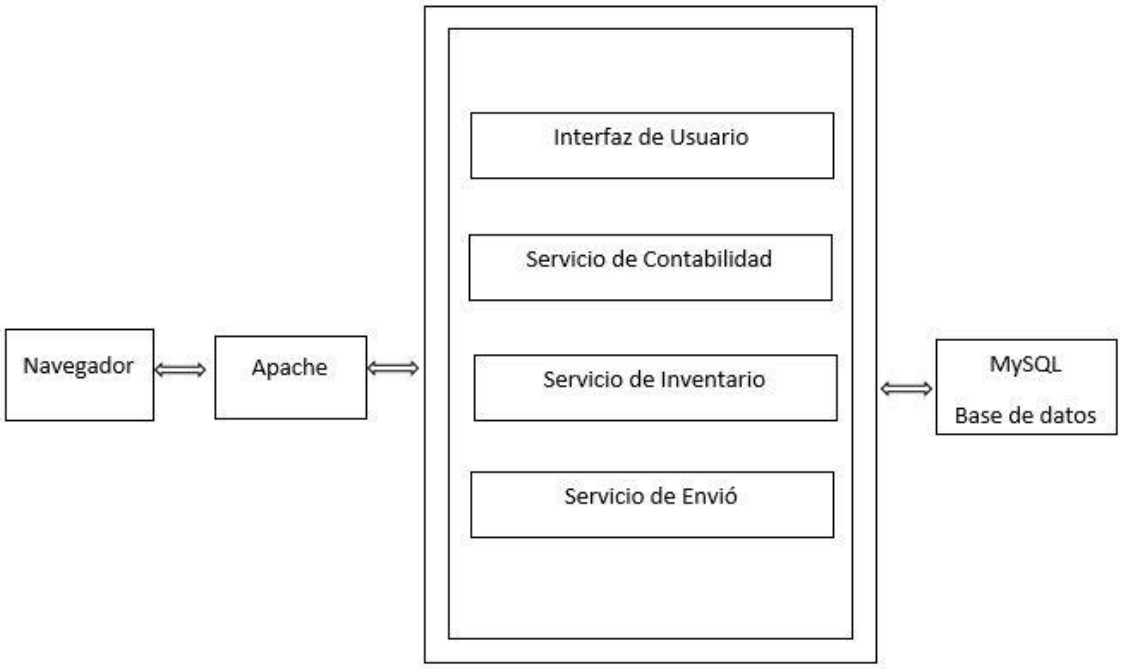
2.2.11. Arquitectura Monolítica

La arquitectura monolítica es un enfoque tradicional de diseño de software en el que todos los módulos o servicios de un sistema están contenidos en una aplicación que se implementa como un solo artefacto. La Figura 2 a continuación muestra un ejemplo de una aplicación monolítica. Contiene varios servicios de

lógica empresarial en una aplicación que se empaqueta y se implementa como un solo archivo.

Figura 2

Ejemplo de una arquitectura Monolítica.



Fuente. Obtenido *(Richardson, 2017)*.

El enfoque arquitectónico monolítico es el más adecuado para ciertos objetivos y particularmente conveniente para aplicaciones pequeñas. No solo es fácil de desarrollar, sino también fácil de probar. El despliegue de un monolito implica la creación de un único artefacto y su traslado al servidor, un proceso que es relativamente sencillo. El escalado se logra fácilmente mediante la replicación de la aplicación en varios servidores y el uso de un equilibrador de carga para la distribución del tráfico (Kharbuja, 2016).

Con el tiempo, a medida que la aplicación crece, la base del código se hace más grande y surgen desafíos de desarrollo, entre los que se encuentran:

Complejidad del software: debido a las cambiantes demandas y requisitos de los usuarios, el software cambia constantemente y su funcionalidad se expande. A medida que la base de código del monolito se hace más grande, la aplicación se vuelve cada vez más difícil de evolucionar y mantener debido a su complejidad (Dragoni et al., 2017).

Agilidad mínima: el desarrollo de un monolito involucra a todos los equipos que trabajan en la misma base de código. La introducción de una nueva función requiere que la tarea se distribuya a diferentes equipos y que las funciones se integren en el sistema. Además, la implementación de un pequeño cambio en la aplicación requiere que se implemente toda la aplicación. El proceso completo de agregar incluso la característica más pequeña y entregarla en producción puede llevar mucho tiempo. La entrega lenta y continua puede hacer que una empresa pierda una oportunidad de mercado (Kharbuja, 2016).

Fragilidad: en una aplicación monolítica, los módulos tienden a estar estrechamente acoplados y con el tiempo se acumulan interdependencias. La realización de cambios simples en un módulo de la aplicación puede desencadenar cambios en otro módulo dependiente y provocar la rotura de todo el sistema (Dragoni et al., 2017).

Escalabilidad limitada: Los monolitos suelen proporcionar muchos servicios dentro de una sola aplicación. Los desafíos surgen cuando algunos servicios tienen una gran demanda y es necesario ampliarlos. Las estrategias tradicionales para gestionar el aumento del tráfico en un monolito incluyen replicar aplicaciones entre servidores y utilizar equilibradores de carga para distribuir el tráfico. Debido a que todos los servicios están dentro de una

aplicación, se debe escalar toda la aplicación, lo que consume una gran cantidad de recursos del servidor, incluso si no es estrictamente necesario. (Kharbuja, 2016).

Bloqueo de tecnología: La decisión sobre qué tecnología utilizar para desarrollar una aplicación monolítica se toma al principio del análisis de requisitos. Los equipos están atrapados en un bloqueo técnico que los obliga a utilizar el mismo lenguaje y marco durante todo el ciclo de vida del proyecto. Las necesidades en constante evolución y las nuevas características pueden impedir que su pila de tecnología proporcione la solución óptima (Kharbuja, 2016).

2.2.12. Programación orientada a Objetos

En el campo del diseño de software, se han realizado extensas investigaciones para encontrar técnicas para gestionar el desarrollo de software a gran escala. Una extensa investigación sobre la arquitectura de software y la aplicación práctica de sus conceptos en el desarrollo de software condujo al surgimiento de la programación orientada a objetos (POO) en la década de 1980. El paradigma de diseño de programación orientada a objetos proporciona varios enfoques para convertir diseños de software en código utilizando un conjunto de soluciones persistentes, conceptos conocidos como patrones. Un ejemplo notable de un patrón de diseño arquitectónico OOP comúnmente utilizado por los desarrolladores para administrar software a gran escala es el patrón Modelo-Vista-Controlador (MVC) (Dragoni et al., 2017).**(Dragoni et al., 2017)**

2.2.13. Arquitectura Orientada a Servicios

El concepto de separación de preocupaciones es un principio de diseño que se enfatiza mucho en

OOP y ha ganado mucha consideración como un concepto profundamente recomendado para diseñar sistemas a gran escala. Como consecuencia, ha surgido el desarrollo basado en componentes, que proporciona una mayor capacidad de gestión y un fácil mantenimiento para sistemas complejos. La Arquitectura Orientada a Servicios (SOA) nació de una mezcla de los conceptos de POO y la creación de componentes de aplicaciones. En SOA, una aplicación se fragmenta en varias porciones acreditadas como servicios. Un servicio proporciona una funcionalidad a la que pueden acceder otros servicios a través de varios protocolos, como el Protocolo simple de acceso a objetos (SOAP). Se utiliza un sistema de comunicación como el Enterprise Service Bus (ESB) (Dragoni et al., 2017).

2.2.14. Nuevas demandas de mercado

El aumento de las demandas y expectativas de los consumidores ha obligado a las empresas a reorientarse para prosperar y sobrevivir en el mercado impulsado por el consumidor actual. En consecuencia, la adopción de la computación en la nube por parte de las empresas se ha vuelto inminente. La computación en la nube facilita la implementación de aplicaciones de una manera que puede escalar los recursos informáticos, permitir implementaciones en caliente y permitir la entrega continua. La necesidad de las empresas de tener sus aplicaciones ejecutándose en la nube se ha visto acelerada por la creciente demanda de eficiencia en las operaciones y alta disponibilidad. Por la misma razón, las empresas necesitan innovar lo más rápido posible y, por lo tanto, existe una mayor necesidad de implementación continua. Al adoptar la computación en la nube, la mayoría de las empresas intentan implementar aplicaciones monolíticas

que se han traducido en grandes desafíos y la imposibilidad de aprovechar al máximo la potencia de la computación en la nube (Goetsch, 2017).

2.2.15. Microservicios

La arquitectura de microservicios es una forma de desarrollar aplicaciones en el estilo de software o arquitectura que estructuran una aplicación como una colección de servicios débilmente acoplados, donde cada servicio se puede implementar de forma independiente, modular y pequeña, implementando capacidades comerciales (Newman, 2015). Cada servicio lleva a cabo un proceso único y se comunica con un mecánico ligero y bien definido. para cumplir un objetivo comercial (Familiar, 2015).

El microservicio es una variante de estilo arquitectónico SOA diseñada para minimizar las limitaciones existentes en aplicaciones monolíticas y de otro tipo basadas en arquitectura. Y mejorar el desarrollo de software utilizando marcos y tecnologías estándar.

Los microservicios dividen la gran aplicación monolítica en varios servicios más pequeños que permiten o mejoran el modularidad de la aplicación y hacen que sea más fácil de entender, desarrollar y probar la aplicación. Cada pequeño servicio tiene su propia base de datos y actúa como un servicio de componentes con lógica empresarial y función específicas que se pueden implementar de forma independiente en el servidor. La descomposición de la aplicación en un servicio de componentes individuales más pequeños permite al servicio seleccionar cualquier tecnología y marcos adecuados y desplegarlos individualmente en el servidor (Newman, 2015).

La arquitectura de microservicios no restringe la selección de tecnologías y marcos. Los desarrolladores pueden seleccionar cualquier tecnología emergente, plataforma y marcos de desarrollo de software según su suite y habilidades. Por lo tanto, un servicio se puede programar usando el lenguaje de programación PHP usando el marco sublime, mientras que otro servicio puede usar JavaScript con el marco node.js. De manera similar, como cada servicio se puede desarrollar, implementar y escalar de forma independiente sus respectivos servicios, el desarrollo paralelo es posible utilizando microservicios donde cada servicio puede ser controlado por pequeños equipos autónomos, lo que permite la entrega e implementación continuas.

2.2.15.1. Características de microservicios

Las principales características de los microservicios son:

a) Flexibilidad: un servicio o sistema es lo suficientemente flexible como para admitir todas las características necesarias para mantener la competitividad empresarial en el entorno empresarial dinámico.

b) Modularidad: cada servicio es independiente y se concentra en algunas funciones comerciales que contribuyen al comportamiento general del sistema en lugar de la funcionalidad completa de un solo servicio (Dragoni et al., 2017).

c) Evolución: Es posible agregar nuevas funcionalidades a cada servicio y mantenible. El servicio no debería afectar al otro servicio, incluso si el servicio está inactivo (Dragoni et al., 2017).

2.2.15.2. Arquitectura Monolítica Versus Microservicios

La arquitectura monolítica se ha explicado ampliamente en el capítulo anterior, detallando sus beneficios y advertencias. La arquitectura monolítica está estructurada de forma totalmente diferente a la arquitectura de microservicios en términos de aspectos tanto tecnológicos como organizativos. La siguiente tabla resume las diferencias más significativas entre las dos arquitecturas.

Tabla 1

Principales diferencias entre arquitecturas monolíticas y de microservicios

Software Monolítico	Microservicios
Una aplicación simple	Arreglo de muchos servicios pequeños con funcionalidades limitadas
Es necesaria toda la aplicación para ser desplegada	Los microservicios se pueden desplegar separadamente
Un almacén de datos para toda la aplicación	Cada microservicio tiene su propio almacén de base de datos
Comunicación al interior de la aplicación	Llamadas remotas, usualmente llamadas REST vía HTTP
Separación entre desarrolladores y Operarios	Cooperación entre los desarrolladores y los operarios para mantener las operaciones estables
El estado se encuentra en la aplicación externa en tiempo de ejecución	Los estados son almacenados centralizados, las instancias individuales son retiradas

2.2.15.3.SOA versus Microservicios

Desde un punto de vista de definición general, los microservicios y SOA (Arquitectura Orientada a Servicios) parecen similares. Ambos ponen énfasis en dividir grandes sistemas en servicios distribuidos. Los servicios se convierten en los componentes principales de las arquitecturas

y para la implementación del sistema y las funcionalidades comerciales. Desde esta perspectiva, es difícil distinguir entre ellos. Demostraremos las diferencias entre microservicios y SOA comparando las características de sus servicios y sus capacidades.

2.2.15.4. Adopción de Microservicios en las empresas

Según la Encuesta de desarrollo de aplicaciones de NGINX 2015, el porcentaje de organizaciones que utilizan o investigan microservicios fue del 68 por ciento. La Tabla 2 a continuación brinda los detalles sobre el resultado de la encuesta. Según LeanIX Microservices Survey 2017, la cantidad de porcentajes aumentó al 80 por ciento en solo 2 años y la mayoría de estas organizaciones tiene planeado intensificar el uso de microservicios en los próximos días.

Figura 3

Usos de microservicios por empresas

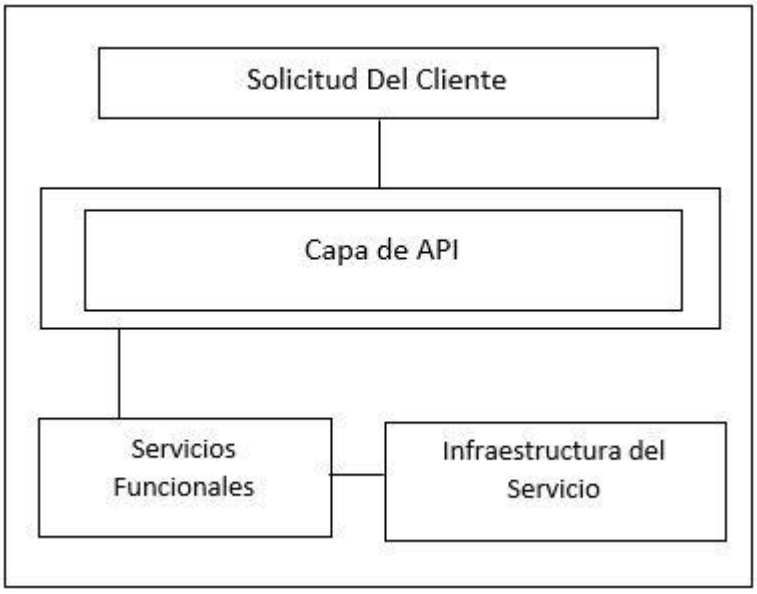
Fuente. Obtenido de (Survey, 2017).

2.2.16. Taxonomía de servicios

La taxonomía de servicios se relaciona con la clasificación de servicios en un patrón de arquitectura. Hay varias formas de clasificar los servicios. Los servicios se pueden clasificar de acuerdo con las tareas que realizan en la arquitectura, por ejemplo, algunos servicios implementan funcionalidades comerciales mientras que otros implementan funcionalidades no comerciales como la seguridad o el registro (Richards, 2016).

Figura 4

Taxonomía de servicios en la arquitectura de microservicios



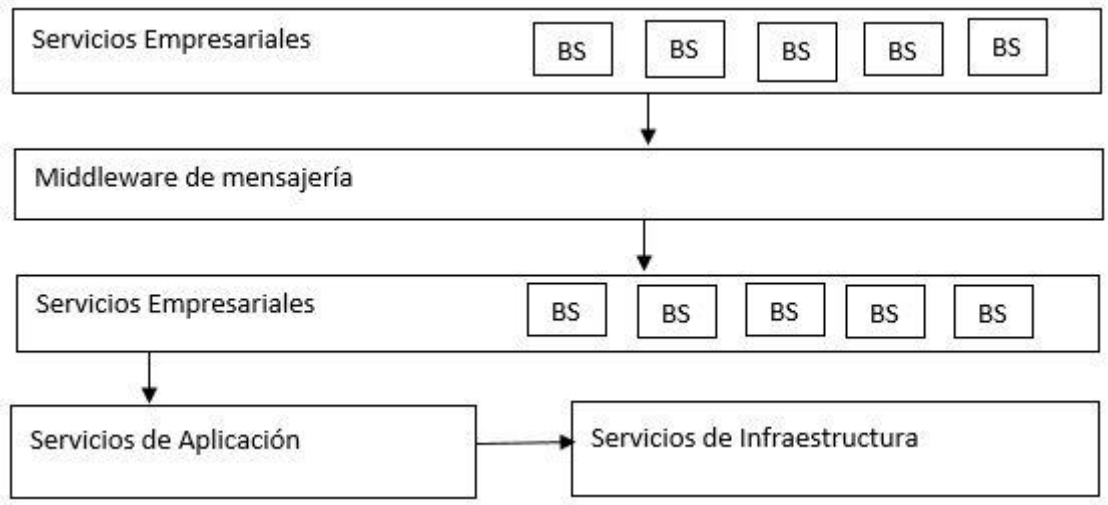
Fuente. Obtenido de (Richards, 2016).

La taxonomía de servicios de la arquitectura de microservicios se limita a solo dos clasificaciones de servicios, a saber, los servicios funcionales y los servicios de infraestructura. Los servicios funcionales consisten en servicios que implementan y respaldan funciones u operaciones comerciales particulares. Por otro lado, los servicios de infraestructura realizan tareas de soporte como autorización, autenticación, monitoreo y registro. Importante para esta

comparación es el hecho de que los servicios de infraestructura en microservicios no son accesibles fuera de la arquitectura y solo se comparten internamente entre servicios. Los servicios funcionales son accesibles al mundo exterior a través de una implementación de API (Richards, 2016).

Figura 5

Taxonomía de servicios en SOA



Fuente. Obtenido de (Richards, 2016).

La taxonomía de servicios en la arquitectura SOA es formal y clasifica distintamente sus servicios en cuatro tipos, como se muestra en la Figura anterior. Los Servicios Empresariales son servicios de alto nivel que definen las principales operaciones comerciales. Su representación es generalmente a través de XML, Business Process Execution Language (BPEL) o Web Services Definition Language (WSDL). Los servicios empresariales son servicios específicos pero concretos que implementan la funcionalidad empresarial definida por los servicios empresariales y, por lo general, se comparten entre otros servicios de la arquitectura.

Los servicios empresariales dependen de la infraestructura y los servicios de aplicaciones para completar las solicitudes. Los servicios de aplicaciones son

servicios detallados que implementan funciones comerciales que son muy específicas en el contexto de la aplicación. Se puede acceder a los servicios empresariales a través de una interfaz de usuario o mediante un servicio empresarial. El servicio de infraestructura, al igual que en la arquitectura de microservicio, se utiliza para implementar tareas de soporte no funcionales como la auditoría y el monitoreo (Richards, 2016).

2.2.17. Compartir componentes

SOA está construido fundamentalmente para maximizar el uso compartido de componentes mediante el uso de servicios compartidos de nivel empresarial. Si bien el uso compartido de componentes en SOA ayuda a reducir la duplicación de funcionalidades, conduce a un acoplamiento estrecho entre los componentes, lo que aumenta los riesgos asociados con la realización de cambios. En arquitecturas distribuidas, compartir algunos servicios siempre será inevitable, un ejemplo son los servicios de infraestructura. Sin embargo, mientras SOA intenta compartir tanto como sea posible, los microservicios intentan minimizar el uso compartido de componentes utilizando el concepto de "contexto limitado". Por lo tanto, los servicios son autosuficientes y contienen dependencias mínimas (Richards, 2016).

2.2.18. Beneficios de los microservicios

Los beneficios de utilizar la arquitectura de microservicios durante el desarrollo de software son significativamente más fuertes que la arquitectura monolítica existente. Debido a sus importantes características y beneficios, algunas grandes empresas como Amazon, eBay y Netflix están utilizando la

arquitectura de microservicios para su negocio empresarial. Algunos de los beneficios clave descubiertos hasta ahora se mencionan a continuación:

a) Escalabilidad: el escalado se puede hacer solo a aquellos servicios que necesitan escalado sin afectar al resto de los servicios. A diferencia de los servicios grandes y monolíticos, los servicios más pequeños se pueden ejecutar en hardware más pequeño y menos potente (Alpers et al., 2015).

b) Límites de módulo sólidos: cada servicio es pequeño y se centra en la capacidad de una sola empresa. Los límites del servicio se convierten en el mamparo obvio, que es el concepto clave en la ingeniería de resiliencia. A diferencia del sistema monolítico, si alguno de los componentes falla, todo el sistema se detendrá. Sin embargo, en microservicios, si un servicio falla y no afectará al resto o a todo el sistema, ese servicio puede aislarse y el resto del sistema puede continuar el trabajo.

c) Desarrollo e implementación independientes: cada servicio se puede desarrollar e implementar de forma independiente en cualquier servidor.

d) Diversidad tecnológica: a diferencia del servicio monolítico, el microservicio facilita al desarrollador seleccionar la tecnología, las herramientas y los marcos en función de sus habilidades y voluntad. No se limitará a marcos o lenguajes de programación únicos.

e) Simplicidad y facilidad de mantenimiento: la aplicación de microservicio es pequeña y se centra en la función empresarial única. Dado que es pequeño, es más fácil para un nuevo desarrollador para comprender la funcionalidad de un servicio y mantener la corrección de errores.

f) Gestión de datos descentralizada: cada servicio tiene su propia base de datos y la gestión de datos eficaz se puede realizar fácilmente utilizando microservicios (Lewis, James; Fowler, 2014).

g) Capacidad de composición: los microservicios brindan la oportunidad de reutilizar la funcionalidad de la manera más diferente posible para diferentes propósitos. La reutilización de la funcionalidad es la promesa clave del sistema distribuido y la arquitectura orientada a servicios (*Monolith Vs Microservices: Which is the Best Option for You? | Webdesigner Depot Webdesigner Depot » Blog Archive*, s. f.).

2.2.19. Granularidad del servicio

La granularidad del servicio entre las dos arquitecturas es significativamente diferente. Como sugiere el prefijo "micro" en microservicios, los servicios en la arquitectura son pequeños y detallados. Son servicios de un solo propósito y tienen como objetivo hacer una sola cosa y hacerlo bien. Su alcance se limita a la implementación de una única función comercial. Por otro lado, el tamaño de los servicios SOA varía desde servicios de aplicaciones detallados hasta servicios empresariales grandes y detallados. No es inusual encontrarse con un servicio SOA que representa un producto de gran empresa o incluso un subsistema completo. SOA depende de varios servicios para ejecutar una solicitud comercial individual (Richards, 2016).

2.2.20. Inconvenientes de los Microservicios

a) La complejidad de un sistema distribuido: cada servicio es ahora un servicio independiente que necesita manejar cuidadosamente la solicitud que viaja entre los módulos. La complejidad puede aumentar debido a la

latencia de la red en llamadas remotas, tolerancia a fallas, control de versiones, serialización de mensajes, etc. (Alpers et al., 2015).

b) Complejidad de la gestión de la base de datos: Varias bases de datos que residen en diferentes servicios y plataformas pueden crear dificultades en la gestión de transacciones. La recuperación de datos pertenecientes a varios servicios puede resultar difícil (*Monolith Vs Microservices: Which is the Best Option for You? | Webdesigner Depot Webdesigner Depot » Blog Archive*, s. f.).

c) Dificultades en la implementación: a diferencia del servicio monolítico, los microservicios que tienen varios servicios pueden tener complejidad durante la implementación. Cada servicio debe implementarse individualmente y puede estar en un servidor diferente.

d) Pruebas: antes de comenzar las pruebas, cada servicio dependiente debe estar operativo. Las pruebas pueden ser difíciles o imposibles a menos que todo el servicio dependiente sea completamente funcional y esté disponible (*Monolith Vs Microservices: Which is the Best Option for You? | Webdesigner Depot Webdesigner Depot » Blog Archive*, s. f.).

e) Coherencia eventual: Una limitación clave existente en el sistema de distribución es mantener una coherencia sólida (*The Strengths and Weaknesses of Microservices*, s. f.). A diferencia de otros sistemas distribuidos y SOA, existen problemas para mantener una coherencia sólida en los microservicios hasta cierto punto. Sin embargo, los microservicios utilizan una metodología basada en eventos para minimizar el problema de la coherencia.

2.2.21. Metodología de Desarrollo de Software

Metodología ágil En febrero de 2001, después de una conferencia celebrada en Utah, EE. UU., se acuñó el término "ágil" y se aplicó al desarrollo de software. A esta reunión asistieron 17 expertos de la industria del software, entre ellos algunos creadores o promotores de técnicas de software. El objetivo es delinear valores y principios que permitan a los equipos desarrollar software rápidamente y responder a los cambios que puedan ocurrir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas (Pressman, 2009).

Esta reunión condujo a la creación de Agile Alliance3, una organización sin fines de lucro dedicada a promover el concepto de desarrollo de software ágil y ayudar a las organizaciones a adoptar estos conceptos. El punto de partida es el Manifiesto Ágil, un documento que resume la filosofía “Agile” (Canós & Letelier, 2012).

2.2.21.1. Manifiesto Ágil

Según el manifiesto se valora:

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.**
- Las personas son un factor clave en el éxito de los proyectos de software. Formar un buen equipo es más importante que crear un entorno. Con demasiada frecuencia, el error es crear el entorno primero y esperar que el equipo se adapte automáticamente.
- **Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.**

Desarrollar software que actúa más que conseguir una humana documentación. La norma a perseguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

- **La colaboración con el cliente más que la negociación de un contrato.**

Las personas son un factor clave en el éxito de los proyectos de software. Formar un buen equipo es más importante que crear un entorno. Con demasiada frecuencia, el error es crear el entorno primero y esperar que el equipo se adapte automáticamente.

- **Responder a los cambios más que seguir estrictamente un plan.**

La capacidad de responder a los cambios que puedan ocurrir a lo largo del proyecto (en requisitos, tecnología, equipos, etc.) también determina su éxito o fracaso. Por lo tanto, la planificación no debe ser rígida, sino flexible y abierta. Los valores anteriores inspiraron los doce principios de la declaración. Estas son las características que distinguen los procesos operativos de los procesos tradicionales. Los dos primeros principios son universales y resumen gran parte de la ética ágil. El resto tiene que ver con el proceso y equipo de desarrollo a seguir, los objetivos a cumplir y su organización. El principio es:

- La primera prioridad es satisfacer las necesidades de los clientes mediante la entrega oportuna y continua de software que cree valor.

- Bienvenido el cambio. Adopte el cambio para brindar a los clientes una ventaja competitiva.

- Entregar software con frecuencia con plazos de entrega de semanas a meses y mantener el tiempo entre entregas lo más corto posible.

- Profesionales y desarrolladores deben trabajar juntos durante todo el proyecto.

- Proyectos basados en el desarrollo proactivo de la personalidad. Bríndeles el entorno y el apoyo que necesitan y confíe en ellos para hacer el trabajo.

- El diálogo cara a cara es la forma más eficiente y eficaz de proporcionar información en el equipo de desarrollo.

- El software que funciona es la principal medida del progreso.

- Los procesos ágiles contribuyen al desarrollo sostenible. Los anunciantes, desarrolladores y usuarios deben poder mantener una paz duradera.

- La atención constante a la calidad técnica y al buen diseño mejora la flexibilidad. - La simplicidad es fundamental.

- La mejor arquitectura, requisitos y diseño provienen de equipos autoorganizados.

- El equipo reflexiona constantemente sobre cómo ser más eficiente y adapta su comportamiento en consecuencia.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. LUGAR DE ESTUDIO

La Empresa GROUP D´ MERCADO. Está ubicado en el distrito de Juliaca, provincia de San Román, del departamento de Puno, exactamente en Jr. Cahuide con el Jr. Ramón Castilla. Tiene también sucursales en el Distrito de Juliaca Jr. Tumbes Nº 1278, en Puno Av. Floral Nº 395 y otra sucursal en la Provincia de Espinar Cusco. Anexo B.

3.2. POBLACIÓN Y MUESTRA DE INVESTIGACIÓN

3.2.1. Población

La población fue constituida, por la cantidad de peticiones realizadas en el periodo de evaluación de un mes (30 días).

3.2.2. Muestra

El diseño muestral utilizado es el método muestral no probabilístico en la empresa GROUP D´ MERCADO. La muestra se obtuvo por conveniencia, un total de 20 peticiones en el periodo de evaluación.

3.3. MÉTODO DE RECOLECCIÓN DE DATOS

La información para este estudio fue recopilada directamente por ingenieros profesionales a través de observaciones, entrevistas directas con gerentes clave y empleados responsables de las respectivas áreas de la empresa, y datos secundarios como formularios circulares y documentos de ventas. (Formato de llamada de cliente, mensaje

activo, formato de llamada adjunta, etc.) todo ello obtenido de la sede central y sucursales. De la empresa del que se hizo el análisis documental.

Para la recolección de datos para el diseño del sistema se utilizó las historias de usuario, instrumento recomendado para el desarrollo de sistemas de información según metodología de ingeniera de software.

3.4. MÉTODO DE INVESTIGACIÓN

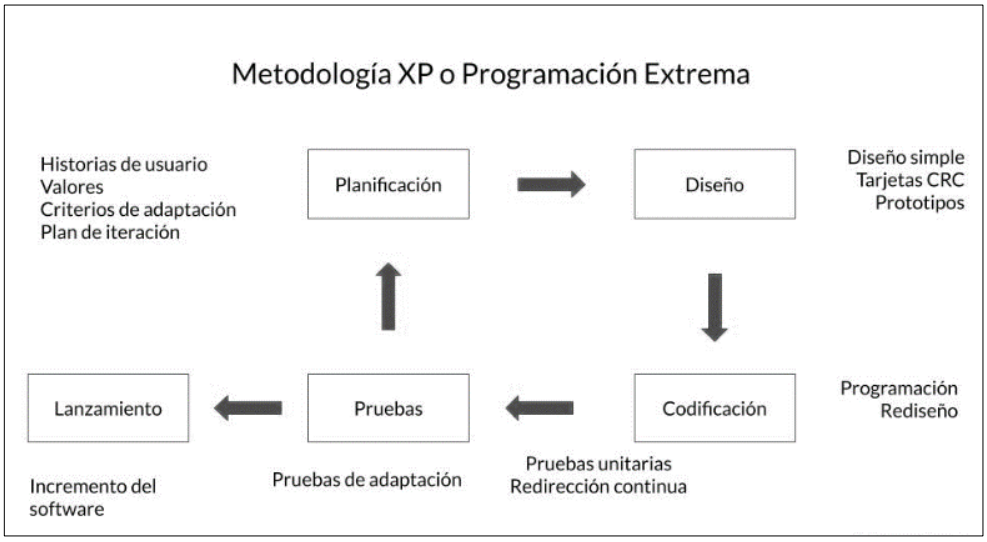
Este estudio pertenece al tipo de experimento técnico porque manipula directamente las variables independientes, el modelo de composición de microservicios para medir el efecto sobre las variables dependientes, la implementación de aplicaciones web de comercio electrónico, el propósito de utilizar este método es sacar conclusiones. . y resumir los resultados de la investigación de manera cuantitativa.

3.4.1. Métodos empleados

Metodología XP La metodología ágil es un método de gestión de proyectos que utiliza ciclos de desarrollo cortos para centrarse en la mejora continua en el desarrollo de un producto o servicio en lugar de gestionar el proyecto en sí. El método XP (Xtreme Programming), o método de Programación Extrema, es una metodología de desarrollo de software flexible que se centra en la programación de soluciones de software.

Figura 6

Fases de la metodología XP



Fuente. Sánchez (2007).

Sánchez (2007) menciona que el ciclo de vida de un proyecto XP es flexible y dinámico y considera las siguientes fases que se muestra en la Figura 3 y se detallan a continuación.

En la fase de planificación se definen historias de usuario que describen las características que debe tener el sistema de software, pero con menos atención al detalle; Estas historias de usuario constan de tres o cuatro líneas y están escritas en un lenguaje no técnico para que el equipo de desarrollo de software pueda entenderlas y recordarlas.

En la fase de diseño del método XP, se recomienda que el diseño sea simple, directo y lo menos complicado posible para lograr un diseño que sea fácil de entender e implementar, y en el proceso tomará menos tiempo y esfuerzo desarrollarlo. El objetivo de la fase de codificación es establecer estándares de codificación para que el código sea coherente, fácil de leer y modificable, lo que ayuda a los desarrolladores a comprender los requisitos. La programación en pares

es un método que caracteriza el enfoque XP. Cada par de programadores escribe su propio código y luego lo integra.

La fase de prueba es uno de los pilares del método XP y consiste en el uso de pruebas para verificar la funcionalidad del software implementado.

3.5. DISEÑO ESTADÍSTICO A UTILIZAR

En este trabajo se ha realizado un análisis descriptivo en la evaluación de sistemas de información y una comparación de las hipótesis de investigación mediante pruebas de hipótesis de diferencia de medias, donde se pudo observar el tiempo de venta del producto y la calidad del software utilizado. Cada uno de los métodos desarrollados se utilizan indicadores ISO 9126 en todas las etapas.

3.6. DESCRIPCIÓN DE MÉTODOS POR OBJETIVOS ESPECÍFICOS

3.6.1. Objetivos específicos 2: Diseñar el modelo de composición de microservicios de la Empresa GROUP D'MERCADO

Modelo BPMN de la composición de servicios

Se utiliza la notación del modelo de proceso de negocios BMPN para especificar la composición de los servicios prestados por la empresa GROUP D'MERCADO.

Diseño arquitectónico de la composición de microservicios

El diseño arquitectónico permitirá especificar gráficamente la lógica del modelo propuesto para el sistema de la empresa GROUP D'MERCADO.

Diseño de la seguridad para acceder a los recursos de microservicios

Se utiliza el diagrama de seguridad de la composición de los microservicios.

3.6.2. Objetivos específicos 1: Analizar los requerimientos arquitectónicos del sistema de software de la Empresa GROUP D'MERCADO,

Para realizar el Análisis de los requerimientos arquitectónicos del sistema propuesto para la Empresa GROUP D'MERCADO se utilizó:

Modelado del proceso de negocio

El modelado de los procesos correspondientes al negocio se utiliza el Modelo y Notación de Procesos de Negocio en inglés Business Process Model and Notation (BPMN), para el modelado de los procesos de negocio y que constituye un estándar en la industria, es en un formato de flujo de trabajo (workflow).

Historias de usuario

Esta técnica recaba los requerimientos de software, mediante un descripción concisa y breve del comportamiento deseado del software, se recomienda utilizar cada historia por cada característica del Software considerada como principal (Canós & Letelier, 2012)..

Tareas de ingeniería

Las tareas de ingeniería son producto de la transformación de las historias de usuario en tareas, estas describen que actividades son las que se realizarán para cada historia de usuario.

Casos de uso

Se utilizan los casos de uso como herramienta para la identificación de los requerimientos de software necesarios para el desarrollo del sistema.

3.6.3. Objetivos específicos 3: Implementar la arquitectura basada en microservicios de la Empresa GROUP D'MERCADO

Implementación de los microservicios

La implementación de los microservicios está basada en el paradigma de la Programación Orientada a Objetos.

Despliegue de los microservicios

Los microservicios implementados son desplegados sobre API basado en su API de alto nivel para proporcionar contenedores livianos que ejecutan procesos de manera aislada

Evaluación de calidad del sistema implementado basado en microservicios

La evaluación de la calidad de los microservicios implementados se realiza en términos de granularidad, acoplamiento, cohesión, complejidad y reusabilidad.

3.6.4. Objetivos específicos 4: Evaluar el diseño de la arquitectura basada en microservicios de la empresa GROUP D'MERCADO

La evaluación del diseño de la arquitectura propuesta para los microservicios es comparada con la arquitectura monolítica en los siguientes indicadores:

- Evaluación del Tiempo de respuesta
- Evaluación de la Disponibilidad
- Evaluación del Rendimiento

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

El presente capítulo muestra los resultados obtenidos en función de los objetivos planteados para el desarrollo de la arquitectura de software basada en microservicios para el desarrollo de aplicaciones web de la empresa GROUP D'MERCADO, JULIACA.

4.1. RESULTADOS CONFORME AL OBJETIVO ESPECÍFICO I: REQUERIMIENTOS ARQUITECTÓNICOS DEL SISTEMA DE SOFTWARE DE LA EMPRESA GROUP D'MERCADO

Los resultados están constituidos por el modelado del proceso de negocio, las historias de usuario, tareas de ingeniería y los casos de uso identificados

4.1.1. Modelado del proceso de negocio

Figura 7

Modelamiento del proceso del negocio 1

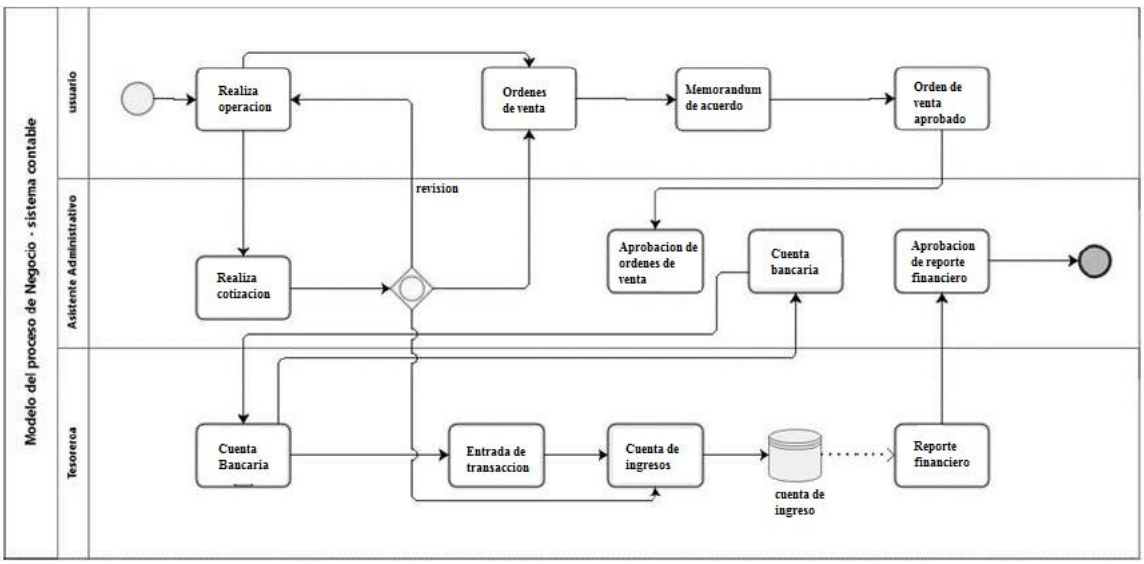
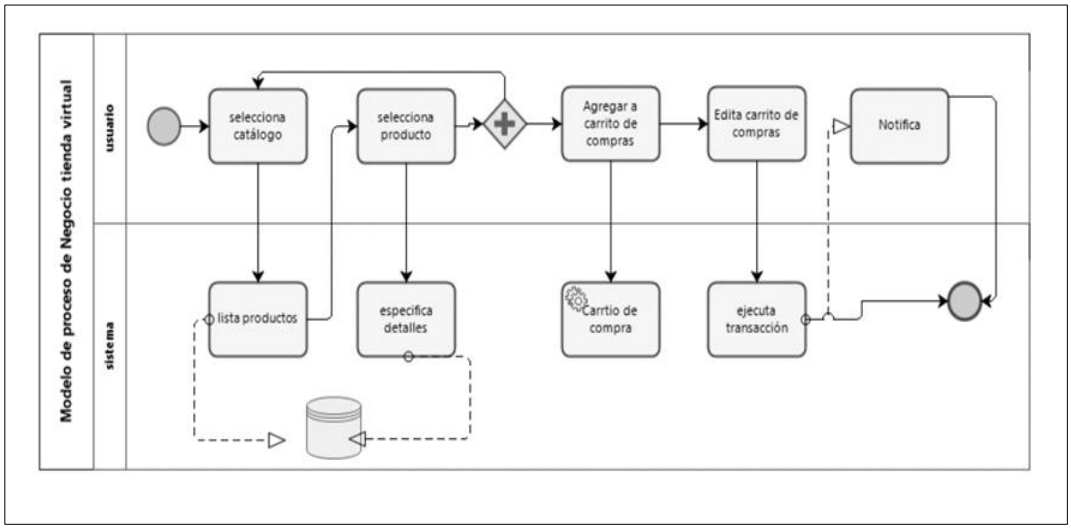


Figura 8

Modelamiento del proceso del negocio 2.



4.1.2. Historias de usuario

Las tareas de ingeniería se identificaron a partir de las historias de usuario, se identificó que el cliente requiere que la aplicación web, pueda ser accedida en cualquier momento prioriza la seguridad del sistema todas estas reflejadas en la historia de usuarios, sobre las transacciones, además el usuario desea realizar ventas en línea que involucra el uso de dinero por lo que se remarca la necesidad de la seguridad de la aplicación web.

A continuación, se especifican las historias de usuarios:

Tabla 2

Historia de Usuario 1

Número 01	**Historia de Usuario**
Nombre de historia:	Disponibilidad del Sistema
Como	Usuario
Quiero:	Acceder a la aplicación en cualquier momento del día por 24 horas por todos los días del año
Para:	Poder realizar ventas a través de la aplicación Web

Tabla 3

Historia de Usuario 2

Número 02	**Historia de Usuario**
Nombre de historia:	Seguridad de las transacciones
Como	Usuario
Quiero:	Una plataforma digital con seguridad en las operaciones transaccionales
Para:	Poder realizar transacciones comerciales digitales

Tabla 4

Historia de Usuario 3

Número 03	**Historia de Usuario**
Nombre de historia:	Rapidez del Sistema
Como	Usuario
Quiero:	Que la velocidad de la aplicación sea rápida
Para:	Organizar las actividades de forma eficiente

Tabla 5

Historia de Usuario 4

Número 04	**Historia de Usuario**
Nombre de historia:	Funciones nuevas
Como	Usuario
Quiero:	Que la aplicación web pueda actualizarse con nuevas funcionalidades.
Para:	Poder manejar la lista de clientes

Tabla 6

Historia de Usuario 5

Número 05	**Historia de Usuario**
Nombre de historia:	Atención concurrente
Como	Usuario
Quiero:	Que el sistema no se cuelgue por la concurrencia de varios usuarios al mismo tiempo, que funcione bien
Para:	Poder acceder las nuevas funcionalidades como promociones

4.1.3. Tareas de ingeniería

Tabla 7

Tarea de ingeniería 1

Tarea de ingeniería	
Numero de tarea: 01	Número de historia: 01
Nombre de la tarea: Disponibilidad prioritaria del sistema	
Tipo de tarea: No funcional	Puntos estimados: 5
Fecha de inicio: 10/05/2023	Fecha fin: 25/05/2023
Programador responsable: Programador 1	
Descripción:	
La aplicación Web deberá de funcionar 24 horas por los 7 días de la semana durante todo el año.	

Tabla 8

Tarea de ingeniería 2

Tarea de ingeniería	
Numero de tarea: 02	Número de historia: 02
Nombre de la tarea: Sistema de seguridad	
Tipo de tarea: No funcional	Puntos estimados: 5
Fecha de inicio: 26/05/2023	Fecha fin: 24/06/2023
Programador responsable: Programador 2	
Descripción	
La seguridad de la autenticación de los usuarios del sistema debe ser realizados utilizando certificados digitales.	

Tabla 9

Tarea de ingeniería 3

Tarea de ingeniería	
Numero de tarea: 03	Número de historia: 03
Nombre de la tarea: Escalabilidad	
Tipo de tarea: No funcional	Puntos estimados: 6
Fecha de inicio: 01/07/2023	Fecha fin: 27/07/2023
Programador responsable: Programador2	
Descripción	
El sistema está en la capacidad de agregar componentes nuevos al sistema adaptarse a las necedades futuras de los usuarios.	

Tabla 10

Tarea de ingeniería 4

Tarea de ingeniería	
Numero de tarea: 04	Número de historia: 04
Nombre de la tarea: Eficiencia del sistema	
Tipo de tarea: No funcional	Puntos estimados: 6
Fecha de inicio: 02/08/2023	Fecha fin: 18/08/2023
Programador responsable: Programador 2	
Descripción	
El sistema Web podrá responder oportunamente las peticiones realizadas por los usuarios en un tiempo máximo de 3 segundos.	

Tabla 11

Tarea de ingeniería 5

Tarea de ingeniería	
Numero de tarea: 05	Número de historia: 05
Nombre de la tarea: Rendimiento	
Tipo de tarea: No funcional	Puntos estimados: 6
Fecha de inicio: 02/09/2023	Fecha fin: 15/09/2023
Programador responsable: Programador 1	
Descripción	
El sistema aceptará peticiones y serán atendidas de forma eficiente hasta un total de 20 personas por segundo.	

4.1.4. Casos de uso

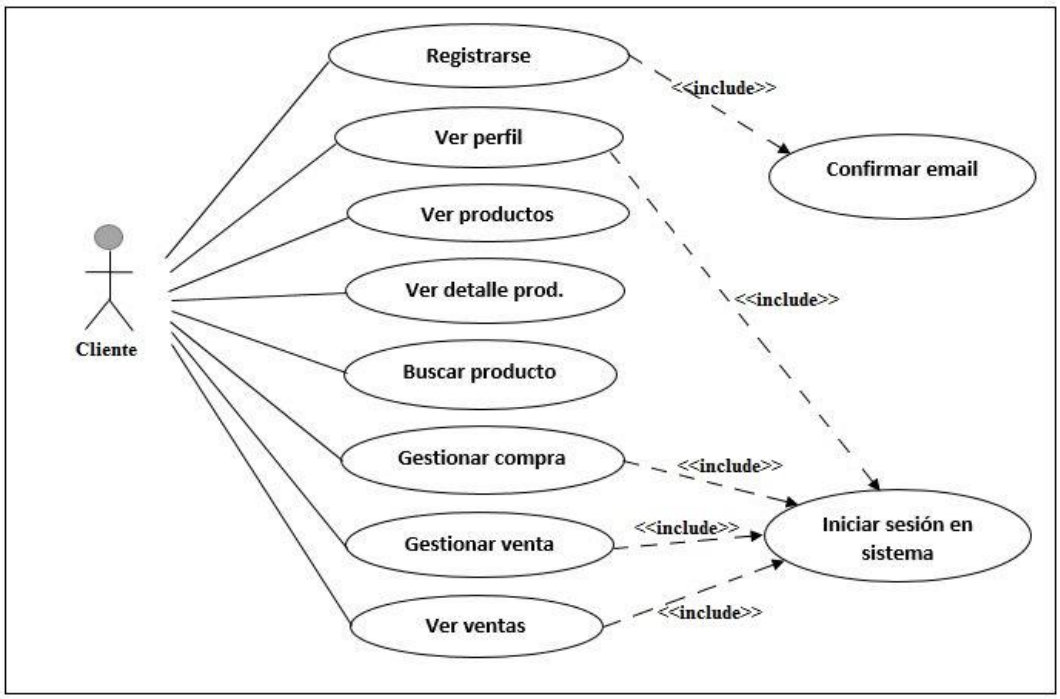
Los casos de uso proporcionan un resumen gráfico de los requerimientos funcionales para llevar a cabo el proceso de implementación.

Los casos de uso del sistema en línea representan las actividades principales como son ingresar al sistema realizar su registro, visualizar productos, buscar productos, realizar venta de producto seleccionado, comprar y ver las compras realizadas.

Para la identificación de los microservicios se utilizó como referencia los casos de uso presentados.

Figura 9

Requerimiento funcional mediante casos de usos



La Tabla 15: resume los microservicios identificados mediante funcionalidades que fueron expresadas en las tareas que la plataforma o aplicación debe realizar, se identificó el valor de la tarea, escalabilidad y entidad de datos para los microservicios usuarios, productora y ventas.

Tabla 12

Resumen de requerimientos identificados

Microservicios	Tareas	Valor de la tarea	Escalabilidad	Entidad de datos
Usuarios	Registro	1	1	Usuarios
	Inicio de sesión,	2	2	
	confirmar email	1	1	
	ver email	1	1	
Productos	Ver lista productos,	2	3	Productos
	ver detalle de producto,	2	3	
	buscar producto	2	3	
Ventas	Seleccionar producto,	3	3	Ventas
	Comprar producto,	3	3	
	cantidad	2	2	

La arquitectura que se propone está basada en atributos de calidad estos son, eficiencia, rendimiento, disponibilidad, escalabilidad y seguridad.

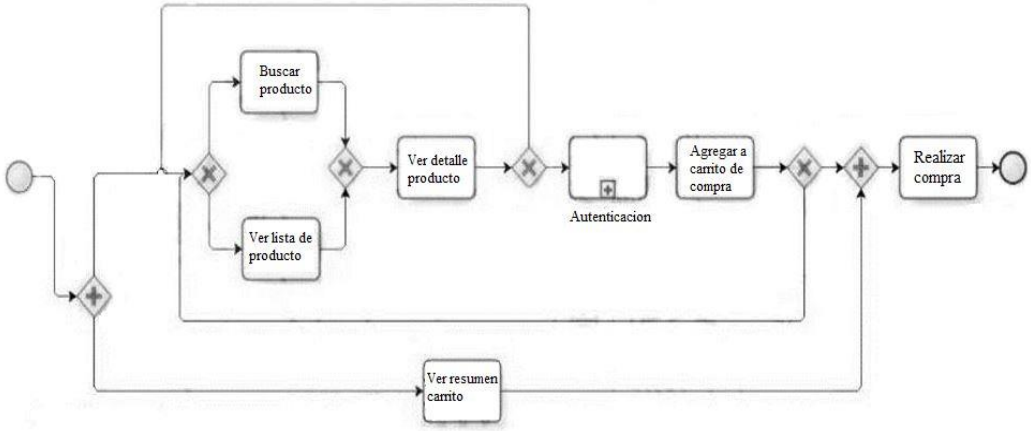
4.2. RESULTADOS CONFORME AL OBJETIVOS ESPECÍFICOS 2: MODELO DE COMPOSICIÓN DE MICROSERVICIOS DE LA EMPRESA GROUP D'MERCADO

4.2.1. Modelo BPMN de la composición de servicios

Se han identificado siete tareas principales para la composición del servicio de ventas en línea, es decir tienda virtual

Figura 10

Modelo BPMN



Sobre estos microservicios se elaboró la tabla a continuación según el tipo de petición y rutas finales.

Tabla 13

Petición y rutas finales

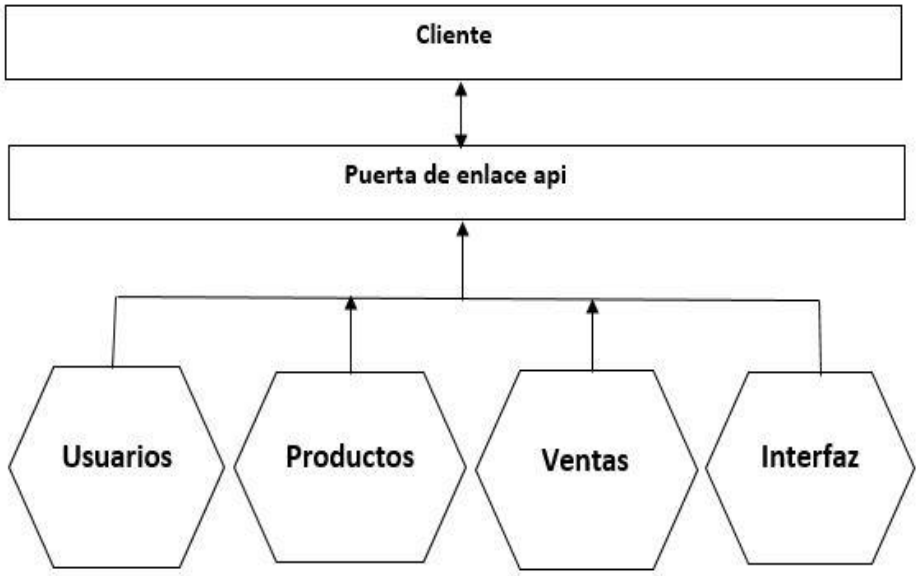
Microservicios	Tareas	Petición	Destino
Usuarios	Inicio de sesión	POST	/Inicio Sesión
productos	Ver lista productos	GET	/productos
	Ver detalle de producto	GET	/productos/detalle/var
frontend	Buscar	POST	/productos/buscar
ventas	Agregar a lista de compras	POST	/ventas/listar
	Compra	POST	/ventas/validar
Front end	Cantidad de artículos en lista	GET	/ventas/productos

4.2.2. Diseño arquitectónico de la composición de microservicios

El modelo arquitectónico de la composición de microservicios consta de 4 componentes principales lógicos.

Figura 11

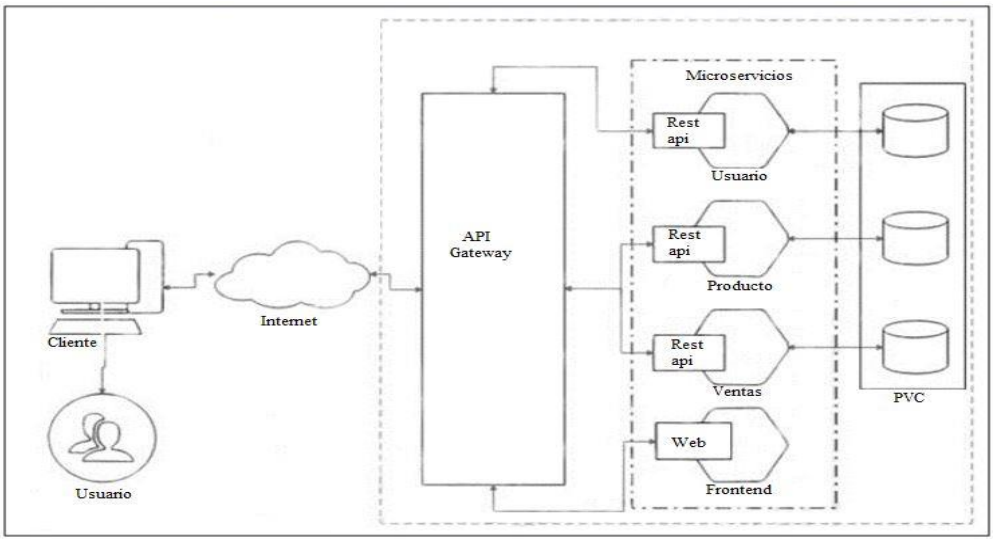
Modelo arquitectónico de microservicios



La arquitectura propuesta soporta consultas independientes donde cada uno de los componentes físicos esta implementado independientemente constituyendo un microservicio. La solución para la persistencia a la base de datos es realizada por un PVC (Persistent Volume Claim). El acceso a los microservicios es realizado por un API Gateway. Los detalles de la relación mencionada se detallan en la siguiente figura.

Figura 12

Accesos a los microservicios



4.3. RESULTADOS CONFORME AL OBJETIVOS ESPECÍFICOS 3: IMPLEMENTACIÓN DE LA ARQUITECTURA BASADA EN MICROSERVICIOS DE LA EMPRESA GROUP D'MERCADO

Implementación de aplicaciones web basadas en microservicios utilizando el espacio de trabajo Echo para implementar de forma independiente microservicios de usuarios, productos y ventas y utilizando el lenguaje de programación PHP para la codificación del lado del servidor. Los microservicios front-end se implementan utilizando el excelente banco de trabajo. El servicio JWT se utiliza para proteger las comunicaciones de microservicios.

4.3.1. Evaluación de calidad del sistema implementado basado en microservicios

Los resultados de la evaluación de los atributos propuestos: granularidad, cohesión, reusabilidad y complejidad; atributos de calidad de la implementación de los microservicios se detallan a continuación:

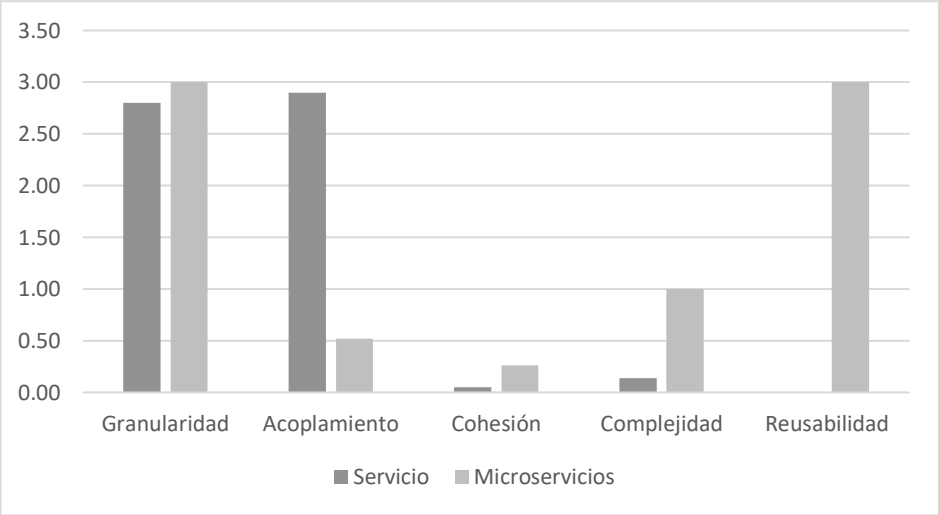
Tabla 14

Evaluación de calidad del sistema implementado basado en microservicios

Atributo	Servicio	Microservicios
Granularidad	2.80	3.00
Acoplamiento	2.90	0.52
Cohesión	0.05	0.26
Complejidad	0.14	1.00
Reusabilidad	0.00	3.00

Figura 13

Evaluación de la calidad del sistema



Los resultados muestran con respecto a la granularidad es superior en la arquitectura que utiliza los microservicios, con respecto al acoplamiento se aprecia un acoplamiento altamente inferior al modelo monolítico demostrando la independencia de los microservicios. Respecto a la cohesión de los microservicios implementados en el modelo se aprecia una mayor cohesión en esta arquitectura, La complejidad es mayor también en los microservicios y con respecto a la última

métrica la reusabilidad la independencia de los microservicios permite una mayor reusabilidad de estos.

4.4. OBJETIVOS ESPECÍFICOS 4: EVALUACIÓN DEL DISEÑO DE LA ARQUITECTURA BASADA EN MICROSERVICIOS DE LA EMPRESA GROUP D'MERCADO

4.4.1. Evaluación del Tiempo de respuesta

La siguiente tabla muestra los resultados de la medición del tiempo de respuesta en milisegundos para el modelo de composición de microservicios propuesto y el modelo monolítico, donde el tiempo de respuesta promedio de diferentes microservicios y el tiempo de respuesta promedio del modelo monolítico.

