FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Implementación del modelo CALMS de DevOps para la optimización del proceso de desarrollo de software: caso de estudio empresa Ayarcode

PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO DE SISTEMAS

Autores

Bach. Albarran Salazar Harvy Jefferson
https://orcid.org/0000-0002-5369-3205

Bach. Delgado Farro Javier Manuel
https://orcid.org/0000-0003-0241-973X

Asesor

Mag. Juan Carlos Arcila Diaz
https://orcid.org/0000-0002-7788-951X

Línea de Investigación

Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú

2024

IMPLEMENTACIÓN DEL MODELO CALMS DE DEVOPS PARA LA OPTIMIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE: CASO DE ESTUDIO EMPRESA AYARCODE

Aprobación del jurado

Mg. Bances Saavedra David Enrique

Presidente del Jurado de Tesis

Mg. Asenjo Carranza Enrique David

Secretario del Jurado de Tesis

Mg. Guevara Alburqueque Laurita Belen

Vocal del Jurado de Tesis

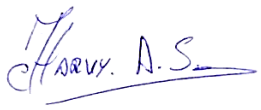
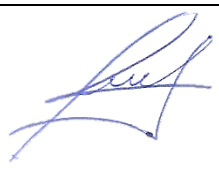
DECLARACIÓN JURADA DE ORIGINALIDAD

Quienes suscriben la DECLARACIÓN JURADA, somos **Harvy Jefferson Albarran Salazar, Javier Manuel Delgado Farro,** del Programa de Estudios de **Ingeniería de Sistemas** de la Universidad Señor de Sipán S.A.C, declaramos bajo juramento que somos autores del trabajo titulado:

IMPLEMENTACIÓN DEL MODELO CALMS DE DEVOPS PARA LA OPTIMIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE: CASO DE ESTUDIO EMPRESA AYARCODE

El texto de mi trabajo de investigación responde y respeta lo indicado en el Código de Ética del Comité Institucional de Ética en Investigación de la Universidad Señor de Sipán, conforme a los principios y lineamientos detallados en dicho documento, en relación con las citas y referencias bibliográficas, respetando el derecho de propiedad intelectual, por lo cual informo que la investigación cumple con ser inédito, original y autentico.

En virtud de lo antes mencionado, firman:

Harvy Jefferson Albarran Salazar	DNI: 72121734	
Delgado Farro Javier Manuel	DNI: 70922936	

Pimentel, 18 de junio de 2023.

Dedicatoria

Dedico esta investigación, fruto de esfuerzo y dedicación, principalmente a Dios, quien me ha guiado en mi camino académico, otorgándome la sabiduría necesaria para llevar a cabo mi formación profesional. A mis padres Jaime y Lizbeth, a mis hermanos Anhely y Josias, mis abuelos Jesús y Amelia, y mi tío Lenin que han sido el pilar fundamental en mi vida y gracias a ellos me he podido formar profesionalmente aportando cada uno de una u otra manera, a Patric que estuvo para alentarme y darme esas fuerzas para seguir adelante; les agradezco a todos ustedes, por su amor incondicional, su apoyo constante, su sacrificio inquebrantable y cada uno de sus consejos; mi gratitud eterna y mi amor más profundo siempre hacia ustedes.

Harvy Jefferson Albarran Salazar

La presente investigación está dedicada a mi amada familia, en especial a mis padres Manuel y María, gracias a quienes he podido dar cada paso en la vida. Agradezco profundamente su apoyo constante y por siempre creer en mí, permitiéndome alcanzar mis metas profesionales. También quiero expresar mi gratitud a Janella, Mauricio y Marisol por sus valiosos consejos, paciencia, apoyo incondicional, aliento y fortaleza para seguir adelante, pues cada uno de ustedes ha sido una fuente de inspiración y motivación, permitiéndome crecer día a día. Mi cariño y agradecimiento es para ustedes.

Javier Manuel Delgado Farro

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que contribuyeron de manera significativa en la realización de esta tesis. Este trabajo representa un esfuerzo colectivo que no habría sido posible sin su invaluable apoyo y orientación.

En primer lugar, agradecemos a Dios por brindarnos la sabiduría y las fuerzas necesarias para perseverar a lo largo de este desafiante proceso. También queremos extender nuestro agradecimiento a nuestras familias, cuyo amor incondicional, paciencia y comprensión nos han sustentado en cada etapa de esta travesía académica. Su apoyo constante ha sido fundamental para nuestro éxito.

Asimismo, deseamos expresar nuestro reconocimiento a la empresa Ayarcode Consulting S.A.C, donde tuvimos la oportunidad de adquirir conocimientos a nivel profesional y llevar a cabo la investigación que dio origen a esta tesis. Su colaboración y disposición fueron indispensables para nuestro crecimiento y desarrollo profesional.

Agradecemos sinceramente a cada uno de nuestros distinguidos docentes, mentores y amigos cuya dedicación y orientación fueron fundamentales en la elaboración de este trabajo. En particular, queremos destacar la invaluable contribución de Evert Sánchez y Alejandro Llontop, cuyo compromiso y guía fueron de suma importancia en nuestro camino hacia la culminación de esta investigación.

Finalmente, esta tesis es el resultado de un esfuerzo conjunto en el que muchas personas han dejado una huella imborrable en nuestras vidas. A todos y cada uno de ustedes, nuestro más profundo agradecimiento por su inestimable contribución y por formar parte de este importante logro en nuestro camino académico.

Los autores.

Índice

Índice de tablas

Índice de figuras

Resumen

En la actualidad, el uso del software es esencial en las organizaciones y en nuestras actividades diarias, evidenciando la necesidad de contar con productos y servicios tecnológicos confiables, útiles y seguros en todas las etapas de las operaciones comerciales. Las empresas buscan métodos más eficientes para mejorar la efectividad en la creación de software, centrándose en mantener la excelencia del software y mejorar los procesos internos mediante una gestión más eficaz de los recursos a su disposición. Este desafío de mejorar el proceso de desarrollo de software y cerrar brechas entre las áreas de operaciones y desarrollo se ha convertido en un obstáculo significativo en la industria, donde el 47% de los directores ejecutivos enfrentan la presión de transformación digital. En este entorno, el propósito de este estudio fue implementar el modelo CALMS de DevOps en el ciclo de desarrollo de software de una empresa de pequeña escala; estandarizando, automatizando y estructurando de manera iterativa cada etapa del proceso. La investigación comprendió cuatro fases: Establecer la línea base de desempeño, seleccionar herramientas y metodología, construir, adaptar e implementar el modelo CALMS de DevOps, y evaluar los resultados. Los resultados demuestran una disminución en la tasa de falla de cambios, pasando del 55.56% al 11.11%. Además, la frecuencia de liberación de código y la implementación también experimentaron mejoras significativas, con aumentos del 28.32% y 25.47%, respectivamente. La percepción de los colaboradores mejoró en un 67.62%. En conclusión, su implementación exitosa generó beneficios tangibles para la empresa considerada como caso de estudio.

Palabras Clave: DevOps, optimización, desarrollo de software, Metodologías de desarrollo, Enfoque ágil.

Abstract

Today, the use of software is essential in organizations and in our daily activities, evidencing the need for reliable, useful and secure technological products and services in all stages of business operations. Companies are looking for more efficient methods to improve the effectiveness of software creation, focusing on maintaining software excellence and improving internal processes through more effective management of the resources at their disposal. This challenge of improving the software development process and closing gaps between operations and development has become a significant hurdle in the industry, with 47% of CEOs facing digital transformation pressure. In this environment, the purpose of this study was to implement the CALMS model of DevOps in the software development cycle of a small-scale enterprise; standardizing, automating and iteratively structuring each stage of the process. The research comprised four phases: Establishing the performance baseline, selecting tools and methodology, building, adapting and implementing the DevOps CALMS model, and evaluating the results. The results show a decrease in the change failure rate from 55.56% to 11.11%. In addition, code release frequency and deployment also experienced significant improvements, with increases of 28.32% and 25.47%, respectively. Employee perception improved by 67.62%. In conclusion, its successful implementation generated tangible benefits for the case study company.

Keywords: DevOps, optimization, software development, development methodologies, agile approach.

I. INTRODUCCIÓN

1.1. Realidad problemática

Al día de hoy, la mayoría de las empresas, en diferente proporción, hacen uso de herramientas tecnológicas para el funcionamiento de sus operaciones comerciales; por tal motivo, el uso de software ha pasado a ser una actividad predominante en el ámbito empresarial y en la vida cotidiana, demostrando así la necesidad e importancia de tener productos y servicios dependientes de esta herramienta tecnológica que brinde fiabilidad, utilidad y seguridad en todo momento durante la ejecución de las actividades u operaciones [1]. La ingeniería de software continúa transformando y acelerando organizaciones de todo tipo, donde las prácticas y capacidades a favor de las mejoras en las industrias es un tema necesario e indispensable; en el Perú, según la segmentación empresarial realizada por el INEI, el 27.6% de las entidades grandes, el 16% de las entidades medianas y el 9% de las entidades pequeñas desarrollan su propio software; y de todas las empresas peruanas, el 67% cuenta con sistema de gestión logrando mejorar los procesos de su organización [2].

Las empresas y organizaciones están cada vez más dependientes de la industria del software para mejorar sus procesos comerciales [3]. Esto ha generado la necesidad de tener métodos más eficientes para realizar las actividades de desarrollo de software, no solo para mantener la calidad del software, sino también para incrementar la calidad de sus procesos internos [4]. La industria del software siempre está buscando maneras efectivas y flexibles de desarrollar software de calidad dentro de tiempos y presupuestos limitados [5]. Esta necesidad de proporcionar continuamente nuevas funciones de aplicaciones que sean innovadoras y fáciles de usar, y que a su vez ofrezcan la calidad correspondiente, ha convertido la optimización en un factor relevante en el desarrollo de software, considerando que, con un tiempo razonable y una organización adecuada, cualquier desarrollo de software tiene una mayor probabilidad de brindar aplicaciones y servicios de alta calidad [1].

En el informe de Forrester [6], se dio a conocer que el 47% de los directores ejecutivos estadounidenses se enfrentan a la presión de su directorio para transformarse digitalmente

dentro de las organizaciones, donde a menudo sobreestiman el progreso del despliegue ágil. Además, se señala que los directivos son especialmente propensos a pasar por alto el progreso de la digitalización, en comparación con aquellos que realmente están llevando a cabo las actividades comerciales [7].

El proceso de desarrollo de software se conceptualiza como la suma de actividades informáticas necesarias con el fin de generar un software de calidad. Los métodos empleados cumplen un rol importante en el ciclo de desarrollo. En un enfoque tradicional como el modelo Waterfall (metodología de cascada o secuencial), la comunicación entre los clientes o partes interesadas y la empresa de software tiende a ser deficiente, lo que resulta en ciclos más prolongados donde los tiempos de desarrollo suelen ser más largos [3]. Los desafíos más comunes asociados con este enfoque incluyen la actividad de planificar los proyectos, la estimación, gestionar cambios y garantizar la calidad [8]. En los distintos procesos de desarrollo de software, se pueden identificar actividades genéricas comunes como la definición, la construcción, la verificación y la evolución del software. Durante este proceso, se produce una interacción entre los usuarios y los desarrolladores de software, los usuarios y la tecnología, y los ingenieros de sistemas y las tecnologías de la información. En esencia, el proceso de desarrollo de software es de constante aprendizaje interactivo en el cual se recopila, transforma y organiza el conocimiento a medida que se implementa el software. A lo largo de este proceso, se incorpora y utiliza el conocimiento adquirido para mejorar y optimizar el desarrollo de software [9].

Para resolver los problemas de comunicación entre áreas y acelerar el desarrollo, nació un nuevo método llamado Agile, que significa ‘moverse rápido y fácilmente’, el cual mejora la interacción entre los clientes o partes interesadas y los desarrolladores. Algunos de los enfoques que implementan el método Agile incluyen Scrum, Extreme Programming (XP), Lean y Kanban [10]. Las metodologías ágiles son implementadas ampliamente para aumentar la eficiencia de proyectos y también para satisfacer los requisitos competitivos del cliente. Por lo tanto, los equipos de desarrollo están adoptando cada vez más el enfoque ágil, que permite

abordar los problemas y desafíos del modelo tradicional, además de brindar múltiples oportunidades a los equipos del proyecto durante la construcción del software, fomentando una transición de comunicaciones formales a comunicaciones frecuentes [8]. Por ejemplo, con Scrum, la brecha de comunicación se mitiga mediante la realización de reuniones diarias, planificación estratégica y revisión constante de los avances individuales y globales [11].

A pesar de que el método Agile permite reducir el tiempo de un ciclo de desarrollo de software, aún existe una serie de inconvenientes entre los integrantes del equipo que influye negativamente en dicho proceso, por lo cual su abordaje es necesario e indispensable [12]. Estos inconvenientes se deben en parte a las diferentes prioridades de los equipos de desarrollo y operaciones. Mientras que el equipo de desarrollo se enfoca principalmente en producir nuevas características y asegurar que los clientes o las partes interesadas puedan utilizar el software lo antes posible, el equipo de operaciones ve el problema desde una perspectiva distinta. Ellos se concentran en mantener un software confiable y libre de errores, lo que a veces conduce a una mala organización del trabajo en equipo [13].

Tras el desarrollo del software Agile se obtuvo una nueva manera de realizar las mediciones de productividad, en muchas escuelas de Agile, las actividades se subdividen en historias, luego, los programadores realizan estimaciones de las historias asignándoles una cantidad de "puntos", que declara el esfuerzo relativo esperado para completarlas. Al final de una iteración, se realiza el registro del total de puntos aprobados por el cliente: esta es la velocidad del equipo. Velocity está hecho para ser usado como un instrumento de organización de capacidad; por ejemplo, se puede utilizar para extrapolar cuánto tiempo le llevará al grupo realizar el trabajo planificado y estimado. Sin embargo, algunos gerentes también lo han usado como una forma para medir la productividad del equipo o incluso para compararlos [14]. Ante estas problemáticas, se introdujo el enfoque DevOps como una extensión del método Agile, para cambiar la manera en cómo se desarrolla el software y, por lo tanto, afectar directamente la forma en que se gestionan los cambios, desarrollando aplicaciones en plazos cortos (Sprints) e involucrando de manera diferente a colaboradores,

reuniendo al equipo de desarrollo y al de operaciones de manera compacta, intentado eliminar brechas entre estas áreas [8]. La automatización es un pilar de DevOps donde se busca que el proceso de integración y evaluación de código e implementación en el método Agile se pueda realizar automáticamente y de la mejor manera posible [3].

Se logró identificar que las organizaciones de software enfrentan varias barreras al adoptar prácticas de DevOps debido a que el trabajo en conjunto entre los grupos de desarrollo y operaciones requieren la fusión de diferentes procesos, herramientas y conjuntos de habilidades. Para abordar este desafío, se llevó a cabo una exploración sistemática para delimitar el conjunto de las prácticas más efectivas de DevOps, en la cual se realizó un estudio de encuesta a los profesionales de la industria sobre la mejor práctica identificada y se aplicó la técnica fuzzy-AHP (metodología utilizada para decidir con multicriterio en circunstancias de incertidumbre en la información disponible) para priorizar las mejores prácticas con respecto a la importancia para el proceso DevOps; además se identificó un total de 48 mejores prácticas, las cuales, están en línea con las prácticas del mundo real, en los que destacan: las organizaciones inician prácticas de desarrollo con proyectos pequeños, incluir modelado para infraestructura y aplicaciones heredadas en sus planes devops y considerar los cambios en la arquitectura de las aplicaciones basados en las instalaciones, la nube y los contenedores desde las primeras etapas del proceso [5].

Además, se realizó un mapeo de las mejores prácticas en diferentes categorías del marco CALMS (cultura, automatización, medición e intercambio). Esto conduce a la conclusión de que las mejores prácticas identificadas, su categorización y el marco basado en AHP difuso, en conjunto, ayudarán a los expertos de la industria a revisar y mejorar sus estrategias para hacer el proceso DevOps más eficiente; además, se presentó una explicación detallada sobre DevOps y se exploraron los desafíos a nivel cultural que las empresas enfrentan al implementarlo. Mediante una revisión literaria, se identificaron 10 desafíos relevantes, algunos de ellos son: cultura organizacional, liderazgo y patrocinio ejecutivo, procesos y prácticas de desarrollo y operaciones; los cuales fueron analizados de

manera exhaustiva en base a diversas regiones geográficas para ser considerados al adoptar esta cultura [15]. Asimismo, se dio a conocer e ilustrar cuál es el impacto de Agile y DevOps en las prácticas de administración de proyectos y la estructura del equipo en los proyectos de desarrollo, concluyendo que las implicaciones prácticas indican que los sectores de software pueden implementar metodologías Agile y DevOps para su proceso de construcción de software, lo que puede mejorar sus prácticas de dirección de proyectos y también organizar la estructura del equipo [8]. Además, se presentó un análisis detallado de 5 entornos de desarrollo distintos que implementaron con éxito DevOps, ellos son: desarrollo de software empresarial, desarrollo de aplicaciones web y móviles, desarrollo de infraestructura de TI, operaciones en la nube, desarrollo de productos de software de consumo; alcanzando beneficios esenciales como lanzamientos rápidos y mínimos errores de implementación, dicha información fue recopilada a través de entrevistas con 26 profesionales y estudios realizados en las organizaciones. Cada caso fue analizado individualmente utilizando un conjunto de temas predefinidos y luego se realizó una síntesis cruzada de los casos. Se concluyó que en DevOps es crucial que el equipo de desarrollo de software asuma la propiedad y la responsabilidad de implementar cambios en el software en producción, haciendo uso de herramientas y recibiendo soporte en el proceso de canalización de ejecución para acelerar la entrega de cambios, corregir errores y controlar incidentes de producción. Se encontró que el tiempo de entrega se ve afectado por indicadores contextuales, como las aprobaciones manuales del propietario del producto. Este análisis contribuye a una mejor comprensión del concepto, las prácticas y los impactos percibidos de DevOps, especialmente en pequeñas y medianas empresas [16].

Las grandes empresas con una larga historia y tecnologías con décadas de antigüedad, también obtienen importantes beneficios, como una entrega acelerada y costos más bajos. En comparación con las altas entidades que implementan DevOps (Netflix, Amazon, Google y Facebook) y entidades que demuestran un bajo rendimiento, se encontró que los de alto rendimiento tienen implementaciones de código 46 veces más frecuentes, un

tiempo medio para recuperación del tiempo de inactividad 170 veces más rápido, un tiempo de entrega 440 veces más veloz y una tasa de fallo 5 veces menor (1/5 de posibilidad de que falle un cambio) [17].

En Holanda, se presentó el desarrollo de contratos de los diversos servicios para microservicios de IoT desde la perspectiva de DevOps, con ello, se analizó las partes interesadas para ayudar a programar contratos de este tipo [16]. Mientras que, en Portugal, se exploraron las ventajas que se obtuvieron al optar por usar DevOps y Agile, en los resultados se obtuvieron doce (12) características como lo son la automatización del desarrollo de los procesos y el progreso de los equipos en cuanto a su comunicación [9].

En Indonesia, se utilizaron métodos cuantitativos para analizar los datos de la investigación mediante regresión lineal simple. La técnica del cuestionario se utilizó para recuperar los datos de los encuestados mediante 62 preguntas, que constan de 42 preguntas de calidad del trabajo en equipo de 6 indicadores y 20 preguntas de DevOps para 4 indicadores. Los resultados de las diversas pruebas de calidad dan a conocer que todas las herramientas son válidas y confiables, mientras que las pruebas de hipótesis mostraron que la variable del uso de DevOps influye en la variable calidad del trabajo en equipo en un 75,6%. La puesta en marcha de DevOps en el desarrollo de software tiene correlación positiva con la calidad del trabajo en equipo [3].

La presente investigación propone la adopción de DevOps y el pensamiento ágil con la finalidad de superar las brechas existentes entre el área de Desarrollo de Software (Dev) y el área de Operaciones (Ops) en la empresa caso de estudio seleccionada, en la cual se identificó que la problemática se presenta con mayor énfasis en el transcurso de creación e integración de software, donde el tiempo que se estima para su realización es mucho más amplio, debido a que se trabaja mediante una metodología tradicional, lo cual no sería lo adecuado ni lo más recomendado. La colaboración y comunicación entre estas áreas es fundamental para que el proyecto sea accesible para los clientes bajo los requerimientos establecidos.

La implementación de la cultura de DevOps y sus mejores prácticas, fundamentadas en sus bases de automatización, colaboración, comunicación e integración, se propone con la finalidad de aumentar la efectividad en el proceso de construcción de software, reducir gastos asociados a los entregables y mejorar los procesos para optimizar el tiempo de entrega, seguridad y solidez del software, generando méritos constantes en la organización. Todo ello, mediante la optimización que trae consigo la implementación de DevOps, accediendo a los distintos roles como el desarrollo, ingeniería de calidad, operaciones de TI y seguridad, trabajando en forma conjunta para ofrecer proyectos confiables en tiempos reducidos.

1.2. Formulación del problema

¿De qué manera la implementación del modelo CALMS de DevOps puede optimizar el proceso de desarrollo de software en la empresa AYARCODE?

1.3. Hipótesis

Mediante la implementación del modelo CALMS de DevOps se mejorará el proceso de desarrollo de software en la empresa AYARCODE.

1.4. Objetivos

Objetivo general

Implementar el modelo CALMS de DevOps en el proceso de desarrollo de Software en la empresa AYARCODE.

Objetivos específicos

- Establecer la línea base del desempeño de la empresa AYARCODE en el proceso de desarrollo de software.
- Seleccionar la metodología ágil y las herramientas como parte de las buenas prácticas de DevOps.
- Implementar la metodología ágil y las herramientas como parte de las buenas prácticas de DevOps
- Aplicar los instrumentos para la medición del desempeño de la empresa AYARCODE

en el proceso de desarrollo de software.

1.5. Teorías relacionadas al tema

DevOps:

El término DevOps se originó en la exposición de Patrick Debios y Andrew Clay Shafer, en la conferencia Ágil del 2008 [19], donde Humble y Molesky definieron al término DevOps como la fusión de los términos Development (desarrollo) y Operations (operaciones) para tratar de alinear todo lo vinculado referente a la entrega de software, en especial a los desarrolladores, personal de pruebas y de operaciones; donde tras el debate sostenido en la conferencia, los autores consolidaron que DevOps es una mentalidad que fomenta la aportación interfuncional entre grupos, con mayor énfasis en las áreas de desarrollo y operaciones, con el objetivo de operar sistemas resistentes y así acortar tiempos en la entrega de cambios; donde este enfoque puede ser descrito como un movimiento, filosofía, metodología, conjunto de prácticas óptimas, una mentalidad o cultura, a pesar de sus diversas opiniones en cuanto a si su enfoque se centra principalmente en la cultura o si es un enfoque técnico, lo cual ha generado varias discusiones en la comunidad de desarrolladores [20]; asimismo, se define como un esfuerzo colaborativo y multidisciplinario que busca la automatización en la implementación constante, asegurando su corrección y confiabilidad; a través de la unión de procesos, herramientas y cultura que promuevan la cooperación entre los equipos de desarrollo y operaciones en una organización [13].

DevOps es un marco de desarrollo diseñado para superar la separación tradicional entre las etapas de desarrollo y operaciones. Su principal propósito es fomentar la comunicación, colaboración e integración continua entre estas dos áreas [14]. Al implementar un conjunto de prácticas de desarrollo sólidas, DevOps posibilita la entrega de servicios de software de manera más rápida, puntual y con una calidad superior [16]. Estas buenas prácticas facilitan el desarrollo, prueba e implementación ágil y confiable de software al impulsar el trabajo colaborativo entre desarrolladores, evaluadores y operadores. El objetivo final es reducir el tiempo necesario para la implementación de cambios en un sistema y

transformar esos cambios en un entorno de producción efectivo [21]; en otras palabras, es la erudición que suscita la cooperación entre el personal de desarrollo y el de pruebas para desplegar un software de manera rápida, haciendo uso de un método repetitivo y automatizado [22], donde se elimina aislamientos entre los equipos de desarrollo y operaciones de TI, con la intención de colaborar entre sí y presentar valor a los usuarios de forma permanente [23].

DevOps no se clasifica como un marco o un estándar de referencia, si no que constituye un movimiento destinado a la compilación, proceso de pruebas y publicación de software generado con mayor velocidad y confiabilidad [20]. Ante ello, [24] describe algunas capacidades y habilitadores de DevOps comunes de la implementación de los procedimientos de DevOps en la tabla siguiente:

Tabla I Capacidades Y Habilitadores De Devops

Capacidades	Factores culturales	Facilitadore tecnológicos
- Planificación continua. - Colaboración y despliegue continuo - Publicación y despliegues continuos - Supervisión constante y optimización de la infraestructura. - Rápida recuperación de fallos en el servicio.	- Metas compartidas, definición de éxito y motivadores. - Comunicación fluida. - Experimentación continua y adquisición de conocimiento. - Desarrollo de automatización.	- Automatización de pruebas - Automatización de la monitorización - Automatización de la Infraestructura - Gestión de la configuración del código y la infraestructura.

Fuente: [24]

DevOps ofrece una serie de modelos y métodos que facilitan la mejora de las actividades realizadas por los equipos de desarrollo de software y tecnología de la información (TI). El objetivo es agilizar y asegurar los procesos de compilación, evaluación y publicación del software. Estas prácticas se agrupan en cinco dimensiones: colaboración, automatización, cultura, supervisión y medición [14]. A continuación, se detallan estas dimensiones junto con sus prácticas y patrones correspondientes en la siguiente tabla:

Tabla II Dimensiones, patrones y prácticas de DevOps

Dimensión	Patrones de prácticas	Ejemplos de prácticas
Colaboración	Replanteamiento y reorientación a los equipos encargados de las actividades de desarrollo y operaciones	Aumentar el alcance de las responsabilidades. Intensificar la cooperación y la implicación en el trabajo diario de los demás.
Automatización	Automatización de infraestructura y procesos de implantación	Infraestructura como código Automatización del proceso de implantación
Cultura	Empatía, apoyo y buen ambiente de trabajo entre desarrollo y operaciones	Comunicación efectiva entre áreas Respeto mutuo, apoyo y voluntad de trabajar en equipo Compartir responsabilidades
Supervisión	Instrumentación de la aplicación y agregación de los datos supervisados para obtener información	Retroalimentación continua que va desde el entorno de producción hasta el inicio del ciclo de desarrollo Detección proactiva con la finalidad de prever los posibles problemas.
Medición	Métricas útiles	Uso de métricas para evaluar e incentivar a los equipos

Fuente: [16]

a) Colaboración:

La colaboración entre miembros de un equipo es vista como una forma de empoderamiento, particularmente para los desarrolladores, quienes pueden obtener un mayor control en el funcionamiento del sistema, permitiéndoles ampliar sus habilidades y conocimientos [13], La colaboración es muy importante al compartir información y en el desarrollo de las competencias y aptitudes de los integrantes del equipo. Para implementar DevOps de manera efectiva, es necesario un cambio cultural completo, basado en principios como la confianza, respeto y comunicación [25].

Se refiere a la necesidad de promover e impulsar el trabajo colaborativo entre los diferentes equipos de la organización de Tic, como desarrollo, operaciones, seguridad, pruebas, etc., [26]. Algunas sugerencias para ejecutar esta dimensión son:

i. Promover la cooperación entre los equipos mediante la comunicación y la transparencia para compartir información y conocimientos, trabajando en conjunto

para resolver problemas y alcanzar objetivos comunes.

ii. Implementar herramientas y prácticas que promuevan la cooperación, como reuniones periódicas, tableros de seguimiento, sistemas de gestión de incidencias y chats en línea.

iii. Determinar objetivos y metas compartidas para motivar a los equipos a colaborar y trabajar juntos.

iv. Brindar oportunidades de capacitación y aprendizaje para que los integrantes del equipo puedan adquirir conocimientos y habilidades que les permitan colaborar de manera más efectiva.

v. Fomentar la cultura de aprendizaje continuo y experimentación, donde los equipos prueben nuevas ideas y soluciones y compartan sus resultados con los demás colaboradores involucrados.

b) Automatización

Es esencial mejorar la automatización de las pruebas y los despliegues para mantenerse al día con la tendencia creciente del desarrollo ágil de software. En cuanto a la implementación de herramientas y prácticas para automatizar las actividades vinculadas al desarrollo, prueba, implementación y operación del software, se mencionan varias estrategias y prácticas para abordar esta dimensión [26], entre las cuales se destacan:

i. Identificar los procesos manuales y repetitivos que pueden ser automatizados, para mejorar la eficiencia y reducir errores.

ii. Implementar herramientas de integración y entrega continua para la automatización del proceso de construcción, prueba y distribución de software.

iii. Utilizar herramientas de infraestructura como código para automatizar el proceso de crear y gestionar en entornos de desarrollo, prueba y producción.

iv. Implementar pruebas automatizadas para garantizar la calidad del software y disminuir el tiempo de prueba.

v. Utilizar herramientas de monitorización y registro para obtener una visibilidad completa del rendimiento del software y poder detectar problemas en tiempo real.

Para la fase de implementación existen tres términos vinculados directamente, los cuales son: Integración Continua, Entrega Continua y Despliegue Continuo [27].

Integración continua (CI) Consiste en que los miembros del equipo fusionen regularmente sus copias de seguridad de trabajo en una rama principal compartida [28]. Se practica la validación continua del código a través de pruebas y fusiones de cambio, donde el sistema es capaz de detectar automáticamente los cambios realizados por los desarrolladores, lo que desencadena una serie de procesos que incluyen la compilación del código, pruebas automáticas y la publicación de la compilación en el repositorio [29].

Entrega Continua, es un conjunto de capacidades que permiten generar cambios de todo tipo como: modificaciones de configuración, características, solución de errores y experimentos en producción o en manos de los usuarios de manera segura, rápida, y sostenible. Bajo los principios de la incorporación de calidad, trabajo en pequeños lotes, planificación de tareas repetitivas por computadora, persecución incansable de la mejora y la responsabilidad holística [17]. Para la implementación de mejora continua se debe crear los cimientos de la gestión integral de la configuración, la integración continua y las pruebas continuas. Considerado un procedimiento de ingeniería de software, lo que favorece que múltiples equipos continúen generando software de valor en tiempos breves y repetitivos, garantizando que la liberación del software pueda realizarse de forma confiable en cualquier momento [30].

El Despliegue Continuo (CD), extiende los principios de la integración continua, al automatizar la liberación del software después de pruebas exhaustivas [28]; no se requiere la intervención humana para implementar los cambios una vez que un desarrollador ha comprobado el código y ha pasado todas las pruebas automatizadas, para lo cual se pueden utilizar diversas herramientas de automatización, como Azure DevOps, Jenkins u otras, que permiten conectar, orquestar y activar las aplicaciones

necesarias [27].

El Despliegue Continuo es un componente fundamental de DevOps, donde cada cambio en el código es sometido a un flujo de pruebas automatizado; si el cambio supera exitosamente todas las pruebas, se despliega de forma automática en el entorno de producción [29], este proceso es crucial para optimizar el proceso de entrega de software y a menudo se complementa con la entrega continua [31].

c) Cultura

En DevOps, muchas de las prácticas implican un cambio en la cultura y mentalidad de los colaboradores encargados del proceso de desarrollo y entrega de software. Se fomenta la empatía, el apoyo y la creación de un ambiente laboral saludable para las personas que participan en estos procesos [16].

La presente dimensión implica la creación de un ambiente de colaboración, aprendizaje, confianza y responsabilidad, que permita a los equipos trabajar juntos de manera eficaz y lograr los objetivos de manera sostenible, ante ello sugiere un listado de prácticas para su implantación de manera correcta [32], las cuales se mencionan a continuación:

i. Impulsar el trabajo integrado y en conjunto entre las áreas de desarrollo y operaciones, eliminando las barreras organizacionales y mejorando la comunicación y la transparencia.
ii. Alentar la experimentación y el aprendizaje continuo, permitiendo a cada una de las áreas probar nuevas ideas y soluciones, y aprender de los errores.
iii. Generar un ambiente de apoyo y confianza, donde los colaboradores puedan brindar libremente sus ideas y opiniones, y puedan trabajar juntos para resolver problemas y alcanzar objetivos comunes.
iv. Establecer una visión compartida y un conjunto de valores y principios, para que todos trabajen hacia los mismos objetivos y sepan cómo tomar decisiones y resolver problemas.

v. Generar un contexto en el que se promueva la responsabilidad y la propiedad entre los miembros del equipo, brindándoles la libertad necesaria para decidir y solucionar los problemas de manera autónoma. Al permitirles asumir un compromiso activo con el éxito del proyecto, se estimula su motivación y empoderamiento, lo que a su vez contribuye a un mayor nivel de eficacia y logro de los objetivos establecidos. Alentando la autonomía y la toma de decisiones, se fortalece la capacidad del equipo para adaptarse a los desafíos y encontrar soluciones innovadoras, generando así un ambiente de trabajo más colaborativo y productivo.

d) Supervisión

Para la detección y corrección oportuna de problemas, que es esencial en DevOps, es importante realizar una supervisión constante de los sistemas y la infraestructura por parte del personal de operaciones; el monitoreo continuo también ayuda a asignar y gestionar de manera adecuada los recursos necesarios, todo ello se consigue al utilizar herramientas de monitoreo y realizar registros automatizados [24]. No obstante, puede resultar complicado para los desarrolladores buscar anomalías en la gran cantidad de registros disponibles en los sistemas si no están diseñados para mostrar los errores automáticamente [16].

Para realizar una correcta supervisión, [32], sugiere el siguiente conjunto de prácticas:

i. Llevar a cabo revisiones periódicas de los objetivos y los indicadores clave de rendimiento para medir el avance y la eficacia de la implementación de DevOps.
ii. Realizar pruebas continuas para evaluar la efectividad de la automatización y detectar cualquier posible problema de calidad del software.
iii. Implementar una estrategia de monitoreo constante para supervisar el rendimiento del software y detectar posibles problemas en tiempo real.
iv. Realizar auditorías de seguridad regulares para garantizar que el software esté desarrollándose y operando de acuerdo con los estándares de seguridad

establecidos.

v. Fomentar la colaboración y el flujo de conocimiento entre los equipos de desarrollo y operaciones con el fin de mejorar la eficiencia en la supervisión y la solución de problemas.

e) Medición

En DevOps, se utilizan diferentes métricas para supervisar y evaluar el rendimiento de los procesos en las actividades de desarrollo y operaciones. Sin embargo, en lugar de métricas indirectas, se sugiere hacer uso de métricas comunes que se centren en los objetivos del negocio para evaluar e incentivar a los equipos de desarrollo y operaciones. Los comentarios de producción también pueden utilizarse para impulsar decisiones, mejoras y cambios en el sistema, mediante la recopilación de datos cuantificables sobre cómo está funcionando el sistema, según la tasa de conversión o cualquier otra métrica que la empresa utilice para determinar el éxito [16].

La implementación del enfoque ágil DevOps requiere la configuración de un entorno de calidad que permita la automatización de los escenarios de prueba. En este proceso, es necesario utilizar una serie de distintas herramientas, ya sean Jenkins para la integración continua, Cucumber para BDD, Junit para TDD, GIT para la gestión de la configuración, Quality Center, JIRA y ALM para gestionar pruebas y defectos, así como Selenium, QTP y UFT para la automatización, entre otras [33].

Para evidenciar la correcta implementación de DevOps se consideran principalmente a las métricas de frecuencia de implementación, plazo de entrega de cambios, tiempo para restaurar el servicio, tasa de fallas en los cambios [26].

1. **Frecuencia de implementación:**

   Frecuencia en que las aplicaciones se implementan en producción, por lo general es representada en porcentajes y también es reconocido como rendimiento [32]. Está enfocada en reducir el tamaño de los lotes en un proyecto, lo cual es una parte fundamental del enfoque Lean; dado que medir esto en el software es

complicado, se utiliza la frecuencia con la que se implementa el software en producción como una medida sustituta [34].

Es posible determinar la frecuencia de implementación de un proyecto de desarrollo de software mediante la división del número total de despliegues o de implementaciones exitosas realizadas durante un período específico por la duración de ese período. Cabe señalar que la frecuencia de implementación puede fluctuar dependiendo del tipo de proyecto y del nivel de implementación de DevOps en la organización [35]. Por lo tanto, la fórmula para calcular la frecuencia de implementación es la siguiente:

$$\text{Frecuencia de implementación} = \frac{\text{Número total de despliegues exitosos}}{\text{Cycletime}}$$

Donde:

$$\text{Cycletime} = \text{Tiempo de Desarrollo} + \text{Tiempo de Testing} + \text{Tiempo de Despliegue}$$

Ejemplo:

Si un equipo de desarrollo realiza 40 despliegues exitosos durante un período de tiempo de un mes (30 días), la frecuencia de implementación sería: Frecuencia de implementación = 40/30 = 1.33 despliegues por día.

2. Plazo de entrega de cambios

Tiempo transcurrido desde el compromiso del código hasta la implementación y la producción, representado como una duración [32], es crucial en el proceso de desarrollo de software. Se refiere al lapso de tiempo que transcurre desde el compromiso inicial del código hasta su exitosa implementación en producción [34]. Un lapso breve en este proceso es deseable, ya que permite una retroalimentación y corrección de errores más rápidas, así como una entrega expedita de soluciones a problemas o defectos identificados. Por lo tanto, se considera que un tiempo de implementación más corto es óptimo para lograr una mayor eficiencia en la gestión de proyectos de desarrollo de software. Factores como la complejidad del proyecto y la

disponibilidad de recursos del equipo de desarrollo pueden afectar este tiempo, es importante establecer plazos realistas y acordados por ambas partes para mejorar la eficiencia y evitar retrasos [35].

La fórmula para calcular plazo de entrega de cambios es:

$$\textbf{Plazo de entrega de cambios} = \text{Fecha de entrega acordada} - \text{Fecha real de entrega}$$

Ejemplo:

Si se acuerda entregar un cambio el 20 de mayo y se entrega el 25 de mayo, el plazo de entrega de cambios es de 5 días (25 de mayo - 20 de mayo = 5 días).

3. **Tiempo para restaurar el servicio**

Time to Restore Service o TTRS evalúa el tiempo que un equipo de desarrollo y operaciones necesita para reparar un servicio que ha dejado de funcionar adecuadamente. Se mide desde la detección del problema hasta el momento en que el servicio se restablece [32]. La fórmula para calcular plazo de entrega de cambios es la siguiente:

$$\text{TTRS} = \text{Hora de resolución} - \text{Hora de inicio del incidente}$$

Donde la Hora de inicio del incidente es el momento en que se detecta el problema en el servicio y la Hora de resolución es el momento en que se logra solucionarlo y se restablece el servicio [26].

Ejemplo:

Se presenta un incidente a las 9 de la mañana del día 20 de mayo y se le da solución a las 11 de la mañana del mismo día, el Tiempo para restaurar el servicio es de 2 horas (11:00 am - 9:00 am = 2 horas).

4. **Tasa de fallas en los cambios**

Indicador que muestra el porcentaje de cambios realizados en una aplicación o servicio de software que terminan afectando su desempeño o funcionamiento, y que requieren reparaciones o correcciones posteriores para solucionar problemas como

interrupciones o degradación del servicio; esto puede incluir acciones de reversión, parcheo, corrección o solución de problemas urgentes [34]. Por lo tanto, según [26], la fórmula para calcular la tasa de fallas en los cambios es la siguiente:

$$\text{Tasa de fallas en los cambios} = \frac{\text{Despligues fallidos}}{\text{Número total de despliegues}} x\ 100$$

Ejemplo:

> Si en un mes se implementaron 200 cambios en un sistema y de esos cambios, 10 resultaron en fallos que requirieron reparación, entonces la tasa de fallas en los cambios sería del 5%.

Es relevante resaltar que el principal objetivo de DevOps es incrementar la eficiencia y la calidad del proceso de desarrollo de software, sin priorizar exclusivamente la reducción de costos [36], aunque las organizaciones que adoptan DevOps pueden experimentar una disminución significativa en los costos de infraestructura, así como mejoras en la calidad del software y en la velocidad de entrega [20], es importante recordar que el ahorro de costos no es el objetivo principal de esta metodología [24]. Si bien la implementación de DevOps puede resultar en beneficios económicos, es crucial destacar que estos son una consecuencia de los aspectos fundamentales de la metodología y no su objetivo principal.

Metodologías de desarrollo ágiles:

La metodología de desarrollo ágil es muy popular en el sector de las tecnologías de la Información y de las Comunicaciones, por su adaptabilidad a cambios y priorizar los requisitos de manera recurrente [37]. Se trata de un enfoque general que comprende diferentes técnicas en el desarrollo de software, todas ellas organizadas de manera que el software sea desarrollado por equipos multifuncionales y autoorganizados. Entre las prácticas más comunes de la metodología se encuentran la programación en pares, el desarrollo de pruebas y la planificación de funciones; la estimación es un aspecto crucial en la metodología ágil, ya que su precisión puede aumentar significativamente la probabilidad de éxito del software que se está desarrollando [38].

La metodología Agile es un enfoque para gestionar proyectos, típicamente utilizado

en el desarrollo de software, la cual se ha introducido en respuesta a los inconvenientes de los modelos de procesos rígidos impulsados por planes (como Waterfall, RUP, etc.) y otras metodologías resistentes [33]. Surgen de la necesidad de superar las dificultades y desventajas de aplicar metodologías tradicionales en la gestión e implementación de proyectos [9]. Además, están destinadas a mejorar la destreza de un equipo de desarrollo, para incrementar su capacidad de producir y replicar el cambio, fundamentándose en el desarrollo iterativo [39]. El método Agile mejora la interacción entre los interesados y los desarrolladores, siendo algunos de los enfoques que implementan el método Agile, Scrum, Extreme Programming (XP), Lean y Kanban. El método ágil fomenta una transición de comunicaciones formales a comunicaciones frecuentes. En Scrum, por ejemplo, la brecha de comunicación se mitiga mediante la realización de scrum diarios, planificación de sprints, revisión de sprints [3]. Siendo un aspecto clave de Scrum y Kanban, dos metodologías ágiles populares, la constante expansión y mejora de sus herramientas [40].

Las metodologías ágiles de desarrollo de software más comunes son programación extrema (XP), metodología Crystal, desarrollo basado en funciones (FDD), método Kanban y Scrum [41]; donde Scrum es la metodología ampliamente utilizada como práctica ágil de desarrollo de software; comprende el poder gestionar proyectos como parte de sus prácticas, incluye propietarios de productos y desarrolladores que permitirán obtener una visión amplia del desarrollo; sin embargo, no es adecuado para productos que se centran en la usabilidad porque el propietario del producto se centra en el aspecto comercial. La otra práctica Agile más popular es la Programación Extrema (XP), la cual consta de cuatro fases: Planificación, codificación, diseño y prueba [10] [42]. Estos métodos apuntan a mejorar la agilidad de un equipo de desarrollo de software basándose en el desarrollo iterativo, equipos autoorganizados, artesanía y procesos que son ligeros y maniobrables pero que brindan suficiente coordinación para los comportamientos del proyecto [39].

Para el desarrollo de software existen metodologías como extreme programming, scrum, kanba y otras; las cuales demandan de un trabajo supervisado por todo el equipo,

debido a que los avances obtenidos son a partir de las iteraciones que deben ser contrastadas con la entrega planificada desde un inicio, para lo cual se sugiere el uso del tablero Kanban debido a su fácil implementación y a que permite agilizar los procesos y obtener un sistema de mejor calidad en tiempo reducido, independientemente de la metodología de desarrollo ágil a utilizar [43]. Siendo una de las metodologías más aplicadas es Scrum, considerada como un marco de trabajo de desarrollo, un enfoque de gestión de proyectos y una metodología ágil adaptable, incremental e iterativa en la que se utilizar varios procesos y técnicas [44]. Es el método más popular de la familia ágil, en el enfoque basado en Scrum se despliegan los llamados sprints, que suelen tener un tiempo de duración entre 2 y 4 semanas; su planificación comienza con cada sprint y es utilizado para determinar que entrega se va a realizar a continuación y cómo se va a conseguirlo. Al finalizar cada sprint, se lleva a cabo una retrospectiva del sprint para brindar apoyo al aprendizaje y la mejora continua en el equipo; asimismo ofrece herramientas para rastrear la productividad del equipo midiendo la velocidad de la tarea que se puede trazar en el llamado gráfico de quemado [45].

Para implementar una metodología ágil de forma secuencial, es necesario que la organización establezca un modelo de desarrollo ágil como primer paso. En segundo lugar, es importante considerar la posibilidad de utilizar la computación en la nube para trasladar la infraestructura, el software y la plataforma. En tercer lugar, la organización debe adaptar sus operaciones al flujo de trabajo de canalización de CI/CD. En cuarto lugar, es recomendable automatizar el proceso de implementación de software para reducir las interacciones humanas y mejorar la rapidez en la entrega de productos al mercado. Finalmente, es importante automatizar las pruebas de software para identificar errores y defectos en el código [46]; asimismo que para una correcta implementación metodológica se establezca reuniones diarias, establecer el entorno y alcance del trabajo de desarrollo, considerar retrospectiva donde se comparta conocimientos y experiencias, dar a conocer el incremento de resultados, conformar un equipo multidisciplinario en conocimientos técnicos, establecer ciclos semanales y trimestrales de evaluación, contar con el compromiso y formación requerida de

los desarrolladores y orientar a todo el equipo para conformar un diseño incremental [47]. Ante ello, las principales formas de implementar un desarrollo ágil integrado con usabilidad son vinculadas al enfoque combinado y prácticas, procesos, tecnología, personas y social [47], las cuales se detallan a continuación:

Tabla III Implementación de desarrollo ágil

Forma de Integración	¿Cómo lograr la integración?
Enfoque Combinado	Desarrollo iterativo
	Reunión de equipo multidisciplinario
	Colaboración de todas las partes interesadas
	Combinación de pruebas
	Integración de historias de usuario basado en escenarios
Diferentes enfoques (prácticas, procesos, tecnología, personas y social)	Adopción de prácticas por una o todas las áreas
	Combinación de procesos ágiles
	Combinación entre tecnología utilizada en cada dominio
	Incorporación de miembros con habilidades específicas
	Integración de interacción en los miembros
Prácticas (para hacer frente a los factores que restringen la integración del diseño ágil y centrado en el usuario)	Diseño inicial por parte de los equipos de desarrollo y control de calidad
	Diseño de características funcionales y sus funciones relacionadas
	Asignación de responsabilidades de productos de UX por separado
	Comprensión de la visión del diseño compartida, con los esfuerzos de los profesionales sincronizados
	Preparación de uso de usuarios para evaluación usando técnicas de ingeniería.
	Sesiones de demostración y revisión
	Planificación por adelantado para la inclusión de usuarios y ejecución de pruebas
	Profesionales de UX como clientes ágiles para validación
	Preparación de desarrolladores y distribución de equipos
	Documentación específica de integración

Fuente: [48]

La automatización completa puede convertirse en un obstáculo que dificulta en gran medida la adopción total de DevOps. Por lo tanto, para la propuesta del modelo se considera los siguientes principios de Lean, donde es fundamental centrarse en mantener lo mínimo pero útil.

1. **Eliminar el exceso:** Es importante eliminar la documentación superflua, los

requerimientos sin solicitar textualmente por el cliente, las transferencias de desarrollo entre diferentes grupos y el tiempo excesivo de desarrollo, entre otros aspectos. La clave está en entregar precisamente lo que los clientes desean.

2. **Entregar lo más rápido posible:** Por lo general se prioriza un enfoque sin errores en el desarrollo, donde a su vez permita obtener retroalimentación más rápidamente y ayuda para mejorar el producto; siendo este uno de los componentes más importantes de DevOps.

3. **Empoderar al equipo**: La toma de decisiones deben incluir la participación de las personas encargadas de ejecutar los planes. Al posponer las decisiones, se pierde el tiempo para que las autoridades coordinen las actividades propuestas a sus colaboradores.

Al seguir estos principios para la implementación de DevOps, se pretende evitar la sobre automatización y hacer que el equipo tenga la capacidad de respuesta y flexibilidad necesaria de adaptabilidad a los cambios y entregar un producto de valor a los clientes.

Modelo CALMS de DevOps:

El modelo CALMS de DevOps, que proviene de las iniciales en inglés; las cuales son: cultura, automatización, lean, métricas y compartir, busca transformar el pensamiento de los grupos de desarrollo y operaciones para cumplir con las expectativas empresariales en el ámbito tecnológico. Su objetivo se fundamenta en la optimización y automatización máxima de las actividades que se realizan en el proceso de cración de software, permitiendo una implementación rápida en producción de las funcionalidades requeridas por el negocio [53].

CALMS, por sus siglas comprende los términos de Cultura, Automatización, Lean, Medición, Compartir, los cuales se definen de la siguiente manera [54]:

Cultura: Si se pudiera resumir la cultura de DevOps, en una palabra, sería "colaboración" y, aún mejor, "colaboración transversal". Es fundamental que los profesionales de Desarrollo y TI/Operaciones trabajen juntos de manera efectiva, ya que DevOps no se trata solo de herramientas, sino de solucionar problemas humanos.

Automatización: La inversión en automatización posibilita la eliminación de tareas manuales repetitivas, la creación de procesos reproducibles y el establecimiento de sistemas confiables. La automatización generalmente se inicia con la implementación continua, donde cada modificación en el código se somete a pruebas automatizadas y se despliegan compilaciones exitosas de manera automatizada.

Lean: Dentro de DevOps, el principio de lean se centra en la mejora continua y la aceptación de errores. El objetivo es eliminar actividades de poco valor y avanzar rápidamente, para lo cual es necesario la identificación de las oportunidades de mejora en todos los aspectos, desde realizar revisiones constantes para mejorar los procesos del equipo hasta llevar a cabo pruebas en diferentes mecanismos de integración para nuevos usuarios del software.

Medición: La medición es fundamental para demostrar que los esfuerzos continuos de mejora realmente están generando resultados. Existen diversas herramientas y tecnologías para medir el rendimiento, y se recomienda comenzar por métricas básicas, como el tiempo que lleva pasar del desarrollo al despliegue, la frecuencia de errores o fallos, el tiempo de recuperación después de un fallo del sistema y el número de usuarios del producto.

Compartir: En los equipos que implementan DevOps, es común encontrar una función rotativa donde los desarrolladores se encargan de manejar las incidencias reportadas por los usuarios finales mientras resuelven problemas de producción. Esta estrecha colaboración genera confianza y respeto mutuo, y es fundamental el reconocimiento público de los logros y contribuciones de los colaboradores para mantener la motivación.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación

- **Tipo de Investigación**

El presente estudio se llevó a cabo siguiendo un enfoque cuantitativo, que se fundamenta en el razonamiento deductivo y lógico. Este enfoque parte de una teoría previamente establecida y formula hipótesis que se someten a prueba mediante la recopilación y el análisis de datos numéricos con el fin de abordar el problema planteado [49]. Durante la investigación, se emplearon datos numéricos para ejecutar el proceso de análisis estadístico, centrándose en las variables definidas en la sección de operacionalización.

- **Diseño de la investigación**

En la presente investigación, se empleó un diseño cuasi-experimental con el propósito de explorar las relaciones de causa y efecto entre las variables independientes y dependientes. Este diseño se utiliza para examinar cómo las modificaciones en la variable independiente [50], en este caso, la implementación del modelo CALMS de DevOps, afectan en la variable dependiente, que en este estudio corresponde al proceso de desarrollo de software. En otras palabras, se busca analizar el impacto del conjunto de buenas prácticas DevOps en el proceso de desarrollo de software.

2.2. Variables, Operacionalización

Variable Independiente:

Implementación del modelo CALMS de DevOps

Variable Dependiente:

Optimización del proceso de desarrollo de software

Tabla IV Operacionalización de variables

Variable de estudio	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Ítems	Instrumento	Valores finales	Tipo de variable	Escala de medición
Optimización del proceso de desarrollo de software	Número de despliegues exitosos de software realizados en un periodo de tiempo específico	$\frac{\text{Número total de despliegues exitosos}}{\text{Cycletime}}$	Despliegue e Implementación	Frecuencia de implementación	Implementación y despliegue	Ficha de registro	valores decimales	Cuantitativa	**Bajo:** 0-32% **Medio:** 33-74% **Alto:** 75 - 100%
	Evalúa la frecuencia y velocidad de los despliegues de software realizados durante un sprint	Cantidad de Despliegues por sprint		Frecuencia de liberación de Código	Implementación y despliegue	Ficha de registro	valores numéricos	Cuantitativa	**Bajo:** 0 - 1 **Medio:** 2 - 3 **Alto:** 4 - 5
	Tiempo que transcurre desde la finalización del desarrollo o la corrección hasta su implementación y disponibilidad.	Fecha de acordada − Fecha de entrega	Velocidad de desarrollo	Plazo de entrega de cambios	Eficiencia de entregas	Ficha de registro	valores numéricos	Cuantitativa	**Bajo:** 3 días a más **Medio:** 1-3 días **Alto:** Menos de 1 día
	Tiempo que el equipo necesita para reparar un servicio que ha dejado de funcionar adecuadamente.	Hora de resolución − Hora de inicio del incidente	Monitoreo	Tiempo para restaurar el servicio	Monitoreo	Ficha de registro	valores numéricos	Cuantitativa	**Bajo:** <1semana **Medio:** <1día **Alto:** <1 hora

	Porcentaje de cambios realizados en una aplicación que terminan afectando su desempeño o funcionamiento	$\frac{\text{Despliegues fallidos}}{\text{Número total de despliegues}} x1$		Tasa de fallas en los cambios	Monitoreo	Registro de monitoreo	valores porcentuales	Cuantitativa	**Bajo:** 41>% **Medio:** 21-40% **Alto:** 0-20%
	Evalúa la cantidad de código desplegado exitosamente por cada sprint	$\frac{\text{Despliegues exitosos}}{\text{Número total de despliegues}} x$	Calidad	Ratio de Éxito	Monitoreo	Ficha de registro	valores porcentuales	Cuantitativa	**Bajo:** 0-33% **Medio:** 34-66% **Alto:** 67-100%
Implementación del modelo CALMS de DevOps	Medida del grado de complacencia del colaborador con la orquestación y gestión de los proyectos	A partir de las respuestas, se puede calcular un promedio que representa la satisfacción general del colaborador.	Experiencia del colaborador	Percepción del colaborador	Desarrollo	Encuesta	valores numéricos etiquetados	Cuantitativa	**Escala 0-1:** 1: Si Cumple 0: No Cumple

2.3. Población de estudio, muestra, muestreo y criterios de selección

Población

La población de estudio se definió teniendo en cuenta el conjunto de unidades de análisis que comparten características relevantes y proporcionan datos pertinentes para esta investigación [49]. La unidad de análisis seleccionada para este estudio es el proceso de desarrollo de software en la empresa Ayarcode, ya que su examen es fundamental para responder a la pregunta de investigación que busca mejorar de manera óptima este proceso. En consecuencia, la población de estudio se compone de (04) cuatro proyectos de desarrollo de software.

Muestra

La muestra representa un individuo específico dentro de la población, seleccionado por el investigador debido a las características adecuadas que presenta en beneficio de la investigación [49]. En este caso, se empleó el método no probabilístico de selección por conveniencia para elegir (01) un proyecto de desarrollo de software, considerando factores determinantes como el tamaño y la complejidad, como sugiere [35]. A continuación, se presenta la caracterización de los proyectos.

Tabla V Resumen de la descripción del contexto de los proyectos

Aspecto	Proyecto 1	Proyecto 2	Proyecto 3	Proyecto 4
Nombre del proyecto	MHO	Acciones	Requisitos Legales	School Report Card App Web
Tipo del proyecto	Sistema de gestión documental y reportería	Sistema de control de calidad	Sistema de envío masivo de requisitos legales	Sistema académico
Descripción	Sistema que permite la gestión, almacenamiento y seguimiento de documentos y reportes relacionados con la higiene ocupacional, con el objetivo de centralizar la información relacionada con los riesgos laborales para facilitar la toma de decisiones y la prevención de accidentes laborales.	Sistema para la planificación y ejecución de pruebas, seguimiento de errores y defectos, gestión de requisitos y casos de prueba, y generación de informes y análisis de métricas de calidad. Incluye herramientas de desarrollo y gestión de proyectos para facilitar la colaboración y la automatización de los procesos	Sistema encargado de almacenar y mantener actualizada la información de los requisitos legales aplicables, así como de facilitar la búsqueda y selección de los requisitos que son relevantes para la organización.	Reporte de escuelas de diferentes regiones, de estudiantes, docentes y de las calificaciones obtenidas.
Usuarios Finales	Empresa Pacasmayo- Área de Higiene Ocupacional	Empresa Pacasmayo- Área de control de calidad	Empresa Pacasmayo- Área de gestión legal	Departamento de Educación- Puerto Rico
Tamaño del equipo de desarrollo	2 desarrolladores	2 desarrolladores	2 desarrolladores	2 desarrolladores
Lenguaje de programación	Typescript	php	Typescript	C#
Tiempo estimado de desarrollo	2 meses y medio	4 meses	5 meses	7 meses

Fuente: Elaboración Propia

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad

Técnicas

Observación

La técnica propuesta consistió en realizar una visita al área de desarrollo para observar y analizar el proceso de desarrollo de software, incluyendo la integración de los procesos de desarrollo y operaciones. Durante esta supervisión, se prestó especial atención a las interacciones y comunicación entre los integrantes de cada equipo, observando cómo se resolvían los problemas y se tomaban las decisiones. Además, se analizaron las herramientas utilizadas en el proceso de desarrollo y su aplicación correspondiente, así como las prácticas de colaboración y trabajo en equipo entre los miembros de desarrollo y operaciones. Para la evaluación del progreso y la eficacia del proceso de desarrollo de software, se utilizaron métricas establecidas, como cycletime, incidentes, despliegues, defectos de producción, mejoras propuestas y tiempos de respuesta. Estos datos se emplearon para medir y evaluar el rendimiento del proceso de desarrollo de software y su eficacia en función de los objetivos establecidos. Se esperaba obtener los resultados de la implementación de DevOps y compararlos con los objetivos y metas establecidos; para así llevar a cabo un análisis crítico de la implementación de DevOps y proponer recomendaciones para futuras implementaciones. Para llevar a cabo esta observación, se utilizó un instrumento de ficha de observación cuyo objetivo era el registro de datos relevantes y de gran importancia para la investigación dentro de la organización. Esta ficha se utilizó para recopilar información mediante la interacción directa entre el investigador y la realidad que se presentaba, plasmándola en tarjetas o cuadros conocidos como fichas de observación [51].

Entrevistas

Se realizaron entrevistas a los desarrolladores y miembros del equipo de operaciones para recopilar sus opiniones y percepciones sobre la implementación de

DevOps y su impacto en el proceso de desarrollo de software. Estas entrevistas se fundamentaron en un listado de interrogantes incluidas en el instrumento de recolección de datos, que se diseñó en forma de encuesta.

Instrumentos de recolección de datos

Encuesta:

La encuesta se empleó con el propósito de recopilar datos cuantitativos relacionados con la implementación de DevOps, abordando aspectos tales como la eficacia de las herramientas y prácticas implementadas, así como la percepción de los colaboradores, donde el rango de posibles valores está comprendido entre 0 y 1 como se detalla en el anexo N° 04 la cual va orientada a la evaluación de DevOps en empresas de desarrollo de software [54].

Ficha de observación

Mediante el presente instrumento, se registró los datos relevantes de la organización en beneficio de la investigación, donde se va a indicar a detalle la aplicación de la técnica de observación pudiendo incluir evidencias como grabaciones, fotos, videos, entre otros. Se aplica al medir el indicador de Tiempo para restaurar el servicio, en el cual se describe el tiempo considerado y características adicionales como la manera y actitud en que se realiza dicha solución, como se muestra en el Anexo N° 08.

2.5. Procedimiento de análisis de datos

Los datos recopilados mediante la aplicación de los instrumentos según las técnicas planteadas, fueron preparados mediante limpieza, normalización de datos y priorización para así corroborar la validación de resultados tras su medición. De acuerdo al tema de investigación se evaluó la optimización del proceso de desarrollo de software mediante la implementación del modelo CALMS de DevOps en el proyecto seleccionado mediante los siguientes indicadores:

1. **Frecuencia de implementación**

Definición:

Se trata de la frecuencia con la que las aplicaciones se despliegan en producción, generalmente expresada en porcentaje y conocida como rendimiento. Esta métrica permite evaluar la capacidad de entregar software de manera ágil y constante, lo que, a su vez, facilita la obtención de feedback temprana por parte de los usuarios y la detección temprana de errores en el proceso [32].

Fórmula:

$$\text{Frecuencia de implementación} = \frac{\text{Número total de despliegues exitosos}}{\text{Cycletime}}$$

Términos:

Despliegues exitosos: Número de veces que se ha implementado una nueva versión de un software con éxito, es decir, sin errores o problemas significativos.

Cycletime: Comprende el tiempo entre el inicio del desarrollo hasta el despliegue en producción, responde a la siguiente fórmula:

$$\textbf{Cycletime} = \text{Tiempo de Desarrollo} + \text{Tiempo de Testing} + \text{Tiempo de Despliegue}$$

Escalas:

Bajo: .0-32%

Medio: 33-74%

Alto: 75 - 100%

Instrumento:

Ficha de registro de cambios

2. **Frecuencia de liberación de código:**

Definición:

Número de historias de usuario por sprint subidas a producción

Fórmula:

$$\text{CD} = \text{Cantidad de Despliegues por sprint}$$

Escalas:

Bajo: 0 - 1

Medio: 2 - 3

Alto: 4 - 5

Instrumento:

Ficha de observación

3. Plazo de entrega de cambios

Definición:

Es el tiempo que transcurre desde el compromiso del código, hasta la implementación y la producción, representado como una duración.

Fórmula:

$$\textbf{PEC} = \text{Días acordados para la entrega} - \text{Días empleados para la entrega}$$

Términos:

- *PEC: Plazo de entrega de cambios*
- **Días acordados para la entrega:** Días límite en la que se acuerda que un determinado cambio o mejora de software debe estar completado y estar listo para su implementación.
- **Días empleados para la entrega:** Número de días considerados para finalizar el trabajo y se implementó el cambio en el sistema.

Escalas:

Bajo: 3 días a más

Medio: 1-3 días

Alto: Menos de 1 día

Instrumento:

Ficha de registro de entrega de cambios

4. **Tiempo para restaurar el servicio:**

Definición:

Evalúa el tiempo que un equipo de desarrollo y operaciones necesita para reparar un servicio que ha dejado de funcionar adecuadamente. Se mide desde la detección del problema hasta el momento en que el servicio se restablece.

Fórmula:

$$\text{TTRS} = \text{Hora de resolución} - \text{Hora de inicio del incidente}$$

Términos:

- **Hora de resolución:** Hora en la que se soluciona el incidente o problema que afectó al servicio.
- **Hora de inicio del incidente:** hora en la que se detectó el incidente o problema que afectó al servicio.

Escalas:

Bajo: Menos de una semana

Medio: Menos de un día

Alto: Menos de una hora

Instrumento:

Ficha de observación

5. **Tasa de fallas en los cambios**

Definición:

Indicador que muestra el porcentaje de cambios realizados en una aplicación o servicio de software que terminan afectando su desempeño o funcionamiento, y que requieren reparaciones o correcciones posteriores para solucionar problemas como interrupciones o degradación del servicio; esto

puede incluir acciones de reversión, parcheo, corrección o solución de problemas urgentes [34].

Fórmula:

$$\text{Tasa de fallas en los cambios } = \frac{\text{Despligues fallidos}}{\text{Número total de despliegues}} x\ 100$$

Términos:

- **Despliegues fallidos:** Número total de despliegues implementados que causaron una degradación del servicio, es decir, que generaron problemas en la disponibilidad, confiabilidad, seguridad o cualquier otro aspecto del servicio en producción.
- **Número total de Despliegues:** Cantidad de cambios que se han implementado en producción en un periodo de tiempo determinado, incluyendo aquellos que han causado problemas y aquellos que no.

Escalas:

Bajo: >41%

Medio: 21-40%

Alto: 0-20%

Instrumento:

Registro de monitoreo mediante el uso de la Herramienta Jenkins

6. Ratio de éxito:

Definición:

Evalúa la proporción de código implementado de manera exitosa en cada sprint.

Fórmula:

$$\text{RE } = \frac{\text{DE}}{\text{TD}} x\ 100$$

Donde:

TD = DE+DF

DE = Despliegue exitosos por sprint

DF = Despliegues fallidos por sprint

TD = Total de despliegues por sprint

RE = Ratio de éxito

Escalas:

Bajo: 0-33%

Medio: 34-66%

Alto: 67-100%

Instrumento:

Ficha de observación

7. **Percepción del colaborador**

Definición:

Medida del grado de complacencia del colaborador con la orquestación y gestión de los proyectos.

Escalas:

Si: 1

No: 0

Instrumento:

Encuesta

2.6. Criterios éticos

En este estudio, se consideraron los aspectos de validez, transferibilidad, neutralidad y confirmación, ya que estos criterios se ajustaron de manera óptima al estudio. En cuanto a la validez, se realizó un análisis exhaustivo del contexto inicial y de los resultados obtenidos para responder a la pregunta sobre la optimización del proceso de desarrollo de software en el caso de estudio.

Mientras que, referente al criterio de transferibilidad, permitió generalizar los hallazgos del presente estudio, donde a través de la descripción detallada de los contextos y de las variables trabajadas se espera que los conocimientos plasmados sean adquiridos y/o tomados en cuenta por investigadores que se encuentran en posiciones similares para trabajos futuros.

Por otro lado, el criterio de neutralidad mantendrá una posición parcial, asegurando que los resultados obtenidos no sean modificados según la opinión propia por parte del investigador. Y, por último, el criterio de confirmación asegurará que los resultados sean publicados, compartidos y replicados en diferentes contextos de forma veraz y directa.

III. RESULTADOS Y DISCUSIÓN

3.1. Resultados

RESULTADOS DE LA ENCUESTA APLICADA ACERCA DE LA IMPLEMENTACIÓN DE DEVOPS

Las siguientes tablas contienen los resultados de la encuesta aplicada en el pre test y postest, donde se detalla la percepción de los colaboradores acerca de los procesos vinculados al desarrollo de software en la empresa considerada como caso de estudio, donde los valores más cercanos a 0 indica la carencia o ausencia de lo señalado y las respuestas más cercanas a 1 precisan que el cumplimiento de las mismas.

Resultados de la encuesta aplicada antes de la implementación de devops

3.1.1. Dimensión de integración continua

Tabla VI Percepción acerca de la integración continua-Pretest

Fase	Enfoque de Preguntas	%
Integración continua	Definición de políticas, procedimientos o estrategias para control de versiones	33.33%
	Repositorios para automatizar el control de versiones	0.00%

	Uso de herramientas para garantizar la integridad de código	50.00%
	Procedimientos para pruebas unitarias	0.00%
	Automatización del código fuente	0.00%
	Identificación de errores durante la integración de nuevo código	0.00%
	Políticas o procedimientos para recuperación del estado estable del sistema ante fallos	50.00%
	Acceso al control de versiones para los equipos de desarrollo y operaciones	66.67%
	PROMEDIO	25.00%

3.1.2. Dimensión de Entrega continua

Tabla VII Percepción acerca de la entrega continua-Pretest

Fase	Enfoque de Preguntas	%
Entrega continua	Ambientes de producción	33.33%
	Ambientes Productivos	0.00%
	Políticas, mecanismos o procedimientos para entrega a stage, preproducción o producción.	0.00%
	Uso de herramientas que automaticen la entrega	0.00%
	Criterios de verificación de funcionalidades a liberar	0.00%
	PROMEDIO	6.67%

3.1.3. Dimensión de pruebas continuas

Tabla VIII Percepción acerca de las pruebas continuas-Pretest

Fase	Enfoque de Preguntas	%
Pruebas continuas	Ambientes de pruebas	0.00%
	Pruebas que garantizan la consistencia del sistema	66.67%
	Pruebas con escenarios definidos	0.00%

	Pruebas de verificación de integridad de requerimientos	50.00%
	Pruebas según un plan definido	0.00%
	Políticas, mecanismos o procedimientos para la solución de incidentes	0.00%
	Pruebas en un espacio común definido	0.00%
	Ambientes de stage son un espejo de ambientes productivos	33.33%
	PROMEDIO	18.75%

3.1.4. Dimensión de monitoreo y observabilidad continua

Tabla IX Percepción acerca del monitoreo y observabilidad continua-Pretest

Fase	Enfoque de Preguntas	%
Monitoreo y observabilidad continua	Seguimiento a los logs implementados en el proyecto	0.00%
	Mecanismos para realizar el monitoreo de la infraestructura implementada	0.00%
	Mecanismos para monitorear el cumplimiento	0.00%
	Mecanismos de la detección de problemas en ambiente productivo	0.00%
	PROMEDIO	0.00%

3.1.5. Dimensión de Educación entorno a DevOps

Tabla X Percepción acerca de la educación en entorno DevOps-Pretest

Fase	Enfoque de Preguntas	%
Educación en torno a DevOps	Capacitación de partes interesadas en DevOps	0.00%
	Seguimiento de aplicación de Devops	0.00%
	Aplicación de Devops	0.00%
	PROMEDIO	0.00%

3.1.6. Dimensión de Realimentación continua e innovación

Tabla XI Percepción acerca de la realimentación continua e innovación-Pretest

Fase	Enfoque de Preguntas	%
Realimentación continua e innovación	Espacios para compartir conocimientos	0.00%
	Apertura a nuevas ideas para mejorar	83.33%
	Mecanismos para garantizar la continuidad de procesos	0.00%
	Espacios para compartir conocimientos de interés común	0.00%
	Espacios de retrospección para ver los puntos a mejorar	0.00%
	PROMEDIO	16.67%

3.1.7. Dimensión de cultura

Tabla XII Percepción acerca de la dimensión cultura-Pretest

Fase	Enfoque de Preguntas	%
Medición de la cultura	Mecanismo de medición de cultura	0.00%
	Seguimiento a la cultura organizacional	0.00%
	Comprensión y aceptación de la cultura actual	0.00%
	Procedimientos claros para planes de acción	0.00%
	PROMEDIO	0.00%

3.1.8. Dimensión de Despliegue continuo

Tabla XIII Percepción acerca del despliegue continuo-Pretest

Fase	Enfoque de Preguntas	%
Despliegue continuo	Procedimientos para la integridad de artefactos	0.00%
	Herramientas de automatización para el despliegue	50.00%
	Orquestación para despliegues de nuevos artefactos	0.00%
	Aprobación para desplegar	33.33%
	PROMEDIO	20.83%

3.1.9. Dimensión de satisfacción laboral

Tabla XIV Percepción acerca de la satisfacción laboral-Pretest

Fase	Enfoque de Preguntas	%
Satisfacción laboral	Identificación del nivel de satisfacción laboral	0.00%
	Incentivos para el nivel de satisfacción laboral	0.00%
	Incentivo para colaboración y coordinación asertiva	33.33%
	Espacios de realimentación	33.33%
	PROMEDIO	16.67%

Previo a la implementación de las prácticas de DevOps basadas en el modelo CALMS, los resultados promedio en las dimensiones evaluadas fueron los siguientes: Integración continua (25%), Entrega continua (6.67%), Pruebas continuas (18.75%), Monitoreo y observabilidad continua (0%), Educación en el ámbito de DevOps (0%), Retroalimentación continua e innovación (16.67%), Medición de la cultura (0%), Despliegue continuo (20.83%) y Satisfacción laboral (16.67%).

Resultados de la encuesta aplicada después de la implementación de devops

3.1.10. Dimensión de integración continua

Tabla XV Percepción acerca de la integración continua-Postest

Fase	Enfoque de Preguntas	%
Integración continua	Definición de políticas, procedimientos o estrategias para control de versiones	83.33%
	Repositorios para automatizar el control de versiones	100.00%
	Uso de herramientas para garantizar la integridad de código	100.00%
	Procedimientos para pruebas unitarias	0.00%
	Automatización del código fuente	100.00%
	Identificación de errores durante la integración de nuevo código	100.00%
	Políticas o procedimientos para recuperación del estado estable del sistema ante fallos	50.00%
	Acceso al control de versiones para los equipos de desarrollo y operaciones	100.00%

	PROMEDIO	79.17%

3.1.11. Dimensión de Entrega continua

Tabla XVI Percepción acerca de la entrega continua- Postest

Fase	Enfoque de Preguntas	%
Entrega continua	Ambientes de producción	66.67%
	Ambientes Productivos	83.33%
	Políticas, mecanismos o procedimientos para entrega a stage, preproducción o producción.	33.33%
	Uso de herramientas que automaticen la entrega	100.00%
	Criterios de verificación de funcionalidades a liberar	83.33%
	PROMEDIO	73.33%

3.1.12. Dimensión de pruebas continuas

Tabla XVII Percepción acerca de las pruebas continuas- Postest

Fase	Enfoque de Preguntas	%
Pruebas continuas	Ambientes de pruebas	66.67%
	Pruebas que garantizan la consistencia del sistema	83.33%
	Pruebas con escenarios definidos	0.00%
	Pruebas de verificación de integridad de requerimientos	100.00%
	Pruebas según un plan definido	83.33%
	Políticas, mecanismos o procedimientos para la solución de incidentes	0.00%
	Pruebas en un espacio común definido	83.33%
	Ambientes de stage son un espejo de ambientes productivos	100.00%
	PROMEDIO	64.58%

3.1.13. Dimensión de monitoreo y observabilidad continua

Tabla XVIII Percepción acerca del monitoreo y observabilidad continua- Postest

Fase	Enfoque de Preguntas	%
Monitoreo y observabilidad continua	Seguimiento a los logs implementados en el proyecto	100.00%
	Mecanismos para realizar el monitoreo de la infraestructura implementada	100.00%
	Mecanismos para monitorear el cumplimiento	83.33%
	Mecanismos de la detección de problemas en ambiente productivo	100.00%
	PROMEDIO	95.33%

3.1.14. Dimensión de Educación entorno a DevOps

Tabla XIX Percepción acerca de la educación en entorno DevOps- Postest

Fase	Enfoque de Preguntas	%
Educación en torno a DevOps	Capacitación de partes interesadas en DevOps	100.00%
	Seguimiento de aplicación de Devops	100.00%
	Aplicación de Devops	83.33%
	PROMEDIO	94.44%

3.1.15. Dimensión de Realimentación continua e innovación

Tabla XX Percepción acerca de la realimentación continua e innovación- Postest

Fase	Enfoque de Preguntas	%
Realimentación continua e innovación	Espacios para compartir conocimientos	100.00%
	Apertura a nuevas ideas para mejorar	100.00%
	Mecanismos para garantizar la continuidad de procesos	100.00%
	Espacios para compartir conocimientos de interés común	0.00%
	Espacios de retrospección para ver los puntos a mejorar	0.00%

PROMEDIO	60.00%

3.1.16. Dimensión de cultura

Tabla XXI Percepción acerca de la dimensión cultura- Postest

Fase	Enfoque de Preguntas	%
Medición de la cultura	Mecanismo de medición de cultura	66.67%
	Seguimiento a la cultura organizacional	100.00%
	Comprensión y aceptación de la cultura actual	100.00%
	Procedimientos claros para planes de acción	33.33%
	PROMEDIO	75.00%

3.1.17. Dimensión de Despliegue continuo

Tabla XXII Percepción acerca del despliegue continuo-Pretest

Fase	Enfoque de Preguntas	%
Despliegue continuo	Procedimientos para la integridad de artefactos	66.67%
	Herramientas de automatización para el despliegue	100.00%
	Orquestación para despliegues de nuevos artefactos	100.00%
	Aprobación para desplegar	100.00%
	PROMEDIO	91.67%

3.1.18. Dimensión de satisfacción laboral

Tabla XXIII Percepción acerca de la satisfacción laboral- Postest

Fase	Enfoque de Preguntas	%
Satisfacción laboral	Identificación del nivel de satisfacción laboral	66.67%
	Incentivos para el nivel de satisfacción laboral	100.00%
	Incentivo para colaboración y coordinación asertiva	100.00%

Espacios de realimentación	50.00%
PROMEDIO	79.17%

Después de implementar DevOps, se evidencian mejoras en todas las dimensiones evaluadas: integración continua (79.17%), entrega continua (73.33%), pruebas continuas (64.58%), monitoreo y observabilidad continua (95.33%), educación en el ámbito de DevOps (94.44%), retroalimentación continua e innovación (60%), medición de la cultura (75%), despliegue continuo (91.67%) y satisfacción laboral (79.17%). Estos resultados muestran un incremento significativo en la adopción y efectividad de prácticas DevOps en la organización, reflejando un mejor desempeño en las áreas evaluadas de desarrollo y operaciones.

A continuación, se presenta un análisis comparativo entre el pretest y postest, que muestra que antes de implementar DevOps, la empresa tenía un cumplimiento del 11.62%. Después de la implementación, este cumplimiento se incrementó en un 67.62%, alcanzando un cumplimiento actual del 79.24%.

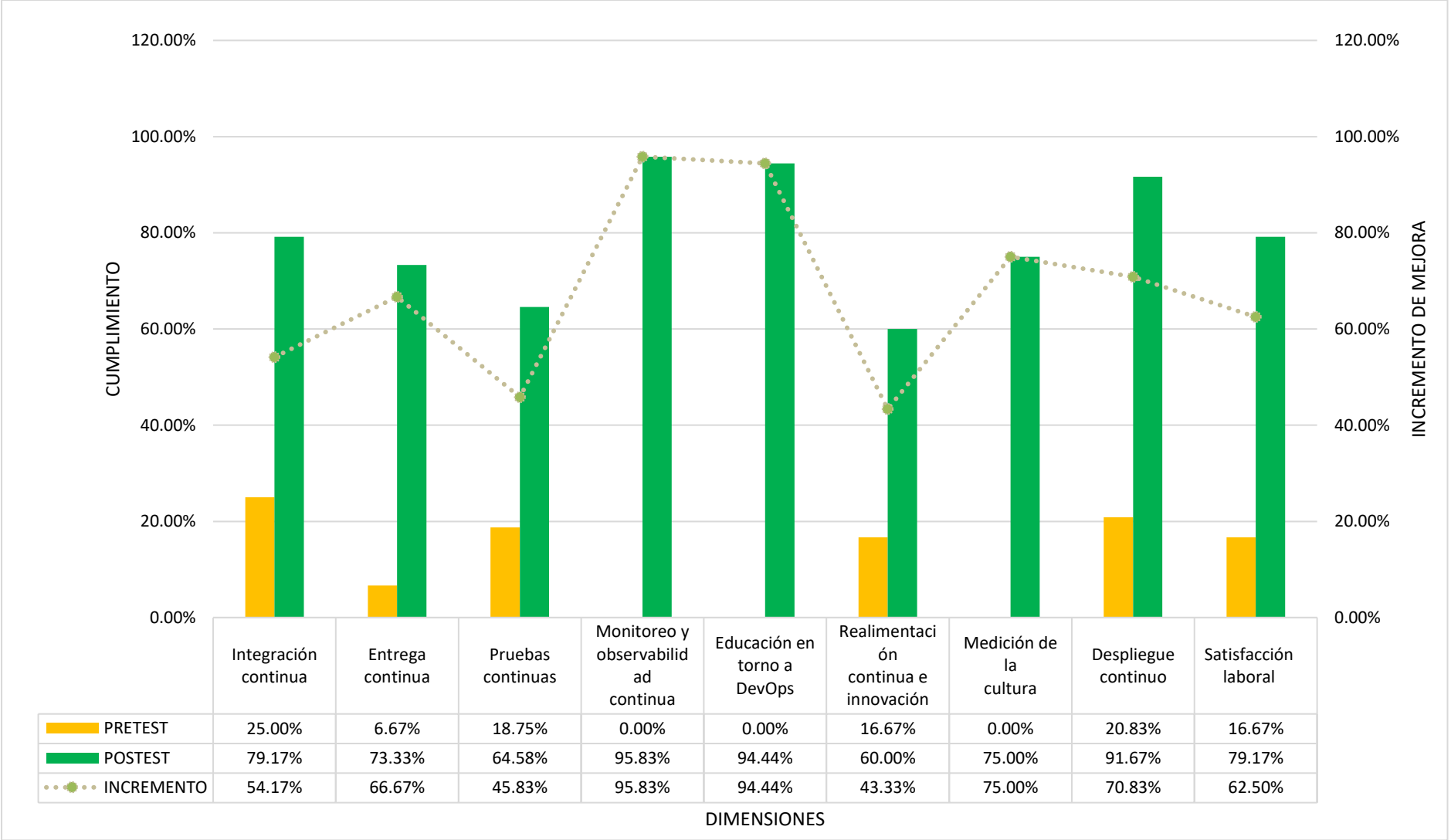


Fig. 1 Resultados de la encuesta aplicada a los colaboradores
Fuente: Elaboración propia

RESULTADOS DE LA OBSERVACIÓN REALIZADA

A continuación, se presentan la siguiente tabla con los resultados de la observación realizada. Estos resultados descriptivos de la investigación incluyen los siguientes indicadores: frecuencia de implementación, plazo de entrega de cambios, tiempo para restaurar el servicio, tasa de fallas en los cambios, frecuencia de liberación de código y ratio de éxito.

Tabla XXIV Resumen de observación realizada

DIMENSIÓN	INDICADOR	NIVELES		PRETEST	POSTEST	MEJORA
Despliegue e implementación	Frecuencia de implementación	Bajo	*[0% - 32%]*	*16.65%*	*42.12%*	*25.47%*
		Medio	*[33% - 74%]*			
		Alto	*[75% - 100%]*			
		Condición Final		**BAJO**	**MEDIO**	
	Frecuencia de liberación de Código	Bajo	*[0 - 1] despliegues*	*1.67*	*2.33*	*0.66*
		Medio	*[2 - 3] despliegues*			
		Alto	*[4 - 5] despliegues*			
		Condición Final		**MEDIO**	**ALTO**	
Velocidad	Plazo de entrega de cambios	Bajo	*[>3] día*	*1*	*-0.5*	*-1.5*
		Medio	*[1 - 3] días*			
		Alto	*0 días (el mismo día o antes)*			
		Condición Final		**MEDIO**	**ALTO**	
Monitoreo	Tiempo para restaurar el servicio	Bajo	[<1 semana]	*1.83*	*0.17*	*-1.66*
		Medio	[<1 día]			
		Alto	[<1 hora]			
		Condición Final		**MEDIO**	**ALTO**	
	Tasa de fallas en los cambios	Bajo	[> 41%]	*55.56%*	*11.11%*	*-44.45%*
		Medio	[21% - 40%]			
		Alto	[0 - 20%]			
		Condición Final		**BAJO**	**ALTO**	
Calidad	Ratio de Éxito	Bajo	[0% - 33%]	*44.44%*	*88.89%*	*44.45%*
		Medio	[34% - 66%]			
		Alto	[67% - 100%]			
		Condición Final		**MEDIO**	**ALTO**	

Fuente: Elaboración propia

Después de implementar DevOps, se observó una mejora significativa en los indicadores

evaluados. La Frecuencia de Implementación aumentó en un 25.47% (de 16.65% a 42.12%) y la Frecuencia de Liberación de Código mejoró en 0.66 (de 1.67 a 2.33). El Plazo de Entrega de Cambios se redujo de 1 a -0.5 días y el Tiempo para Restaurar el Servicio mejoró en 1.66 días (de 1.83 a 0.17). La Tasa de Fallas en los Cambios disminuyó en un 44.45% (de 55.56% a 11.11%). El Ratio de Éxito aumentó en un 44.45% (de 44.44% a 88.89%). Estos resultados reflejan una mejora significativa en los procesos de desarrollo de software, demostrando el impacto positivo de la implementación de DevOps en la eficiencia y calidad del trabajo.

3.2. Discusión

Las métricas seleccionadas demuestran que la adopción de DevOps mejora significativamente el proceso de desarrollo de software. Esto ha impactado positivamente en lapercepción de los colaboradores y la aceptación del modelo planteado, con un incremento de hasta el 67.62%. Asimismo, se observa una influencia positiva del 75.6% en la calidad del trabajo en equipo, como se ha evidenciado en la implementación de DevOps en empresas de Indonesia [3].

La implementación de DevOps reduce el ciclo de desarrollo (Cycletime), lo cual aumenta la frecuencia de implementación de código; lo cual mejoró del 16.65% al 42.12%, mostrando un avance del 25.47%. Estos resultados son congruentes con los resultados en una empresa colombiana en la que experimentó una mejora del 70% en su Cycletime, un 60% de mejora en percepción de colaboradores, un 90% de mejora en tiempos de respuesta y mantenimientos y en una disminución de errores por reléase de un 95% [58]

Con la adopción de DevOps, la frecuencia de liberación de código aumentó de 1.67 a 2.33 despliegues por sprint, mejorando en 0.66. El plazo de entrega de cambios también mejoró, pasando de un aplazamiento de un día a una entrega anticipada de 0.5 días, lo que resulta en un adelanto de 1.5 días. Esto se traduce en un ahorro significativo de recursos como tiempo y dinero. Similarmente en otra investigación se demostró un incremento en el número y en la calidad de los entregables en diversas etapas del ciclo de desarrollo del software [59]. Asimismo, [60] reportó mejoras en los tiempos de despliegue, con una

reducción del 95%. En el estudio de arquitectura de microservicios se evidenció que la implementación de DevOps mejora el tiempo de despliegue y liberación de código mediante la orquestación de contenedores y microservicios [61].

La adopción de DevOps redujo significativamente la tasa de fallas en los cambios, especialmente aquellos causados por conflictos en la fusión manual de código. La frecuencia de fallas disminuyó de un 55.56% a un 11.11%, resultando en una reducción del 44.45%. Además, el tiempo para restaurar dichas fallas se redujo notablemente, pasando de 1.83 días a solo 0.17 días. El Ratio de éxito de los despliegues también mejoró considerablemente, aumentando del 44.44% al 88.89%, lo que representa una mejora del 44.45% en el número de despliegues exitosos. Estos resultados son consistentes con una investigación en Lisboa que reportó mejoras de hasta un 500% en los tiempos de resolución de incidentes y una mejora del 83.33% el número de problemas mensuales, donde tras la implementación del monitoreo constante se puso disminuir de tres a solo 0.5 problemas mensuales [60].

De manera similar en otras investigaciones se reportó una reducción del 90% en la tasa de fallos de cambios, una disminución del 50% en el tiempo de liberación de código y una reducción del 71% en las solicitudes de cambio al producto, estos hallazgos destacan el impacto positivo de la implementación del enfoque ágil en términos de calidad, eficiencia y control de cambios en el desarrollo de software. Además, comparando con grandes empresas que utilizan DevOps como Netflix, Amazon, Google y Facebook, se encontró que estas realizan implementaciones de código 46 veces más frecuentes, tienen un tiempo medio de recuperación de inactividad 170 veces más rápido, un tiempo de entrega 440 veces más veloz y una tasa de fallos cinco veces menor (1/5 de posibilidad de que falle un cambio) [17].

3.3. Aporte a la investigación

En el contexto de la presente investigación, el primer paso fundamental fue establecer una sólida línea de base. Para lograrlo, se procedió a delimitar claramente la empresa objeto de estudio. Esto se logró mediante la aplicación de una encuesta y la utilización de fichas de observación como herramientas clave. Estas estrategias permitieron recopilar información

detallada relacionada con la percepción de los colaboradores, sin importar el área en la que se desempeñaban.

3.3.1. Aplicación de Encuesta

Con el propósito de establecer una línea base precisa, se consideró de gran importancia medir la percepción de los colaboradores en relación a los equipos de desarrollo y operaciones en la empresa. Para lograrlo, se implementó una encuesta titulada "Encuesta realizada a los equipos de desarrollo y operaciones en la empresa" (Anexo N°04), compuesta por un total de 45 preguntas. Estas preguntas fueron diseñadas para evaluar dimensiones clave del proceso, como la integración continua, entrega continua, pruebas continuas, monitoreo y observabilidad continua, educación en entorno DevOps, realimentación continua, innovación, medición de la cultura, despliegue continuo y satisfacción laboral. Con el fin de garantizar resultados significativos, el instrumento consideró dos escalas, donde "Sí" (1) indicaba el cumplimiento del ítem y "No" (0) hacía referencia al incumplimiento y ausencia del mismo; de esta manera, se obtuvo una visión clara de la percepción de los colaboradores en cada área. La encuesta se llevó a cabo de forma virtual a través de un formulario alojado en Google Drive. Para asegurar la calidad de los datos recopilados, se validó que cada pregunta fuera de respuesta obligatoria y que se respondiera utilizando las escalas establecidas, lo que garantizó la consistencia en las respuestas y evitó la presencia de valores atípicos que pudieran distorsionar el análisis posterior. La aplicación de esta encuesta basada en escalas permitió medir la percepción de los colaboradores en relación a diferentes dimensiones del proceso de desarrollo y operaciones. