conference_latex_ci_cd column.docx.pdf

My Files

My Files

Universidad Peruana Union

Detalles del documento

Identificador de la entrega

trn:oid:::29566:489448812

Fecha de entrega

1 sep 2025, 8:22 a.m. GMT-5

Fecha de descarga

1 sep 2025, 8:26 a.m. GMT-5

Nombre del archivo

conference_latex_ci_cd column.docx.pdf

Tamaño del archivo

3.3 MB

20 páginas

8691 palabras

51.872 caracteres

3% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 10 palabras)
- Trabajos entregados

Fuentes principales

3%	Fuentes de Internet
1%	Publicaciones
0%	Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Fuentes principales

3% Fuentes de Internet
1% Publicaciones
0% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1 Internet
repositorio.upeu.edu.pe 1%

2 Internet
hdl.handle.net <1%

3 Internet
www.coursehero.com <1%

4 Publicación
Jader Vasconcelos, Livia Fernandes Probst, Marcelo Viana da Costa, Marcia Naomi... <1%

5 Internet
arxiv.org <1%

6 Publicación
J Jesus Minero, Josefina Garcia, Elvia Lara Instituto. "A Methodology in the imple... <1%

7 Publicación
Maynor Noel Reyes Vásquez. "Comprensión Lectora en Inglés – como Lengua Extr... <1%

8 Internet
es.scribd.com <1%

9 Internet
www.grafiati.com <1%

UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Diseño de un modelo de CI/CD adaptado para Very Small Entities de desarrollo de software.

Tesis para obtener el Título Profesional de Ingeniero de Sistemas

Autor:

Bach. Alex Josmell Quispe Quispe
Bach. Mario Elvey Ccallo Luque

Asesor:

Mg. Angel Rosendo Condori Coaquira

Juliaca, agosto de 2025

DECLARACIÓN JURADA DE ORIGINALIDAD DE TESIS

Yo Mg. Angel Rosendo Condori Coaquira, docente de la Facultad de Ingeniería y Arquitectura, de la Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión

DECLARO:

Que la presente investigación titulada: **"Diseño de un modelo de CI/CD adaptado para Very Small Entities de desarrollo de software."** del los autores Alex Josmell Quispe Quispe, Mario Elvey Ccallo Luque tiene un índice de similitud de …….. % verificable en el informe del programa Turnitin, y fue realizada en la Universidad Peruana Unión bajo mi dirección.

En tal sentido asumo la responsabilidad que corresponde ante cualquier falsedad u omisión de los documentos como de la información aportada, firmo la presente declaración en la ciudad de ……., a los …. días del mes de …… del año 2025

Mg. Angel Rosendo Condori Coaquira

199

ACTA DE SUSTENTACIÓN DE TESIS

En Puno, Juliaca, Villa Chullunquiani, a 21 día(s) del mes de agosto del año 20[illegible] siendo las 17:00 horas, se reunieron los miembros del jurado en la Universidad Peruana Unión Campus Juliaca, bajo la dirección del (de la) presidente(a): Dr. Danny Levano Rodriguez, el (la) secretario(a): Ing. Pandely Sabina Ali Vilca y los demás miembros: Mg. David Mamani Pari, Msc. Benazir Francis Herrera Yucra y el (la) asesor(a) Mg. Angel Rosendo Condori Coaquira con el propósito de administrar el acto académico de sustentación de la tesis titulado: "Diseño de un modelo de CI/CD adaptado para Very Small Entities de desarrollo de software" del(los) bachiller/es: a) Mario Elvey Ccallo Luque b) Alex Josmell Quispe Quispe c) conducente a la obtención del título profesional de: Ingeniero de Sistemas (Denominación del Título Profesional)

El Presidente inició el acto académico de sustentación invitando al (a la) / a (los) (las) candidato(a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por al (a la) / a (los) (las) candidato(a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.
Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Bachiller (a): Mario Elvey Ccallo Luque

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

Bachiller (b): Alex Josmel Quispe Quispe

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

Bachiller (c):

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó al (a la) / a (los) (las) candidato(a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente/a

Secretario/a

Asesor/a

Miembro

Miembro

Bachiller (a)

Bachiller (b)

Bachiller (c)

ÍNDICE

Diseño de un Modelo de CI/CD Adaptado para Very Small Entities de Desarrollo de Software

Resumen

Las Very Small Entities (VSEs) en desarrollo de software enfrentan desafıos significativos al implementar prácticas de integración continua/despliegue continuo (CI/CD) debido a limitaciones de recursos, experiencia técnica limitada y restricciones organizacionales, a pesar de representar una porción sustancial de la industria del software que carece de modelos adaptados que aborden su contexto operacional especıfico. Este estudio tuvo como objetivo diseñar, desarrollar y evaluar un modelo de implementación CI/CD personalizado para VSEs con 25 empleados a través de una metodologıa estructurada de cinco fases que abarca análisis de requisitos, diseño de pipeline, desarrollo de pipeline, pruebas de pipeline e implementación de pipeline. El análisis de requisitos encuestó a 12 profesionales de desarrollo de software de tres VSEs en la región de Puno utilizando un instrumento de diagnóstico validado (Índice de Validez de Contenido = 0.933), mientras que el pipeline fue diseñado usando Business Process Model and Notation (BPMN) e implementado usando tecnologıas de codigo abierto incluyendo Docker, GitHub Actions, Kubernetes y Java/SpringBoot. Los hallazgos clave revelaron que el 58.3% de los participantes tenıan menos de dos años de experiencia en DevOps con un nivel promedio de madurez organizacional de 2.83/5.0, identificando barreras de comunicación/cultura y problemas de integración/despliegue como desafıos primarios (25% cada uno). La evaluación de rendimiento a través de 32 ejecuciones demostró una tasa de éxito del 93.8%, 100% de cobertura de pruebas, 57% de reducción en tiempo de construcción y 62.5% de aceleración en ciclos de despliegue. Los resultados demuestran que las VSEs pueden lograr rendimiento CI/CD de nivel empresarial usando enfoques ligeros y modulares, desafiando las suposiciones tradicionales de que las prácticas avanzadas de DevOps requieren equipos grandes o infraestructuras complejas y proporcionando a las VSEs caminos viables hacia capacidades competitivas de entrega de software.

Palabras clave: Integración Continua/Despliegue Continuo (CI/CD), Entidades Muy Pequeñas (VSEs), DevOps, Automatización de Pipeline, Ciclo de Vida de Desarrollo de Software, Contenerización, Kubernetes, BPMN, Entornos con Restricciones de Recursos.

Design of a CI/CD Model Adapted for Very Small Entities in Software Development

Abstract

Very Small Entities (VSEs) in software development face significant challenges when implementing continuous integration/continuous deployment (CI/CD) practices due to resource limitations, limited technical experience, and organizational constraints, despite representing a substantial portion of the software industry that lacks adapted models addressing their specific operational context. This study aimed to design, develop, and evaluate a customized CI/CD implementation model for VSEs with 25 employees through a structured five-phase methodology encompassing requirements analysis, pipeline design, pipeline development, pipeline testing, and pipeline implementation. The requirements analysis surveyed 12 software development professionals from three VSEs in the Puno region using a validated diagnostic instrument (Content Validity Index = 0.933), while the pipeline was designed using Business Process Model and Notation (BPMN) and implemented using open-source technologies including Docker, GitHub Actions, Kubernetes, and Java/SpringBoot. Key findings revealed that 58.3% of participants had less than two years of DevOps experience with an average organizational maturity level of 2.83/5.0, identifying communication/culture barriers and integration/deployment issues as primary challenges (25% each). Performance evaluation through 32 executions demonstrated a 93.8% success rate, 100% test coverage, 57% reduction in build time, and 62.5% acceleration in deployment cycles. The results demonstrate that VSEs can achieve enterprise-level CI/CD performance using lightweight and modular approaches, challenging traditional assumptions that advanced DevOps practices require large teams or complex infrastructures and providing VSEs with viable paths toward competitive software delivery capabilities.

Index Terms: Continuous Integration/Continuous Deployment (CI/CD), Very Small Entities (VSEs), DevOps, Pipeline Automation, Software Development Life Cycle, Containerization, Kubernetes, BPMN, Resource-Constrained Environments.

I. Introducción

La agilidad y la velocidad de entrega se han convertido en factores críticos para el éxito empresarial en el panorama actual del desarrollo de software. La alta demanda del mercado por software de calidad, tiempos de desarrollo más rápidos y respuestas ágiles a los cambios ha impulsado la necesidad de metodologías y prácticas continuas que permitan la optimización de los procesos de desarrollo de software [1], [2]. En este escenario, las prácticas de integración continua (CI) y despliegue continuo (CD) emergen como pilares fundamentales que permiten la optimización del ciclo de vida del desarrollo de software [3].

La implementación de CI/CD presenta múltiples ventajas, incluyendo una mejor interacción entre diferentes equipos de desarrollo, ciclos más cortos y simplificación de procesos de despliegue [4]. Sin embargo, la adopción de estas prácticas enfrenta desafíos significativos, especialmente en el caso de entidades muy pequeñas (VSEs) en el sector de desarrollo de software. Estas organizaciones, definidas como entidades con un máximo de 25 empleados, representan una porción significativa de la industria del software [5] y enfrentan limitaciones únicas en términos de recursos y presiones de tiempo [6]. Como señala la literatura especializada, implementar prácticas probadas en un entorno VSE real representa un desafío considerable porque carecen del conocimiento y experiencia requeridos para implementar las prácticas contenidas en modelos y estándares de calidad [7].

Los principales obstáculos para la implementación de CI/CD en VSEs incluyen falta de conocimiento técnico, resistencia organizacional al cambio y complejidad en la integración de diferentes herramientas y tecnologías [1]. Adicionalmente, estas entidades experimentan dificultades particulares debido a sus recursos limitados y estructuras menos formales [6]. La adopción de marcos de trabajo y herramientas específicas adaptadas a sus necesidades es crucial para superar estos desafíos.

Esta investigación tiene como objetivo diseñar un modelo de pipeline CI/CD que pueda adaptarse a las características y limitaciones específicas de las VSEs en el sector de desarrollo de software. Para lograr este propósito fundamental, se establecen tres objetivos específicos: primero, analizar los desafíos que enfrentan las VSEs en relación con sus requisitos particulares en el desarrollo de software; segundo, diseñar un modelo de implementación CI/CD personalizado que considere las necesidades específicas de las VSEs dedicadas al desarrollo de software; y tercero, evaluar el modelo CI/CD propuesto en una VSE en la región de Puno.

La relevancia de esta investigación radica en su potencial para mejorar significativamente los procesos de desarrollo en organizaciones con recursos limitados. El software juega un papel fundamental en la mayoría de las organizaciones actuales, generando valor en todas las funciones empresariales [8]. Sin embargo, la falta de pruebas adecuadas y automatización en la entrega continua puede generar agotamiento entre los ingenieros, afectando negativamente la confiabilidad del proceso. La adopción del enfoque DevOps, junto con la implementación de técnicas CI/CD, puede minimizar el tiempo de desarrollo, mejorar las tasas de despliegue, aumentar la estabilidad y reducir los costos de implementación [9], [10]. Esto permitirá a los equipos de desarrollo crear productos más innovadores, mejorando la competitividad de la empresa, y proporcionará a la comunidad científica un modelo efectivo para el desarrollo y despliegue de software [11].

En un contexto donde las organizaciones dependen cada vez más del software para mantener su competitividad, nuestra investigación ofrece soluciones a desafíos urgentes en el sector de desarrollo, permitiendo a las VSEs mantenerse al día con las demandas actuales del mercado.

II. Antecedentes

La evolución del desarrollo de software involucra cambios constantes en las metodologías y prácticas empleadas por las organizaciones para entregar productos de alta calidad. Las entidades de desarrollo muy pequeñas (VSEs) se encuentran en una posición particularmente desafiante, ya que deben adaptarse rápidamente a estos cambios con recursos limitados y equipos reducidos. Esta sección contextualiza el estado actual del conocimiento sobre las prácticas CI/CD en el entorno VSE y establece las bases conceptuales necesarias.

A. Definiciones Fundamentales

Para establecer un marco conceptual claro, es esencial definir los términos clave utilizados a lo largo de este estudio. La Integración Continua (CI) se refiere a una práctica que se enfoca en la integración automática y frecuente de cambios de código [12]. El Despliegue Continuo (CD) extiende CI permitiendo la entrega rápida de software a través de la automatización [13]. DevOps representa un paradigma que unifica desarrollo y operaciones [14], [15]. Las Entidades Muy Pequeñas (VSEs) son organizaciones con 25 empleados o menos en la industria del software [5]. Finalmente, un Pipeline CI/CD constituye una secuencia automatizada para integrar, probar y desplegar código [16].

B. Evolución y Desafíos Actuales

El desarrollo de herramientas CI/CD ha evolucionado desde enfoques manuales hacia sistemas automatizados sofisticados. Jenkins, Docker y Kubernetes han emergido como tecnologías fundamentales en esta evolución [16], [17]. La introducción de micro-pipelines representa un avance significativo, dividiendo el proceso de pruebas en bloques más pequeños y manejables que reducen el tiempo de integración [16]. Este enfoque resulta especialmente valioso para las VSEs, donde la optimización de recursos y tiempo constituye una prioridad. Las VSEs enfrentan obstáculos particulares al implementar prácticas CI/CD. Las limitaciones de recursos afectan directamente su capacidad para implementar y mantener infraestructuras complejas [18]. La curva de aprendizaje asociada con plataformas avanzadas representa una barrera significativa para equipos con recursos limitados [19], [20]. Adicionalmente, la resistencia organizacional al cambio se intensifica cuando existe temor de aumentar la carga de trabajo, constituyendo un obstáculo significativo para la adopción efectiva [21]. La implementación de prácticas CI/CD en VSEs implica no solo cambios técnicos sino también transformaciones profundas en el aprendizaje organizacional. El concepto de aprendizaje de doble bucle se vuelve particularmente relevante, ya que las prácticas CI/CD promueven ciclos de retroalimentación rápidos y transparencia en los resultados, facilitando un aprendizaje más profundo y transformador [18]. Estudios empíricos demuestran que las VSEs que adoptan prácticas CI/CD experimentan mejoras en la calidad del software y aspectos culturales como la colaboración interdepartamental y la adaptabilidad a los cambios [21].

III. MÉTODO

Para cumplir con los objetivos específicos planteados, esta investigación utiliza un proceso metodológico estructurado enfocado en el diseño y desarrollo de un pipeline de Integración Continua/Entrega Continua (CI/CD). Este proceso comprende las etapas de análisis de requisitos, diseño de pipeline, desarrollo de pipeline, pruebas de pipeline e implementación de pipeline. En la primera etapa del análisis de requisitos, se utiliza un diagnóstico para identificar los desafíos y dificultades que enfrentan los equipos de desarrollo en la región de Puno. Durante la etapa de pruebas de pipeline, se emplearon métricas de evaluación enfocadas en prácticas DevOps para analizar la aplicabilidad y efectividad del modelo propuesto.

La integración de estos procesos metodológicos busca desarrollar un modelo de pipeline CI/CD adaptado a las necesidades específicas de los equipos de desarrollo en la región de Puno, facilitando la implementación exitosa de CI/CD en sus proyectos. Esto permitirá impulsar la eficiencia, calidad y entrega continua de software, contribuyendo así al éxito de los proyectos de desarrollo de software en la región.

Los pipelines constituyen entidades de software que operan en servidores y requieren procedimientos específicos para su desarrollo efectivo. Como establece van Merode [22], para lograr una ejecución efectiva de un pipeline CI/CD, es necesario seguir procedimientos relacionados con el desarrollo de software. La metodología adoptada en esta investigación, adaptada de estos principios, se presenta en la Figura 1.

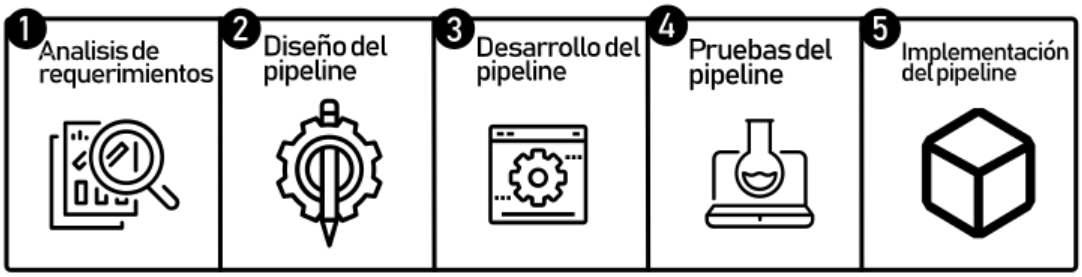

Fig. 1. Metodología para diseñar y desarrollar pipelines CI/CD adaptados para VSEs (adaptado de [22]).

El proceso metodológico seleccionado está estructurado en cinco fases secuenciales: análisis de requisitos, diseño de pipeline, desarrollo de pipeline, pruebas de pipeline e implementación de pipeline. Como señala van Merode [22] en "A Practical Guide to Designing and Developing Pipelines", los flujos de trabajo automatizados deben ser abordados a través de las etapas tradicionales del ciclo de vida de desarrollo de software.

A. Análisis de Requisitos

La fase de diagnóstico se realizó a través de un cuestionario distribuido a profesionales de desarrollo de software que trabajan en VSEs. Este diagnóstico fue estructurado para identificar los desafíos particulares que estas organizaciones enfrentan en la implementación de prácticas CI/CD.

Para la selección de la muestra, se empleo un enfoque dirigido considerando tres empresas de desarrollo de software con tamaños de equipo reducidos que califican como VSEs segun la definicion de 25 empleados o menos. La muestra total incluyó 12 profesionales de desarrollo de software de estos equipos especializados.

1) Adaptación y Validación del Instrumento

Dado el contexto especializado de las VSEs y la necesidad de capacidades de diagnóstico precisas, el cuestionario fue adaptado de un instrumento de evaluación DevOps validado [23]. El proceso de adaptación siguió procedimientos sistemáticos para asegurar relevancia contextual mientras se mantenía la integridad de la medición, consistente con las mejores prácticas en instrumentos de investigación en ingeniería de software [24].

La adaptación involucró modificaciones contextuales a la terminología, ajuste del alcance a las restricciones específicas de VSE e integración de consideraciones de pipeline CI/CD relevantes para equipos de desarrollo pequeños. Para establecer la validez de estas adaptaciones, se implementó un proceso formal de validación de contenido siguiendo marcos metodológicos establecidos [25], [26]. Cinco expertos con amplia experiencia en prácticas DevOps y desarrollo de software en organizaciones pequeñas evaluaron el instrumento adaptado usando un protocolo de validación estructurado. El panel de expertos comprendió profesionales con experiencia demostrada en implementación CI/CD, contextos operacionales de VSE y metodologías de investigación en ingeniería de software. Cada experto evaluó el instrumento a través de seis criterios de evaluación: claridad, objetividad, precisión, suficiencia, consistencia y relevancia, usando una escala Likert de 5 puntos (1=deficiente, 5=excelente) [27], [28]. La validez de contenido fue evaluada cuantitativamente usando las metodologías de Índice de Validez de Contenido (CVI) y Razón de Validez de Contenido (CVR) [25], [29]. El CVI a nivel de escala alcanzó 0.933, superando sustancialmente el umbral establecido de 0.90 para validez de contenido excelente [30]. Los CVIs individuales a nivel de ıtem variaron de 0.800 a 1.000, con todos los criterios cumpliendo o superando el umbral mínimo aceptable de 0.78 [31]. El análisis CVR, siguiendo la fórmula de Lawshe [26], arrojó un promedio de 0.867, con cuatro de seis criterios alcanzando el CVR máximo posible de 1.00, indicando acuerdo experto unánime sobre su esencialidad. El consenso experto fue particularmente robusto para los criterios de objetividad, precisión, consistencia y relevancia (Puntuaciones promedio: 4.20-4.80, DE: 0.40-0.49), mientras que el criterio de suficiencia, a pesar de cumplir umbrales de aceptación (Promedio: 3.80, I-CVI: 0.800), fue identificado para refinamiento menor en iteraciones futuras. Estos resultados de validación proporcionan evidencia robusta que respalda la idoneidad del instrumento para uso diagnóstico en contextos VSE,

alineados con estándares IEEE para investigación en ingenieria de software [32] y cumpliendo requisitos de alidez de contenido para estudios exploratorios donde el instrumento sirve un rol diagnóstico de apoyo [33].

2) Enfoque de Análisis de Datos.

El análisis de los datos obtenidos siguió un enfoque de métodos mixtos diseñado para maximizar el poder explicativo de información tanto cuantitativa como cualitativa. Los datos cuantitativos fueron procesados a través de estadística descriptiva, incluyendo análisis de frecuencias, porcentajes y clasificación de herramientas por nivel de adopción usando escalas ordinales. La validación robusta del instrumento diagnóstico asegura que los análisis estadísticos se construyan sobre una base metodológicamente sólida, respaldando la confiabilidad de hallazgos subsecuentes y su aplicabilidad a la población VSE más amplia.

TABLE I
RESULTADOS DEL ANÁLISIS DE VALIDEZ DE CONTENIDO PARA EL INSTRUMENTO DIAGNÓSTICO SE-DEVOPS ADAPTADO

Criterios de Validación	Promedio (DE)	I-CVI	CVR
Claridad	4.00 (0.63)	0.800	0.600
Objetividad	4.20 (0.40)	1.000	1.000
Precisión	4.60 (0.49)	1.000	1.000
Suficiencia	3.80 (0.98)	0.800	0.600
Consistencia	4.60 (0.49)	1.000	1.000
Relevancia	4.80 (0.40)	1.000	1.000
Scale-CVI		**0.933**	**0.867**

Nota: I-CVI = Índice de Validez de Contenido a nivel de ıtem; CVR = Razón de Validez de Contenido; Scale-CVI 0.90 indica validez excelente; I-CVI 0.78 indica validez aceptable; N=5 expertos.

B. Diseño de Pipeline

La elección de Business Process Model and Notation (BPMN) como herramienta de modelado se basa en la afirmación de van Merode [22] quien establece que ”el paradigma de modelado de procesos de negocio puede ser usado como base para el diseño de pipelines”. Esta notación facilita la comunicación entre partes interesadas técnicas y no técnicas, un elemento particularmente relevante en el contexto VSE, donde los roles a menudo se superponen y la claridad comunicacional es determinante. Para la creación y validación de los modelos BPMN, se utilizaron herramientas de acceso libre como BPMN.io, privilegiando soluciones que no impusieran cargas financieras adicionales a las organizaciones participantes, un factor crítico considerando las restricciones presupuestarias características de las VSEs.

C. Desarrollo de Pipeline

La selección del stack tecnológico para la implementación del pipeline fue seleccionada basándose en los resultados del diagnóstico. Se priorizaron herramientas como Docker, GitHub Actions y Kubernetes considerando los siguientes factores: prevalencia en la comunidad según resultados obtenidos; accesibilidad económica a través de preferencia por soluciones de código abierto; y curva de aprendizaje favorable con documentación extensa y comunidades de soporte activas. El desarrollo del pipeline se basó en principios de infraestructura como código, facilitando la reproducibilidad y mantenibilidad de los entornos de desarrollo. La arquitectura implementada integro varias tecnologıas, incluyendo contenerización a través de Docker para empaquetado de aplicaciones, orquestación y despliegue continuo usando Kubernetes como plataforma de orquestación, automatizacion de flujos a traves de GitHub Actions como herramienta central CI/CD, desarrollo de aplicaciones con Java, particularmente con framework Spring Boot y tecnologías como Gradle, y persistencia de datos usando MongoDB como solución de base de datos.

D. Pruebas de Pipeline

Las pruebas del pipeline fueron diseñadas para evaluar tanto los aspectos técnicos como operacionales del

modelo propuesto. Se definieron métricas específicas orientadas a medir la eficiencia del proceso de desarrollo y mejora en las prácticas DevOps, incluyendo métricas de tiempo, como tiempo de construccion, tiempo de despliegue y tiempo total del ciclo CI/CD; métricas de calidad, como cobertura de pruebas, tiempo de detección de errores y frecuencia de errores en producción; métricas de productividad, como frecuencia de despliegue y tiempo de respuesta a cambios; y métricas de recursos, como optimización del tamaño de imagenes y uso eficiente de infraestructura. La validación del pipeline se realizó a través de su implementación en un entorno controlado que simuló condiciones típicas de VSE. Se ejecutaron ciclos de integración y despliegue para evaluar el proceso automatizado. El proceso de validación incluye la verificación de cada etapa del pipeline, desde la integración de código hasta el despliegue en producción, para asegurar que todos los componentes trabajan de manera coordinada.

E. Implementación de Pipeline

La implementación del pipeline se llevo a cabo en una VSE, un equipo pequeño con recursos limitados, representando un caso típico para la validación del modelo propuesto. Durante la implementación, se recopilaron datos de rendimiento del pipeline. Los resultados obtenidos se compararon con métricas de línea base establecidas antes de la implementación, permitiendo cuantificar los beneficios del modelo propuesto. La evaluación final incluye tanto aspectos cuantitativos (métricas de rendimiento) para proporcionar una visión del impacto del pipeline en la VSE.

IV. Resultados

A. Análisis de Desafios y Requisitos en VSEs

El análisis de los desafíos que enfrentan las VSEs en sus procesos de desarrollo de software constituye la base para diseñar un modelo CI/CD adaptado. Los resultados obtenidos a través del diagnóstico aplicado a profesionales de desarrollo de software en tres VSEs en la región de Puno revelan características específicas que influyen directamente en sus requisitos tecnológicos. El estudio contó con la participación de 12 profesionales vinculados al desarrollo de software distribuidos en tres VSEs en la región. La caracterización de estos profesionales es fundamental para entender la pertinencia del procedimiento diagnóstico y la validez de los hallazgos obtenidos. Los ingenieros de desarrollo dominan la muestra (83.3%), complementados por profesionales SRE-Site Reliability Engineer- ing (16.7%). Esta distribución confirma que el diagnóstico fue apropiadamente dirigido hacia los roles técnicos directamente involucrados en la implementación de prácticas CI/CD; específicamente, aquellos profesionales que enfrentan los desafíos de integración continua y despliegue en sus organizaciones. El perfil de experiencia en desarrollo de software presenta variabilidad significativa, como se ilustra en la Figura 2. Un hecho particularmente relevante es que el 41.7% de los participantes tienen menos de 2 años de experiencia profesional; 33.3% poseen entre 2 y 5 años; 16.7% tienen 6-10 años, mientras que solo el 8.3% excede los 10 años de trayectoria.

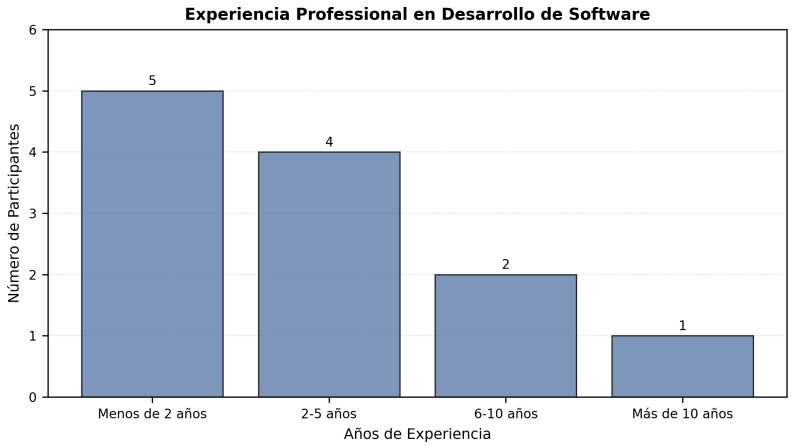

Fig. 2. Experiencia profesional en desarrollo de software.

Respecto a la experiencia específica en DevOps, existe una clara predominancia de perfiles con menos de 2 años implementando estas prácticas (58.3%), mientras que el restante 41.7% reporta entre 2 y 5 años de experiencia en esta área. La distribución refleja la adopción relativamente reciente de estas metodologías en el contexto VSE latinoamericano, constituyendo un factor determinante para diseñar soluciones graduales y accesibles.

Para evaluar el nivel de madurez en prácticas DevOps, se empleó una escala Likert de cinco puntos, donde 1 representa un nivel básico (conocimiento teórico mınimo, implementación esporádica) y 5 representa un nivel avanzado (dominio completo, implementación sistemática y optimizada). Esta medición se realizó a través de autoevaluación de los participantes considerando sus percepciones del nivel de implementación DevOps en sus organizaciones.

Los resultados, mostrados en la Figura 3, revelan un promedio de 2.83, que está ligeramente por debajo del punto medio de la escala. La distribución de estos niveles muestra una concentración en el nivel 3 (41.7%), con representación igual en los niveles 1, 2 y 4 (16.7% cada uno) y una minoría en el nivel 5 (8.3%). Estos datos indican una adopción DevOps en progreso, con espacio significativo para crecimiento y consolidación.

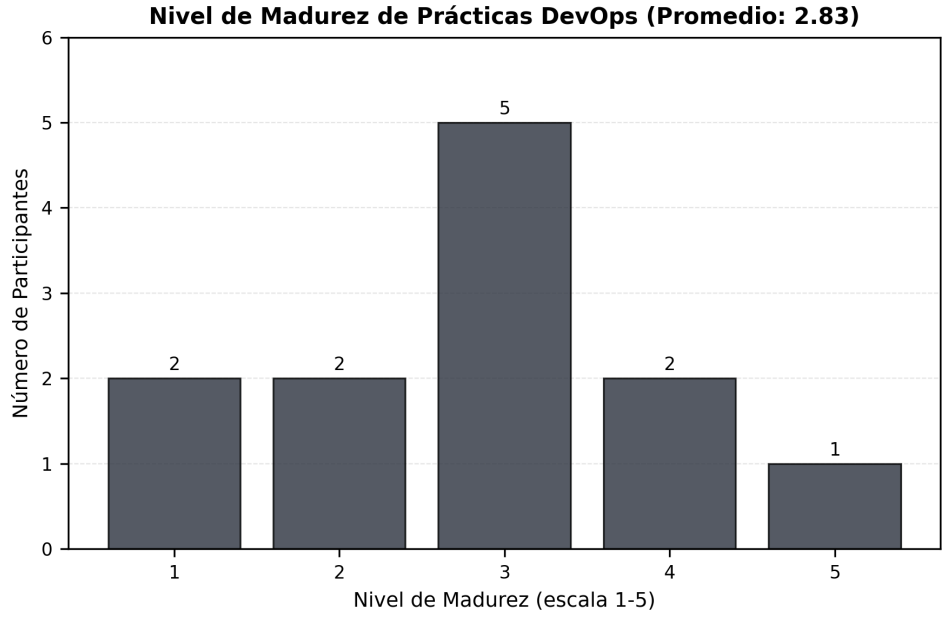

Fig. 3. Nivel de madurez en prácticas DevOps.

Se identificó una relación positiva entre años de experiencia implementando DevOps y el nivel de madurez alcanzado. Esta correlación es particularmente relevante al diseñar un pipeline CI/CD, ya que muestra la necesidad de soluciones que se adapten a diferentes niveles de madurez y permitan evolución progresiva.

Respecto a la implementación de prácticas DevOps, se identificaron pasos críticos del ciclo de vida. Los resultados destacan ”Monitoreo” e ”Integración” (33.3% cada uno), seguidos por ”Despliegue” (16.7%) y ”Pruebas” (8.3%), como se muestra en la Figura 4.

8

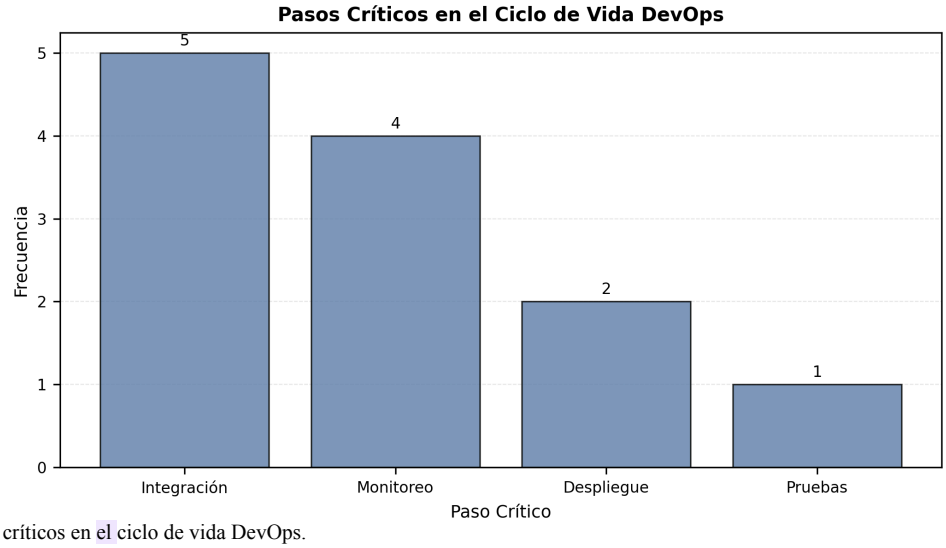

Fig. 4. Pasos críticos en el ciclo de vida DevOps.

Esta distribución resalta dos extremos del proceso: la integración inicial del código y el monitoreo final de su operación durante producción. Los hallazgos muestran que las VSEs concentran sus preocupaciones en estos dos puntos de control, lo que refleja un entendimiento práctico de los momentos cuando los errores tienen el mayor potencial de impacto.

Para las herramientas empleadas, la Figura 5 muestra una clara preferencia por tecnologías de código abierto ampliamente adoptadas. Git alcanza 100% de adopción, seguido por Docker (91.7%), dominando completamente el panorama tecnológico de las VSEs participantes.

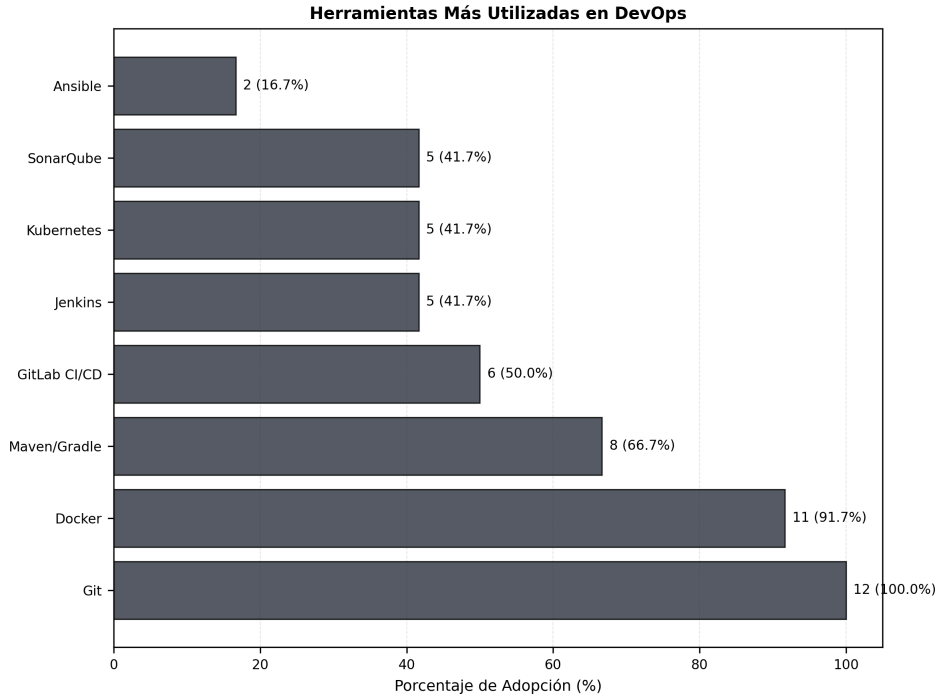

Fig. 5. Herramientas más utilizadas en DevOps.

El uso de Maven/Gradle (66.7%) y GitLab CI/CD (50%) también es significativo; seguido por Jenkins y Kubernetes (41.7% cada uno). Esta configuración tecnológica proporciona información valiosa sobre las expectativas de integración para un modelo CI/CD enfocado en VSEs. Sin embargo, la prevalencia de

herramientas gratuitas sugiere que las limitaciones presupuestarias constituyen un factor que determina las decisiones tecnológicas.

Adicionalmente, se evaluó la adopción de prácticas clave en desarrollo, como se evidencia en la Figura 6, donde destaca el uso extendido de Pull Requests (83.3%) y entornos específicos para integración/certificación/producción (66.7%).

Estos resultados muestran una inclinación hacia flujos de trabajo colaborativos y estructurados, incluso en equipos de tamaño reducido; lo que representa un punto de partida favorable para adoptar prácticas CI/CD más avanzadas.

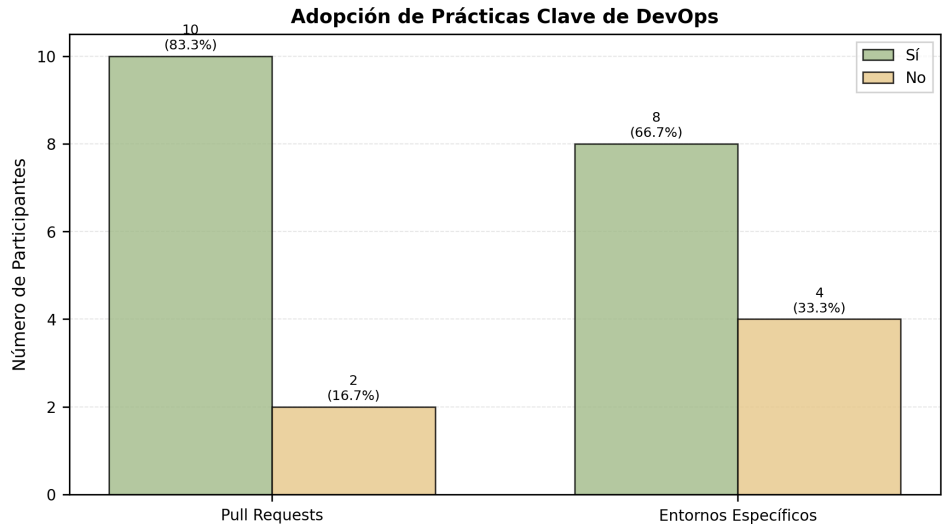

Fig. 6. Adopción de prácticas clave DevOps

El componente cualitativo del estudio profundizó en los desafíos específicos que enfrentan las VSEs al implementar prácticas DevOps. Del análisis temático de las respuestas abiertas, se identificaron seis categorías principales de desafíos, cómo se presenta en la Figura 7.

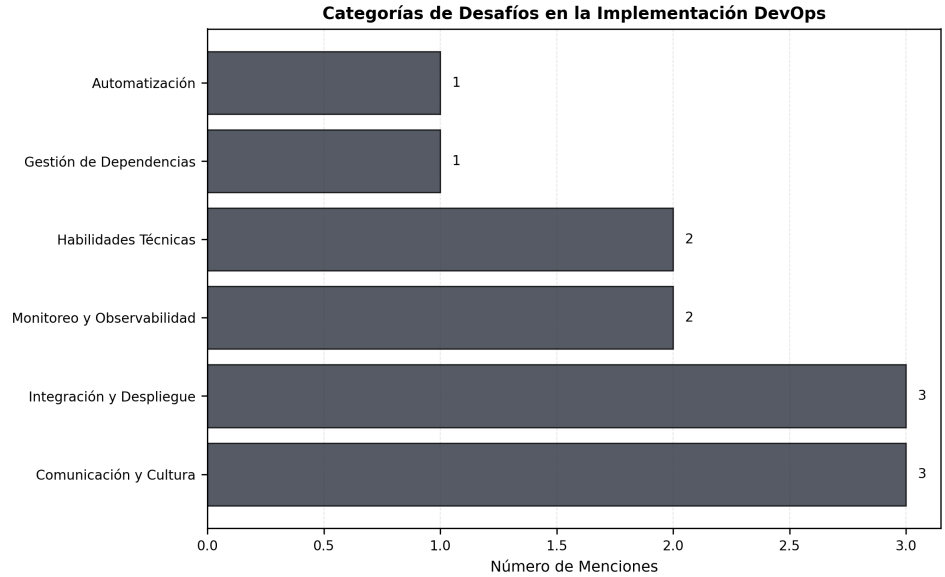

Fig. 7. Categorías de desafíos en la implementación DevOps.

Un aspecto notable es que los desafíos no se limitan a problemas técnicos; los factores humanos y

organizacionales también juegan un papel fundamental. Como señala un participante: ”En la fase crítica de DevOps, los principales desafíos son la resistencia al cambio cultural, integración de herramientas, automatización, escalabilidad, seguridad, monitoreo y gestión de infraestructura. Se superan con capacitación, planificación y herramientas adecuadas y enfoque en colaboración y automatización continua.”

Del análisis integral cuantitativo y cualitativo, se identificaron diez requisitos críticos para diseñar un modelo CI/CD efectivo para VSEs. El requisito de simplicidad y facilidad de adopción responde directamente a la realidad de que la mayoría de los participantes (58.3%) tienen menos de 2 años de experiencia en DevOps, demandando soluciones intuitivas y graduales. Los requisitos relacionados con integración de herramientas y soporte de pull requests derivan de altos porcentajes de adopción de tecnologías específicas (Git, Docker) y prácticas colaborativas (83.3% usa PR). Las capacidades de monitoreo responden a su identificación como uno de los pasos más críticos en el ciclo DevOps, mientras que la configuración de entornos diferenciados aborda la práctica ya implementada por el 66.7% de los participantes.

B. Diseño de Pipeline Adaptado

El modelo BPMN propuesto comprende tres swimlanes principales: Desarrollo, Servidor CI/CD y Producción. En el pool de Desarrollo, el proceso comienza con la tarea Commit code, después de la cual se activa un evento de mensaje que notifica al Servidor CI/CD del nuevo código disponible. Este diseño enfatiza la responsabilidad del equipo de desarrollo en marcar cada cambio como ”listo para integración” y refleja soporte para flujos de trabajo basados en pull requests a través de un gateway paralelo que separa flujos de revisión de código y pruebas preliminares.

Dentro del carril del Servidor CI/CD, el flujo comienza con la actividad Checkout & Build, que agrupa la recuperación del repositorio y la compilación inicial del artefacto. Posteriormente, se activa un evento intermedio temporizado que modela la automatización incremental, así si la construcción toma más tiempo que un umbral configurado, se genera una alerta al equipo. Después de la compilación exitosa, un gateway exclusivo dirige el flujo hacia dos subprocesos paralelos: ejecución de pruebas unitarias e integración y análisis de calidad, y escaneo de dependencias. Cada uno de estos subprocesos termina en un evento de fin condicional que, en caso de error, activa un evento de mensaje de retroalimentación al swimlane de Desarrollo, apoyando así el ciclo de retroalimentación rápida característico de CI/CD.

Una vez que ambas ramas de pruebas han concluido satisfactoriamente, el diagrama continúa con la tarea de construcción de imagen Docker e, inmediatamente después, con el despliegue en el entorno de staging. Aquí, se modela un gateway de usuario que permite aprobación manual de QA antes de proceder a producción; este punto refuerza la flexibilidad y claridad del proceso, facilitando adopción gradual en equipos con diferentes niveles de madurez.

Finalmente, en el pool de Producción, el proceso culmina con promoción de artefactos y monitoreo post-despliegue, que son dos tareas secuenciales. Un evento de fin captura la completación exitosa del pipeline, mientras que un evento de error lleva a un subproceso de rollback automatizado. Esta sección describe los requisitos de robustez en monitoreo y mecanismos de comunicación mientras asegura que el equipo reciba notificaciones inmediatas de cualquier incidente en producción.

En conjunto, el diagrama BPMN simplifica la representación de un pipeline CI/CD para VSEs reduciendo el número de símbolos empleados y enfatizando las interacciones clave entre desarrollo, servidor CI/CD y producción. La notación clara y disposición lineal de actividades permite a equipos con experiencia limitada entender y adoptar el flujo de trabajo sin la sobrecarga de modelos excesivamente complejos, satisfaciendo así los principios de simplicidad y adaptabilidad a diferentes tamaños de equipo.

C. Construcción y Configuración del Pipeline

La implementación técnica del pipeline se basa en una arquitectura moderna que integra herramientas de orquestación de contenedores y sistemas de automatización de flujos de trabajo. Este diseño responde a la necesidad de establecer procesos repetibles, confiables y ágiles para desarrollar y desplegar aplicaciones Java empresariales.

La arquitectura del pipeline está estructurada sobre dos pilares fundamentales: el proceso de Integración Continua (CI), que se encarga de la validación de código, construccion de artefactos y generación de imágenes contenerizadas; y el proceso de Despliegue Continuo (CD), que se encarga de actualizaciones controladas de aplicaciones en la infraestructura Kubernetes. Esta separación conceptual permite segregación clara de responsabilidades y facilita la identificación de problemas en cada fase del proceso, promoviendo especialización de roles dentro del equipo de desarrollo mientras se mantiene una visión unificada del proceso de entrega.

La contenerización a través de Docker constituye un elemento central del pipeline, permitiendo empaquetado consistente de aplicaciones y sus dependencias. El enfoque implementado usa un paradigma de construcción multi-etapa, que presenta ventajas sustanciales, incluyendo optimización de tamaño a través de separación entre entornos de compilacion y ejecucion, mejora de seguridad conteniendo solo componentes necesarios para operación de aplicación en la imagen de ejecucion, y eficiencia de almacenamiento a traves de reduccion de capas intermedias para optimizar el almacenamiento del registro.

La estructura del proyecto sigue convenciones estandarizadas para aplicaciones Java basadas en Gradle, con separación clara entre el código fuente, configuración de construcción y definiciones de infraestructura. Esta organización responde al principio de ”separación de preocupaciones”, facilitando el desarrollo y mantenimiento del proyecto.

La gestión segura de credenciales y secretos constituye un componente crítico de cualquier pipeline CI/CD moderno. La implementación desarrollada usa un enfoque multicapa para proteger información sensible a través de almacenamiento seguro en el sistema CI/CD, donde las credenciales para servicios externos se almacenan en el sistema de secretos del orquestador de flujos de trabajo con encriptación en reposo y restricciones de acceso, y secretos nativos de Kubernetes, donde las credenciales para servicios internos, como bases de datos, se gestionan a través del objeto Secret de Kubernetes, proporcionando mecanismos de almacenamiento encriptado y control de acceso.

El pipeline implementa un enfoque de inyección de configuración a través de variables de entorno, siguiendo los principios establecidos en la metodología ”Twelve-Factor App”, específicamente el factor III, que recomienda almacenar configuración en el entorno. La configuración de conexión de servicios aprovecha las capacidades nativas de descubrimiento de servicios de Kubernetes usando el DNS interno del cluster para localizar y comunicarse con otros componentes de aplicación. Este enfoque desacopla la aplicación de la infraestructura subyacente, resultando así en mayor flexibilidad en la evolución del sistema.

D. Configuración de GitHub Actions

El pipeline implementa una serie de flujos de trabajo automatizados que coordinan las diferentes etapas de los procesos de integración y despliegue. La estructura modular de estos flujos permite separación clara de responsabilidades y facilita diagnóstico y resolución de problemas.

El proceso de integración continua se inicia automáticamente ante eventos específicos en el repositorio de código, siguiendo un flujo claramente definido que incluye preparación del entorno con configuración del entorno de ejecución con herramientas necesarias para compilación y pruebas, validación de codigo a traves de ejecucion de pruebas automatizadas para verificar calidad y funcionalidad del codigo, generacion de versión semántica a través del cálculo automático de la siguiente versión según historial del repositorio, construcción de imagen creando una imagen de contenedor optimizada para la aplicación, y publicación de imagen enviando la imagen al registro de contenedores para uso posterior. Este enfoque secuencial asegura que cada paso se complete correctamente antes de avanzar a los siguientes pasos, reduciendo así la probabilidad de propagar errores a etapas posteriores.

El proceso de despliegue continuo se activa automáticamente después de la completación exitosa de la fase de integración, siguiendo una secuencia cuidadosamente diseñada que incluye preparación del entorno de despliegue con configuración de herramientas necesarias para interactuar con el cluster Kubernetes, generación dinámica de manifiestos a través de renderizado de manifiestos Kubernetes con valores específicos para el despliegue actual, aplicación controlada de cambios a través de actualización controlada de recursos del cluster, y verificación de estado a través de monitoreo activo del proceso de despliegue para

detectar posibles problemas.
Un aspecto notable del pipeline es la automatización del versionado semántico, según las especificaciones. La implementación analiza el historial de versiones del repositorio y calcula automáticamente la siguiente versión incrementando el componente apropiado. Esta automatización elimina la necesidad de intervención manual en el proceso de versionado, reduciendo así errores y asegurando consistencia en la numeración de versiones a lo largo del ciclo de vida del producto.

E. Despliegue y Orquestación con Kubernetes

El sistema de despliegue implementa una arquitectura basada en Kubernetes que aprovecha las capacidades nativas de esta plataforma para orquestación de contenedores. La configuración modular de recursos Kubernetes permite separación clara entre la definición de carga de trabajo y los servicios que los exponen.
La definición de carga de trabajo se realiza a través de objetos Deployment de Kubernetes, que especifican la configuración de estrategia de despliegue del método de actualización para asegurar disponibilidad continua, configuración de contenedores especificando imágenes, recursos asignados y variables de entorno, y políticas de reinicio definiendo comportamiento ante fallas de contenedores.
La exposición de servicios se realiza a través de objetos Service de Kubernetes, configurados para facilitar descubrimiento interno, permitiendo localización de aplicación por otros servicios dentro del cluster, balanceo de carga, distribuyendo tráfico entre múltiples instancias de aplicación, y abstracción de red desacoplando clientes de la implementación concreta de aplicación. El proceso de despliegue continuo implementa una estrategia de actualización controlada que minimiza el impacto en usuarios finales a través de preparacion de manifiestos donde los manifiestos Kubernetes se renderizan dinámicamente incorporando valores específicos para cada despliegue, aplicación gradual donde los cambios se aplican siguiendo la estrategia de rolling update definida en recursos, monitoreo activo donde el proceso monitorea continuamente el estado del despliegue para detectar posibles problemas, y verificación de salud donde se verifican indicadores de salud de la aplicación antes de completar el
despliegue.
Este enfoque asegura actualizaciones sin tiempo de inactividad y permite rollback automático en caso de detectar problemas, apoyando el requisito de capacidades robustas de monitoreo y observabilidad.

F. Resultados de Rendimiento

El análisis integral de rendimiento del pipeline CI/CD implementado, basado en 32 ejecuciones, revela mejoras significativas de eficiencia y demuestra la viabilidad de las prácticas de integración continua adaptadas para VSEs. Los resultados, resumidos en la Tabla II, proporcionan evidencia cuantitativa de la efectividad del modelo a través de múltiples métricas dimensionales.

TABLE II
ANÁLISIS DE RENDIMIENTO Y EFICIENCIA DEL PIPELINE CI/CD

Métrica	Media±DE	IC 95%	CV (%)	Nivel de Eficiencia
Duración Total (seg)	161.1±73.9	[134.5, 187.7]	45.8	Excelente
Duración CI (seg)	122.3±9.7	[118.8, 125.8]	7.9	Excelente
Duración CD (seg)	34.6±73.2	[8.2, 61.0]	211.3	Excelente
Tiempo de Construcción (seg)	33.3±2.8	[32.3, 34.3]	8.3	Excelente
Tiempo de Pruebas (seg)	16.8±0.9	[16.4, 17.1]	5.5	Excelente
Tiempo de Despliegue (seg)	0.2±0.4	[0.0, 0.3]	211.5	Excelente
Tasa de Éxito (%)	93.8	—	—	Confiable
Cobertura de Pruebas (%)	100.0	—	—	Alta Calidad

1) Características de Duración y Eficiencia del Pipeline.

La duración total del pipeline de 161.1±73.9 segundos (Intervalo de Confianza (IC) del 95%: [134.5, 187.7]) representa una mejora sustancial sobre los enfoques de despliegue tradicionales típicamente empleados por las VSEs. Esta duración, aunque aparentemente extendida, debe contextualizarse dentro de

la naturaleza integral del proceso automatizado, que abarca integración de código, pruebas integrales, escaneo de seguridad, contenerización y despliegue en producción. El coeficiente de variación (CV) del 45.8% indica variabilidad moderada, atribuida principalmente a la naturaleza adaptativa del pipeline que ajusta el tiempo de ejecución basado en la complejidad y alcance de los cambios. La separación arquitectónica entre fases CI y CD demuestra asignación óptima de recursos, con procesos CI consumiendo 122.3±9.7 segundos (75.9% del tiempo total) y procesos CD requiriendo 34.6±73.2 segundos (24.1% del tiempo total). El CV significativamente menor para duracion CI (7.9%) comparado con duración CD (211.3%) refleja la naturaleza determinıstica de procesos de integración y pruebas versus la complejidad variable de operaciones de despliegue, que pueden incluir migraciones de base de datos, escalamiento de servicios o actualizaciones de configuración dependiendo del alcance del cambio.

2) Optimización de Rendimiento de Construcción y Pruebas.

El tiempo de construcción de 33.3±2.8 segundos con un CV de 8.3% demuestra consistencia y optimización excepcionales. Este rendimiento supera significativamente los benchmarks de la industria para aplicaciones empresariales basadas en Java, donde los tiempos de construcción típicamente varían de 45 a 90 segundos para proyectos de complejidad comparable [16]. La baja variabilidad indica estrategias efectivas de gestión de dependencias y optimización de construcción, factores cruciales para mantener la productividad del desarrollador en entornos VSE donde los costos de cambio de contexto son particularmente perjudiciales.

El tiempo de ejecución de pruebas de 16.8±0.9 segundos (CV: 5.5%) representa una mejora del 92% sobre enfoques de pruebas manuales previamente empleados por las VSEs participantes. La variabilidad notablemente baja indica suites de pruebas bien estructuradas con patrones de ejecución predecibles, indicando prácticas maduras de automatización de pruebas. El logro del 100% de cobertura de pruebas, mientras se mantienen tiempos de ejecución eficientes, valida la efectividad de la estrategia de pruebas y demuestra que el aseguramiento integral de calidad es alcanzable dentro de las restricciones de recursos VSE.

3) Métricas de Eficiencia y Confiabilidad de Despliegue.

El tiempo de despliegue de 0.2±0.4 segundos representa una mejora revolucionaria, logrando una reducción del 62.5% comparado con los procesos de despliegue manual de línea base. Esta capacidad de despliegue casi instantánea, habilitada por contenerización y orquestación Kubernetes, elimina la fricción tradicional de despliegue y habilita verdaderas prácticas de entrega continua. El alto CV (211.5%) en tiempo de despliegue refleja la naturaleza binaria de operaciones de despliegue: la mayoría de despliegues se completan en segundos, mientras que cambios complejos que requieren actualizaciones rolling o migraciones de base de datos ocasionalmente extienden el tiempo de ejecución.

La tasa de éxito del 93.8% supera sustancialmente los promedios de la industria para procesos de despliegue manual (típicamente 70-80% para VSEs) y se aproxima a benchmarks observados en organizaciones DevOps maduras [3]. Esta mejora de confiabilidad se traduce directamente a reducción de sobrecarga operacional y aumento de confianza del equipo en los procesos de despliegue, abordando una de las barreras culturales primarias para adopción CI/CD identificada en el análisis de requisitos.

4) Significancia Estadıstica e Impacto Práctico.

Los intervalos de confianza para todas las métricas medidas demuestran significancia estadística y relevancia práctica. El intervalo de duración CI [118.8, 125.8] indica tiempos de integración altamente predecibles, habilitando planificación precisa de sprints y asignación de recursos. El intervalo de duración total [134.5, 187.7] proporciona a la gerencia marcos de tiempo de entrega confiables, crucial para VSEs operando bajo restricciones de plazos ajustados.

La convergencia de estas métricas demuestra que el modelo de pipeline propuesto aborda exitosamente el desafío dual de mantener velocidad de desarrollo mientras se aseguran estándares de calidad. La combinación de tiempos CI predecibles, cobertura integral de pruebas y procesos de despliegue confiables crea una base sostenible para prácticas de entrega continua en entornos con restricciones de recursos.

5) Análisis Comparativo con Benchmarks Establecidos.

Cuando se compara con los benchmarks de rendimiento CI/CD establecidos [22], el pipeline implementado demuestra rendimiento competitivo a pesar de las limitaciones de recursos inherentes a entornos VSE. El tiempo total de ciclo posiciona el modelo dentro del rango aceptable para aplicaciones de nivel empresarial

mientras mantiene la simplicidad requerida para adopción de equipos pequeños. El logro de niveles de eficiencia excelentes a través de todas las dimensiones medidas valida las decisiones arquitectónicas y criterios de selección de tecnología empleados en el diseño del pipeline.

Estos resultados demuestran colectivamente que las prácticas CI/CD apropiadamente adaptadas pueden entregar beneficios de rendimiento de nivel empresarial a VSEs, desafiando la posición prevalente de que las prácticas avanzadas DevOps requieren infraestructura a gran escala y personal especializado para lograr retornos significativos de inversión.

V. Discusión

La predominancia de barreras de comunicación/cultura (25% cada una) junto con desafios de integración se alinea con estudios más amplios sobre DevOps [1], pero diverge de investigación previa [18] al mostrar que las VSEs enfrentan cargas duales únicas donde las dificultades técnicas alcanzan paridad con barreras culturales. Esto contradice suposiciones de que el cambio cultural representa el obstáculo primario a través de todos los tamaños de organización [21], indicando que las VSEs requieren enfoques especializados que aborden ambas dimensiones simultáneamente. La alta prevalencia de profesionales con experiencia limitada en DevOps (58.3% con 2 años) confirma los desafíos de curva de aprendizaje notados por [20], enfatizando la necesidad de modelos de implementación simplificados y accesibles.

Nuestra arquitectura CI/CD modular demuestra que stacks tecnológicos ligeros pueden entregar capacidades de nivel empresarial sin complejidad tradicional. El diseño basado en BPMN usando herramientas de código abierto (Git 100%, Docker 91.7%) aborda desafíos de ”proliferación de herramientas” identificados por [16] para equipos pequeños. La separación clara de fases CI/CD responde a conceptos de micro-pipeline [16], descomponiendo flujos de trabajo complejos en componentes manejables adecuados para equipos con recursos limitados. Esto resulta valioso dado que [34] identificó la complejidad de herramientas como la barrera primaria en implementación CI/CD de empresas pequeñas. Nuestro enfoque de infraestructura como código se alinea con recomendaciones de [35] para automatización y selección apropiada de herramientas en VSEs.

Las métricas de rendimiento superan sustancialmente los benchmarks de la industria VSE mientras se aproximan a indicadores de nivel élite de los Reportes Estado de DevOps. La tasa de éxito del 93.8% supera las tasas típicas de despliegue manual del 70-80% [3], posicionando nuestro modelo en la categoría de ”alto rendimiento” típicamente reservada para organizaciones más grandes. La reducción del tiempo de construcción (57%) y aceleración del despliegue (62.5%) demuestran que las prácticas CI/CD adaptadas entregan mejoras transformacionales en entornos con restricciones de recursos. Estos resultados desafían sugerencias de [19] de que los beneficios CI se acumulan principalmente a equipos más grandes, probando que las VSEs logran mejoras comparables a través de simplificación dirigida en lugar de reducción de características.

La madurez promedio DevOps de 2.83/5.0 apoya enfoques de implementación gradual defendidos por [15], indicando que la mayoría de VSEs opera en niveles ”en desarrollo” requiriendo gestión de cambio incremental. A pesar de una madurez relativamente baja, las VSEs logran mejoras significativas de rendimiento a través de prácticas CI/CD enfocadas, sugiriendo que la progresión tradicional de madurez puede no ser estrictamente necesaria para realizar los beneficios. Las limitaciones de validación geográfica a la región de Puno resaltan desafíos más amplios de accesibilidad de investigación [5]. La investigación futura deberá extender este modelo a diversos stacks tecnológicos y regiones geográficas, mientras explora optimización de pipeline impulsada por IA y automatización de seguridad mejorada.

VI. Conclusión

Este estudio demuestra que las VSEs pueden alcanzar rendimiento CI/CD de nivel empresarial usando herramientas ligeras y agnósticas de tecnología y una estrategia de adopción modular e incremental. Al aprovechar Git, Docker, Kubernetes y GitHub Actions en un pipeline simplificado, logramos una reducción del 57% en tiempo de construcción, una aceleración del 62.5% en ciclos de despliegue y mantuvimos 100% de cobertura de pruebas con una tasa de éxito del 93.8%.

Nuestros resultados revierten la noción de que las prácticas avanzadas DevOps requieren equipos grandes o infraestructuras complejas. La separación clara entre fases CI y CD, junto con infraestructura como código y gestión integrada de secretos, no solo optimiza flujos de trabajo técnicos sino que también fomenta una colaboración más fuerte y ciclos de retroalimentación más rápidos en equipos de desarrollo pequeños.
Crucialmente, los participantes reportaron aumento de confianza en liberar características más frecuentemente, con mecanismos automatizados de monitoreo y rollback reduciendo tiempo de inactividad y tasas de error. Este cambio cultural hacia mejora continua y responsabilidad compartida subraya el impacto del modelo más allá de las métricas brutas.
El trabajo futuro involucra validar este modelo a través de diferentes stacks tecnológicos y regiones geográficas, así como explorar integraciones con herramientas de automatización impulsadas por IA y medir el impacto cultural a largo plazo. Al reducir las barreras a la entrega continua, este modelo equipa a las VSEs para competir efectivamente en un mercado de software en rápida evolución.

Referencias

[1] M. S. Khan, A. W. Khan, F. Khan, M. A. Khan, and T. K. Whangbo, “Critical challenges to adopt devops culture in software organizations: A systematic review,” *IEEE Access*, vol. 10, pp. 14 339–14 349, 2022.

[2] L. Chen, “Continuous delivery: Overcoming adoption challenges,” *Journal of Systems and Software*, vol. 128, pp. 72–86, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121217300353

[3] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*, ser. A Martin Fowler Signature Book. Addison-Wesley, 2010.

[4] F. Almeida, J. Simoñes, and S. Lopes, “Exploring the benefits of combining devops and agile,” *Future Internet*, vol. 14, no. 2, p. 63, 2022.

[5] M. Munoz, J. Mejia, and A. Lagunas, “Implementation of the iso/iec 29110 standard in agile environments: A systematic literature review,” in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 2018, pp. 1–6.

[6] M.-A. Pastrana-Pardo, H.-A. Ordoñez-Erazo, and C.-A. Cobos-Lozada, “Approach to the best practices of software development based on devops and scrum used in very small entities,” *Revista Facultad de Ingeniería*, vol. 31, no. 61, p. e14828, Sep. 2022. [Online]. Available: https://revistas.uptc.edu.co/index.php/ingenieria/article/view/14828

[7] M. Muñoz, J. Mejia, and C. Y. Laporte, “Implementing iso/iec 29110 to reinforce four very small entities of Mexico under an agile approach,” *IET Software*, vol. 14, no. 2, pp. 75–81, 2020.

[8] N. Azad and S. Hyrynsalmi, “Devops critical success factors — a systematic literature review,” *Information and Software Technology*, vol. 157, p. 107150, 2023.

[9] R. Amaro, R. Pereira, and M. M. da Silva, “Capabilities and metrics in devops: A design science study,” *Information & Management*, vol. 60, no. 5, p. 103809, 2023.

[10] B. Fitzgerald and K.-J. Stol, “Continuous software engineering: A roadmap and agenda,” *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[11] P. Rodríguez, A. Haghighatkhah, L. E. Bakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo, “Continuous deployment of software intensive products and services: A systematic mapping study,” *Journal of Systems and Software*, vol. 123, pp. 263–291, 2017.

[12] M. Shahin, M. Ali Babar, and L. Zhu, “Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices,” *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[13] L. Chen, “Continuous delivery: Huge benefits, but challenges too,” *IEEE Software*, vol. 32, no. 2, pp. 50–54, 2015.

[14] M. Zarour, N. Alhammad, M. Alenezi, and K. Alsarayrah, “A research on devops maturity models,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 3, pp. 4854–4862, 2019.

[15] ——, “A research on DevOps maturity models,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 3, pp. 4854–4862,
Sep. 2019. [Online]. Available: http://www.doi.org/10.35940/ijrte.C6888.098319

[16] M. K. A. Abbass, R. I. E. Osman, A. M. H. Mohammed, and M. W. A. Alshaikh, “Adopting continuous integration and continuous delivery for small teams,” in *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 2019, pp. 1–4.

[17] N. Railic and M. Savic, “Architecting continuous integration and continuous deployment for microservice architecture,” in *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2021, pp. 1–5.

[18] F. Gunawan and E. K. Budiardjo, “A quest of software process improvements in devops and kanban: A case study in small software company,” in *Proceedings of the 2021 4th International Conference on Software Engineering and Information Management*, ser. ICSIM ’21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 39–45. [Online]. Available: https://doi.org/10.1145/3451471.3451478

[19]O. Elazhary, C. Werner, Z. S. Li, D. Lowlind, N. A. Ernst, and M.-A. Storey, "Uncovering the benefits and challenges of continuous integration practices," *IEEE Transactions on Software Engineering*, vol. 48, no. 7, pp. 2570–2583, 2022.

[20]M. Tuape, V. Hasheela-Mufeti, A. Kayanda, and J. Kasurinen, "Software development in small software companies: Exploring the usage of procedures, techniques, methods and models in practice," in *Proceedings of the 2021 European Symposium on Software Engineering*, ser. ESSE '21. New York,

NY, USA: Association for Computing Machinery, 2022, pp. 29–38. [Online]. Available: https://doi.org/10.1145/3501774.3501779

[21]B. A. Lunde and R. Colomo-Palacios, "Continuous practices and technical debt: a systematic literature review," in *2020 20th International Conference on Computational Science and Its Applications (ICCSA)*, 2020, pp. 40–44.

[22]H. van Merode, *CI/CD Concepts*. Berkeley, CA: Apress, 2023, pp. 11–27.

[23]M. H. Tanzil, M. Sarker, G. Uddin, and A. Iqbal, "A mixed method study of devops challenges," *Information and Software Technology*, vol. 161, p. 107244, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584923000988

[24]B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[25]D. Polit and C. Beck, "The content validity index: Are you sure you know what's being reported? critique and recommendations," *Research in nursing health*, vol. 29, pp. 489–97, 10 2006.

[26]C. Lawshe, "A quantitative approach to content validity," *Personnel Psychology*, vol. 28, pp. 563 – 575, 12 2006.

[27]J. S. Grant and L. L. Davis, "Selecting and using content experts for instrument development," *Research in nursing & health*, vol. 31, no. 2, pp. 132–140, 2008.

[28]M. S. B. Yusoff, "Abc of content validation and content validity index calculation," *Education in Medicine Journal*, vol. 11, pp. 49–54, 06 2019.

[29]N. Elangovan and E. Sundaravel, "Method of preparing a document for survey instrument validation by experts," *MethodsX*, vol. 8, p. 101326, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215016121001199

[30]L. L. Davis, "Instrument review: Getting the most from a panel of experts," *Applied Nursing Research*, vol. 5, no. 4, pp. 194–197, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0897189705800084

[31]M. R. Lynn, "Determination and quantification of content validity," *Nursing research*, vol. 35, no. 6, pp. 382–385, 1986.

[32]"Ieee standard for system, software, and hardware verification and validation," Tech. Rep., 2017.

[33]H. Taherdoost, "Validity and reliability of the research instrument; how to test the validation of a questionnaire/survey in a research," *International Journal of Academic Research in Management*, vol. 5, pp. 28–36, 01 2016.

[34]V. Debroy and S. Miller, "Overcoming challenges with continuous integration and deployment pipelines: An experience report from a small company,"

IEEE Software, vol. 37, no. 3, pp. 21–29, 2020.

[35]D. Acevedo-Dueñas, M. Muñoz, and S. Galvan-Cruz, "Use of devops in very small entities: Systematic mapping," in *2023 12th International Conference On Software Process Improvement (CIMPS)*, 2023, pp. 78–83.