Tabla 15

Tiempo de respuesta del modelo de composición de Microservicios propuesto y la arquitectura monolítica

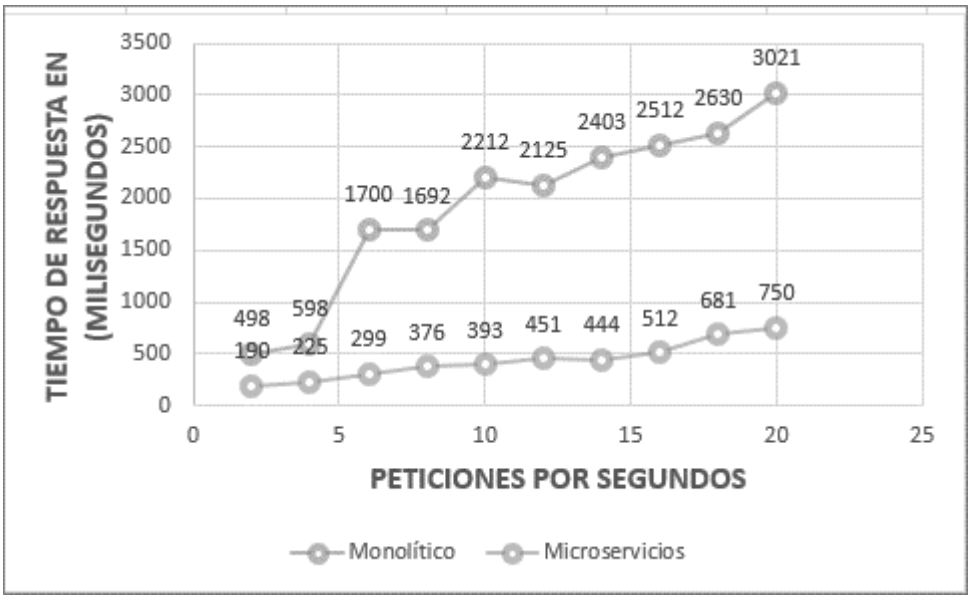
Tiempo de respuesta(ms)			
peticiones/segundo	**Monolítico**	**Modelo propuesto**	**% de mejora**
2	498	190	62
4	598	225	62
6	1700	299	82
8	1692	376	78
10	2212	393	82
12	2125	451	79
14	2403	444	82
16	2512	512	80
18	2630	681	74
20	3021	750	75

Los resultados muestran que, en comparación con el modelo monolítico, el modelo de composición de microservicios propuesto mejora el tiempo de respuesta en un 75,17% cuando se emiten 20 solicitudes por segundo. Puede ver que las solicitudes en la arquitectura de microservicios están descentralizadas. cálculo, pero debido al ancho de banda que nos dio el cliente, este está limitado a 20 solicitudes por minuto, y las simulaciones se realizan desde máquinas con servidores donde están agrupados, permitiendo realizar tareas más rápido de lo que requeriría un solo modelo sólido. . menos tiempo porque es autónomo y tiene una función definida, por lo que tiene menos tareas que un modelo monolítico. Además, estos microservicios residen en sus propios hosts, es decir, se ejecutan en sus propios procesos.

Tiempo De Respuesta Del Modelo De Composición De Microservicios Propuesto Y La Arquitectura Monolítica

Figura 14

Tiempo de respuesta del modelo



En la figura se muestra el tiempo de respuesta del modelo de composición de Microservicios propuesto y la arquitectura monolítica.

4.4.2. Evaluación de la Disponibilidad

Proporciona un resumen de las métricas de disponibilidad que se relacionan con la capacidad de una aplicación web para brindar un servicio 24 horas al día, 7 días a la semana a los usuarios que acceden a ella, es decir. Las 24 horas del día, los 7 días de la semana, en diferentes regiones.

Una hora. Esta tabla muestra un resumen de las métricas de disponibilidad para el modelo de composición de microservicios y el modelo monolítico, donde la disponibilidad se calcula como la cantidad de solicitudes a la aplicación web y los errores que ocurrieron durante esas solicitudes. Se realizaron pruebas en aplicaciones web de comercio electrónico desarrolladas en ambos modelos.

Tabla 16

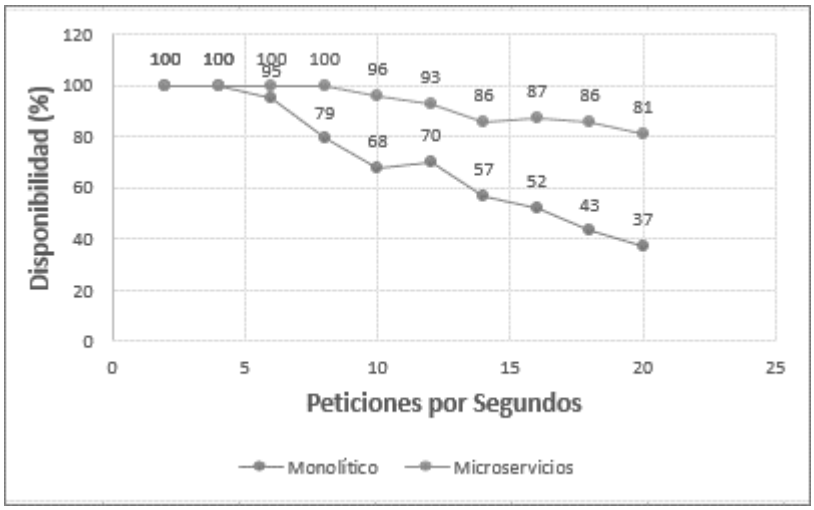
Disponibilidad del modelo de composición de Microservicios propuesto y la arquitectura monolítica

Disponibilidad (%)			
peticiones/ segundo	**Monolítico**	**Modelo propuesto**	**% de Mejora**
2	100	100	0
4	100	100	0
6	95	100	5
8	79	100	21
10	68	96	28
12	70	93	23
14	57	86	29
16	52	87	35
18	43	86	43
20	37	81	44

Como se puede observar en los resultados, inicialmente ambos modelos respondieron con un 100% de disponibilidad, pero cuando se realizaron 6 o más solicitudes por segundo se observó una menor disponibilidad del modelo monolítico, perdiendo incluso un 63% de disponibilidad. En este caso el modelo emite 20 solicitudes por segundo, y también se puede observar que en el modelo de microservicio al emitir 20 solicitudes por segundo solo se pierde el 20% de las solicitudes, como se puede observar en la curva, el modelo de microservicio muestra más alta disponibilidad que el modelo general; esto significa que la aplicación web de GROUP D'MERCADO realiza continuamente sus funciones y si ocurre un error al solicitar un servicio, el servicio será copiado a otro servidor, asegurando una alta disponibilidad del servicio. Las aplicaciones web se diferencian de los modelos monolíticos en que hay una pérdida de disponibilidad en caso de falla debido a la necesidad de resolver problemas del servidor que interfieren con su funcionalidad.

Figura 15

Disposición del modelo de composición



En la figura se muestra la disponibilidad del modelo de composición de microservicios propuesto y la arquitectura monolítica.

4.4.3. Evaluación del Rendimiento

La siguiente tabla muestra un resumen del rendimiento del modelo de composición de microservicios propuesto y el modelo monolítico, donde el rendimiento se calcula evaluando 60 segundos para obtener los datos expuestos, y la prueba de carga se realiza bajo demanda en el segundo.

Tabla 17

Rendimiento del modelo de composición de microservicios propuesto y la arquitectura monolítica

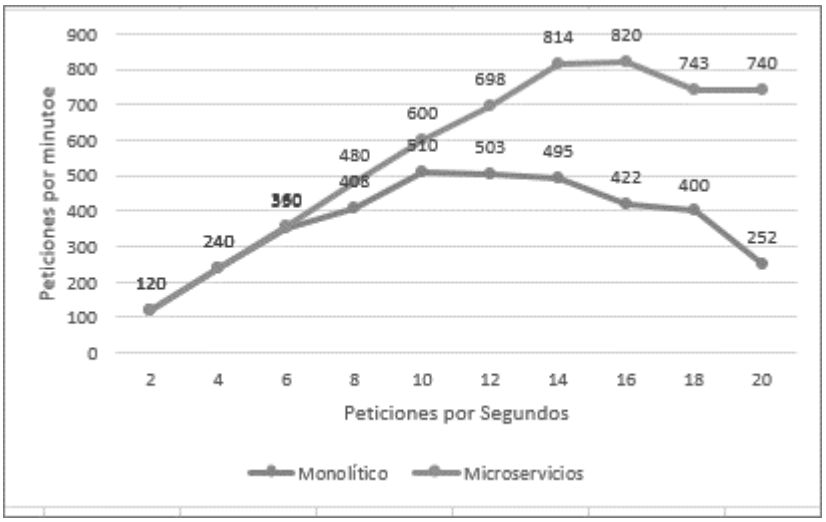
Rendimiento(peticiones/minuto)			% de mejora
peticiones/segundo	**Monolítico**	**Modelo propuesto**	
2	120	120	0
4	240	240	0
6	350	360	3
8	408	480	18
10	510	600	18
12	503	698	39
14	495	814	64
16	422	820	94
18	400	743	86
20	252	740	194

La siguiente figura muestra que ambos modelos lograron el mejor rendimiento durante el tiempo estimado, pero a partir de 6 solicitudes por segundo, el modelo base experimentó una degradación del rendimiento, que se debe a su arquitectura y degradación del rendimiento a partir de ese punto. . del modelo; Respecto al modelo de composición de microservicios, se puede observar que su rendimiento baja de 12 solicitudes a una por segundo, una de las limitaciones observadas es el ancho de banda, ya que no se observa saturación en términos de procesamiento y memoria.

Rendimiento del modelo de composición de microservicios propuesto y la arquitectura monolítica.

Figura 16

Rendimiento del modelo de composición de microservicios propuesto y la arquitectura monolítica



Se pudo apreciar en la implementación que al igual que Ruelas (2017), es conveniente un modelo basado en microservicios que un modelo molítico. Similar a López (2017), nuestra propuesta utiliza la cualidad de granularidad haciendo su diseño más versátil. También se pudo observar que al igual que Xiao (2016), es necesario rescata conceptos como SOA, API y Microservicios, para aprovechar lo mejor de cada propuesta y mejorar la capacidad de respuesta. Con respecto a Dragoni (2017), los esfuerzos orientados a la seguridad en la comunicación entre los microservicios es un aspecto que necesita de mayor investigación. Al igual que los hallazgos de Chura (2015), Quispe Hernández y Vargas Chavarri (2007), Huayanca Quispe y Huaman Varas (2017), la propuesta en microservicios permite automatizar, reducir y mejorar la efectividad en la ejecución de los procesos sobre todos los de compra y venta.

4.5. PRUEBA DE HIPÓTESIS

Como método de prueba de hipótesis se utilizó la denominada prueba t-student para probar las hipótesis del modelo de composición de microservicios, la implementación de la aplicación web GROUP D'MERCADO, el proceso se realizó mediante SPSS, para lo cual se plantearon las siguientes hipótesis presentar:

Con respecto al tiempo de respuesta se ha formulado las siguientes hipótesis estadísticas:

- H_0: El tiempo de respuesta de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes es mayor o igual que el modelo monolítico.
- H_a: El tiempo de respuesta de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes, es menor al modelo monolítico.

Tabla 18

Prueba de muestras independientes con respecto al tiempo de respuesta

		Prueba de Levene de calidad de varianzas		prueba t para la igualdad de medias		
		F	Sig.	t	gl	Sig. (bilateral)
tiempo respuesta	Se asumen varianzas iguales	11.340	.003	5.570	18	.000
	No se asumen varianzas iguales			5.570	9.838	.000

En la tabla 21, se muestra que Sig = 0.000, es decir Sig <0.05 entonces se rechaza la hipótesis nula y se acepta la hipótesis alterna que afirma que el tiempo de respuesta de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes es menor que el implementado con el modelo monolítico.

Con respecto disponibilidad se ha formulado las siguientes hipótesis estadísticas:

- HO: La disponibilidad de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes es menor o igual que el modelo monolítico.
- Ha: La disponibilidad de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes es mayor al modelo monolítico.

Tabla 2

Prueba de muestras independientes con respecto a la disponibilidad

		Prueba de Levene de calidad de varianzas		**prueba t para la igualdad de medias**		
		F	**Sig.**	**T**	**gl**	**Sig. (bilateral)**
disponibilidad	Se asumen varianzas iguales	9.885	.006	-2.970	18	.008
	No se asumen varianzas iguales			-2.970	10.785	.013

Con respecto al rendimiento se ha formulado las siguientes hipótesis estadísticas:

- Ho: El rendimiento de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes, es menor o igual que el modelo monolítico.
- Ha: El rendimiento de la aplicación Web de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes, es mayor al modelo monolítico.

Tabla 20

Prueba de muestras independientes con respecto al rendimiento

		Prueba de Levene de calidad de varianzas		**prueba t para la igualdad de medias**		
		F	**Sig.**	**T**	**gl**	**Sig. (bilateral)**
rendimiento	Se asumen varianzas iguales	5.983	.025	-2.152	18	.045
	No se asumen varianzas iguales			-2.152	13.535	.050

En la siguiente tabla, se muestra que Sig = 0.045, es decir Sig <0.05 entonces la regla de la decisión rechaza la hipótesis nula y se acepta la hipótesis alterna, que afirma que el rendimiento de la aplicación Web de de la empresa GROUP D'MERCADO basado en el modelo de composición de microservicios utilizando Kubernetes es mayor que el implementado bajo el modelo monolítico.

Tabla Resumen de la Prueba de Hipotesis:

		Prueba de Levene de calidad de varianzas		prueba t para la igualdad de medias		
		F	**Sig.**	**t**	**gl**	**Sig. (bilateral)**
tiempo respuesta	Se asumen varianzas iguales	11.34	0.003	5.57	18	0
disponibilidad	Se asumen varianzas iguales	9.885	0.006	-2.97	18	0.008
rendimiento	Se asumen varianzas iguales	5.983	0.025	-2.152	18	0.045

V. CONCLUSIONES

PRIMERA: El modelo de composición de microservicios, basado en la arquitectura de microservicios, para la implementación de una aplicación Web de la empresa GROUP D'MERCADO utilizando Kubernetes funciona significativamente en comparación con un modelo monolítico en un 100% con respecto a los indicadores de rendimiento, disponibilidad y tiempo de respuesta

SEGUNDA: El análisis de los requerimientos arquitectónicos permitió comprender cómo la aplicación Web de la empresa GROUP D'MERCADO funciono significativamente, para lo cual se identificó cinco requerimientos que son la alta disponibilidad, seguridad, rendimiento, escalabilidad, eficiencia y rendimiento. Los requerimientos funcionales del sistema se identificaron a partir del mapa de proceso del negocio de la empresa GROUP D'MERCADO a través del modelo BPMN.

TERCERA: El diseño del modelo propuesto se realizó mediante el diseño de una arquitectura lógica y física, identificándose cuatro microservicios que son productos, ventas, usuarios y Frontend, estos microservicios fueron orquestados a través de una API Gateway, que actuó como una puerta de comunicación entre los diferentes microservicios y el orquestador.

CUARTA: GROUP D'MERCADO introdujo un modelo de composición de microservicios en su aplicación web, facilitando la implementación de cada microservicio., utilizando la tecnología de forma independiente.

Se utilizó el lenguaje de programación PHP para la implementación del lado del servidor y el marco de trabajo sublime, la construcción del lado del cliente se ha vuelto más fácil. De manera similar, se utilizó el servicio de autenticación JWT para habilitar la seguridad al acceder a los microservicios. El otro es Kubernetes, una herramienta tecnológica que facilita la orquestación de contenedores en los que se implementan microservicios.

VI. RECOMENDACIONES

PRIMERA: Se prefiere el modelo de composición de servicios propuesto basado en microservicios al modelo de arquitectura monolítica., para el desarrollo de aplicaciones Web de la empresa GROUP D'MERCADO, ya que los usuarios esperan de esta alta disponibilidad, mejor rendimiento y mayor tiempo de respuesta.

SEGUNDA: Recomendamos que la fase de análisis de la metodología XP se centre en identificar los requisitos arquitectónicos, ya que esto garantiza una funcionalidad de alta calidad del sistema de software.

TERCERA: La fase de diseño de la metodología XP recomienda centrarse en diseñar una arquitectura lógica y física esquemática para abstraer el comportamiento de los componentes. que conformarán un sistema software. Así mismo se recomienda el modelo BPMN y el álgebra para el diseño de composición de servicios.

CUARTA: Recomendamos implementar microservicios de forma independiente y validarlos en paralelo utilizando métricas de calidad. en los atributos de calidad de granularidad, cohesión, complejidad, reusabilidad y acoplamiento.

VII. REFERENCIAS BIBLIOGRÁFICAS

Alvarado Zambrano, E. A. (2023). *Aplicación web para el servicio de trámites académicos de la UNACH usando una arquitectura basada en microservicios.* Riobamba, Universidad Nacional de Chimborazo.

Arboleda Cola, C. A. (2017). *Propuesta metodológica para migración de sistemas web con arquitectura monolítica hacia una arquitectura basada en microservicios.* Quito, 2017.

Arcila Díaz, J. C. (2021). Arquitectura de software basada en microservicios para mejorar la disponibilidad de historias clínicas electrónicas odontológicas, Chiclayo–Lambayeque, 2020.

Avila, P. Q., Mora, M. G. Z., & Sacoto, A. S. Q. (2020). Arquitectura de microservicios para compras en línea: caso de uso" ala orden". *Polo del Conocimiento: Revista científico-profesional*, *5*(1), 151-162.

Calderón-Gómez, H., Navarro-Marín, F., Gómez-Pulido, J. M., Castillo-Sequera, J. L., Garcés-Jiménez, A., Polo-Luque, M.-L., . . . Vargas-Lombardo, M. (2019). Desarrollo de aplicaciones eHealth basadas en microservicios en una arquitectura de Cloud. *Revista Ibérica de Sistemas e Tecnologias de Informação*(E23), 81-93.

Calla Charrez, V. (2000). Modelo de pronóstico para número de egresos hospitalarios usando redes neuronales - Hospital Regional Manuel Núñez Butrón - Puno, 2009 - 2018 (Universidad Nacional del Altiplano Puno). Retrieved from http://repositorio.unap.edu.pe/handle/UNAP/11474

Chura (2015), Sistema De Administración De Ventas De una Micro y Pequeña Empresa en Azángaro, 2015

De la Cruz May, M. E., Castellanos, L. J. R., Jiménez, M. B. J. C., Coello, M. E. A. P., & Alí, B. Diseño de un Sistema de Gestión Energética Para Redes Eléctricas Inteligentes

Dragoni, N., Giallorenzo, S., Lluch Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, Today, and Tomorrow. Obtenido de SpringerLink: https://link.springer.com/chapter/10.1007/978-3-319-67425-4_12

Gómez Gallego, J. P. (2017). Definición de una arquitectura para la transformación de software centralizado a software basado en microservicios en el ámbito web.

Hernández, L. M. A., Romero, V. A. P., González, S. A. S., & Rodríguez, J. A. V. (2021). Arquitectura REST para el desarrollo de aplicaciones web empresariales. *Revista Electrónica sobre Tecnología, Educación y Sociedad, 8*(15).

Herrera Salazar, J., Huaman Varas, J., & Huayanca Quispe, C. (2017). *Desarrollo e implementación de un sistema de información para mejorar los procesos de compras y ventas en la empresa Humaju.* Tesis de Licenciatura, Universidad Autónoma del Perú]. http://repositorio ….

Loayza, W. J. T., & Rodríguez, F. S. (2022). Herramienta para el modelado y generación de código de Arquitecturas de Software basadas en Microservicios y Diseño guiado por el dominio (DDD). *Revista peruana de computación y sistemas, 4*(2), 3-14.

López, D., & Maya, E. (2017). Arquitectura de software basada en microservicios para desarrollo de aplicaciones web.

López Hinojosa, J. D. (2017). *Arquitectura de software basada en microservicios para desarrollo de aplicaciones web de la Asamblea Nacional.*

Loza Peralta, C. V. (2020). Arquitectura de Software basada en microservicios para el uso en dispositivos de internet de las cosas.

Lozano Ramos, M. (2023). Arquitecturas de microservicios para el desarrollo de aplicaciones web.

Paja Domínguez, H. E. (2015). Predicción de rendimiento académico mediante regresión y redes neuronales en los estudiantes de la Escuela profesional de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano - Puno, 2015. (Universidad Nacional del Altiplano Puno). Retrieved from http://repositorio.unap.edu.pe/handle/UNAP/6337.

Pareja Valerio, C. B., & Burgos Robles, L. J. (2019). La arquitectura de software basada en microservicios: Una revisión sistemática de la literatura.

Quispe, A. (2018). Formulación de un modelo de red neuronal artificial para el pronóstico de concentración de oxígeno disuelto y clorofila en la bahía interior de Puno Universidad Nacional del Altiplano Puno.

Rodríguez, J. Á. I., Padilla, J. I., & Parra, H. A. (2020). Arquitectura basada en micro-servicios para aplicaciones web. *Tecnología Investigación y Academia Universidad Distrital*, *7*(2), 10.

Rodríguez, Z. M., Rodríguez, L. D. P., & Suarez, J. C. G. (2020). Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua. *Industrial Data*, *23*(2), 141-149.

Ruelas Acero, D. A. (2017). Modelo de composición de microservicios para la implementación de una aplicación web de comercio electrónico utilizando kubernetes.

Salgado Reyes, N., Beltrán Morales, J., Guaña Moya, J., Escobar Terán, C., Nicolalde Rodríguez, D., & Chafla Altamirano, G. (2018). Modelo para predecir el rendimiento académico basado en redes neuronales y analítica de aprendizaje. Revista Ibérica de Sistemas y Tecnologías de Información, 258–266. Retrieved from https://search.proquest.com/openview/5720c78f2e17a27355a8766fe81feb1a/1?pq origsite=gscholar&cbl=1006393

Sánchez Nina, A. (2017). Modelo estadístico para determinar la deserción estudiantil de las escuelas profesionales de la UNA - PUNO, 2017 (Universidad Nacional del Altiplano Puno). Retrieved from http://repositorio.unap.edu.pe/handle/UNAP/6297

Sotolongo-Aguilar, G., & Guzmán-Sánchez, M. V. (2001). Aplicaciones de las redes neuronales. El caso de la Bibliometría. Ciencias de la Información, 32(1), 27-34.

Trebejo Loayza, W. J. (2023). Herramienta para el modelado y generación de código de Arquitecturas de Software basadas en Microservicios.

Terán-Villanueva, J. D., Ibarra-Martínez, S., Laria-Menchaca, J., Castán-rocha, J. A., García-Ruiz, A. H., & Martínez-infante, J. E. (2019). Estudio de redes neuronales para el pronóstico de la demanda de asignaturas. Tunja-Boyaca, 28(50). Retrieved from http://www.scielo.org.co/pdf/rfing/v28n50/0121-1129-rfing-28-50-00034.pdf

Vásquez, J. (2016). Modelo Predictivo para Estimar la Deserción de Estudiantes en una Institución 614 de Educación Superior (Universidad de Chile). Retrieved from http://repositorio.uchile.cl/bitstream/handle/2250/144169/Vásquez Verdugo Jonathan.pdf?sequence=1(de la Cruz May, Castellanos, Jiménez, Coello, & Alí)

Vargas Chavarri, F., & Quispe Hernández, A. Á. Implementación de un sistema de información web para optimizar la gestión administrativa de la empresa comercial angelito de la ciudad de chepén.

Velasco, J. I. P., Ruiz, A. I. R., & Alvira, H. A. P. (2019). Arquitectura basada en microservicios para aplicaciones web. *Tecnología Investigación y Academia, 7*(2), 12-20.

Vera-Rivera, F. H., Astudillo, H., & Gaona, C. (2019). Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*(E23 (2019)), 107-120.

Villaizán Yamamoto, H. R. (2019). Arquitectura de software basada en microservicios para implementación de la aplicación web de cobranza digital en Financial Systems Company SAC.

Xiao, G., & Martina, M. (2016). *VLSI architectures design for encoders of High Efficiency Video Coding (HEVC) standard.* Tesis doctoral, Politecnico di Torino.

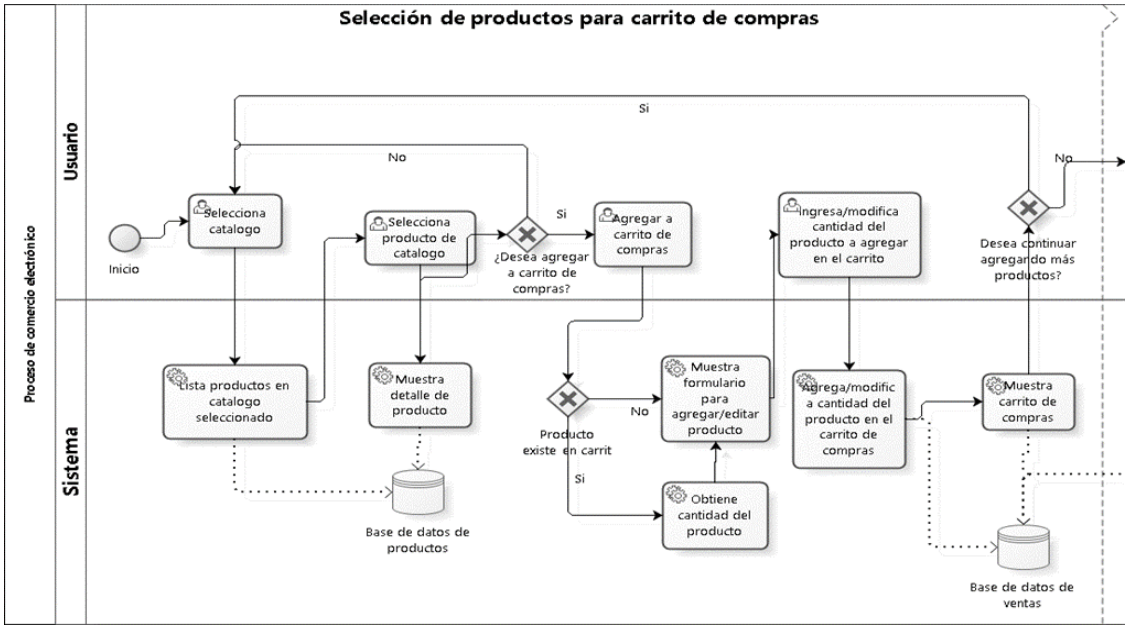
Zarria Torres, C., Arce Ramos, C., & Lam Moraga, J. (2016). Estudio de variables que influyen en la deserción de estudiantes universitarios de primer año, mediante minería de datos. 6(1), 73–84. Retrieved from https://dialnet.unirioja.es/servlet/articulo?codigo=5608574

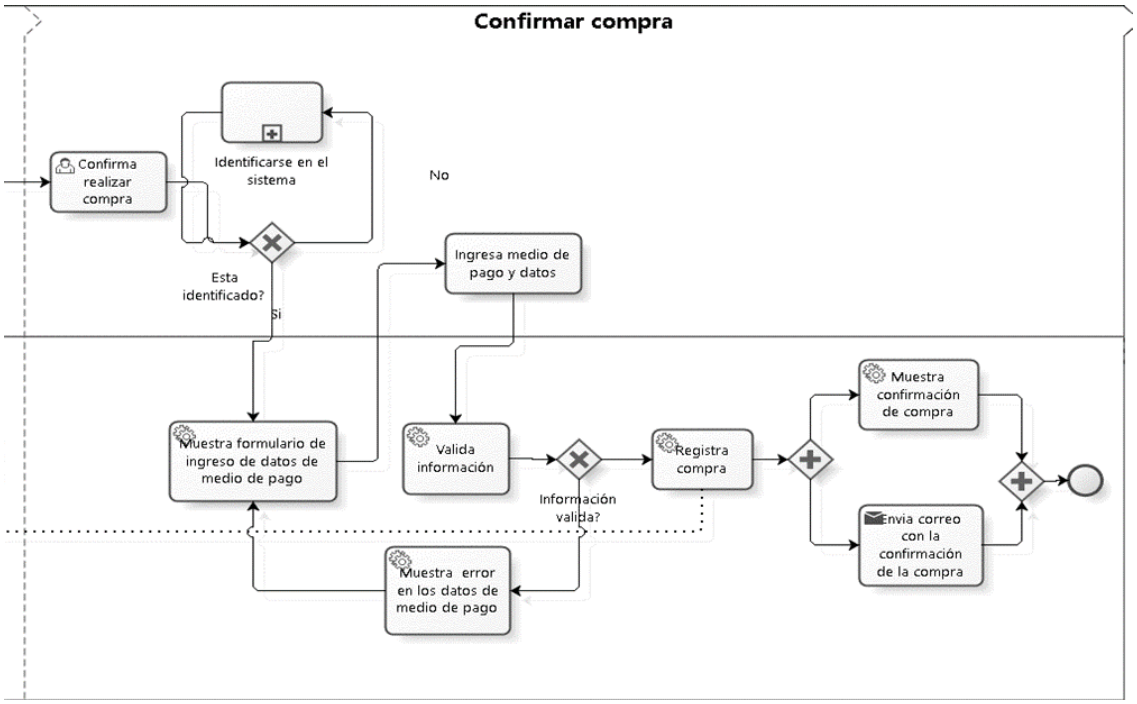
Zevallos Salazar, R. J. (2017). Predicción del rendimiento académico mediante redes neuronales (Universidad Nacional del Altiplano Puno). Retrieved from http://repositorio.unac.edu.pe/bitstream/handle/UNAC/2728/Zevallos Salazar_TESIS_2017.pdf?sequence=1&isAllowed=y

ANEXOS

ANEXO 1. Ubicación geográfica de la sede central Juliaca GROUP D´ MERCADO

ANEXO 2. Diagrama del proceso de negocio de comercio electrónico





ANEXO 3. Manual de usuario

ANEXO 4. Código fuente del sistema.

1) **Configuración API:**

```
<?php
   require_once 'product_service.php';

   header("Content-Type: application/json");

   $method = $_SERVER['REQUEST_METHOD'];

   switch ($method) {
      case 'GET':
         try {
            $productos = obtenerProductos();
            $productosArray = [];
            foreach ($productos as $producto) {
               $productosArray[] = $producto;
            }
            echo json_encode($productosArray);
         } catch (Exception $e) {
            http_response_code(500);
            echo json_encode(['error' => $e->getMessage()]);
         }
         break;

      case 'POST':
         try {
            $datosJson = file_get_contents('php://input');
            $datos = json_decode($datosJson, true);

            if (!isset($datos['nombre']) || !isset($datos['descripcion'])) {
               http_response_code(400);
               echo json_encode(['error' => 'Faltan datos necesarios para la
creación del producto.']);
               break;
            }

            $resultado = crearProducto($datos['nombre'], $datos['descripcion']);
            if ($resultado) {
               http_response_code(201);
               echo json_encode(['mensaje' => 'Producto creado con éxito.']);
            } else {
               http_response_code(500);
               echo json_encode(['error' => 'Error al crear el producto.']);
            }
```

```
            } catch (Exception $e) {
                http_response_code(500);
                echo json_encode(['error' => $e->getMessage()]);
            }
            break;
        }
    ?>
```

2) Código Conexión Base De Datos:

```
<?php
$host = "localhost"; // o la IP del servidor de la base de datos
$user = "root";
$password = "";
$dbname = "davsol";

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $user, $password);
    // Configurar el PDO error mode a excepción
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    die("ERROR: No se pudo conectar. " . $e->getMessage());
}
?>
```

3) Código De Productos:

```
<?php
require_once 'db.php';

/**
 * Obtiene todos los productos de la base de datos.
 *
 * @return array
 */
function obtenerProductos() {
    global $pdo;
    try {
        $consulta = "SELECT * FROM articulo";
        $sentencia = $pdo->query($consulta);
        return $sentencia->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        // Manejar el error adecuadamente en la práctica real
        exit("Error al obtener productos: " . $e->getMessage());
    }
}

/**
 * Crea un nuevo producto en la base de datos.
 *
 * @param string $nombre Nombre del producto.
```

```
 * @param string $descripcion Descripción del producto.
 * @return bool Retorna true si la creación fue exitosa, de lo contrario false.
 */
function crearProducto($nombre, $descripcion) {
    global $pdo;
    try {
        $consulta = "INSERT INTO articulo (nombre, descripcion) VALUES (?, ?)";
        $sentencia = $pdo->prepare($consulta);
        return $sentencia->execute([$nombre, $descripcion]);
    } catch (PDOException $e) {
        // Manejar el error adecuadamente en la práctica real
        exit("Error al crear producto: " . $e->getMessage());
    }
}

?>
```

ANEXO 5. Declaración jurada de autenticidad de tesis

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo QUISPE CANSAYA DIEGO DEYVIS,
identificado con DNI 46728412 en mi condición de egresado de:

☑ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

INGENIERÍA ESTADÍSTICA E INFORMÁTICA,

informo que he elaborado el/la ☑ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP DE MERCADO, JULIACA 2022"

Es un tema original.

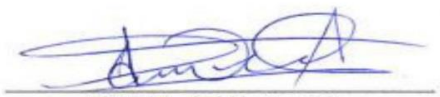
Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno 04 de enero del 2024



FIRMA (obligatoria)

Huella

ANEXO 6. Declaración jurada de autenticidad de tesis.

Universidad Nacional del Altiplano Puno | VRI Vicerrectorado de Investigación | Repositorio Institucional

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo COSME ESCALERA SONIA YESENIA,
identificado con DNI 43162494 en mi condición de egresado de:

☑ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

INGENIERÍA ESTADÍSTICA E INFORMÁTICA,

informo que he elaborado el/la ☑ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP DE MERCADO, JULIACA 2022"

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

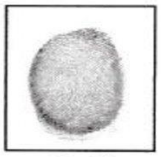
Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno 04 de enero del 2024

FIRMA (obligatoria)



Huella

ANEXO 7. Autorización para el depósito de tesis en el Repositorio Institucional

Universidad Nacional del Altiplano Puno | VRI Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo QUISPE CANSAYA DIEGO DEYVIS,
identificado con DNI 46728412 en mi condición de egresado de:

☑ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

INGENIERÍA ESTADÍSTICA E INFORMÁTICA,

informo que he elaborado el/la ☑ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP DE MERCADO, JULIACA 2022"

para la obtención de ☐**Grado,** ☑ **Título Profesional** o ☐ **Segunda Especialidad.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno 04 de enero del 2024

FIRMA (obligatoria)

Huella

ANEXO 8. Autorización para el depósito de tesis en el Repositorio Institucional.

Universidad Nacional del Altiplano Puno | VRI Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo COSME ESCALERA SONIA YESENIA,
identificado con DNI 43162494 en mi condición de egresado de:

☑ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

INGENIERÍA ESTADÍSTICA E INFORMÁTICA,

informo que he elaborado el/la ☑ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA EL DESARROLLO DE APLICACIONES WEB DE LA EMPRESA GROUP DE MERCADO, JULIACA 2022"

para la obtención de ☐**Grado,** ☑ **Título Profesional** o ☐ **Segunda Especialidad.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

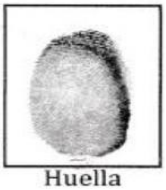
En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno 04 de enero del 2024



FIRMA (obligatoria)

Huella