Universidad Nacional Federico Villarreal

VRIN | VICERRECTORADO DE INVESTIGACIÓN

ESCUELA UNIVERSITARIA DE POSGRADO

METODOLOGÍA DEVOPS Y SU INFLUENCIA EN EL PROCESO DE DESPLIEGUE DE SOFTWARE EN UNA UNIVERSIDAD PÚBLICA

Línea de investigación:
Ingeniería de software, simulación y desarrollo de TICs

Tesis para optar el Grado de Académico de Doctor en Ingeniería de Sistemas

Autor

Inquilla Quispe, Ricardo Carlos

Asesor

Hilario Falcón, Francisco Manuel

ORCID: 0000-0003-3153-9343

Jurado

Coveñas Lalupu, José

Petrlik Azabache, Iván Carlo

Lira Camargo, Jorge

Lima - Perú

2025

METODOLOGIA DEVOPS Y SU INFLUENCIA EN EL PROCESO DE DESPLIEGUE DE SOFTWARE EN UNA UNIVERSIDAD PÚBLICA

INFORME DE ORIGINALIDAD

13%	12%	3%	4%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorio.ucv.edu.pe Fuente de Internet	2%
2	hdl.handle.net Fuente de Internet	1%
3	repositorio.unfv.edu.pe Fuente de Internet	1%
4	www.coursehero.com Fuente de Internet	1%
5	1library.co Fuente de Internet	<1%
6	repositorio.espe.edu.ec Fuente de Internet	<1%
7	Submitted to Universidad Nacional Federico Villarreal Trabajo del estudiante	<1%
8	www.cacic2016.unsl.edu.ar Fuente de Internet	<1%

Universidad Nacional
Federico Villarreal

VRIN | VICERRECTORADO DE INVESTIGACIÓN

ESCUELA UNIVERSITARIA DE POSGRADO

METODOLOGIA DEVOPS Y SU INFLUENCIA EN EL PROCESO DE DESPLIEGUE DE SOFTWARE EN UNA UNIVERSIDAD PÚBLICA

Línea de Investigación:
Ingeniería de Software, simulación y desarrollo de TICs
Tesis para optar el Grado de Académico de Doctor en Ingeniería de Sistemas

Autor
Inquilla Quispe, Ricardo Carlos

Asesor
Hilario Falcón, Francisco Manuel
ORCID: 0000-0003-3153-9343

Jurado

Coveñas Lalupu, José
Petrlik Azabache, Iván Carlo
Lira Camargo, Jorge

Lima – Perú
2025

DEDICATORIA

Materializan la esencia del Dios Creador de todos los Sistemas dentro del vasto y contemplativo Universo, ellos son mis padres Miguel Inquilla Choqueña y Fermina Quispe Quispe, así como mi hermana Luz Inquilla Quispe, a quienes constantemente me han deseado lo mejor en la vida.

AGRADECIMIENTO

Al Dios Todopoderoso, que creó todo lo que existe con sus matemáticas espirituales, primero. Me dio a mis padres, quienes siempre me dan ánimo. A mis amigos y colegas de posgrado que me brindaron apoyo y motivación en momentos difíciles.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El objetivo fue implementar una metodología Devops para mejorar el proceso de despliegue de software en una Universidad Pública, a través de sus métricas analizadas “Frecuencia de Implementación”, “Tiempo de Entrega de Cambios”, “Tiempo de Recuperación de Fallas” y “Tasa de Fallas de Implementación”. Se ejecuto un diseño que incluyo experimentación en los momentos de pretest y postest, se verificó la prueba de normalidad para seleccionar finalmente el estadígrafo de Wilcoxon lo cual permitió validar los cambios observados. Los datos se recolectaron en un entorno controlado, con mediciones antes y después del experimento. Sus resultados fueron más pronunciados en todos los indicadores clave. Así, la “Frecuencia de Implementación” evidencia una disminución de errores después de la intervención y una disminución promedio de 416 minutos del “Tiempo de Entrega de Cambios”, lo que demuestra una mayor eficiencia del proceso de desarrollo. Al mismo, el “Tiempo de Recuperación de Fallas” disminuyó en promedio 57 minutos, lo que aumenta la capacidad de respuesta a incidentes. Además, la “Tasa de Fallas de Implementación” disminuyó notablemente, lo que evidencia una mejora en la calidad de las implementaciones. Se concluye que la ejecución de prácticas DevOps generó muchas mejoras de en el desarrollo y entregas de software.

Palabras clave: devops, frecuencia de despliegue, tiempo de entrega de cambios, tiempo de recuperación ante fallos, tasa de fallos en despliegues

ABSTRACT

The objective of this study was to implement a DevOps methodology to improve the software deployment process in a public university, through the analysis of the following metrics: Deployment Frequency, Lead Time for Changes, Mean Time to Recovery, and Change Failure Rate. An experimental design with pretest and posttest measurements was applied. A normality test was conducted to select the Wilcoxon statistical test, which allowed the validation of the observed changes. Data were collected in a controlled environment, with measurements taken before and after the intervention. The results were more pronounced across all key indicators. Specifically, Deployment Frequency showed a reduction in errors after the intervention, along with an average decrease of 416 minutes in Lead Time for Changes, demonstrating greater efficiency in the development process. Likewise, Mean Time to Recovery decreased by an average of 57 minutes, improving incident response capability. In addition, the Change Failure Rate decreased significantly, indicating an improvement in the quality of deployments. It is concluded that the implementation of DevOps practices generated substantial improvements in software development and delivery processes.

Keywords: change delivery time, deployment failure rate, deployment frequency, devops, failure recovery time

I. INTRODUCCIÓN

1.1. Planteamiento del problema

En los últimos años, el enfoque Devops ha tenido importante participación en la industria del desarrollo de software el cual promueve la articulación entre el desarrollo de software y las operaciones de TI. DevOps es una ideología y un marco que trabaja con las mentes de la colaboración global a través de una asociación ecológica que fusiona la construcción de relaciones ágiles, el servicio (gestión) y las operaciones hacia un objetivo común de entregar software de calidad de manera rentable y a tiempo. Existe un creciente interés en la investigación con temática de DevOps, con estudios centrados en las pruebas automatizadas en entornos DevOps e industrias que prosperan con aplicaciones web y arquitecturas orientadas a servicios (Pando & Dávila, 2023).

La transformación que está experimentando la TI hacia nuevas arquitecturas web como microservicios, nube y contenedores dicta la transformación correspondiente de las actividades de construcción de software, implementación y soporte (Benni et al., 2019). La investigación empírica para corporaciones multinacionales con uso intensivo de software revela desafíos en la adopción de DevOps y por lo tanto, la necesidad de más evidencia empírica sobre los factores impulsores de la adopción de DevOps por parte de las organizaciones y los resultados esperados, como un software de mejor calidad, una entrega efectiva con menos fallas y eficiencia en la entrega (Trigo et al., 2022). Todos estos hallazgos narran las prácticas de DevOps y su efecto en optimización de las actividades de construcción de software a nivel internacional.

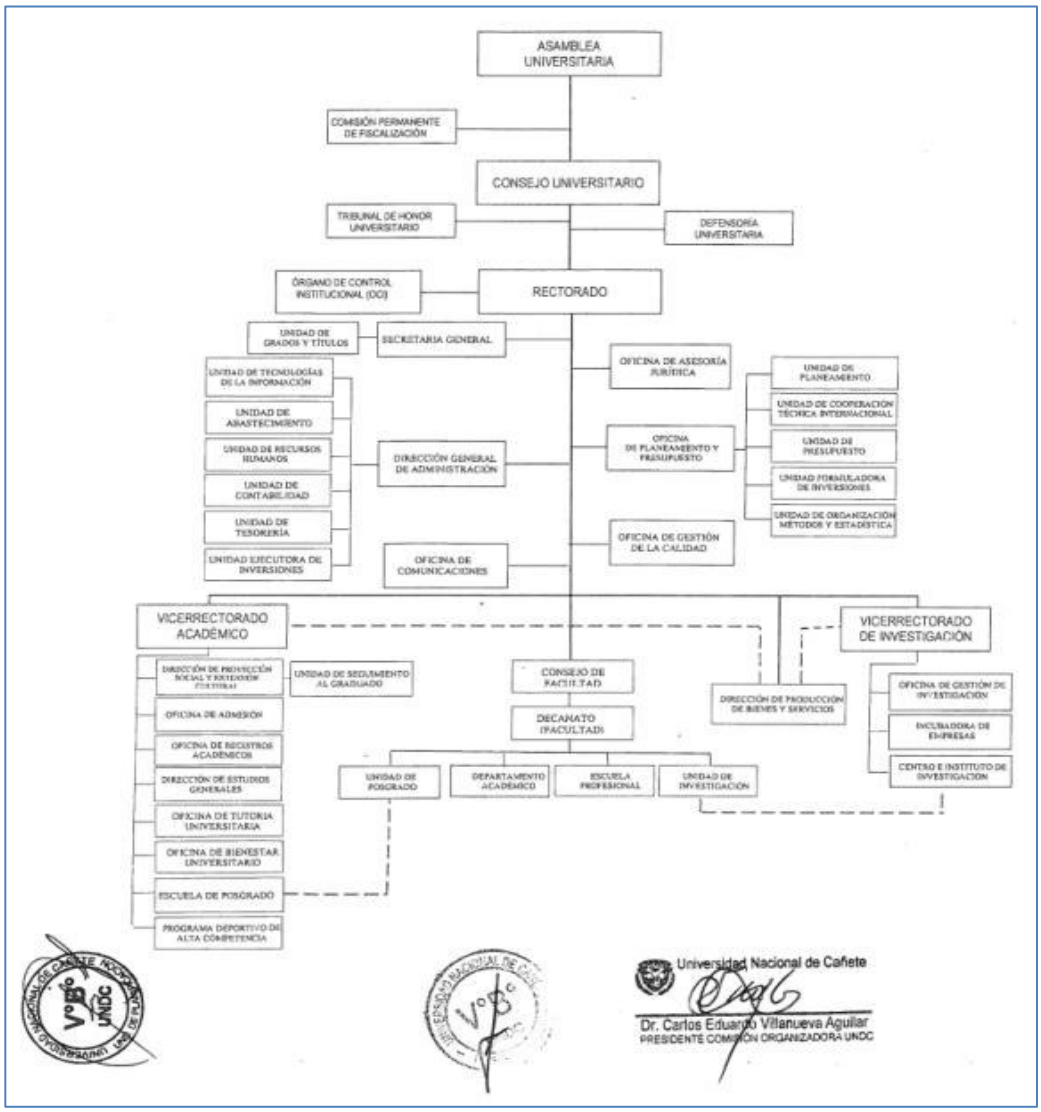
Con el brote de COVID 19, las organizaciones se ven empujadas repentinamente a la transformación digital y hemos visto la forma en que se ha demostrado que las empresas trabajan de manera deslocalizada, por lo que COVID 19 ha acelerado la adopción de DevOps

aprovechando la ola de su ideología que conduce a la primera práctica con el movimiento global al trabajo remoto, por lo que DevOps perpetúa una cultura continua de transparencia y experimentación a través de las organizaciones que permite a los equipos localizar más fácilmente las ineficiencias dentro de un proceso y cambiarlo a un ritmo más rápido.

Con respecto a la Universidad Nacional de Cañete, las limitaciones de tiempo y la creciente demanda de servicios requieren la necesidad de un conjunto de nuevos requisitos que influyen en los procesos académicos y administrativo. Esta situación ha derivado en un conjunto de fallas que han tenido un impacto negativo en los servicios a los usuarios. Un inconveniente en las actividades de construcción y despliegue de software es el tiempo de espera de los usuarios que puede salirse de control, también puede afectar la falta de confianza que se pierde. Por lo tanto, la finalidad principal de esta investigación es aplicar un enfoque de entrega continua al proceso de implementación de software que actúe como catalizador de una revolución en sus prácticas de desarrollo e implementación de software que mejore la forma en que la universidad lo lleva a cabo actualmente. Esta investigación está orientada a fomentar la adopción de nuevas estrategias tecnológicas como aliadas en la automatización y agilización, y para las actividades de implementación de software destinadas a aumentar la satisfacción en la comunidad universitaria.

Figura 1

Organigrama de la Universidad Nacional de Cañete

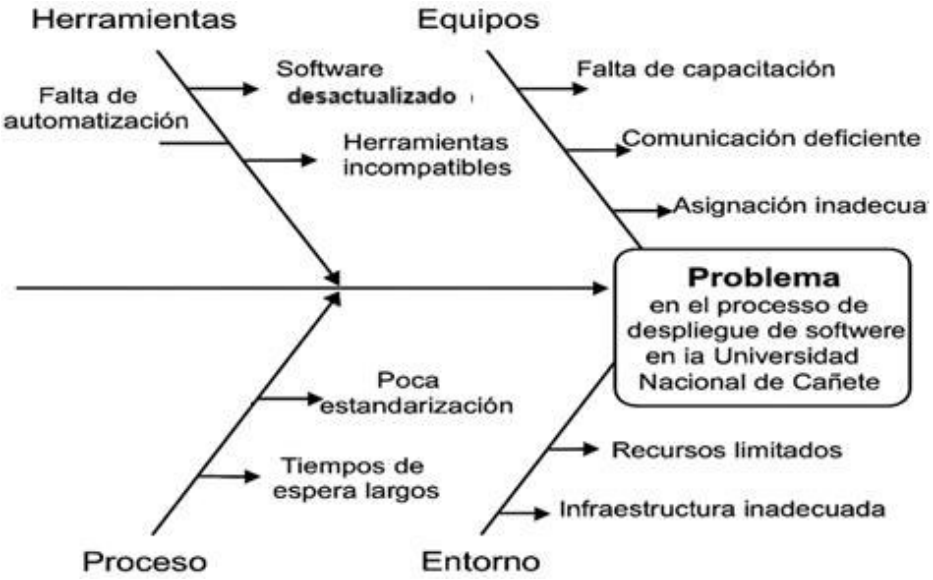


1.2. Descripción del problema

La Universidad Nacional, ofrece una formación en el ámbito profesional, científico, tecnológico y humanístico reforzado con valores y principios, liderazgo, investigación y responsabilidad social. Desde que surgió el COVID-19 y los nuevos requerimientos de software, se presentan dificultades para la entrega final y despliegue de proyectos de sistemas web en la universidad. Se han identificado errores que generen retrasos en el lanzamiento o transición a producción. Estas deficiencias afectan la performance de los servicios ofrecidos a la comunidad universitaria, generan tiempo excesivo para los usuarios y debilitan la confianza en los sistemas de información institucionales.

