Revista Peruana de Computación y Sistemas 2022 4(2):3-14
https://doi.org/10.15381/rpcs.v4i2.24855

ISSN Online: 2617-2003
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

Artículo de Contribución

Herramienta para el modelado y generación de código de Arquitecturas de Software basadas en Microservicios y Diseño guiado por el dominio (DDD)

Tool for the modeling and code generation of Software Architectures based on Microservices and Domain Driven Design (DDD)

Waldo Jefferson Trebejo Loayza [1,a], Fany Sobero Rodríguez [1,b]

[1] Universidad Nacional Mayor de Marcos, Facultad de Ingeniería de Sistemas e Informática. Lima, Perú

[a] Autor de correspondencia: waldo.trebejo@unmsm.edu.pe, ORCID: https://orcid.org/0009-0008-5011-6315

[b] E-mail: fsoberor@unmsm.edu.pe, ORCID: https://orcid.org/0000-0002-0323-6110

Resumen

Diferentes empresas que brindan servicios digitales deben contar con software sofisticados que permitan realizar diversas operaciones de manera oportuna y segura, manteniendo el programa en constante mejora, actualización y opciones para incorporar nuevas tecnologías del mercado. Muchas empresas aún utilizan sistemas heredados como su principal herramienta para los procesos de negocios, mientras que otras utilizan tecnologías obsoletas que limitan su capacidad para realizar actualizaciones o migraciones a tecnologías modernas, convirtiendo este escenario en una amenaza para la seguridad de los sistemas informáticos con los que operan, que pone en peligro el rendimiento y la productividad de la empresa. El presente artículo implementa una herramienta de modelado y generación de código de arquitectura de software basada en Microservicios y Diseño Dirigido por Dominio (DDD), para facilitar y acelerar el desarrollo de proyectos de software y la migración de sistemas heredados a nuevos servicios independientes, al tiempo que, utilizando tecnologías modernas para dar cuenta de la interoperabilidad, seguridad, escalabilidad, modularidad entre otros atributos de calidad. Para validar la contribución de la herramienta, se validó con expertos en diseño y construcción de software, lo que reveló la eficiencia y eficacia de la generación del modelo y código, así como la aceptación de la usabilidad de la herramienta.

Palabras clave: Arquitectura de software; microservicios; patrones arquitectónicos; diseño guiado por el dominio.

Abstract

Different companies that provide digital services must have sophisticated software that allows various operations to be carried out in a timely and safe manner, keeping the program in constant improvement, updating and options to incorporate new market technologies. Many companies still use legacy systems as their main tool for business processes, while others use outdated technologies that limit their ability to upgrade or migrate to modern technologies, turning this scenario into a threat to the security of IT systems with which they operate, which jeopardizes the performance and productivity of the company. This article implements a software architecture modeling and code generation tool based on Microservices and Domain Driven Design (DDD), to facilitate and accelerate the development of software projects and the migration of legacy systems to new independent services, by while using modern technologies to account for interoperability, security, scalability, modularity among other quality attributes. To validate the contribution of the tool, it was validated with experts in software design and construction, which revealed the efficiency and effectiveness of the model and code generation, as well as the acceptance of the tool's usability.

Keywords: Software Architecture; microservices; architectural patterns; domain driven design.

Recibido: 13/11/2022 - Aceptado: 15/12/2022 - Publicado: 31/12/2022

Citar como:
Trebejo, W. & Sobero, F. (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. https://doi.org/10.15381/rpcs.v4i2.24855

1. Introducción

En la actualidad, las organizaciones a nivel mundial cuentan con sistemas de información automatizados, conjunto de aplicaciones de software para controlar, agilizar, automatizar y optimizar los procesos de negocio, todo ello obliga a ser dependientes de la tecnología y sus avances de manera continua. Este hecho genera la necesidad de usar el internet para la comunicación y distribución de información entre todas las áreas que forman parte de la organización.

Emplear la tecnología es muy importante para los negocios, [1] hace mención que "Incorporar tecnología es fundamental para mejorar el servicio y aumentar la competitividad". Esta manifestación ayuda a entender que las organizaciones empresariales que invierten en tecnologías como apoyo a los procesos de negocio podrán marcar la diferencia de los demás teniendo mayor competitividad que los respalde para asegurar su crecimiento económico.

En tiempos actuales, la tecnología apoyada en el software está sumergida en varias economías, su uso permite influir en pacientes en clínicas, prevenir accidentes de tránsito, controlar aeronaves, interferir en sistemas de transporte, aeropuertos y más [2].

El objetivo principal en la ejecución de un proyecto de software es desarrollar un software sólido con atributos de calidad y capacidad para satisfacer las necesidades comerciales de la empresa; sin embargo, los pasos para el desarrollo de software son flexibles si se utiliza un enfoque de microservicios [3]. Sin embargo, no siempre se logra el objetivo deseado debido a varios factores que pueden estar presentes en el proceso de análisis conceptual, funcional, ecológico, técnico y formal, diseño arquitectónico y construcción de software [4].

Las inversiones en tecnologías amigables ayudan a optimizar servicios y hacer crecer la capacidad de las empresas [1]. Cada empresa tiene implementado más de dos tecnologías diferentes como herramientas tecnológicas, desde sistemas de software más antiguos hasta aplicaciones cada vez más modernas, debido que en el transcurso del tiempo han buscado estar actualizados con lo último del mercado para estar a la vanguardia. Muchas empresas tienen separados los departamentos o áreas con una tecnología diferente, solo en algunos casos cuentan con software que cubren todas las áreas. Todo ello genera la necesidad de realizar mantenimientos, actualizaciones y migraciones a nuevas tecnologías para ofrecer servicios disponibles a toda hora.

Actualmente, el desarrollo de software está orientado a entornos web, que requieren una serie de características, como mensajería o seguridad, control de usuarios, etc., que dificultan el desarrollo y el mantenimiento [5]. Por lo tanto, es fundamental comenzar el proyecto con un modelo de arquitectura de software que asegure y cumpla con todos los requisitos de un software de alta calidad para comprender las características de las diversas tecnologías disponibles en el mercado y comprender el modelo de negocio que será automatizado.

Para diseñar una arquitectura de software exitosa y desarrollar un producto de software de alta calidad, primero se deben comprender las necesidades de los usuarios. Asimismo, algunos investigadores han propuesto técnicas y modelos para obtener información sobre requerimientos, técnicas y recomendaciones más detalladas para mantenerlos, pero pocos han especificado cómo documentarlos durante esta etapa [6].

Cada año que pasa los softwares van quedando obsoletos en cuanto a la tecnología y se necesitan de personales especialista para su mantenimiento y funcionalidad adecuada para obtener un rendimiento adecuado hacia los clientes. Considerar un reemplazo, actualización o migración sería un desafío importante y una tarea costosa para la empresa, transformando esta situación en una amenaza para seguridad del sistema informático, causando inestabilidad y poniendo en peligro la eficiencia y la productividad de las empresas.

Durante la construcción del software, el desarrollador debe lidiar con una variedad de dificultades, como la velocidad del software, el alto nivel de avance y complejidad de los sistemas actuales y otras molestias que surgen a medida que avanza la tecnología. [7]. En este entorno, tener un modelo arquitectónico en etapas tempranas ya no es una cuestión; la aspiración de requisitos del sistema permite un modelo más avanzado de solución alternativa a los requisitos generados, lo que conduce a un producto final con una estética mejorada [8].

Con todo ello, muchas empresas se enfrentan a la necesidad de encontrar una solución para acelerar el desarrollo de proyectos de software y la migración de sistemas heredados hacia soluciones modernas, y poder hacer los mantenimientos y mejoras en un menor tiempo sin impactar a la productividad empresarial y tener una respuesta inmediata a las consultas generadas por los clientes y ser competitivos en el mercado, sacando ventaja a sus competidores. Por lo tanto, para desarrollar un producto de software con atributos de calidad como escalabilidad, interoperabilidad, desplegabilidad, seguridad y modularidad, entre otros, es fundamental diseñar correctamente la arquitectura de software y comprender el negocio lo suficientemente bien como para dividirlo en varias partes y comprender su proceso operativo con el cual se podrá crear o migrar cada parte a nuevas tecnologías que se requiera.

Como propuesta de solución, en este artículo se desarrolla una herramienta de modelado y generación de código para arquitecturas de software orientadas a microservicios y (DDD), el objetivo es modelar y generar código utilizando el diseño estratégico y táctico de DDD, cuyos procesos se desarrollan a partir de la herramienta. Para la construcción de la herramienta, se empleó una metodología ágil, el marco de trabajo Scrum, con el cual se dividieron las tareas a desarrollar mediante

historias de usuario, cada historia de usuario se planifico en sprint para controlar la construcción y entrega final.

El presente artículo, está organizado como se describe a continuación: El marco teórico se desarrolla en la Sección 2, con definiciones y conceptos de los temas de investigación, la Sección 3, muestra la exploración de literatura relacionada con tema de descomposición del dominio usando DDD y la construcción de microservicios, la Sección 4, muestra la metodología manejada para la validación de la herramienta, en la Sección 5, con el diseño e implementación del aporte, y en la Sección 6, con la validación y presentación del resultado obtenido. Finalmente, la Sección 7, expone las conclusiones de la investigación.

2. Marco Teórico

A. Arquitectura de Software

Esta actividad enlaza los requerimientos con el diseño de software, siendo de vital importancia su planteamiento adecuado para así poder satisfacer los requerimientos definidos [9].

La arquitectura de software representa las decisiones de diseño en relación a la estructura y el funcionamiento general del sistema, además, ayuda a comprender y analizar cómo el sistema logrará obtener atributos de calidad de modificabilidad, disponibilidad y seguridad [10].

En otra definición para arquitectura de software, indica que cuando las personas en la industria del software usan el término "arquitectura", se refieren a una definición específica de aspectos más importantes del bosquejo de software interno. Una buena arquitectura es importante; de lo contrario, agregar nuevas capacidades será más lenta y costoso en el futuro [11].

B. Diseño Arquitectónico

El diseño arquitectónico busca entender cómo debe estar organizado un sistema y cómo se diseña la estructura general de ese sistema [9]. Hacer un buen diseño de la arquitectura es fundamental, ya que permite definir la forma y estructura del producto software a desarrollar.

El diseño arquitectónico es el componente esencial de un sistema de software en el cual es necesario determinar cómo se debe estructurar el sistema a desarrollar y cómo se deben conectar los componentes que lo componen. Es fundamental comprender la definición de arquitectura para obtener los conocimientos necesarios, que ayudarán en el diseño del modelo a proponer como solución [9].

C. Patrones de Diseño

Según la definición de los autores [12], un patrón de diseño nombra, complementa y define aspectos de la estructura general del diseño y utilizarlos para crear un diseño reutilizable y encaminado a objetos. El modelo de proyecto define las clases y casos involucrados, sus roles, colaboración y división de deberes. Cada patrón de diseño se centra en un problema específico y describe cuándo y por qué debe usarse, así como sus ventajas y desventajas.

D. Modelo

Según la definición de [13], el modelo es una representación del sistema que se desarrollará para ayudar a la comprensión en la finalización del diseño final del sistema. Esto ayudará a comprender cómo se construye cada componente del sistema que debe implementarse, permitiéndose realizar las modificaciones y pruebas necesarias disminuyendo el riesgo de fallos que puedan darse en el sistema real.

E. Microservicios

La arquitectura de microservicios propone descomponer o particionar aplicaciones en pequeños componentes, donde cada componente va a realizar una función individual del proceso de negocio, esta forma de trabajo difiere de las arquitecturas monolíticas los cuales son un solo componente, la separación de componentes en microservicios aumenta la complejidad de su integración, por lo que se decidió utilizar comunicación asíncrona basada en eventos para desacoplar las interfaces de integración entre servicios. Un aspecto importante que deben tener en cuenta los que implementan esta arquitectura de microservicios es que fue diseñada para definir los límites de cada microservicio más aun los que se comunican entre ellos mediante las interfaces, donde cualquier cambio podría afectar a los otros microservicios [14].

Según la definición de [15], las arquitecturas basadas en microservicios están emergiendo como una solución viable para los problemas de escalado de aplicaciones, particularmente cuando se trata de aplicaciones distribuidas.

F. Domain Driven Design – DDD

Se puede definir como una orientación en la construcción del software complejo basado en el negocio principal de la empresa, explora el modelo con la ayuda de expertos en dominios y desarrolladores de software, y utiliza un lenguaje ubicuo para comprender las definiciones. DDD se divide en diseño estratégico y táctico, y ambos deben combinarse para lograr el éxito en la entrega de un sistema [16].

El DDD, está dividido en dos partes, el Diseño Estratégico está conformado por el dominio, lenguaje ubicuo, subdominio, contexto limitado, mapa de contexto, etc. los cuales van a definir el diseño y la división del modelo de dominio en varias partes, por otro lado, está el Diseño Táctico conformado por entidades, objetos de valor, agregados, servicios, repositorios, etc. que define como se va implementar el modelo de dominio definido.

El objetivo de (DDD) es crear un modelo enfocado a objetos que manifieste el conocimiento de un área específica, totalmente independiente de cualquier concepto de comunicación que pueda haber con elementos de infraestructura como interfaces gráficas, etc. [17].

3. Revisión de la literatura

Existen diferentes modelos como referencia para implementar arquitectura orientada a Microservicios y DDD, el cual se propone utilizar como diseño arquitectónico en este artículo para acelerar el desarrollo de proyectos de software y la migración de sistemas heredados, así como mejorar el mantenimiento y la escalabilidad del software. Por esta razón, es importante evaluar y analizar otros estudios que se han realizado.

En una publicación del articulo [18], los autores realizan una investigación basada en el software existente de una empresa de tecnología que ha desarrollado un sistema básico de aplicaciones de contratación electrónica para satisfacer sus necesidades de automatización en el proceso de contratación interno de una organización, donde detallan diversos inconvenientes que se presentan al intentar mejorar y actualizar el software. Observando las dificultades que crea una arquitectura monolítica ante grandes cambios y avances para nuevos objetivos de desarrollo, surge la necesidad de trasladar la complejidad al nivel de coordinación con la arquitectura de microservicios, lo que ayudaría a la empresa a integrar los procesos comerciales existentes, respaldar la infraestructura de tecnología de la investigación, reutilización y combinación de componentes de servicio. Como resultado de la investigación y discusión, los clasifican en cuatro grupos, comenzando con el análisis comercial de contratos electrónicos, análisis de dominio, definición del contexto acotado y diseño arquitectónico de microservicios. Concluyen mencionando que el análisis y bosquejo de la arquitectura de microservicios utilizando el enfoque de diseño basado en dominios puede ayudar en el proceso de identificación de microservicios, y que la incorporación de microservicios en el sistema de investigación de la empresa es de gran ayuda porque permite un fácil mantenimiento y desarrollo de nuevas aplicaciones.

En un artículo [19], los autores proponen un esquema de diseño de generador de modelo de negocio que se realizará mediante el diseño de modelos controlados por dominio y la implementación de código controlado por modelo. Se basan en el enfoque DDD, que compara los resultados de la arquitectura del sistema creado con DDD con los de la arquitectura heredada, y que la escalabilidad, el rendimiento y la calidad del código del nuevo sistema han mejorado significativamente.

Utilizaron el Modelo Operativo del Formulario Inteligente en la primera fase; Con esta tecnología, los desarrolladores y administradores pueden crear sistemas de aplicaciones de forma rápida y sencilla. En la segunda fase, utilizaron el Modelo de Operación de Flujo de Trabajo, que puede pensarse como una operación compuesta a partir de las operaciones del objeto comercial. Como ejemplo, sugieren que un proceso de aprobación de formulario puede verse como la inclusión de una operación de proceso durante la creación de un nuevo formulario del objeto de negocio, después de lo cual el objeto de formulario se crea correctamente. Cuando falla la aprobación, falla la creación del objeto de formulario. En la tercera fase, utilizaron el Modelo de Generador de Negocios DDDBizModel, que se describe como una colección de modelos generadores de negocios basados en un diseño dirigido por dominio que incluye entidad, objeto de valor, agregación, almacenamiento, repositorio y servicio. Entre estos, la entidad, el objeto de valor y la agregación no solo incluyen datos, sino que también tienen comportamientos comerciales atractivos. Finalmente, los autores concluyeron que implementaron el marco de desarrollo y encapsularon las lógicas comerciales públicas, como el flujo de trabajo, las operaciones de formulario y las operaciones de datos de formulario, que pueden respaldar un mejor diseño y desarrollo centrado en el dominio, mejorando la eficiencia del desarrollo y reduciendo los defectos [19].

En una investigación [20], los autores proponen un diseño para la generación automática de código basado en patrones MDA y MVC los cuales se encuentran en la mayoría de los generadores de código comerciales, así como otras herramientas GNU que convierten código en sistemas CRUD transaccionales en plataformas como JSP, ASP, PHP, Ruby y otras plataformas en el entorno web. Se considera un lenguaje prototipo, pero es posible considerar en su implementación otros lenguajes o enumeraciones debido a la alta usabilidad y confiabilidad. La investigación concluye con la introducción de la línea de productos que reutilizar y combinar paquetes es una forma más sencilla y eficiente de administrar cualquier proyecto de software, independientemente de los modelos y los requisitos del modelo. De acuerdo con los requerimientos de los usuarios, también se menciona que la generación automatizada de líneas de productos de software se convierte en una herramienta poderosa que facilita la conversión y creación de una gran cantidad de artefactos, lo que reduce el tiempo de desarrollo y acrecienta la productividad del desarrollo ágil.

En la investigación [21], Los autores identifican nuevas tecnologías, métodos y arquitecturas utilizadas por la institución en estudio (CGTIC) para la construcción de proyectos web, así como tecnologías existentes para la implementación de microservicios como solución a las dificultades encontradas en el área evaluada. Utilizaron un enfoque cualitativo para la investigación descriptiva y el diseño de documentos como base de su investigación (CGTIC), en la revisión de su estado del arte se basaron en identificar los métodos para implementar microservicios, así como sus requerimientos y necesidades para construcción de aplicaciones web. Finalmente concluyeron: El problema actual en desarrollo de software con la arquitectura utilizada es que es una aplicación monolítica, lo que dificulta el mantenimiento, el escalado del sistema y la adición de nuevas funcionalidades. Esto se basa en la evaluación de aplicar nuevas

tecnologías como la arquitectura de microservicios, con la que esperan mejorar como institución.

En la figura 1, se observa el patrón básico de una arquitectura de Microservicios, el cual permite separar el negocio en diferentes servicios independientes que permitirá realizar con facilidad el mantenimiento, la escalabilidad y adicionar nuevas funcionalidades.

En la investigación [23], los autores enumeran los patrones que se pueden usar en proyectos que tienen como objetivo migrar una aplicación desarrollada desde un enfoque monolítico a un nuevo enfoque de microservicios.

La migración del sistema a la arquitectura de microservicios trae muchos beneficios, como la posibilidad de ofrecer disponibilidad y escalabilidad teniendo en cuenta las diversas partes del sistema, por ende, proponer diferentes estructuras de hardware para diferentes partes del sistema de manera independiente y según la necesidad. Como otro beneficio se tiene la posibilidad de manejar diferentes tecnologías para diferentes servicios, obviando así que el sistema esté amarrado a una sola tecnología y beneficios que posee [24].

En el artículo [25], la autora propone un modelo de componente de microservicio para configurar una aplicación web de comercio electrónico que utiliza Kubernetes como tecnología de contenedores. Diseña la arquitectura de la composición del microservicio, evalúa frente a los atributos de calidad propuestos y verifica el modelo mediante pruebas de carga. Las pruebas en una aplicación web de microservicios proporcionan métricas de tiempo, respuesta, disponibilidad y rendimiento como resultados. Finalmente, detalla los resultados de su prueba, donde menciona que configurar la aplicación web de comercio electrónico basada en un modelo de componentes del microservicio ofrece un rendimiento significativo en comparación con una aplicación web monolítica con métricas de 104% de rendimiento, disponibilidad y tiempo de respuesta.

En la investigación [26], los autores proponen una plataforma de desarrollo unificada basada en una arquitectura de microservicios que divide un sistema de aplicación grande y complejo en varios módulos de servicio que se pueden desarrollar de forma independiente, probado, desplegado, operado y actualizado. Concluyen que la arquitectura de microservicios es capaz de resolver nuevos retos porque cuentan con un sistema independiente y una serie de módulos definidos; cada módulo permite desarrollar, probar, implementar, expandir muy diferente a la arquitectura individualizada. Por lo tanto, la arquitectura de microservicios es una excelente opción para el desarrollo de proyectos, afectando positivamente aspectos como el tiempo, el rendimiento o la estabilidad del proyecto.

En la investigación [27], los autores revisaron la literatura para tener conocimiento y describir las direcciones de exploración actuales en la creación de aplicaciones orientadas en microservicios. Luego definieron la granularidad de microservicios, la modularización y la refactorización de los servicios, la composición de la interfaz de usuario, la seguridad, la orquestación, la supervisión, la administración y la gestión de los microservicios y la tolerancia a fallas de los microservicios. Finalmente, como conclusión, mencionan que han demostrado trabajo, comenzando con el análisis de diferentes fuentes de información, estas revisiones se centran principalmente en el modelado y la arquitectura de aplicaciones. Se utiliza para servicios individuales para definir modelos de diseño y de desarrollo, analizando implementaciones e identificando las ventajas y desventajas de los microservicios.

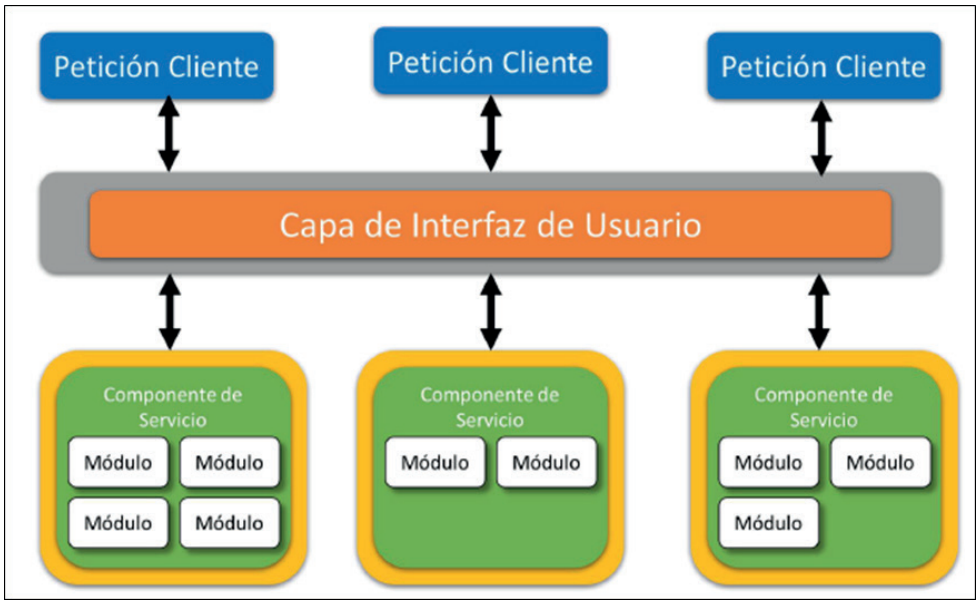

Figura 1. Patrón básico de arquitectura de Microservicios [22]

En el artículo revisado [28], el autor propuso un modelo de especificación para arquitectura de aplicaciones basado en microservicios. Experimentó con la canalización de implementación de Bitbucked y obtuvo la capacidad de realizar pruebas automáticas estáticas, unitarias y funcionales, así como la implementación automática en un servidor de prueba y producción. Concluyó mencionando que el proceso de desarrollo de microservicios se divide en dos partes: el desarrollo de los propios microservicios y el desarrollo de las aplicaciones que utilizan los microservicios, poniendo en práctica los DEVOPS.

4. Metodología

En el presente artículo se empleará el diseño *No Experimental y de tipo aplicado*, como unidad de análisis se trabajará con los *Profesionales de tecnologías de información.*

En los procedimientos estadísticos, la muestra es un subgrupo de la población de la que se coleccionan los datos. Estas muestras deben estar claramente definidas y predefinidas y deben ser representativas de la población. [29].

Para la selección de la muestra, se considera que el profesional debe poseer las siguientes características: un mínimo de 5 años de experiencia en desarrollo de software, al menos 5 años de rutina en diseño de arquitectura de software y conocimientos básicos de DDD.

La población de estudio estuvo constituida por Profesionales de tecnología dedicados al desarrollo de proyectos de software, la muestra fue seleccionada bajo el enfoque No probabilístico mediante la selección por conveniencia, para ello se consideró que los participantes tengan experiencia como arquitectos y desarrolladores de software, se pudo contactar a 10 profesionales que tuvieron disposición para formar parte del estudio.

Se seleccionó una encuesta como método de recolección de datos a través de la cual se recopila la percepción de la usabilidad de la herramienta elaborada mediante 16 preguntas post uso y a su vez se realiza colección de datos que medirán la eficacia y la eficiencia del uso de la herramienta. Los participantes recibieron una inducción sobre el uso de la herramienta para luego realizar un ejercicio practico y de requerir la asistencia se le brindara el soporte requerido.

5. Diseño e Implementación del aporte

El proyecto se implementa en una metodología ágil basada en el marco de trabajo Scrum. Cuando se usa el scrum, debe verse como un método para realizar el trabajo en equipo en pequeños fragmentos al mismo tiempo, con pruebas continuas y períodos de retroalimentación a lo largo del camino para instruir y mejorar a medida que pasa el tiempo. Scrum ayuda a individuos y equipos a crear valor de manera incremental y colaborativa. Scrum, como un marco ágil, proporciona suficiente estructura para que las personas y los equipos se integren en su estilo de trabajo al mismo tiempo que incorpora las mejores prácticas para optimizar sus necesidades específicas. [30].

Para la implementación del aporte, se parte desde el diseño de modelado de procesos con especificación de la funcionalidad de la herramienta para modelado y generación de código de arquitecturas de software centradas en microservicios, donde se describen los pasos que realiza el usuario (Arquitecto/Desarrollador) partiendo desde que se define el dominio y se explora en un taller con todos los involucrados, para luego ingresar a la herramienta web y emplear el diseño estratégico de DDD que abarca desde el registro del dominio junto a los Lenguajes ubicuos definidos, seguido realizar la descomposición del domino, identifica y registra Subdominios, define y configura los Bounded Context. Seguidamente se emplea el diseño táctico del DDD que abarca desde la definición y registro de los Agregados junto a las Entidades y Objetos de valor, para finalmente realizar la configuración y definición de los parámetros con el cual se genera el código de la arquitectura, código de entidades/agregados y la descarga del código fuente generado para luego abrir, compilar y ejecutar el microservicio.

En la figura 2 se detallan el flujo de procesos de la herramienta desde la definición del dominio que abarca

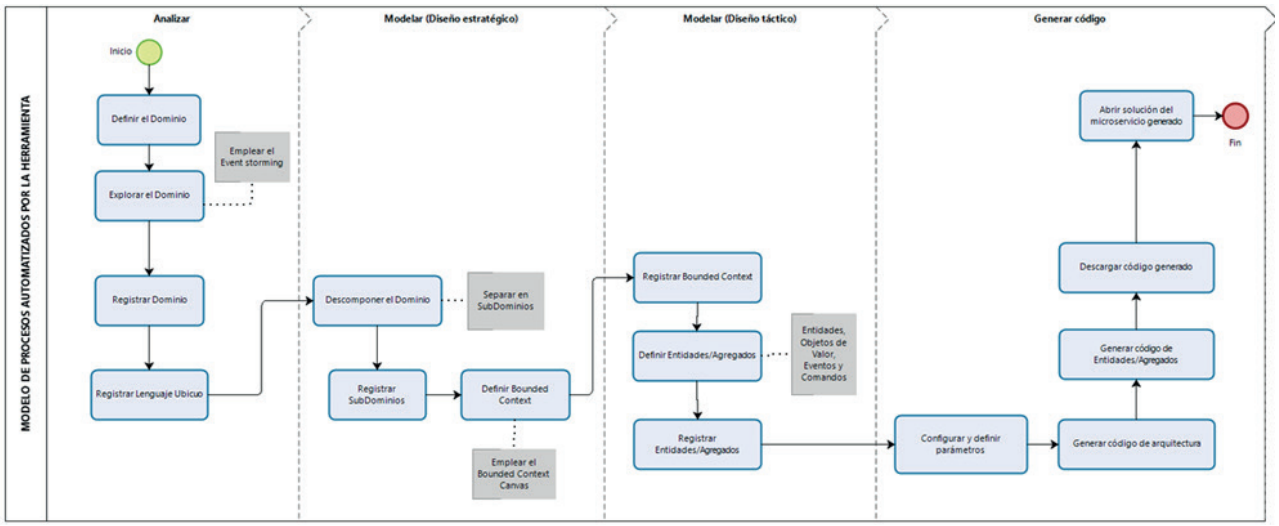

Figura 2. Modelo de procesos automatizados por la herramienta propuesta

la descomposición y modelamiento hasta la generación de código de la arquitectura.

Para identificar los requisitos que debían desarrollarse, se crearon historias de usuarios que coincidían con la descripción de la acumulación de sprint para su desarrollo:

- Aplicación Web: Implementación de la herramienta propuesta.
- Modelado de arquitectura de software con Diseño Estratégico del DDD: Desarrollo de Formularios Web para definición de Dominio, Subdominio y Contexto Limitado.
- Modelado de arquitectura de software con Diseño Táctico del DDD: Desarrollo de formularios web para definición de Agregados, Entidades y Objetos de valor.
- Configuración, definición de parámetros: Ingreso de parámetros de configuración para definir la creación de código de arquitectura.
- Generación de código de Arquitectura: Generación de código de la arquitectura propuesta para Spring Boot.
- Generación de agregados y entidades: proceso de creación de Entidades y Agregados en el código generado del paso anterior.
- Descargar código: Proceso de descarga del código fuente.
- Importación y apertura del código del microservicio: un procedimiento adicional para importar y abrir el código fuente descargado de la herramienta y luego compilar, ejecutar y probar el microservicio.

En base al modelado de proceso mostrado anteriormente, se diseña la arquitectura del software teniendo en cuenta todas las partes a emplear en la construcción y funcionamiento recomendado de la herramienta propuesta. La figura 3 expone la arquitectura diseñada detallando las capas que separan cada parte del proyecto:

La estructura del proyecto propuesto se compone de capas en base a la arquitectura diseñada. Aplicación, Dominio, Persistencia, Utilidad y WebApp (Aplicación Web). La figura 4 muestra la estructura del proyecto con los componentes antes mencionados:

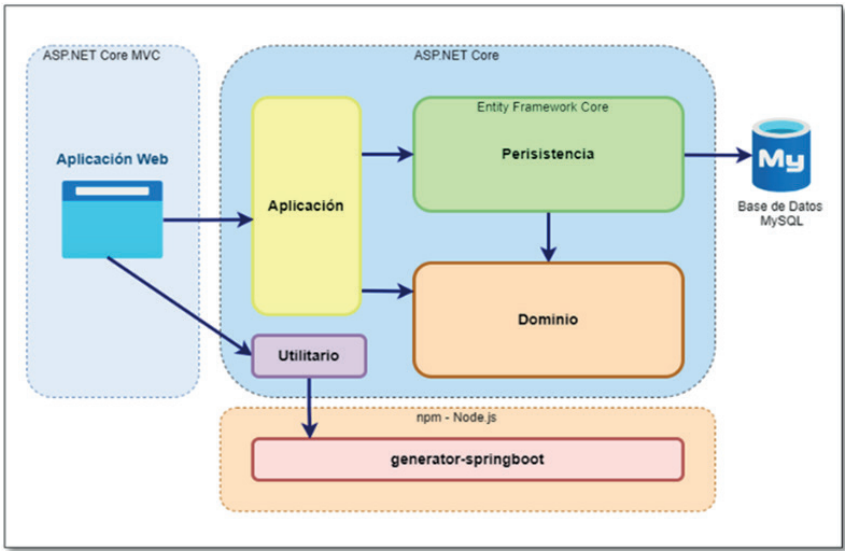

Figura 3. Arquitectura de la herramienta.

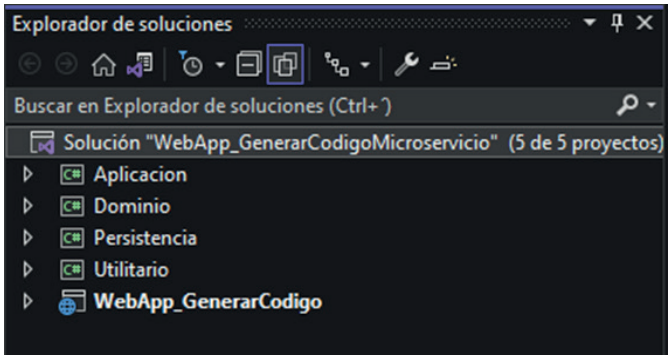

Figura 4. Estructura del proyecto

El proyecto fue desarrollado con NET Core 6.0 para la codificación del backend, y en la parte de frontend se empleó ASP.NET Core MVC.

El patrón MVC es uno de los modelos más importantes de la historia de construcción del software, y todavía se usa considerablemente hoy en día. Corresponde a la colección de patrones de estilo arquitectónico de presentación separada y separa el código responsable de dibujar datos en la interfaz del código responsable de ejecutar el modelo de negocio. Hay tres tipos: modelo, vista y controlador. [17].

La parte de comunicación con la base de datos utilizó Entity Framework Core, que es la versión ligera de código abierto y multiplataforma de la habitual tecnología de acceso a datos de Entity Framework que es compatible con varios motores de bases de datos. [31].

La figura 5, muestra el esquema de base de datos creado desde el proyecto inicial utilizando Entity Framework Core (Code First):

En el componente de generación de código (Capa utilitario), se realizó la integración de la aplicación web con un proyecto externo llamado generator-springboot, el cual es un proyecto generador Yeoman para generar Microservicios con SpringBoot [32].

Finalmente, se logró desarrollar la herramienta completa con las opciones de modelado empleando el diseño estratégico y táctico del DDD y la generación de código de arquitectura. La figura 6 expone la pantalla principal de la herramienta desarrollada:

6. Validación y Presentación de resultados

A. Objetivos de la validación de la herramienta

Primer objetivo: Conocer la eficiencia y eficacia de la herramienta con la medición del tiempo invertido y la culminación de la tarea en los procesos de modelado y generación de código utilizando el enfoque DDD con el uso de la herramienta desarrollada.

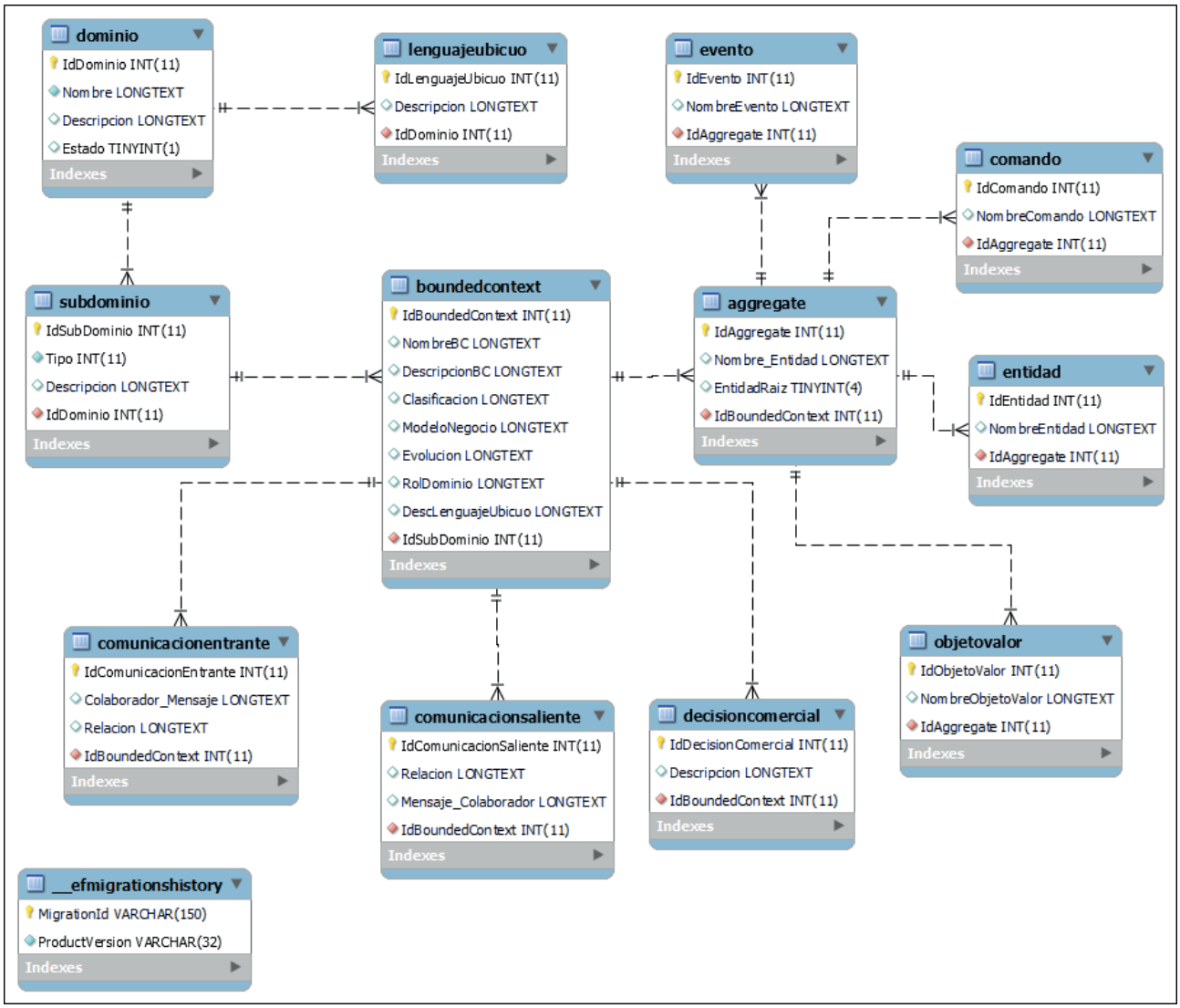

Figura 5. Diagrama de la base de datos

Figura 6. Pantalla principal de la herramienta.

Segundo objetivo: Obtener un porcentaje de satisfacción de los usuarios (Arquitectos/Desarrolladores) en base a su experiencia modelando y creando código de arquitectura de software utilizando el enfoque DDD con la herramienta desarrollada.

B. Diseño de la validación de la herramienta

- La validación de la herramienta se realizó con la asistencia de 10 desarrolladores y arquitectos de software, cada uno con más de 5 años de experiencia.
- Se examinó el dominio definido junto con los subdominios correspondientes, mapa de contexto y contextos acotados preparados en el taller de event storming.
- Cada participante recibió capacitación sobre cómo usar la herramienta de evaluación (desde el ingreso de datos hasta la generación de códigos).

C. Ejecución de la validación

Cada participante utilizó la herramienta y eligió la opción de diseño estratégico, donde registraron el dominio, descompusieron el dominio, registraron cada subdominio identificado y luego definieron y registraron los contextos limitados.

Seguidamente, cada participante eligió la opción de diseño táctico, donde definieron y registraron entidades, objetos de valor y los agregados, finalmente seleccionaron la opción de generación de código para obtener los microservicios.

D. Presentación de resultados

Los resultados de la interacción de cada participante con la herramienta se registraron en la Tabla 1, que se puede ver a continuación:

Tabla 1. *Resultado con Eficacia y Eficiencia de la herramienta con la validación de modelado y generación de código.*

Participante	Uso de la Herramienta Web		
	Eficacia		Eficiencia
	¿Necesitó asistencia?	¿Completó tarea con éxito?	Tiempo de esfuerzo (minutos)
P1	No	Si	30
P2	No	Si	28
P3	No	Si	25
P4	No	Si	26
P5	Si	Si	30
P6	No	Si	24
P7	No	Si	27
P8	Si	Si	25
P9	No	Si	28
P10	No	Si	22
Tiempo promedio			26,5

En la tabla 1 se pueden visualizar los resultados de la validación realizada por los participantes, siendo que solo el 20% requirió asistencia para completar la tarea de validación y el 80% restante la completó sin asistencia, lo que indica que la Eficacia de la herramienta desarrollada es del 80%. Además, se observa que el 100% de los participantes completaron la tarea propuesta con Eficiencia, promediando 26,5 minutos de esfuerzo por participante.

En resumen, al utilizar la herramienta de modelado y generación de código de arquitectura, será posible reducir el tiempo requerido para diseñar una arquitectura de software utilizando el enfoque DDD, a diferencia del método tradicional, que tomaría varias horas.

Por otro lado, para la medición de la usabilidad de la herramienta desarrollada, se elaboró una encuesta de satisfacción del usuario después del uso empleando el instrumento "The Post-Study System Usability Questionnaire (PSSUQ)" traducido al español: El

Cuestionario de Usabilidad del Sistema Después del Estudio (PSSUQ).

En el *Apéndice A*, se visualiza la Encuesta de satisfacción del usuario después del uso que se elaboró y distribuyó a todos los participantes en la validación de la herramienta. La encuesta consta de 16 preguntas categorizados en 3 subescalas y un resultado general.

PSSUQ es un instrumento diseñado para medir el agrado oculto de los usuarios con sus sistemas de información. Nació de un proyecto interno de IBS nombrado SUMS (System Usability MetricS) dirigido por Suzanne Henry a fines de la década de 1980. El objetivo de SUMS era demostrar y validar las medidas de usabilidad del sistema en términos de rendimiento, usabilidad y satisfacción de usuario. [33].

PSSUQ tiene 16 elementos en la versión 3, cada uno con 7 opciones (y una opción NA opcional) para ser seleccionados por la encuesta. Donde 1 es (totalmente de acuerdo) y 7 es (totalmente en desacuerdo). En cuanto a los resultados, cuando son más bajos, la productividad y la satisfacción se consideran excelentes [34].

Los elementos del PSSUQ generan cuatro puntuaciones: una general y tres subescalas [34]. Estos son:

- General: promedio de los elementos 1 a 16 (todos).
- Calidad del sistema (SysQual): puntaje medio de los elementos 1 a 6.
- Calidad de información (InfoQual): puntaje medio de los elementos 7 a 12.
- Calidad de interfaz (IntQual): puntaje medio de elementos 13 a 15

Para recopilación de información, se aplicó la encuesta elaborada a los participantes profesionales en desarrollo de software. Los resultados logrados se exponen en la Tabla 2. Resultado para el PSSUQ promedio general es de 1.77, el cual es muy bueno en términos de desempeño y satisfacción con la herramienta validada.

Tabla 2. *Muestra los resultados de la encuesta*

Puntuación general PSSUQ	Utilidad del Sistema (SYSUSE)	Calidad de la Información (INFOQUAL)	Calidad de la Interfaz (INTERQUAL)
2,13	1,67	2,50	2,33
1,69	1,50	2,17	1,00
1,63	1,50	2,00	1,33
2,13	1,67	2,67	2,00
1,44	1,17	1,67	1,33
1,19	1,00	1,33	1,33
1,81	1,50	2,33	1,67
2,00	2,00	2,00	2,00
1,94	1,67	2,33	1,67
1,75	1,33	2,33	1,33
1,77	1,50	2,13	1,60

Además, se observan las siguientes subescalas: utilidad del sistema (SYSUSE) con una media de 1,50, calidad de información (INFOQUAL) con una media de 2,13 y calidad de interfaz (INTERQUAL) con una media de 1,60. Para interpretar estos resultados, utilizamos las definiciones del PSSUQ, que establecen que cuando el resultado es menor y está cerca al uno (1), la herramienta validada ha recibido el mayor nivel de satisfacción para los usuarios (arquitectos y desarrolladores).

7. Conclusión

Se realizó la revisión de literatura de varios artículos relacionados al tema de implementación microservicios empleando el DDD, donde los autores resaltan la importación del estilo arquitectónico microservicios por su característica de separar el negocio en varios servicios independientes fáciles de hacer mantenimiento y la escalabilidad requerida.

Se logró construir una herramienta para modelar y generar código para arquitecturas de software centrada en microservicios utilizando el enfoque de diseño impulsado por dominio (DDD), que permitió realizar el modelado y la generación de código eficientes y efectivos. Para ello, se realizó la validación de la herramienta con la ayuda de arquitectos y desarrolladores que tienen mayor a 5 años de experiencia en el campo del softwaare. Como consecuencia, los participantes dedicaron un promedio de 26,5 minutos de esfuerzo, lo cual es bastante aceptable en comparación con el tiempo que llevaría diseñar de manera tradicional.

Se realizó una encuesta de satisfacción del usuario después del uso, obteniendo como resultado para Utilidad del Sistema (SYSUSE) un promedio igual a 1.50, Calidad de la Información (INFOQUAL) un promedio igual a 2.13 y Calidad de la Interfaz (INTERQUAL un promedio igual a 1.60 esto podría indicar que hubo una mayor aceptación de la satisfacción del usuario.

Por lo tanto, la herramienta de modelado y generación de código de arquitectura de software facilita y acelera desarrollar un proyecto de software, así como migrar sistemas más antiguos desde la descomposición del dominio a la estructura del proyecto en un menor tiempo.

Referencias

[1] CentroaméricaEconomía. (2016). EL IMPACTO DE LA CONECTIVIDAD EN LOS NEGOCIOS. https://centroamericaeconomia.com/2016/09/27/el-impacto-de-la-conectividad-en-los-negocios-2/, consultado el 22 de noviembre del 2022

[2] Villadiego, R. (2020). ¿Por qué el software conquistó al mundo?. Retrieved 11 2022, from https://forbes.co/2020/08/27/red-forbes/por-que-el-software-conquisto-al-mundo/

[3] López, D., & Maya, E. (2017). Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web. Septima Conferencia de Directores de Tecnologia de Informacion 2017, 7, 1–12. http://dspace.redclara.net:8080/bitstream/10786/1277/1/93 Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.pdf

[4] Gatell, A. (2019). Significado Del Proceso De Diseño En La Formación Del Arquitecto. Particularidades En Cuba. Contexto, 13(19), 73–88. https://doi.org/10.29105/contexto13.19-7

[5] Sayago, J. (2018). Generador de Código Utilizando el Paradigma de Líneas de Producto Software Code Generator Using the Software Product Lines Paradigm. Rev. Hallazgos21, 3(2), 190–212. https://revistas.pucese.edu.ec/hallazgos21/article/view/281

[6] Sommerville, I. (2005). Requerimientos Del Software. Universidad Veracruzana, 109–110. https://www.uv.mx/personal/fcastaneda/files/2015/08/F_Capitulo_5_Requerimientos_del_software.pdf

[7] Bachmann, F., Bass, L., Chastek, G., Donohoe, P., & Peruzzi, F. (2000). The Architecture Based Design Method. Engineering, 1–61. https://www.researchgate.net/publication/235088008_The_Architecture_Based_Design_Method

[8] Meaurio, V., & Schmieder, E. (2013). La Arquitectura de Software en el Proceso de Desarrollo: Integrando MDA al Ciclo de Vida en Espiral. Revista Latinoamericana de Ingenieria de Software, 1(4), 142–146. https://doi.org/10.18294/relais.2013.142-146

[9] Sommerville, I. (2011). Ingeniería de software 9. PEARSON. https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

[10] Software Engineering Institute (sf). Software Architecture, Why Architecture?. Recuperado el 22 de noviembre del 2022 de: https://www.sei.cmu.edu/our-work/software-architecture/

[11] Fowler, M. (2019). Software Architecture Guide. Recuperado el 22 de noviembre del 2022, de https://martinfowler.com/architecture/

[12] Gamma, E., Helm, R., Johnson, R. y Vlissides, J. (2003). Patrones de diseño. Elementos de software orientado a objetos reutilizable. Madrid: Pearson Educación

[13] Pons, C., Giandini, R., & Pérez, G. (2010). DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS Conceptos teóricos y su aplicación práctica. La Plata: Universidad Nacional de La Plata.

[14] Velepucha, V., Flores, P., & Torres, J. (2018). MOMMIV: Modelo para descomposición de una arquitectura monolítica hacia una arquitectura de microservicios bajo el Principio de Ocultación de Información. Quito: RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação

[15] Roldan, D., Valderas, P. & Torres, V. (2018). Microservicios. Un enfoque integrado. Madrid: RA-MA.

[16] Evans, E. (2015). Domain-Driven Design Reference: Definitions and Pattern Summaries. Google Libros. https://books.google.com.pe/books?hl=es&lr=&id=ccRsBgAAQBAJ&oi=fnd&pg=PA25&dq=Domain-Driven+Design+Reference:+Definitions+and+Pattern+Summaries&ots=gNd5VRkihF&sig=57SvvyMd2dWHX-gqqvLITcR5jkg#v=onepage&q=Domain-Driven Design Reference%3A Definitions and Pa

[17] Torre, C., Zorrilla, U., Calvarro, J., & Ramos, M. (2010). Guía de Arquitectura N-Capas Orientada al Dominio con .NET 4.0. Microsoft Architecture. https://resultados.qualab.com.pe/resources/guia.pdf

[18] Fajar, A., Novianti, E., & Firmansyah, F. (2020). Design and Implementation of Microservices System Based on Domain-Driven Design. International Journal of Emerging Trends in Engineering Research, 8(7), 3058–3062. https://doi.org/10.30534/ijeter/2020/30872020

[19] Chen, C., Dong, C., Cai, J., & Cheng, X. (2019). Business Generator Model Based on Domain Driven Design. Journal of Physics: Conference Series, 1168(5), 1–8. https://doi.org/10.1088/1742-6596/1168/5/052028

[20] Gaitán, C. (2017). Líneas de Productos Software: Generando Código a Partir de Modelos y Patrones. Scientia et Technica, 22(2), 175–181. https://doi.org/10.22517/23447214.9131

[21] Lopez D., Maya E. (2017). Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web. Séptima Conferencia de Directores de Tecnología de Información, TICAL 2017 Gestión de las TICs para la Investigación y la Colaboración, San José, del XX al XX de julio de 2017. Pichincha

[22] Richards, M. (2015). Software Architecture Patterns. O'Reilly Media, Inc.

[23] Carvalho, R. y Vendramel, W. (2019). Padrões de migração de sistemas legados para arquitetura baseada em microsserviços. Revista de Sistemas e Computação, Salvador, v. 9, n. 1, p. 58-74, jan./jun. 2019

[24] Newman, S. Building microservices: designing fine-grained systems. 1st ed. O'Reilly Media, 2015.

[25] D. A. Ruelas Acero. MODELO DE COMPOSICIÓN DE MICROSERVICIOS PARA LA IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE COMERCIO ELECTRÓNICO UTILIZANDO KUBERNETES. Revista de Investigaciones de la Escuela de Posgrado de la Universidad Nacional del Altiplano, vol. 7, nº 3, pp. 728-737, 2018.

[26] Zheng, L. y Wei, B. (2018). Application of microservice architecture in cloud environment project development. Beijing: School of North China Electric Power University

[27] Vera-Rivera, F., Gaona, C. y Astudillo, H. (2019). Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación. Revista Ibérica de Sistemas e Tecnologias de Informação

[28] Vera-Rivera, H. (2018). Método de automatización del despliegue continuo en la nube para la implementación de microservicios. Universidad del Valle, Cali, Colombia

[29] Hernández, R., Fernández, C., & Baptista, P. (2014). Metodología de la investigación (6th ed.). Mc Graw Hill. https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf

[30] scrum.org (sf). ¿Qué es Scrum?. Recuperado el 18 de noviembre del 2022 de https://www.scrum.org/resources/what-is-scrum

[31] Microsoft (2022). Entity Framework Core. Recuperado el 21 de noviembre del 2022 de https://learn.microsoft.com/es-es/ef/core/

[32] K. Siva Prasad, R. (2020). generator-springboot. Obtenido de https://github.com/sivaprasadreddy/generator-springboot

[33] Lewis, J. R. (2002). Psychometric Evaluation of the PSSUQ Using Data from Five Years of Usability Studies. International Journal of Human-Computer Interaction, 14(3&4), 463–488. DOI:10.1080/10447318.2002.9669130

[34] Sauro, J., & Lewis, J. R. (2012). Quantifying the User Experience: Practical Statistics for User Research. Elsevier, USA. DOI:10.1016/B978-0-12-384968-7.00001-1

Apéndices

Apéndice A. *Encuesta de Satisfacción de usuario Post-Uso*

Nro.	PSSUQ (Post-Study System Usability Questionnaire) Versión 3	Totalmente de acuerdo				Totalmente en desacuerdo				Subescalas
		1	2	3	4	5	6	7	NA	
1	En general, estoy satisfecho con lo fácil que es usar la herramienta.									Utilidad del sistema (SYSUSE)
2	Fue sencillo utilizar esta herramienta.									
3	Pude completar las tareas y escenarios rápidamente usando esta herramienta.									
4	Me sentí cómodo usando esta herramienta.									
5	Fue fácil aprender a usar esta herramienta.									
6	Creo que podría volverme productivo rápidamente usando esta herramienta.									
7	La herramienta dio mensajes de error que claramente me dijeron cómo solucionar problemas.									Calidad de la información (INFOQUAL)
8	Cada vez que cometía un error al usar la herramienta, podía recuperarme fácil y rápidamente.									
9	La información (como ayuda en línea, mensajes en pantalla y otra documentación) proporcionada con esta herramienta fue clara.									
10	Fue fácil encontrar la información que necesitaba.									
11	La información fue efectiva para ayudarme a completar las tareas y escenarios.									
12	La organización de la información en las pantallas de la herramienta fue clara.									
13	La interfaz de esta herramienta fue agradable.									Calidad de la interfaz (INTERQUAL)
14	Me gustó usar la interfaz de esta herramienta.									
15	Esta herramienta tiene todas las funciones y capacidades que espero que tenga.									
16	En general, estoy satisfecho con esta herramienta.									