Figura 2

Diagrama de Ishikawa



El diagrama de Ishikawa presentado identifica las causas principales que afectan negativamente el proceso de despliegue de software en la Universidad Nacional de Cañete, agrupándolas en cuatro categorías: herramientas, equipos, procesos y entorno. En cuanto a las herramientas, se evidencia la falta de automatización, software desactualizado y herramientas incompatibles. En la dimensión de los equipos, destacan la falta de capacitación, la comunicación deficiente y la asignación inadecuada de personal. Los procesos se ven limitados por la escasa estandarización, preciosos insuficientes y tiempos de espera prolongados. Finalmente, el entorno institucional presenta obstáculos como recursos limitados e infraestructura inadecuada. Estas causas combinadas explican las dificultades en la eficiencia, confiabilidad y agilidad del despliegue de software institucional, justificando la necesidad de implementar una metodología DevOps como solución estratégica.

En efecto, para enfrentar estos desafíos, la Oficina de Tecnologías de la Información de la Universidad Nacional de Cañete pretende replantear las prácticas tradiciones de desarrollo de software mediante la metodología modernas que permitan optimizar el ciclo de vida del

software, facilitar la colaboración entre los equipos de desarrollo y operaciones. La adopción de esta metodología permitirá a la Universidad Nacional de Cañete beneficiar de servicios digitales agiles, estables alineados a los objetivos estratégicos institucionales.

1.3. Formulación de problema

1.3.1. Problema general.

¿De qué manera la implementación de una metodología Devops influye positivamente en el proceso de despliegue de software en una Universidad Pública?

1.3.2. Problemas específicos

¿De qué manera la implementación de una metodología Devops influye positivamente en la frecuencia de despliegue en una Universidad Pública?

¿De qué manera la implementación de una metodología Devops influye positivamente en el tiempo de entrega de cambios en una Universidad Pública?

¿De qué manera la implementación de una metodología Devops influye positivamente en el tiempo de Recuperación ante fallos

en una Universidad Pública?

¿De qué manera la implementación de una metodología Devops influye positivamente en la tasa de fallos en despliegues en una Universidad Pública?

1.4. Antecedentes

1.4.1. Antecedentes nacionales

La investigación "*Marco DevOps en el Proceso de Desarrollo de Software en una Entidad Financiera Privada, Lima 2023*", Moreyra (2023) tuvo como finalidad evaluar el impacto del marco DevOps en el Proceso de Desarrollo de Software dentro de una organización financiera. La investigación se centró en evaluar los resultados del desarrollo de proyectos

obtenidos a través de enfoques tradicionales en contraste con aquellos que utilizan el marco DevOps. Empleando una metodología cuantitativa, este estudio aplicado presentó un diseño pre y post experimental, examinando una muestra de 87 requisitos divididos en dos grupos separados. Los resultados indicaron una mejora del 55,17% en la frecuencia de implementación, una baja del 16% en la densidad de defectos y un aumento del 33,33% en la productividad del proceso. Se concluyo que el despliegue del marco DevOps impactó beneficiosamente en el proceso de actividades de construcción de software dentro de una institución financiera.

En el artículo titulado "*Software de centro de costes multiplataforma para pymes que utiliza DevOps*" de Vigo et al. (2020) se propusieron optimizar el despliegue de software de centro de costes para pymes mediante la introducción de DevOps. Al integrar las operaciones con el desarrollo, DevOps logró un desarrollo de software más eficiente que reduciría el tiempo de implementación en más del 50 %, lo que aumentó la tasa de éxito de la prueba de aceptación realizada por el equipo de desarrollo, que arrojó un 72 %. Se utilizaron diferentes herramientas para crear un software multiplataforma; esto permitirá a las pymes gestionar sus centros de costes de forma eficaz en cualquier dispositivo. El estudio brindó información sobre las fases de desarrollo de DevOps, destacando la importancia de la integración continua a lo largo del proyecto.

En su trabajo "*Implementación de DevOps para mejorar la integración y el despliegue de software en seguros*", Casas (2020) tuvo como objetivo evaluar el impacto de DevOps en la integración y el despliegue de software en la industria de seguros. La investigación se llevó a cabo localmente en Lima, en el distrito de San Isidro, durante el año 2019. En el estudio se dieron a conocer los fundamentos, prácticas y técnicas que sustentan la cultura DevOps: integración continua, despliegue continuo, automatización de pruebas y valores fundamentales (trabajo en equipo, automatización, medición y monitoreo) a través de la colaboración. Se

aplicaron procedimientos con experimentación preexperimental junto con enfoques cuantitativos. La población de estudio incluyó ocho sprints semanales y un total de dieciocho historias de usuario obtenidas de proyectos organizacionales. Las hipótesis se probaron mediante pruebas U de Man-Whitney y T de Student. En calidad de herramienta de implementación DevOps, nuestro tiempo de ciclo promedio se redujo de 18,11 días a 11,61 días, la frecuencia de liberación de código aumentó de 1,63 a 3,25 en promedio y la tasa de éxito de 54,58% a 72,92%. Los hallazgos del estudio sugieren que la velocidad y la calidad del proceso de integración e implementación de software de una firma perteneciente al sector seguros ubicada en San Isidro, Lima fueron mejores durante el año 2019.

En el estudio de Vidal (2020) denominado "*Propuesta DevOps en el proceso de desarrollo de sistemas de información en la financiera Compartamos*", Miraflores afirmó que el estudio tuvo como objetivo desarrollar una propuesta DevOps para optimizar las fases de despliegue de software de la financiera, que acompañado como acelerador de lo que llama "transformación digital", le permite asegurar su competencia, el enfoque del trabajo mencionado es "cualitativo", con tipo de investigación es "aplicada". Se utilizaron "entrevistas en profundidad" y "observación" como técnicas, "guías" como herramientas y "triangulación" como técnicas de revisión de contenido, permitiéndonos interpretarlo y luego verificarlo mediante el uso de "matriz" para analizar lo que se está estudiando. Se concluye que la filosofía DevOps mejora la performance operativa que está estrechamente vinculada a los objetivos comerciales al aprovechar sus capacidades técnicas y sus fuertes raíces en la cultura organizacional. Así mismo el autor que implementar Devops es fundamenta realizar una diagnóstico inicial desde las tres dimensiones de personas, procesos y tecnología para guiar las estrategias de perfeccionamiento y entregar valor a los clientes.

Según Bravo et al. (2020) en su estudio titulado "*Análisis comparativo de Waterfall y DevOps en el desarrollo de microservicios*", tuvieron como objetivo fue realizar una tabla

comparativa de los enfoques Waterfall y DevOps aplicandolo al desarrollo de microservicios para buscar detalles de libros en una tienda virtual. La investigación utilizó un enfoque cuantitativo y un diseño transversal que necesitó experimentación. Como resultado, se elaboró una tabla comparativa identificando el tiempo de entrega para cada etapa de la metodología, donde el uso de la metodología DevOps mejoró el tiempo de entrega en un 66%. El microservicio fue desarrollado usando Postgres Sql, Visual Studio y Apache NetBeans lo que permitió validar el rendimiento de las metodologías comparadas en un entorno técnico funcional.

Según Peralta (2019) en su investigación de 2019 titulada "*DevOps en la Entrega Continua en la Dirección General de Estadística y TIC del Ministerio de Cultura, Lima 2019*", tuvo como objetivo evidencias los efectos positivos de DevOps en la entrega continua de software. La investigación se concentró en analizar los inconvenientes que afectan las implementaciones de producción de aplicaciones web y demuestra que DevOps puede mejorar la frecuencia de implementación, minimizar los cambios fallidos y aumentar la cantidad de modificaciones incorrectas. Esta investigación utilizó un diseño experimental con una tipología preexperimental, facilitando comparaciones objetivas para determinar la aceptación o el rechazo, analizando variables tanto antes como después de la estimulación y realizando pruebas de hipótesis a través de la prueba paramétrica T de Student. Esto hallazgos refuerzan la utilidad de DevOps, que tiene el potencial de mejorar la frecuencia de implementación en un 49,26%, reducir los errores fallidos en 57,42% y mejorar los cambios sin fallas en un 25,6%.

1.4.2. Antecedentes internacionales

Rüegger et al., (2024), en su investigación titulada "*Fully Automated DORA Metrics Measurement for Continuous Improvement*", tenian como objetivo general investigar la viabilidad y los beneficios de automatizar la medición de métricas DORA para mejorar continuamente el rendimiento de la entrega de software en microservicios individuales. La

metodología utilizada incluyó el desarrollo y la evaluación de un sistema automatizado de medición de métricas DORA en un caso de estudio industrial con 37 microservicios durante cuatro semanas, recopilando datos de herramientas DevOps en tiempo real. Los resultados revelaron diferencias significativas con respecto a la frecuencia de implementación, el tiempo de entrega de los cambios, el éxito de los cambios, así como el tiempo de entrega promedio para la recuperación, donde las métricas basadas en encuestas dejan fuera información importante. Su conclusión y relevancia recomiendan que con la medición automatizada de las métricas DORA, habrá más transparencia en la toma de decisiones, así como en la mejora del rendimiento del software con información óptimamente detallada.

Port et al. (2024), en su artículo titulado *"Investigating Effectiveness and Compliance to DevOps Policies and Practices for Managing Productivity and Quality Variability"* tuvieron como objetivo Investigar la eficacia y el cumplimiento de las políticas y prácticas de DevOps para gestionar la variabilidad de la productividad y la calidad específicamente en el sistema MONTE en la Autoridad Portuaria de Tanzania. El diseño de la investigación se basó en un análisis de series temporales cruzadas. Se compararon las características de productividad y calidad del sistema MONTE antes y después de la implementación de DevOps. Esto se hizo utilizando datos recopilados durante más de 15 años durante los cuales el sistema funcionó de manera continua. Los principales resultados demostraron que las políticas y prácticas de DevOps pueden gestionar la variabilidad de la calidad y la productividad, ya que hubo bajos niveles de errores informados y, al mismo tiempo, fueron productivas de manera constante. La resultados demostraron que las prácticas de DevOps no solo son efectivas para preservar la calidad y la productividad de los sistemas críticos, sino que también brindan una base sólida para el progreso continuo en el desarrollo de software.

Díaz et al. (2024), en su artículo titulado "*Harmonizing DevOps taxonomies — A grounded theory study*" el cual tuviero como objetivo general entender mejor la estructura

organizacional y las características de los equipos que adoptan DevOps al armonizar las taxonomías existentes. La metodología usada incluyó la teoría fundamentada con codificación colaborativa, utilizando el Acuerdo de Codificador Intercambiable (ICA) para guiar las discusiones. Los resultados principales revelaron una teoría sustantiva y analítica de las taxonomías de DevOps, identificando diferentes estructuras de equipos y sus características. La conclusión y relevancia del estudio subrayan que la armonización de taxonomías es crucial para definir qué estructuras de equipo son más adecuadas para abordar la cultura y prácticas de DevOps según los objetivos y capacidades de las empresas.

Cuadra et al. (2024), en su artículo titulado "*Enabling DevOps for Fog Applications in the Smart Manufacturing domain: A Model-Driven based Platform Engineering approach*" tuvieron como objetivo general desarrollar una plataforma de ingeniería basada en modelos para aplicaciones de Fog en el dominio de la manufactura inteligente, facilitando la integración y automatización de componentes. La metodología utilizada incluyó el uso de técnicas de ingeniería basada en modelos y un editor visual de flujos para diseñar y desarrollar aplicaciones, implementando estándares OpenFog. Los resultados principales mostraron que la plataforma permite una integración y entrega eficientes de aplicaciones, destacando la utilidad de contenedores para aplicaciones microservicios en entornos de manufactura inteligente. La conclusión y relevancia del estudio subrayan que esta plataforma facilita la separación de preocupaciones y mejora la productividad en el desarrollo de aplicaciones, siendo especialmente relevante para la implementación de aplicaciones en la nube en entornos industriales.

Machuzhak (2023), en la tesis titulada "*Дослідження методології DevOps для розробки та підтримки веб-застосунків (Investigación de la metodología DevOps para el desarrollo y mantenimiento de aplicaciones web)*" tuvo como objetivo investigar la aplicación de metodologías DevOps, como CI/CD, para optimizar las actividades de desarrollo y

despliegue de software en aplicaciones web. Utilizando herramientas como GitHub Actions, Docker y Google Cloud Platform, la metodología utilizada permitió implementar prácticas que resultaron aumentando a 25.47% en la frecuencia de implementación y una mejora en la liberación de código de 1.67 a 2.33 despliegues por sprint, además de reducir la tasa de fallas del 55.56% al 11.11%. La conclusión destaca que la adopción de DevOps mejoró significativamente la eficiencia y calidad del desarrollo de software, estableciendo un marco ágil y efectivo para el despliegue continuo de aplicaciones web, lo que demuestra su relevancia como estrategia para la optimización de procesos en proyectos de software.

Ahmed et al. (2023), en sul artículo titulado "*Optimization of DevOps Transformation for Cloud-Based Applications*" tuvo como objetivo general optimizar la transformación DevOps para aplicaciones en la nube mediante técnicas avanzadas de automatización. La metodología utilizada incluyó el uso de herramientas como Git para control de versiones, Docker para la contenedorización, Jenkins para integración y despliegue continuo, y AWS para el despliegue en la nube. Los resultados principales mostraron una reducción del 60% en los costos y mejoras significativas en la eficiencia del despliegue. La conclusión y relevancia destacan que la optimización de DevOps no solo mejora la eficiencia y reduce costos, sino que también facilita una integración continua y una respuesta eficiente a los cambios en las necesidades del cliente.

En un estudio realizado por Navarro (2020) en la Universidad Autónoma de Bucaramanga, titulado "*Modelo de Desarrollo e Implementación de Software bajo el Marco de Buenas Prácticas CMMI en el Área de Sistemas de Información*", el objetivo que se planteó fue diseñar una estrategia para implementar buenas experiencias en el desarrollo e implementación de aplicaciones dentro del departamento de sistemas de información de la universidad alineado a las prácticas con el marco de referencia CMMI-DEV 1.3 Nivel 3. Utilizó la metodología SCAMPI, se realizó un análisis del proceso de construcción de software de la

oficina de tecnologías de la información de la universidad, basado en el modelo CMMI para desarrollo versión 1.3. Este análisis reveló una brecha entre los procesos actuales implementados para proyectos de ingeniería de software y las recomendaciones de CMMI para el despliegue de Nivel 3. Se concluyó que el departamento de tecnologías de la información de la Universidad Autónoma de Bucaramanga ha logrado avances significativos en su alineación con los lineamientos del CMMI, particularmente en dos categorías específicas. Abordar las brechas restantes en estas áreas ayudará a la universidad a lograr el nivel 3 de cumplimiento del CMMI en la creación de software.

1.5. Justificación de la investigación

1.5.1. Justificación práctica.

El despliegue de una metodología de entrega continua permitirá optimizar de manera eficientes la infraestructura tecnológica para el despliegue de software, debido a que integró a los equipos de desarrollo y operaciones de la Universidad Nacional de Cañete. Esto permitirá automatizar tareas que eran repetitivas, así mismo fomentará la cultura de mejora continua y mayor satisfacción de los usuarios.

1.5.2. Justificación teórica.

El presente proyecto va a aportar conocimiento respecto a la variable independiente metodología Devops respecto a su filosofía y estrategias de desarrollo así mismo respecto a la variable dependientes despliegue de software se medirá el efecto que tiene de la variable independientes para analizar los cambios. Todo ello permitirá enriquecer el conocimiento respecto a nuevas metodologías de desarrollo y despliegue de software, debido a que existen poca investigación en relación a las entregas continuas en Universidades Nacionales lo cual servirá de base para futuras indagaciones investigativas.

1.5.3. Justificación social

La presente investigación se alinea con los principios filosóficos y éticos de la Agenda 2030 de las Naciones Unidas, en particular con los siguientes Objetivos de Desarrollo Sostenible (ODS):

ODS 4 (Educación de calidad): La implementación de DevOps en universidades públicas mejora los procesos académicos y administrativos, potenciando enPtornos de aprendizaje digitales más eficaces y sostenibles.

ODS 9 (Industria, innovación e infraestructura): Promueve la adopción de metodologías innovadoras que fortalecen la infraestructura tecnológica institucional.

ODS 16 (Paz, justicia e instituciones sólidas): Aporta a la consolidación de sistemas eficientes, transparentes y orientados a resultados en el sector público.

Desde una mirada crítica, también se reconoce que la transformación digital no es neutra: requiere una reflexión ética sobre cómo se usan las tecnologías, para qué fines y con qué impacto social. Por ello, esta tesis se sustenta en una filosofía pragmática, humanista y constructivista, en la que la tecnología es entendida como un medio para empoderar a las instituciones educativas, generar valor público y contribuir al desarrollo sostenible.

1.5.4. Justificación Metodológica.

Este estudio es metodológicamente sólido ya que usará una herramienta diseñada para evaluar la variable dependiente a través de evaluaciones previas y posteriores al despliegue de una metodología DevOps en las actividades de implementación de software en una universidad pública.

1.6. Limitaciones de la investigación

Se tendrá como restricción los despliegues de software de los procesos académicos.

1.7. Objetivos

1.7.1. Objetivo general

Implementar una metodología Devops para mejorar el proceso de despliegue de software en una Universidad Pública.

1.7.2. Objetivos específicos

Implementar una metodología Devops para mejorar la frecuencia de despliegue en una Universidad Pública.

Implementar una metodología Devops para mejorar el tiempo de entrega de cambios en una Universidad Pública.

Implementar una metodología Devops para mejorar el tiempo de recuperación ante fallos en una Universidad Pública.

Implementar una metodología Devops para mejorar la tasa de fallos en despliegues en una Universidad Pública.

1.8. Hipótesis

1.8.1. Hipótesis general

Si se implementa una metodología Devops entonces influye positivamente en el proceso de despliegue de Software en una Universidad Pública.

1.8.2. Hipótesis específicas.

Si se implementa una metodología Devops entonces influye positivamente en la frecuencia de despliegue en una Universidad Pública.

Si se implementa una metodología Devops entonces influye positivamente en el tiempo de entrega de cambios en una Universidad Pública.

Si se implementa una metodología Devops entonces influye positivamente en el tiempo de recuperación ante fallos en una Universidad Pública.

Si se implementa una metodología Devops entonces influye positivamente en la tasa de fallos en despliegues en una Universidad Pública.

II. MARCO TEÓRICO

2.1. Bases filosóficas

La historia del pensamiento humano está profundamente ligada al uso de herramientas para transformar su realidad. Desde la prehistoria, el ser humano ha sido un ser técnico y social, capaz de construir artefactos como el fuego, la lanza, la rueda o los primeros sistemas de conteo que marcaron el inicio del dominio de su entorno mediante la razón práctica.

En esa línea, el desarrollo tecnológico ha sido una extensión de la capacidad humana para crear, colaborar y evolucionar. Esta visión corresponde a una mirada filosófica constructivista, donde el conocimiento no es una copia de la realidad, sino una construcción progresiva generada en la interacción entre el individuo, su cultura, su entorno y las herramientas que crea.

El constructivismo epistemológico, como el propuesto por Jean Piaget y ampliado por autores como von Glasersfeld y Vygotsky, sostiene que el conocimiento se construye activamente, y que los entornos de aprendizaje y de trabajo deben facilitar experiencias significativas, colaborativas y adaptables.

Esta perspectiva del conocimiento se articula implícitamente con la metodología DevOps, que promueve la integración, el aprendizaje continuo, la automatización y la colaboración como principios relevantes para la optimización continua de los procesos. Desde esta óptica, la tecnología no es solo un conjunto de medios, sino una práctica social con dimensión pedagógica y transformadora.

2.2. Bases teóricas

2.2.3. Metodología Devops

2.2.3.1. Introducción a Devops. DevOps es una metodología que tiene como objetivo agilizar las actividades de desarrollo y operaciones mediante la promoción de la comunicación y la constante combinación entre los equipos de desarrollo (Dev) y operaciones (Ops), lo que, en última instancia, reduce el tiempo de creación de software y acelerar el lanzamiento de nuevas funciones (Redondo & Cárdenas, 2022). Hace hincapié en el continuo crecimiento y la eficiencia en la creación de software, en consonancia con la necesidad de atender eficientemente a las demandas de los usuarios y, al mismo tiempo, mantener la estabilidad y la previsibilidad de los servicios de TI (Jiménez Marco, 2016) . DevOps promueve una cultura de responsabilidad compartida, en la que los equipos trabajan juntos sin problemas para ofrecer cambios frecuentes y confiables, garantizando un equilibrio entre la innovación y la confiabilidad operativa. Al integrar herramientas y prácticas que facilitan esta estrecha colaboración, DevOps mejora la performance general de las actividades de desarrollo e implementación de software, alineándose con las metodologías ágiles modernas y los principios de la entrega continua.

2.2.3.2. Importancia de Devops. De acuerdo con Forsgren et al. (2018) muestra que las compañías de todos las dimensiones, rubros y tipos de tecnología digital que implementan prácticas DevOps obtiene mejores resultados que aquellas que no se implementan prácticas DevOps. El resultado es inicial o no válido de la siguiente manera:

- El código se implementa 46 veces más frecuentemente.
- El tiempo de confirmación (el tiempo transcurrido entre el momento en que se agrega una línea de código de cambio al código fuente y el instante en que dicha línea de código se despliega en producción) se reduce en 440 veces.

- Mejora de 170 veces el tiempo medio de recuperación (MTTR) de errores implementados en producción.
- Reducción de 5 veces en la tasa de fallas para cambios implementados en producción.

2.2.3.3. Historia de Devops. El resultado en cuestión ha sido influenciado por varios movimientos, como se señala en los hallazgos de Willis (2012), uno de estos movimientos es el Movimiento Lean, que se originó a principios de la década de 1980 con el objetivo de formalizar el Sistema de Producción Toyota. Esta iniciativa puso de relieve técnicas como el Mapa de Flujo de Valor, los tableros Kanban y el Mantenimiento Productivo Total. Además, el Movimiento Agile, que comenzó en 2001 con la publicación del Manifiesto Agile escrito por diecisiete expertos en desarrollo de software, tenía como objetivo agilizar las prácticas y metodologías de desarrollo de software. Un elemento central de este movimiento es el principio de entregar software funcional con frecuencia en plazos más cortos, que normalmente abarcan desde unas pocas semanas hasta unos pocos meses.

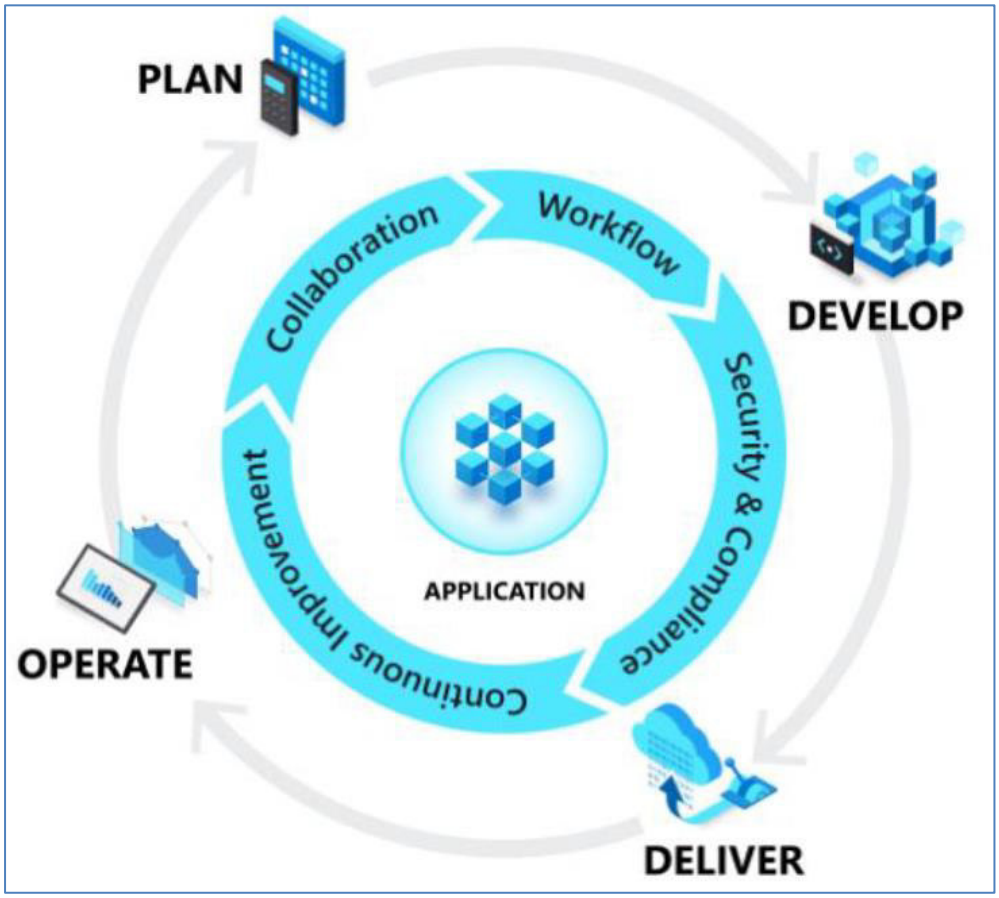
La Conferencia Velocity, que tuvo lugar en 2007, incluyó una presentación innovadora titulada "10 implementaciones por día: cooperación de desarrollo y operaciones en Flickr". Este evento marcó el inicio del Movimiento Velocity. Al año siguiente, en la reunión de Toronto de 2008, Patrick Debois y Andrew Schafer hicieron una socialización sobre la ejecución de principios ágiles a la infraestructura, lo que generó el Movimiento de Infraestructura Ágil. Fue durante esta conferencia que tuvo lugar el primer DevOpsDay en Gante, Bélgica, en 2009, y fue aquí donde se acuñó oficialmente el término DevOps.

En la reunión Agile 2006, Tim Fitz introdujo el concepto de entrega continua en su blog titulado "Despliegue continuo", y más tarde, Jez Humble y David Farley ampliaron esta idea con su patrón conocido como "The Deployment Pipeline".

2.2.3.4. Ciclo de vida de las aplicaciones y DevOps. El flujo de vida de la aplicación, que abarca las fases de planificación, desarrollo, entrega y uso, está muy influenciado por DevOps. Cada fase está interconectada y las etapas no están claramente definidas para ningún rol específico. En una verdadera cultura DevOps, cada rol participa de alguna manera en todas las fases visibilizadas en la Figura 6 (Azure, 2022).

Figura 3

Ciclo de vida de las aplicaciones Devops



Nota. Azure (2022)

Planear

Durante la etapa de planificación inicial, los equipos de DevOps participan en el proceso de conceptualizar, delinear y delinear las características y capacidades de las

aplicaciones y sistemas que pretenden desarrollar. Supervisan el avance a un nivel integral y granular, abarcando tareas individuales, así como aquellas que abarcan múltiples productos dentro de sus carteras.

Al emplear técnicas como la creación de trabajos pendientes, el seguimiento de errores, la gestión ágil de la construcción de software por medio de Scrum, la utilización de tableros Kanban y la representación visual del progreso, los equipos de DevOps planifican de forma eficaz con flexibilidad y transparencia (Azure, 2022).

Desarrollar

La etapa de desarrollo abarca todos los aspectos de la creación de software, incluida la codificación, prueba, revisión y combinación de código de los miembros del equipo, así como la compilación del código para su implementación en varias plataformas de hardware. Los equipos de DevOps se esfuerzan por mantener un ritmo rápido sin comprometer la calidad, la confiabilidad o la eficiencia. Para lograrlo, emplean herramientas eficientes, automatizan tareas rutinarias y refinan incrementalmente el código mediante pruebas automatizadas e integración continua (Azure, 2022).

Entregar

El proceso de entrega de solicitudes de manera confiable y consistente es esencial para garantizar que se reciban y ejecuten correctamente. Esto incluye no solo el acto de enviar y configurar la base central totalmente administrada, sino también establecer las condiciones necesarias para una ejecución exitosa (Azure, 2022).

Durante la fase de entrega, los equipos de DevOps establecen un ciclo de lanzamiento estructurado que incluye fases de soporte manual bien definidas. También crean transiciones programadas para mover soluciones de una etapa a otra hasta que estén listas para los usuarios. La automatización de estos ciclos garantiza que sean inspeccionados, adaptables y repetibles.

Como resultado, los grupos de DevOps pueden ofrecer resultados valiosos con confianza y sin problemas con una sensación de seguridad y tranquilidad (Azure, 2022).

Funcionamiento

Durante la fase operativa, la atención se centra en defender y validar las solicitudes, así como en investigar cualquier problema potencial que pueda surgir en las circunstancias actuales.

Los equipos de DevOps se esfuerzan por garantizar la confiabilidad, maximizar la accesibilidad y eliminar el tiempo de inactividad dentro del sistema, al mismo tiempo que priorizan la seguridad y la administración. Identificar proactivamente y abordar rápidamente los inconvenientes antes de que prejudiquen la experiencia del cliente es un objetivo clave para los equipos de DevOps. Esto requiere el uso de telemetría integral, medidas de precaución y una comprensión profunda tanto de las aplicaciones como del marco subyacente (Azure, 2022).

2.2.2. Principios básicos de Devops

DevOps es un enfoque cultural y colaborativo que tiene como finalidad eliminar la brecha entre la construcción de software y operativas de TI, haciendo hincapié en la comunicación, la colaboración y la integración fluidas entre los equipos multifuncionales (Kadaskar, 2024).

La automatización es un principio básico de DevOps, que permite a las compañias optimizar los procesos, reducir los esfuerzos manuales y mejorar la eficiencia. La automatización continua de tareas como las pruebas, la implementación y la supervisión contribuye a la excelencia operativa y a reducir el tiempo de comercialización

La integración continua (CI) es otro principio clave de DevOps, que se centra en la unificación frecuente de los cambios de código en un repositorio simultáneo.

Esta práctica asegura que las modificaciones de código de los desarrolladores se integren y prueben con regularidad, lo que promueve la detección temprana de los problemas de integración y mantiene la calidad del código (Kadaskar, 2024).

La entrega continua (CD) es esencial en DevOps, ya que permite a las organizaciones ofrecer actualizaciones de software a los clientes de forma rápida y sostenible.

Al automatizar el proceso de lanzamiento y garantizar que el software esté siempre en estado de implementación, la entrega continua permite a las compañías responder rápidamente a los requerimientos del mercado y ofrecer valor de manera eficiente (Kadaskar, 2024).

La supervisión continua es un principio fundamental en DevOps, que hace hincapié en la importancia de la visibilidad en tiempo real del rendimiento y el estado de las aplicaciones de software.

Al monitorear continuamente las métricas clave y el comportamiento del sistema, las organizaciones pueden identificar y abordar los problemas de manera proactiva, garantizando un rendimiento y una confiabilidad óptimos (Kadaskar, 2024).

La naturaleza iterativa de DevOps promueve la agilidad y disponibilidad de respuesta de los requisitos cambiantes, lo que permite a las compañias adaptarse rápidamente y ofrecer valor a los clientes de manera eficiente.

Al adoptar ciclos de desarrollo iterativos y ciclos de retroalimentación, DevOps fomenta una cultura de mejora e innovación continuas dentro de los equipos (Kadaskar, 2024).

2.2.3. Entregas continuas de software

Conceptos y Definiciones

Para implementar un software que cumpla con las reglas esenciales de velocidad, calidad y eficiencia, es crucial no solo desarrollar un código que cumpla con los requisitos tanto de la empresa como de sus clientes, sino también llevar a cabo una serie de actividades

adicionales relacionadas con la construcción, prueba, implementación y lanzamiento de producción del software.

Estos procesos, conocidos como Entrega Continua, se diseñan y ejecutan de manera coordinada y, siempre que sea posible, se automatizan por completo.

Este camino de Ingeniería de Software permite a los grupos generar de manera consistente software valioso para la empresa en ciclos cortos, al tiempo que garantiza que el software pueda lanzarse de manera confiable en cualquier momento (Chen, 2015).

Como afirma Fowler (2013), una compañía logra la entrega continua de acuerdo a los siguientes ciertos criterios:

• El software está permanentemente preparado para su implementación en cualquier etapa de su ciclo de vida.

• El equipo garantiza la preparación del software para su implementación en lugar de concentrarse en el desarrollo de nuevas funciones.

• En cualquier momento, cualquier persona puede recibir comentarios rápidos y automatizados sobre la preparación de los sistemas para la implementación del software después de cualquier modificación realizada en ellos.

• Con solo hacer clic en un botón, es posible implementar cualquier versión del software en cualquier entorno.

2.2.4. Gestión de la configuración

Conceptos y Definiciones

Como señala Humble (2014), la gestión de la configuración a menudo se equipara con el control de versiones y se describe como un procesamiento que implica el almacenamiento, la recuperación, la identificación única y la modificación de todos los artefactos relacionados con el software y sus interconexiones.

Importancia de Gestión de la Configuración

Según Humble (2014), la finalidad principal de la gestión de la configuración son dos. En primer lugar, pretende lograr reproducibilidad, es decir, la capacidad de automatizar la creación de cualquier entorno y garantizar que todos los entornos creados de esta manera tengan configuraciones idénticas.

En segundo lugar, enfatiza la trazabilidad, permitiendo una identificación rápida y precisa de las versiones de cada entorno y facilitando la comparación de versiones anteriores para identificar posibles variaciones.

2.2.5. Integración Continua

La integración continua es un método destinado a mejorar los productos, en el que participan los desarrolladores, que combinan sin problemas todas las modificaciones del código en un repositorio centralizado.

A continuación, se despliegan las pruebas y se crea la versión según un cronograma establecido. El término integración continua, que suele asociarse con la fase de desarrollo o integración del software, incluye un elemento automatizado que facilita la incorporación frecuente.

Los objetivos de la integración continua son acelerar la identificación y resolución de errores, mejorar la calidad del software y reducir el tiempo necesario para validar e implementar nuevos cambios de software. (Olmedo et al., 2018).

2.2.6. Proceso de despliegue

2.2.6.1. Introducción al proceso de despliegue de Devosp. El proceso de despliegue de DevOps es una metodología que integra el desarrollo de software (Dev) y las operaciones de TI (Ops) para mejorar la eficiencia y calidad del ciclo de vida del software. Según Humble y Farley (2010), DevOps se centra en la automatización, la colaboración y la integración continua para lograr un flujo de trabajo más eficiente y reducir el tiempo de entrega de cambios al software.

2.2.6.2. Metricas de Devops. Las métricas en DevOps son esenciales para evaluar la efectividad y el rendimiento de las prácticas de DevOps dentro de las organizaciones. Varios estudios han destacado la importancia de medir las métricas de DevOps para mejorar la eficiencia y analizar el rendimiento (Amaro et al., 2024). Las métricas de investigación y evaluación de DevOps (DORA) son ampliamente aceptadas en la industria para cuantificar el rendimiento de DevOps, pero es necesario adaptar estas métricas a los proyectos de software de código fuente abierto (OSS), centrándose en la frecuencia de lanzamiento, el tiempo de entrega de los cambios publicados, el tiempo de reparación del código y la tasa de problemas de errores para medir el rendimiento y la estabilidad (Ruiz et al., 2023) La implementación de la tecnología blockchain, específicamente Ethereum con programación Solidity, puede mejorar la seguridad y la integridad de los datos métricos y de registro dentro de los procesos de DevOps, garantizando un almacenamiento a prueba de manipulaciones, una auditoría transparente y un control de acceso. preciso (Dhawde, 2024) . Además, el análisis de los proyectos de DevOps mediante métodos bibliométricos ha demostrado que las métricas relacionadas con la integración continua, el diseño y las pruebas de software son de gran interés para los profesionales del software.

2.2.6.3. Frecuencia de Despliegue. La Frecuencia de Implementación, para DevOps, es la rapidez con la cual los cambios del software se publican e implementan en los entornos de producción. La automatización de la publicación e implementación de los cambios del código validados, de las prácticas de integración continua, y despliegue son claves para mejorar esta frecuencia (Istifarulah & Tiaharyadini, 2023). La automatización de la publicación e implementación de cambios de código validados permite a las organizaciones una mayor frecuencia de implementación, acorta los ciclos de lanzamiento y proporciona procesos uniformes para implementar cambios de código en cada versión, lo que conduce en última instancia a un software de mejor calidad (Istifarulah & Tiaharyadini, 2023). Además, la adopción de DevOps como un enfoque cultural facilita la colaboración entre los equipos de desarrollo y operaciones, permitiendo así que sea más fiable y rápido el proceso de implementación (Jha & Khan, 2018). Los estudios han demostrado que la implementación de DevOps puede reducir el tiempo de ciclo, aumentar la frecuencia de publicación del código y mejorar en general la rapidez y la calidad de las actividades de implementación del software (Melgar & Al, 2021). Además, la práctica de la entrega continua en DevOps permite a los grupos entregar versiones de software con frecuencia y eficiencia, incluso en sistemas antiguos, mediante procesos automatizados (Albuquerque & Cruz, 2019) .

La métrica de frecuencia de implementación monitorea la frecuencia con la que se realizan las implementaciones. Esto se vuelve esencial para sitios web y servicios en la nube con mucho tráfico. DevOps apunta a ejecutar implementaciones pequeñas y frecuentes, ya que minimizar el tamaño de cada implementación y la cantidad de cambios por ciclo facilita las pruebas y el lanzamiento. Este enfoque es preferible a tener menos versiones con cambios sustanciales. El seguimiento de esta métrica implica monitorear la cantidad de implementaciones no solo en producción, sino también dentro de los entornos de prueba y ensayo (Sentrio, 2021).

Existen varios métodos para realizar mediciones, entre ellos, los procesos de implementación automatizados, las llamadas a API o los scripts manuales. La frecuencia de implementación sirve como indicador directo e indirecto del tiempo de respuesta, la unidad del grupo, las habilidades de los desarrolladores, la efectividad de las herramientas de despliegue y la eficiencia general del equipo de DevOps (Sentrio, 2021)

2.2.6.4. Tiempo de Entrega de Cambios. Es el tiempo entre la confirmación y el código que ingresa a producción. Se espera que un equipo de alto rendimiento realice cambios en un plazo de entre un día y una semana. El Lead Time for Changes o tiempo de ejecución para cambios mide el tiempo que un commit (una corrección de errores, una nueva funcionalidad u otro cambio en el código) tarda en llegar a producción. Es decir, cuánto tiempo dedica el equipo a implementar, testear y entregar código a los usuarios.

También se conoce como Change Lead Time o Mean Time to Change. Es una de las cuatro medidas clave (Four Key Metrics) para medir el rendimiento del desarrollo de software con un enfoque DevOps y, sin duda, una de las métricas de velocidad más interesantes para que una organización pueda trabajar la reducción de sus tiempos de entrega. Esta métrica sirve como un fuerte reflejo de la eficacia del proceso de construcción, la complejidad del código y las capacidades del equipo.

Un Lead Time for Changes demasiado elevado advierte de que es posible que existan ineficiencias o cuellos de botella. Mientras que un Lead Time reducido muestra que el equipo es capaz de reaccionar rápidamente al *feedback* o los cambios en el mercado. Esta métrica sirve como un indicador eficaz de la eficiencia de construcción de software, la complejidad del código y las capacidades del grupo. Un Lead Time for Changes demasiado elevado advierte de que es posible que existan ineficiencias o cuellos de botella. Mientras que un Lead Time reducido muestra que el equipo es capaz de reaccionar rápidamente al *feedback* o los cambios en el mercado (Sentrio, 2021).

2.2.6.5. Tiempo de Recuperación ante Fallos. La duración promedio de la restauración del servicio después de una falla de producción se conoce como MTTR (tiempo medio de restauración). Esto incluye la recuperación de problemas como una base de datos dañada o una confirmación defectuosa que interrumpe una función. Los equipos de alto rendimiento tienen como objetivo restablecer los servicios en un día o menos. El objetivo es mejorar la velocidad de implementación a través de la automatización y optimizar las pruebas de integración para reducir el tiempo general de implementación. Esto proporciona una métrica clara para evaluar el progreso del equipo en términos de implementaciones, tanto internamente como para clientes externos. MTTR mide el tiempo que lleva restaurar un servicio, desde que se reporta un incidente hasta que se resuelve por completo. Se centra en la capacidad de respuesta de un equipo de DevOps a los problemas de atención al cliente y su capacidad para implementar soluciones efectivas (Sentrio, 2021).

2.2.6.6. Tasa de Fallos en Despliegues. Es la asociación entre cambios fallidos y cambios exitosos en el servicio. Por ejemplo, si implementamos cuatro veces por semana y 3 de ellas fallan por algún motivo (como un error en el código o una tubería estrecha), entonces nuestro CFR será del 75 %. Idealmente, en un servicio de alto rendimiento, la tasa de falla debería ser inferior al 15%. Las altas tasas de fallas pueden indicar en el proceso de DevOps y provocar tiempos de inactividad, lo que resulta en una disminución de ingresos para la empresa. El fracaso, aunque es mejor evitarlo, a veces conduce a contextos que nos impulsan a analizar nuevos paradigmas. (Sentrio, 2021).

2.3. Marco conceptual

Agile, en un enfoque empresarial que enfatiza ciclos cortos, planificación y desarrollo repetitivo para proporcionar mayor control, previsibilidad y soporte para requisitos cambiantes a medida que evoluciona un proyecto.

Arquitectura de software, es la arquitectura lógica, se basa en un conjunto de ideas y modelos abstractos relacionados que abastecen un marco bien definido para articular con el código fuente del software.

Automatización de procesos, implica la mejora, reducción y automatización de procesos esenciales que impulsan un negocio hacia adelante, buscando principalmente reducir gastos a través de la integración de aplicaciones, disminuir los requerimientos de mano de obra, acelerar los tiempos de ejecución de actividades y sustituir procesos manuales por aplicaciones de software.

Compatibilidad de componentes, es la condición que conduce directa o indirectamente (a través de algoritmos) al correcto funcionamiento de programas y sistemas, arquitecturas o aplicaciones.

Docker, es una plataforma de código abierto para construir software, entregar y operativizar aplicaciones que le permite desacoplar las aplicaciones de la infraestructura para poder entregar software eficazmente.

Entornos de desarrollo, es un conjunto de herramientas que automatizan o soportan al menos la mayoría de las fases de desarrollo: análisis de requisitos, diseño arquitectónico, diseño detallado, codificación, pruebas unitarias, pruebas de integración y verificación, gestión de configuración, mantenimiento, etc. Estas herramientas deben estar bien acopladas para que puedan interactuar.

Entrega continua, es una perspectiva de construcción de software en la que los miembros producen software en ciclos óptimos, asegurando que el software pueda implementarse de manera confiable en cualquier momento.

Funciones sin estado, estas funciones se realizan generalmente dentro de contenedores seguros y sin estado. Como resultado, no es posible ejecutar código en un servidor de

aplicaciones de larga ejecución una vez que el evento ha concluido, ni se puede utilizar el contexto de ejecución anterior para cumplir con la solicitud.

Feedback, es un procedimiento de control del sistema; los resultados obtenidos se retroalimentan al sistema para verificar y mejorar su funcionamiento.

Iterativo, es el equivalente a realizar un proceso varias veces para lograr un resultado deseado u objetivo.

Integración Continua, es una práctica de despliegue de software en la que los integrantes del equipo integran su trabajo con frecuencia, al menos una vez al día, lo que resulta en múltiples integraciones diarias.

Software, es un conjunto de programas informáticos, prácticas, estándares que conforman el funcionamiento de un sistema informático.

III. MÉTODO

3.1. Tipo de investigación

Tipo

El estudio emplea un enfoque cuantitativo y se clasifica como investigación aplicada. Esta clasificación surge de la identificación de un problema, lo que lleva al uso de la investigación para abordar indagaciones específicas. Como lo especifica Hernández Sampieri et al. (2014), la investigación aplicada involucra metodologías para evaluar, comparar, interpretar, establecer precedentes y determinar causalidad junto con sus aplicaciones.

Nivel

La investigación que se realiza es de carácter explicativo. Tal como lo define Hernández Sampieri et al. (2014), el propósito de la investigación de nivel explicativo es determinar las causas subyacentes de los eventos o fenómenos investigados.

Diseño

El diseño de la investigación se caracteriza por ser longitudinal y preexperimental. Como lo especifica Hernández Sampieri et al. (2014), un diseño preexperimental se refiere a un enfoque de investigación que implica la administración de estímulos, tratamientos y/o intervenciones con un nivel mínimo de control.

Su representación es la siguiente:

Figura 4

Diseño de preprueba/posprueba con un solo grupo.

G: Representa el grupo al que se aplica el estudio (por ejemplo, un área de desarrollo de software en una universidad pública).

M_1: Es la medición previa (pretest) de la variable dependiente antes de aplicar el tratamiento o intervención.

X: Representa la intervención o tratamiento (Implementación de la metodología DevOps).

M_2: Es la medición posterior (postest) después de aplicar el tratamiento.

Al comienzo, se tomará una medición de los indicadores del proceso: número de depliegues, tiempo desde cambio hasta entrega, tiempo para recuperar errores y cantidad de fallos en lanzamientos antes de realizar la acción (X), que será adoptar de forma gradual métodos DevOps. Luego, se hará otra medición (M_2) de las mismas reglas para notar cualquier cambio o mejora que pudo causar la acción. Este plan deja ver cómo fue el proceso de despliegue de software antes y después de usar DevOps en una universidad pública, en condiciones controladas dentro un lugar real.

3.2. Población y muestra

Población

Según Carrasco (2007), es una colección de todos los elementos en el espacio de dominio al que pertenece la pregunta de investigación y tiene características más específicas que el universo. Se ha identificado como unidad de análisis los procesos de despliegue de software, al estar en constante despliegue no es posible determinarla con precisión su número total en un periodo fijo, considerándose indeterminado.

N = Indeterminado

Muestra

Según Carrasco (2007), fragmentos demográficos caracterizados por la objetividad, obtienen resultados generalizados para todos los items de la población.

La muestra para la presente investigación será intencional y utilizó 16 semanas el cual corresponde a un periodo académico. Esta muestra fue seleccionada por su accesibilidad y pertinencia para evaluar los efectos de la intervención (implementación de la metodología DevOps) bajo condiciones controladas.

3.3. Operacionalización de variables

3.3.1. Variable independiente: Metodología Devops

Definición conceptual

DevOps, combina los términos "desarrollo" y "operaciones", significa la unificación de personas, procesos y tecnología para brindar valor confiable a los clientes. Esta metodología promueve la colaboración entre roles que antes eran distintos, incluidos el desarrollo, las operaciones de TI, la ingeniería de calidad y la seguridad, lo que conduce a la creación de productos de mayor calidad y con alto grado de confiabilidad. Al adoptar una cultura DevOps y emplear herramientas y prácticas DevOps, los equipos pueden satisfacer con éxito los requerimientos de los clientes, mejorar su confianza en las aplicaciones y acelerar la consecución de los objetivos comerciales (Azure, 2022).

Definición operacional

Tabla 1

Operacionalización de la variable independiente: Metodología Devops

Variable	Indicador	Descripción
Metodología Devops	Presencia - ausencia	Ausencia muestra que no se aplica la metodología Devops y Presencia indica que se aplica la metodología Devops

Nota. Elaboración Propia

3.3.2. Variable dependiente: Proceso de despliegue de software

Definición conceptual

El término “despliegue de software” describe las actividades utilizadas para desplegar paquetes de software o aplicaciones. Normalmente, un administrador de la organización o de TI se encarga de ello, ya que no todos los usuarios tienen los conocimientos o la autorización necesarios. El objetivo de la distribución de software es la instalación inicial, la configuración y el mantenimiento de múltiples aplicaciones de software en varios dispositivos de forma eficaz, automatizada, y libre de errores. Esto garantiza que los procesos operativos permanezcan intactos y que su entorno de trabajo sea productivo (Deskcenter, 2022).

Definición operacional

Tabla 2

Operacionalización de la variable dependiente: Proceso de despliegue de software

Variable	Dimensión	Indicador	Unidad de observación
Proceso de despliegue de software	Frecuencia	Frecuencia de Despliegue	Observación directa
	Tiempo	Tiempo de Entrega de Cambios	Observación directa
		Tiempo de Recuperación ante Fallos	Observación directa
	Fallas	Tasa de Fallos en Despliegues	Observación directa

Nota. Elaboración Propia

3.4. Instrumentos

La presente investigación empleó el método de fichas de observación para la recolección de datos dentro de la organización. Como señala Carrasco Díaz (2006), este enfoque implica documentar información importante y relevante para la investigación mediante el uso de fichas o tablas conocidas como fichas de observación. Los datos recolectados surgen del compromiso directo entre el investigador y la realidad observada.

3.5. Procedimientos

Para concretizar los objetivos propuestos de la investigación se inicia con aplicación de un registro pre-test esta información se procesará con un software estadístico para obtener información descriptiva a continuación se ejecuta un registro de datos pretest y postest, seguidamente se procesa software estadístico y obtener información descriptiva, finalmente aplicamos estadística inferencial con el fin de contrastar la hipótesis.

3.6. Análisis de datos

El proceso de análisis de datos estadísticamente se realizará con software que permitirá un análisis estadístico que fundamente el trabajo de investigación. Para realizar el trabajo de análisis utilizaremos la estadística descriptiva y se aplicarán las pruebas estadísticas relevantes para alcanzar el objetivo propuesto mediante la estadística inferencial de acuerdo con la prueba de normalidad para la contratación de las hipótesis.

3.7. Consideraciones éticas

Este estudio se apegó a los estándares éticos de la Asociación Americana de Psicología, incorporando el principio de respeto a las referencias de las cuales se obtuvieron los datos a través de cuestionarios procesados por el autor. Además, defendió el principio de objetividad al basar el análisis contextual en evidencia fáctica, criterios establecidos y metodologías. Por último, enfatizó el principio de originalidad, marcando el inicio del proceso de recuperación de información para la investigación, con énfasis en la citación de fuentes bibliográficas.

IV. RESULTADOS

4.1 Resultados descriptivos

4.1.1 Medidas descripticas del indicador 1: Frecuencia de despliegue

Tabla 3

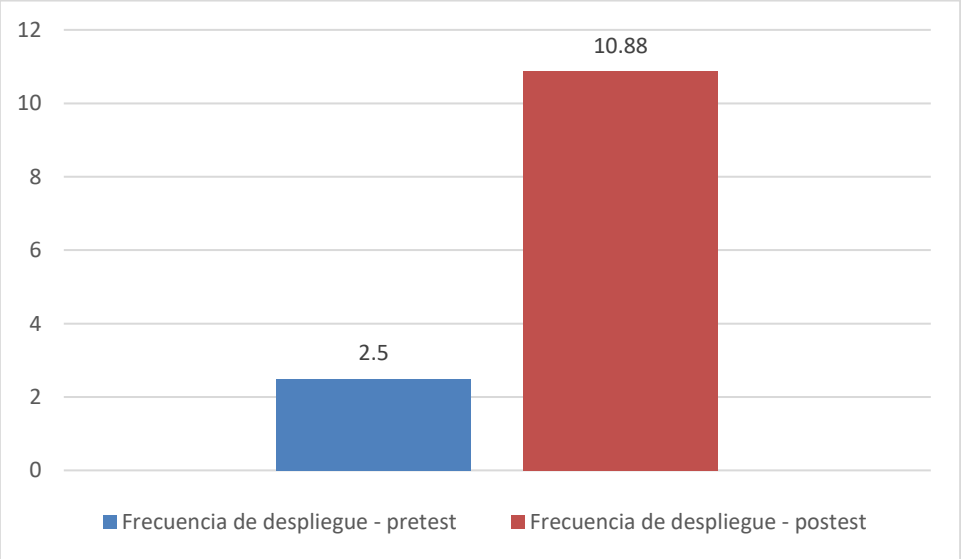
Medidas descripticas del indicador 1: Frecuencia de despliegue

	N	Mínimo	Máximo	Media	Desviación Estándar
Frecuencia de Despliegue - pretest	16	1	4	2,50	,966
Frecuencia de Despliegue - postest	16	5	15	10,88	3,18

De acuerdo con la Tabla 3 los datos del pretest sobre la frecuencia de despliegue revelan una media de 2,50 despliegues, que van de 1 a 4, y una desviación estándar de 0,966. Sin embargo, tras la intervención, la media aumentó a 10,88 despliegues, con un rango de 5 a 15 y una desviación estándar de 3,18. Esto implica que no sólo aumentó la frecuencia de despliegue, sino que también se volvió más consistente, demostrando un mayor nivel de estabilidad.

Figura 5

Comparación de medias del indicador 1: Frecuencia de despliegue



4.1.2 Medidas descripticas del indicador 2: Tiempo de entrega de cambios

Tabla 4

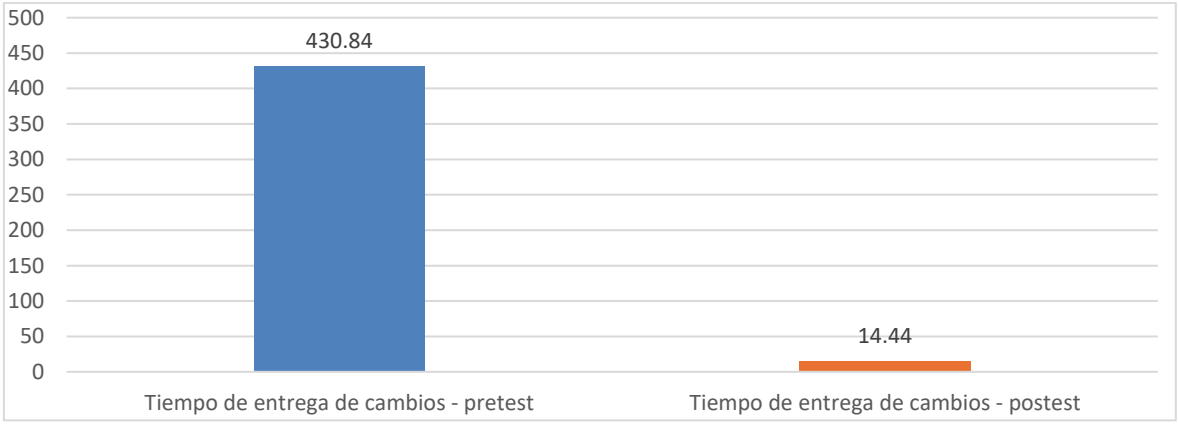
Medidas descripticas del indicador 2: Tiempo de entrega de cambios

	N	Mínimo	Máximo	Media	Desviación estándar
Tiempo de Entrega de Cambios - pretest	16	321,15	590,21	430,84	80,29
Tiempo de Entrega de Cambios - postest	16	10,49	19,14	14,44	2,81

De acuerdo con la Tabla 4 los resultados evidencian una mejora notable en el Tiempo de Entrega de Cambios tras la implementación de Devops. En el pretest, el tiempo promedio fue de 430,84 minutos, con una desviación estándar de 80,29, lo que indica una mayor dispersión en los tiempos (mínimo de 321,15 y máximo de 590,21 minutos). En contraste, en el postest, el tiempo promedio disminuyó significativamente a 14,44 minutos, con una desviación estándar de 2,81, reflejando mayor consistencia en los tiempos (mínimo de 10,49 y máximo de 19,14 minutos).

Figura 6

Comparación de medias del indicador 2: Tiempo de entrega de cambios



4.1.3 Medidas descripticas del indicador 3: Tiempo de recuperación ante fallos

Tabla 5

Medidas descripticas del indicador 3: Tiempo de recuperación ante fallos

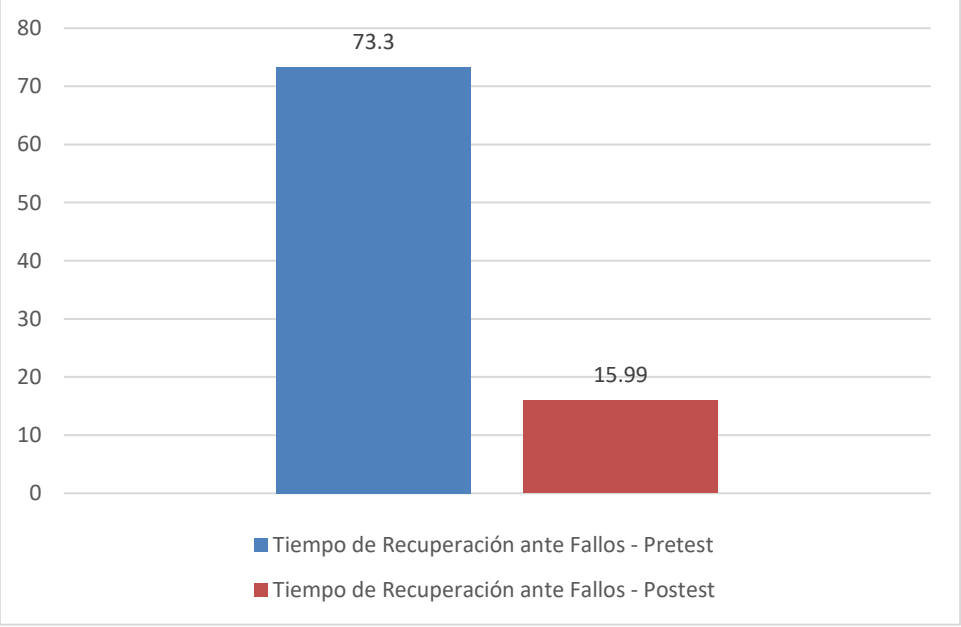
	N	Mínimo	Máximo	Media	Desviación estándar
Tiempo de Recuperación ante Fallos - Pretest	16	40,45	104,82	73,30	16,94
Tiempo de Recuperación ante Fallos - Postest	16	2,24	26,35	15,99	6,15

De acuerdo con la Tabla 5 la intervención dio como consecuencia una rebaja sustancial en la tasa de fallas, lo que marcó una mejora significativa en los tiempos de despliegues.

Antes de la intervención, los tiempos de recuperación ante fallos eran prevalentes, con un valor de 73,30 minutos , sin embargo, tras la intervención descendió al 26,35 con una desviación estándar de 6,70. Esta reducción en el tiempo de recuperación ante fallos refuerza aún más la mayor estabilidad y confiabilidad logradas a través de la intervención.

Figura 7

Comparación de medias del del indicador 3: Tiempo de recuperación ante fallos



4.1.4 Medidas descripticas del indicador 4: Tasa de fallos en despliegues

Tabla 6

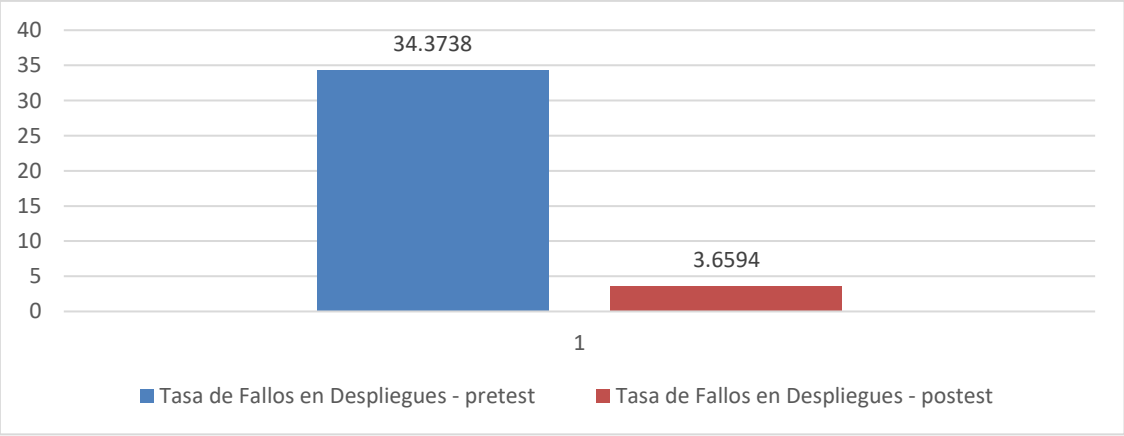
Medidas descripticas del indicador 3: Tasa de fallos en despliegues

	N	Mínimo	Máximo	Media	Desviación estándar
Tasa de Fallos en Despliegues - Pretest	16	,00	50,00	34,38	16,36
Tasa de Fallos en Despliegues - Postest	16	,00	11,11	3,66	4,40

De acuerdo con la Tabla 6 la intervención dio como consecuencia una rebaja sustancial en la tasa de fallas, lo que marcó una mejora significativa en la confiabilidad y consistencia de los despliegues. Antes de la intervención, las fallas eran prevalentes, con un porcentaje promedio de fallas del 34,38% con una desviación estándar de 16,36, lo que indica una alta frecuencia de errores y una gran dispersión entre los valores, sin embargo, tras la intervención, la tasa media de fracaso descendió al 3,66% con una desviación estándar de 4,40 mostrando una disminución significativa en los fallos y una mayor consistencia en los resultados. Esta reducción en la tasa de fallas sugiere una mejora sustancial en el proceso de despliegue a través de la intervención.

Figura 8

Comparación de medias del del indicador 3: Porcentaje de fallas



4.2 Resultados inferenciales

4.2.1 Prueba de Normalidad

Al ser la muestra menor que 50 elementos se empleó la prueba estadística de Shapiro Wilk (Mara, 2011).

Tabla 7

Prueba de normalidad de los indicadores

	Shapiro-Wilk		
	Estadístico	gl	Sig.
Diferencia - Frecuencia de Despliegue	,874	16	,032
Diferencia - Tiempo de Entrega de Cambios	,931	16	,254
Diferencia -Tiempo de Recuperación ante Fallos	,958	16	,622
Diferencia - Tasa de Fallos en Despliegues	,865	16	,023

De acuerdo con la Tabla 6 las salidas muestran que las diferencias en la Frecuencia de Despliegue (p=0,032) y la Tasa de Fallos en Despliegues (p=,023) no siguen una distribución normal, lo que sugiere el uso de estadígrafos no paramétricos como la prueba de Wilcoxon para el análisis. En cambio, las diferencias en el Tiempo de Entrega de Cambios (p=,254) y el Tiempo de Recuperación ante Fallos (p=,622) siguen una distribución normal, por lo que se utilizó los estadígrafos paramétricos como la prueba t de Student para muestras pareadas en estos casos.

4.2.2 Prueba de Hipótesis del indicador 1: Frecuencia de Despliegue

Hipótesis Nula H_0: Si se implementa una metodología Devops entonces no influye positivamente en la frecuencia de despliegue en una Universidad

Hipótesis Alternativa H_1: Si se implementa una metodología Devops entonces influye positivamente en la frecuencia de despliegue en una Universidad

Tabla 8

Estadístico de contraste del indicador 1: Frecuencia de despliegue

	Frecuencia de despliegue (postest -pretest)
Z	-3,529[b]
Sig. asin. (bilateral)	,000

a. Prueba de rangos con signo de Wilcoxon
b. Se basa en rangos negativos.

De acuerdo con la Tabla 8 la prueba de rangos con signo de Wilcoxon para la "Frecuencia de Despliegue" entre pretest y postest muestra un valor Z de -3,529, con un p-valor de <.001. Este resultado es estadísticamente significativo, lo que indica una diferencia notable en la frecuencia de despliegue entre los dos periodos. Esto sugiere que, tras la intervención o cambio implementado, la capacidad para realizar despliegues aumentó, reflejando un proceso más eficiente y posiblemente una mayor confianza en los sistemas de despliegue automatizados.

En consecuencia, se rechaza la hipótesis nula (H0) y se acepta la hipótesis alternativa (H1), concluyendo que la metodología DevOps sí influye positivamente en la frecuencia de despliegue en la Universidad.

4.2.3 Prueba de Hipótesis del indicador 2: Tiempo de entrega de cambios

Hipótesis Nula H_0: Si se implementa una metodología Devops entonces no influye positivamente en el tiempo de entrega de cambios en una Universidad Pública.

Hipótesis Alternativa H_2: Si se implementa una metodología Devops entonces influye positivamente en el tiempo de entrega de cambios en una Universidad Pública.

Tabla 9

Estadístico de contraste del indicador 2: Tiempo de entrega de cambios

	Diferencias emparejadas							
				95% de intervalo de confianza de la diferencia				
	Media	Desviación estándar	Media de error estándar	Inferior	Superior	t	gl	Sig. (bilateral)
Tiempo de entrega de cambios (pretest – postes)	416,39	79.11	19.78	374.24	458.56	21.05	15	.000

De acuerdo con la Tabla 9 muestra el estadístico de contraste del indicador de "Tiempo de Entrega de Cambios" donde se observa una media de 416.39 entre el pretest y el postest, con una desviación estándar de 19.78 minutos y un error estándar de 19.78 minutos. El estadístico t es de 21.05 y 15 grados de libertad, y un p-valor de .000, lo cual es inferior al nivel de significancia de 0.05, permitiendo negar la hipótesis nula y confirma la hipótesis alternativa, concluyendo que la metodología Devops entonces influye positivamente en el tiempo de entrega de cambios en una Universidad Pública.

4.2.4 Prueba de Hipótesis del indicador 3: Tiempo de Recuperación ante Fallos

Hipótesis Nula H_0: Si se implementa una metodología Devops entonces influye positivamente en el tiempo de recuperación ante fallos en una Universidad Pública.

Hipótesis Alternativa H_3: Si se implementa una metodología Devops entonces no influye positivamente en el tiempo de recuperación ante fallos en una Universidad Pública.

Tabla 10

Estadístico de contraste del indicador 3: Tiempo de recuperación ante fallos

	Diferencias emparejadas							
				95% de intervalo de confianza de la diferencia				
	Media	Desviación estándar	Media de error estándar	Inferior	Superior	t	gl	Sig. (bilateral)
Tiempo de recuperación ante fallos (pretest – postes)	57,30	19,85	4,96	46,72	67,88	11,54	15	,000

De acuerdo con la Tabla 9 las salidas de la prueba de diferencias emparejadas para el "Tiempo de Recuperación ante Fallos" entre pretest y postest indican una reducción significativa de 57,30 minutos en promedio, con una desviación estándar de 19,85 minutos y un error estándar de 4,96 minutos.

El valor t obtenido es 11,545 con 15 grados de libertad, y el p-valor es ,000, lo cual es inferior al nivel de significancia de 0,05, permitiendo rechazar la hipótesis nula y confirmando la hipótesis nula que la metodología DevOps sí influye positivamente en la mejora del tiempo de recuperación ante fallos en la Universidad Pública.

4.2.5 Prueba de Hipótesis del indicador 4: Tasa de Fallos en Despliegues

Hipótesis Nula H_0: Si se implementa una metodología Devops entonces influye positivamente en la tasa de fallos en despliegues en una Universidad Pública.

Hipótesis Alternativa H_3: Si se implementa una metodología Devops entonces no influye positivamente en tasa de fallos en despliegues en una Universidad Pública.

Tabla 11

Estadístico de contraste del indicador 4: Tasa de fallos en despliegues

	Tasa de fallos en despliegues (pretest – postest)
Z	-3,376[b]
Sig. asin. (bilateral)	,000

a. Prueba de rangos con signo de Wilcoxon
b. Se basa en rangos negativos.

De acuerdo con la Tabla 11 la prueba de rangos con signo de Wilcoxon para la "Tasa de Fallos en Despliegues" entre pretest y postest indica un valor Z de -3.376 con una significancia asintótica bilateral fue de ,000 lo que indica una diferencia estadísticamente significativa entre los resultados del pretest y el postest. Dado que el resultado se basa en rangos negativos, lo que significa que hubo una disminución de la tasa de fallos en el postest, tras la implementación de la metodología Devops. Por lo tanto, se niega la hipótesis nula (H0) y se acepta la hipótesis alternativa (H3), concluyéndose que la metodología DevOps sí influye positivamente en la reducción de la tasa de fallos en los despliegues en la Universidad Pública.

V. DISCUSIÓN DE RESULTADOS

La interpretación y contextualización de las salidas de la investigación son componentes esenciales de cualquier estudio, y la discusión de los resultados sirve para este propósito. En el presente estudio, examinaremos los resultados obtenidos al implementar la metodología DevOps en una universidad pública, considerando también investigaciones previas y teorías relevantes para proporcionar un análisis integral.

Los resultados logrados en este análisis muestran de forma estadísticamente clara que poner en acción la forma de trabajar DevOps ayuda bastante en los cuatro puntos importantes revisados: número de veces que se lanza al mercado una nueva versión, lo que tarda cambiar algo; lo que cuesta solucionar un problema y cuántos errores tiene cada cambio. Estas mejoras están de acuerdo con varios estudios previos, tanto aquí en Casa como en otros lugares, que han mostrado ventajas parecidas después de usar métodos DevOps.

En cuanto a la rapidez con que se hacen los cambios a los programas, el número Z de -3.529 ($p = .000$) muestra un gran aumento después de poner en uso DevOps. Esto va bien con la revisión por Moreyra (2023), quien encontró que la frecuencia de lanzar nuevas versiones fue más alta por un 55.17% en una empresa financiera. En el 2020, Casas vio que la frecuencia en liberar el código subió de 1.63 a 3.25 por sprint, apoyando así que DevOps ayuda mucho a agilizar el lanzar constante en un lugar de trabajo. A nivel mundial Machuzhak (2023) reportó un mejoramiento del 25.47% en la cantidad de cambios al usar DevOps en aplicaciones web, esto coincide bien con lo hallazgos obtenidos.

Entre el tiempo de entrega de cambios el estudio actual enseña una baja clara (promedio = 416.39 min; $t = 21.05$; $p = .000$), lo cual enseña una mayor eficiencia en la entrega de funcionalidades. Estos resultados son similares a lo que dijo Vigo et al. (2020),

quienes mencionaron una reducción mayor en el tiempo para implementar en empresas pequeñas gracias a la combinación de DevOps. Además, Ahmed et al. (2023) resaltaron grandes mejoras en la rapidez para poner en marcha aplicaciones en la nube después de cambiar procesos con herramientas como Docker y Jenkins.

En cuanto al tiempo de recuperación ante problemas, los resultados muestran una baja promedio de 57.30 minutos ($t = 11.54$; $p = .000$), lo que prueba una manera más veloz contra fallos. Esta mejora concuerda con los hallazgos de Peralta (2019), quien halló una baja del 57.42% en errores fallidos después de poner DevOps en el Ministerio de Cultura. De igual manera Port et al. (2024) concluyeron que las acciones de DevOps son útiles para tener a bajos los niveles de errores en sistemas importantes como el sistema MONTE de la Autoridad Portuaria de Tanzania, confirmando su empleo en campos que necesitan alta disponibilidad.

Por último, sobre la cantidad de errores en despliegues, la revisión con Wilcoxon ($Z = -3.376$; $p = .000$) enseña una baja importante, lo cual sugiere mejor calidad en los depliegues. Este resultado está de acuerdo con los estudios de Machuzhak (2023), quien dice que la tasa de errores bajó del 55,56% al 11,11%, y con lo visto por Casas (2020), donde la tasa de éxito subió de 54.58% a 72.92% en el sector de seguros. De igual forma, Rüegger et al. (2024) mostraron que el seguimiento automático de métricas DORA dejó una reducción de errores y una mejora continua en los procesos de entrega.

Los hallazgos de esta investigación validan la noción de que la introducción de DevOps en una universidad pública tiene la posibilidad de mejorar significativamente la efectividad y la excelencia del procedimiento de implementación de software. La credibilidad de estos resultados se ve reforzada aún más al corroborar la evidencia de otras investigaciones, que sugieren que la implementación de los principios de DevOps puede producir mejoras comparables en entornos educativos. Para mantener la eficiencia operativa

durante las implementaciones, es imperativo priorizar y adoptar la automatización y la integración continua. El cultivo de un entorno de colaboración entre los grupos de desarrollo y operaciones juega un papel fundamental para lograr el éxito con DevOps. Además, es crucial brindar capacitación a los miembros del personal sobre metodologías DevOps y tecnologías emergentes para mantener y mejorar aún más las ventajas observadas.

El logro de este éxito en DevOps implica la necesidad de incorporar la automatización de procesos como un elemento fundamental. Para garantizar la eficiencia y calidad en las implementaciones de software, es vital integrar las tecnologías de CI / CD. Asimismo, es igualmente importante crear una cultura organizacional colaborativa que priorice la comunicación y derribe los silos entre los equipos de desarrollo y los de operaciones La adopción de DevOps implica un cambio en la cultura y en las capacidades, lo que implica el desarrollo de formación constante en metodologías DevOps y la implementación de nuevas tecnologías. Esto garantizará que el equipo esté en condiciones de mantener y mejorar los beneficios obtenidos. Los resultados documentados y su validación con otras investigaciones validan la idea de que DevOps es un enfoque eficaz para mejorar los procedimientos de implementación de la Universidad. La mejora en términos de Este estudio reporta la puntualidad, la regularidad y la calidad de la implementación respaldan los hallazgos anteriores, lo convalida la idea de que DevOps puede ayudar a todas las organizaciones que buscan mejorar su funcionamiento de TI. Los resultados ofrecen un apoyo crucial para el despliegue de DevOps.

VI. CONCLUSIONES

- Respecto al objetivo general el estudio evidencio una influencia positiva al incorporar la metodología DevOps en el proceso de implementación de software en una universidad pública, los resultados del pretest y postest confirman avances notables en sus cuatro métricas cruciales: Frecuencia de Despliegue, Tiempo de Entrega de Cambios, Tiempo de Recuperación ante Fallos, Tasa de Fallos en Despliegues. Las mejoras observadas en estas áreas son estadísticamente significativas, lo que subraya la influencia positiva sustancial de la adopción de DevOps en la eficacia y excelencia del proceso de despliegue de software.
- Respecto a la frecuencia de despliegue, la prueba de rangos con signo de Wilcoxon entre el pretest y el postest arrojó un valor Z de -3.529, con un p-valor de <0.001, lo que indica un aumento significativo en la capacidad para ejecutar despliegues tras la adopción de prácticas DevOps. Los hallazgos confirman un aumento significativo en la frecuencia de despliegue tras la intervención, y respaldan la base teórica que sostiene que la automatización y la integración continua fortalecen la eficiencia operativa del proceso de desarrollo y despliegue de software.
- Respecto a los tiempos de entrega, el estudio evidencio una reducción significativa ($t = 21.053$ y un nivel de significancia de $p < 0.05$). Esta mejora sugiere que ha medida se incorporan orientadas a la automatización, integración continua y mejora de la coordinación entre equipos, se optimiza el flujo de trabajo, admitiendo que los cambios de código se implementen de manera más ágil y eficiente fortaleciendo la capacidad de respuesta tecnológica de la universidad.
- Respecto al tiempo de recuperación de fallos, el estudio evidencio una reducción estadísticamente significativa ($t = 11.545$ y un nivel de significancia de < 0.05). Esto

significa que, la integración de las prácticas de DevOps, como el monitoreo continuo y la automatización de respuestas a incidentes, ha sido efectiva para disminuir el tiempo de inactividad y mitigar de manera efectiva el impacto de los fallos, contribuyendo a una mayor continuidad y estabilidad de los servicios tecnológicos de la universidad.

- Finalmente, respecto a la tasa de fallos en los despliegues mostró una reducción relevante según la prueba de Wilcoxon con un valor Z de -3.376 con un valor p=0.05. Estos resultados confirman que las medidas implementadas han sido efectivas en mejorar la estabilidad y confiabilidad del proceso de despliegue de software. Esto implica que al potenciar las prácticas como como las pruebas automatizadas, la integración continua y el control de calidad, se minimiza los errores, compactando proceso de despliegues seguros y consistentes en la universidad.

VII. RECOMENDACIONES

- Se sugiere al área de desarrollo de software de la universidad asegure una mejora continua en la efectividad y la calidad de los procedimientos de implementación de software. Dentro de la institución, un enfoque DevOps debe ser integrado y mantenido en todo momento en el ciclo de vida del software.
- Se sugiere al área de desarrollo de software de la universidad mejore la adopción de prácticas de calidad, incluidos el análisis de código, pruebas y pruebas automatizadas de integración continua. Dado que estas prácticas permiten identificar y resolver problemas antes de que lleguen a producción, los despliegues son seguros y eficientes.
- Se sugiere área de desarrollo de software de la universidad continue avanzando en la implementación de herramientas de monitoreo y pipeline automático. Se podrían identificar los cuellos de botella rápidamente con estos métodos, optimizando aún más el proceso de despliegue de software
- Se sugiere al área de desarrollo de software de la universidad que el personal de la universidad esté capacitado en herramientas de monitoreo y respuesta a incidentes. Con programas de formación continua en DevOps y nuevas tecnologías pueden ayudar a los equipos a mantenerse actualizados con las mejores prácticas y herramientas del sector.
- Se sugiere al área de desarrollo de software de la universidad que se realice de manera continua las políticas las políticas de implementación en el área de desarrollo de software y mejoras continúas basadas en los datos obtenidos del monitoreo continuo

VIII. REFERENCIAS

Ahmed Mateen Buttar, Adeel Khalid, Mamdouh Alenezi, Muhammad Azeem Akbar, Saima Rafi, Abdu Gumaei, & Muhammad Tanveer Riaz. (2023). Optimization of DevOps Transformation for Cloud-Based Applications. *Electronics*, *12*(2), 357-357. https://doi.org/10.3390/electronics12020357

Albuquerque, A. B., & Cruz, V. L. (2019). Implementing DevOps in Legacy Systems. En R. Silhavy, P. Silhavy, & Z. Prokopova (Eds.), *Intelligent Systems in Cybernetics and Automation Control Theory* (pp. 143-161). Springer International Publishing. https://doi.org/10.1007/978-3-030-00184-1_14

Amaro, R., Pereira, R., & Mira Da Silva, M. (2024). DevOps Metrics and KPIs: A Multivocal Literature Review. *ACM Computing Surveys*, *56*(9), 1-41. https://doi.org/10.1145/3652508

Azure. (2022). *¿Qué es DevOps? Explicación de DevOps | Microsoft Azure*. https://azure.microsoft.com/es-es/overview/what-is-devops/

Bravo Ramírez, M. A., Huasasquiche Calmet, P. M., & Ojeda Monzón, C. P. (2020). *Análisis comparativo entre Waterfall y devOps en el desarrollo del microservicio "Obtener detalle de libros en tienda virtual"*.

Carrasco Díaz, S. (2006). *Metodología de la investigación científica: Pautas metodológicas para diseñar y elaborar el proyecto de investigación*. San Marcos.

Casas, F. (2020). *Implementación de DevOps para mejorar la integración y despliegue de software en el sector de seguros*. Lima. [Tesis de grado, Universidad Cesar Vallejo]. Repositorio Institucional UCV. https://repositorio.ucv.edu.pe/handle/20.500.12692/41593

Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, *32*(2), 50-54. https://doi.org/10.1109/MS.2015.27

Cuadra, J., Hurtado, E., Sarachaga, I., Estévez, E., Casquero, O., & Armentia, A. (2024). Enabling DevOps for Fog Applications in the Smart Manufacturing domain: A Model-Driven based Platform Engineering approach. *Future Generation Computer Systems*, *157*, 360-375. https://doi.org/10.1016/j.future.2024.03.053

Deskcenter. (2022). *Gestione su despliegue de software de forma segura con Deskcenter*. https://www.deskcenter.com/es/deskcenter-management-suite/gestion-de-software/despliegue-de-software/

Dhawde, S. D. (2024). Securing Logs and Metrics in DevOps Using Blockchain: Enhancing Trust and Integrity in Continuous Deployment. *International Journal for Research in Applied Science and Engineering Technology*, *12*(6), 1441-1450. https://doi.org/10.22214/ijraset.2024.63335

DevOps Research and Assessment. (2018). *Accelerate: The science of lean software and DevOps*. IT Revolution Press.

Díaz, J., Pérez, J., Alves, I., Kon, F., Leite, L., Meirelles, P., & Rocha, C. (2024). Harmonizing DevOps taxonomies—A grounded theory study. *Journal of Systems and Software*, *208*, 111908. https://doi.org/10.1016/j.jss.2023.111908

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: Building and scaling high performing technology organizations*.

Fowler, M. (2013). *ContinuousDelivery*. https://martinfowler.com/bliki/ContinuousDelivery.html

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). Metodología de la investigación. En *Metodología de la investigación* (Vol. 6, Número Mc Graw-Hill). McGraw Hill.

Humble, J. (2014). *The Case for Continuous Delivery | Thoughtworks*. https://www.thoughtworks.com/insights/blog/case-continuous-delivery

Istifarulah, M. H. R., & Tiaharyadini, R. (2023). DevOps, Continuous Integration and Continuous Deployment Methods for Software Deployment Automation. *JISA(Jurnal Informatika Dan Sains)*, *6*(2), Article 2. https://doi.org/10.31326/jisa.v6i2.1751

Jha, P., & Khan, R. (2018). A Review Paper on DevOps: Beginning and More To Know. *International Journal of Computer Applications*, *180*(48), 16-20. https://doi.org/10.5120/ijca2018917253

Jiménez Marco, G. (2016). *DevOps, la nueva tendencia en el desarrollo de sistemas TI, un caso práctico en el análisis de incidencias de software*. https://upcommons.upc.edu/handle/2117/85074

Kadaskar, H. R. (2024). UNLEASHING THE POWER OF DEVOPS IN SOFTWARE DEVELOPMENT. *International Journal of Scientific Research in Modern Science and Technology*, *3*(3), 01-07. https://doi.org/10.59828/ijsrmst.v3i3.185

Mara, U. T. (2011). *Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests.*

Melgar, A. S., & Al, E. (2021). DevOps as a culture of interaction and deployment in an insurance company. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(4), Article 4.

Moreyra Reyna, L. C. (2023). *Marco de trabajo DevOps en el proceso de desarrollo de software en una entidad financiera privada, Lima.* [Tesis de grado, Universidad Cesar Vallejo]. Repositorio Institucional UCV. https://repositorio.ucv.edu.pe/handle/20.500.12692/128748

Navarro Ovalle, H. F. (2020). *Modelo de desarrollo e implementación de software bajo el marco de buenas prácticas del CMMI en el área de sistemas de información de la Universidad Autónoma de Bucaramanga*.

Olmedo, P. B., Noel, F., & Moyano, P. (2018). *El rol de Docker para ejecutar pruebas automatizadas como parte de la Integración Continua*.

Pando, B., & Dávila, A. (2023). Una revisión sistemática de la literatura sobre pruebas de software en el contexto de DevOps. *Proceedings of the Institute for System Programming of the RAS*, *35*(1), 163-188. https://doi.org/10.15514/ISPRAS-2023-35(1)-11

Peralta, J. (2019). *DevOps en la entrega continua de la oficina general de estadística y tecnología de la información y comunicaciones del Ministerio de Cultura,* Lima. [Tesis de grado, Universidad Cesar Vallejo]. Repositorio Institucional UCV. https://repositorio.ucv.edu.pe/handle/20.500.12692/39295

Port, D., Taber, B., & Emkani, P. (2024). Investigating effectiveness and compliance to DevOps policies and practices for managing productivity and quality variability. *Journal of Systems and Software*, *213*, 112030. https://doi.org/10.1016/j.jss.2024.112030

Redondo, A. M. F., & Cárdenas, F. de J. N. (2022). DevOps: Un vistazo rápido. *Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla*, *10*(19), Article 19. https://doi.org/10.29057/esh.v10i19.8121

Rüegger, J., Kropp, M., Graf, S., & Anslow, C. (2024). Fully Automated DORA Metrics Measurement for Continuous Improvement. *Proceedings of the 2024 International Conference on Software and Systems Processes*, 36-45. https://doi.org/10.1145/3666015.3666020

Ruiz, J. M. S., Mayo, F. J. D., Oriol, X., Crespo, J. F., Benavides, D., & Teniente, E. (2023). *A Benchmarking Proposal for DevOps Practices on Open Source Software Projects* (Versión 1). arXiv. https://doi.org/10.48550/ARXIV.2304.14790

Sentrio. (2021, abril 20). Cómo medir el éxito DevOps a través de las Four Key Metrics. *Sentrio*. https://sentrio.io/blog/four-key-metrics/

Trigo, A., Varajão, J., & Sousa, L. (2022). DevOps adoption: Insights from a large European Telco. *Cogent Engineering*, *9*(1), 2083474. https://doi.org/10.1080/23311916.2022.2083474

Vidal Béjar, E. R. (2020). *Propuesta DevOps en el proceso de desarrollo de sistemas de información en la empresa financiera Compartamos, Miraflores*. [Tesis de maestría, Universidad Cesar Vallejo]. Repositorio Institucional UCV. https://repositorio.ucv.edu.pe/handle/20.500.12692/49467

Vigo Salinas, J. M., Rodríguez Toral, A. J., & Vigo Salinas, J. M. (2020). *Software multiplataforma de centro de costos para mypes utilizando DevOps*. [Tesis de pregrado, Universidad Privada Antenor Orrego]. Repositorio Institucional UPAO. https://repositorio.upao.edu.pe/item/7ac36ca7-9e89-4b51-8e3c-ca04cfa9f85c

Willis, J. (2012). *The Convergence of DevOps.* IT Revolution. https://itrevolution.com/the-convergence-of-devops/

IX. ANEXOS

Anexo 1 Matriz de consistencia

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLES	INDICADORES	METODOLOGIA
GENERAL ¿De qué manera la implementación de una metodología Devops influye positivamente en el proceso de despliegue de software en una Universidad Pública?	GENERAL Implementar una metodología Devops para mejorar el proceso de despliegue de software en una Universidad Pública.	GENERAL Si se implementa una metodología Devops entonces influye positivamente en el proceso de despliegue de Software en una Universidad Pública.	Metodología de Devops	Presencia - ausencia	**TIPO DE ESTUDIO** El tipo de estudio a realizar es aplicada. **DISEÑO DE ESTUDIO** Pre-Experimental **POBLACIÓN** Indeterminad **TAMAÑO DE MUESTRA** 16 semanas **TÉCNICAS E INSTRUMENTOS** Técnica: La técnica a utilizar será la ficha de recojo de información
ESPECIFICO ¿De qué manera la implementación de una metodología Devops influye positivamente en la frecuencia de despliegue en una Universidad Pública? ¿De qué manera la implementación de una metodología Devops influye positivamente en el tiempo de entrega de cambios en una Universidad Pública? ¿De qué manera la implementación de una metodología Devops influye positivamente en el tiempo de Recuperación ante fallos en una Universidad Pública? ¿De qué manera la implementación de una metodología Devops influye positivamente en la tasa de fallos en despliegues en una Universidad Pública?	ESPECIFICOS Implementar una metodología Devops para mejorar la frecuencia de despliegue en una Universidad Pública. Implementar una metodología Devops para mejorar el tiempo de entrega de cambios en una Universidad Pública. Implementar una metodología Devops para mejorar el tiempo de recuperación ante fallos en una Universidad Pública. Implementar una metodología Devops para mejorar la tasa de fallos en despliegues en una Universidad Pública.	ESPECIFICOS Si se implementa una metodología Devops entonces influye positivamente en la frecuencia de despliegue en una Universidad Pública. Si se implementa una metodología Devops entonces influye positivamente en el tiempo de entrega de cambios en una Universidad Pública. Si se implementa una metodología Devops entonces influye positivamente en el tiempo de recuperación ante fallos en una Universidad Pública. Si se implementa una metodología Devops entonces influye positivamente en la tasa de fallos en despliegues en una Universidad Pública.	Proceso de Despliegue de Software	Frecuencia de Despliegue Tiempo de Entrega de Cambios Tiempo de Recuperación ante Fallos Tasa de Fallos en Despliegues	

Anexo 3 Instrumento (Ficha de recojo de información)

	I1: Frecuencia de despliegue		I2: Tiempo de Entrega de Cambios		I3: Tiempo de Recuperación ante Fallos		I4: Tasa de Fallos en Despliegues	
Item	**Pre-Prueba**	**Post-Prueba**	**Pre-Prueba**	**Post-Prueba**	**Pre-Prueba**	**Post-Prueba**	**Pre-Prueba**	**Post-Prueba**
1.								
2.								
3.								
4.								
5.								
6.								
7.								
8.								
9.								
10.								
11.								
12.								
13.								
14.								
15.								
16.								

Anexo 4 Base de datos de indicadores

	I1: Frecuencia de despliegue (semana)		I1: Tiempo de Entrega de Cambios (minutos)		I2: Tiempo de Recuperación ante Fallos (minutos)		I3: Tasa de Fallos en Despliegues	
Item	**Pre-Prueba**	**Post-Prueba**	**Pre-Prueba**	**Post-Prueba**	**Pre-Prueba**	**Post-Prueba**	**Pre-Prueba**	**Post-Prueba**
1.	2.00	12.00	10.00	448.80	95.28	22.47	0.00	7.14
2.	3.00	14.00	11.00	471.14	68.00	13.97	50.00	0.00
3.	4.00	14.00	10.00	468.93	79.57	16.57	25.00	0.00
4.	2.00	10.00	8.00	466.15	104.82	10.73	50.00	0.00
5.	3.00	10.00	7.00	414.14	97.35	2.24	50.00	0.00
6.	1.00	9.00	8.00	462.16	40.45	18.27	50.00	0.00
7.	3.00	5.00	2.00	350.32	79.00	19.32	50.00	7.14
8.	4.00	5.00	1.00	321.96	56.97	11.29	33.33	0.00
9.	3.00	10.00	7.00	354.73	57.94	26.35	33.33	8.33
10.	1.00	12.00	11.00	528.87	68.21	7.73	33.33	0.00
11.	4.00	15.00	11.00	321.15	62.88	15.23	33.33	10.00
12.	2.00	13.00	11.00	524.77	89.09	14.06	33.33	0.00
13.	2.00	12.00	10.00	369.26	75.22	22.66	0.00	7.14
14.	2.00	6.00	4.00	590.21	62.43	22.35	25.00	11.11
15.	2.00	12.00	10.00	349.66	68.88	15.77	33.33	7.69
16.	2.00	15.00	13.00	451.17	95.28	22.47	50.00	0.00

Anexo 5 Instalación de Jenkins

1. Preparativos Iniciales

Requisitos del Sistema

- **Sistema Operativo**: Ubuntu 24.04 LTS (o equivalente).
- **Memoria RAM**: Al menos 4 GB (se recomiendan 8 GB o más para un mejor rendimiento).
- **Espacio en Disco**: Al menos 10 GB de espacio libre (más si se esperan grandes repositorios o numerosos usuarios).
- **CPU**: Al menos 2 núcleos.
- **Acceso a Internet**: Necesario para descargar GitLab y sus dependencias.

Acceso al Servidor Nutanix

- **Creación de VM en Nutanix**:
    - Utiliza Nutanix Prism para crear una nueva máquina virtual (VM) con los recursos necesarios.
    - Configura la VM con Ubuntu 24.04 LTS como sistema operativo.
    - Asigna suficiente CPU, memoria y almacenamiento según los requisitos de GitLab.

2. Configuración del Entorno

Actualización del Sistema

- Accede a la VM mediante SSH:

  bash

  Copiar código

  ssh usuario@ip_del_servidor

- Actualiza el sistema operativo para asegurar que todos los paquetes estén al día:

bash

Copiar código

sudo apt update && sudo apt upgrade -y

3. Instalación de GitLab

Instalación de GitLab usando el paquete Omnibus

GitLab proporciona un paquete Omnibus que simplifica la instalación al incluir todos los componentes necesarios.

1. **Agregar el repositorio de GitLab y el paquete**:
    - Instala los paquetes necesarios para agregar un repositorio externo:

bash

Copiar código

sudo apt install -y curl openssh-server ca-certificates tzdata

    - Agrega el repositorio oficial de GitLab:

bash

Copiar código

curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.deb.sh | sudo bash

2. **Instalar GitLab**:
    - Instala GitLab utilizando el gestor de paquetes:

bash

Copiar código

sudo apt install gitlab-ee -y

4. Configuración Inicial de GitLab

Configuración de la URL Externa

- Configura la URL externa para GitLab, editando el archivo /etc/gitlab/gitlab.rb:

bash

Copiar código

sudo nano /etc/gitlab/gitlab.rb

- Encuentra y edita la línea:

ruby

Copiar código

external_url 'http://gitlab.example.com'

Reemplaza http://gitlab.example.com con el dominio o la dirección IP del servidor Nutanix.

Reconfigurar GitLab

- Aplica la configuración y asegúrate de que todos los servicios se inicien correctamente:

bash

Copiar código

sudo gitlab-ctl reconfigure

5. Configuración de Firewall y Acceso

Configuración de Firewall

- Asegúrate de que los puertos HTTP (80) y HTTPS (443) estén abiertos para permitir el acceso a GitLab:

bash

Copiar código

sudo ufw allow http

sudo ufw allow https

sudo ufw enable

Acceso Inicial

- Accede a GitLab mediante un navegador web:

arduino

Copiar código

http://ip_del_servidor (o el dominio configurado)

- Usa la contraseña inicial almacenada en:

bash

Copiar código

sudo cat /etc/gitlab/initial_root_password

6. Configuración Adicional de SSL (Opcional)

Configuración de SSL con Let's Encrypt

- Para asegurar el tráfico con HTTPS, puedes utilizar Let's Encrypt:
    - Configura Let's Encrypt en /etc/gitlab/gitlab.rb:

ruby

Copiar código

external_url 'https://gitlab.example.com'

letsencrypt['enable'] = true

letsencrypt['contact_emails'] = ['your_email@example.com'] # Cambia esto por tu correo

letsencrypt['auto_renew'] = true

    - Aplica los cambios:

bash

Copiar código

sudo gitlab-ctl reconfigure

7. Mantenimiento y Gestión de GitLab

Monitoreo y Backups

- Configura monitoreo y backups regulares para mantener la integridad y disponibilidad de los datos.
- GitLab incluye Prometheus y Grafana para monitoreo, que se pueden configurar en /etc/gitlab/gitlab.rb.

Actualizaciones de GitLab

- Para actualizar GitLab, utiliza el siguiente comando:

bash

Copiar código

sudo apt update && sudo apt upgrade -y

8. Optimización y Escalabilidad

Optimización del Rendimiento

- Asegúrate de que la VM tenga suficientes recursos asignados y ajusta la configuración de GitLab según sea necesario.
- Configura GitLab CI/CD runners para distribuir las cargas de trabajo de compilación y pruebas.

Anexo 6. Desarrollo del modelo GODEVOPS

1. Fase 1: Planificación y Definición de Requisitos

Requisitos Funcionales

2. Gestión de Estudiantes:
    - Registro y mantenimiento de perfiles de estudiantes.
    - Seguimiento de inscripciones y asistencia.
3. Gestión de Cursos y Asignaturas:
    - Creación y administración de cursos, incluyendo horarios y detalles del curso.
    - Asignación de profesores a cursos.
4. Evaluación y Calificaciones:
    - Entrada de calificaciones por parte de los profesores.
    - Generación de informes de rendimiento académico.
5. Comunicación:
    - Módulo de mensajería interna entre estudiantes y profesores.
    - Notificaciones de eventos y actividades.

Requisitos No Funcionales Simulados

1. Escalabilidad:
    - Soporte para al menos 1000 usuarios concurrentes sin degradación del rendimiento.
2. Seguridad:
    - Protección de datos personales conforme a la normativa de protección de datos.
    - Implementación de autenticación de dos factores para administradores.
3. Rendimiento:
    - Respuesta del servidor menor a 1 segundo para operaciones comunes.
4. Disponibilidad:
    - Tiempo de actividad del 99.9%.

Selección de Tecnologías

- Framework: Laravel para el backend.
- Control de Versiones: GitLab.
- CI/CD: Jenkins.
- Contenerización: Docker.
- Servidor Web: Nginx.

2. Fase2: Desarrollo

Configuración de Visual Studio Code:

- Instalación de extensiones para Laravel, PHP, Docker y Git.
- Configuración de linters y herramientas de análisis estático de código.

Branching Strategy en GitLab:

- Uso de GitFlow con ramas para desarrollo, características (features), y lanzamientos.
- Uso de merge requests para asegurar revisiones de código antes de la integración.

Normas de Codificación:

- Implementación de normas de codificación para asegurar la consistencia y calidad del código.
- Revisiones de código realizadas por pares para asegurar adherencia a las normas.

3. Fase3: Integración Continua (CI)

Pipeline en Jenkins:

- Extracción de Código: Jenkins extrae el código de la rama principal en GitLab.
- Instalación de Dependencias: Uso de Composer para instalar dependencias de Laravel.
- Pruebas Automatizadas: Ejecución de pruebas unitarias y de integración con PHPUnit.
- Análisis de Calidad: Integración con PHPStan para análisis estático de código.

Notificaciones:

- Configuración de Jenkins para enviar notificaciones por correo electrónico sobre el estado de las compilaciones y pruebas.

Construcción y Almacenamiento

4. Fase 4: Construcción de Imágenes Docker

Creación del Dockerfile:

- Configuración del Dockerfile con la imagen base de PHP, instalación de Laravel, configuración de Nginx, y configuración de permisos.

Construcción de la Imagen Docker:

- Jenkins construye la imagen Docker utilizando el Dockerfile y la etiqueta con un número de versión basado en la fecha y el commit.

Registro en Docker Hub o Nexus Repository:

- Subida de la imagen Docker al registro, asegurando la disponibilidad para el despliegue en otros entornos.

5. **Fase 5: Entrega continua**

Despliegue en Entorno de Prueba:

- Implementación de la imagen Docker en un entorno de prueba para realizar pruebas funcionales y de aceptación.

Despliegue en Entorno de Pre-Producción:

- Despliegue en un entorno que simula las condiciones de producción para pruebas de carga y rendimiento.

Despliegue en Producción:

- Uso de Nginx como servidor web para manejar las solicitudes HTTP. Implementación de estrategias de despliegue como Blue-Green Deployment para minimizar el tiempo de inactividad y los riesgos.

Validación Post-Despliegue:

- Verificación de la integridad de la aplicación y monitoreo de las métricas clave de rendimiento.

6. Fase 6: Monitoreo y Retroalimentación

Configuración de Herramientas de Monitoreo:

- Uso de Prometheus y Grafana para monitorizar el rendimiento del sistema, incluyendo tiempos de respuesta, uso de recursos y errores.

Alertas y Notificaciones:

- Configuración de alertas para incidentes críticos como caídas de servicio o fallas de seguridad.

Revisión de Incidentes:

- Análisis de incidentes para identificar causas raíz y aplicar medidas preventivas.

7. Fase 7: Mantenimiento y Mejora Continua

Actualización de Dependencias:

- Mantener las dependencias del proyecto actualizadas para mejorar la seguridad y el rendimiento.

Optimización de Procesos:

- Revisión continua de los procesos de CI/CD para mejorar la eficiencia del despliegue.

Feedback y Mejoras:

- Recopilar feedback de los usuarios y del equipo para implementar mejoras en el sistema y los procesos.

Figura 9

Modelo de despliegue GoDevops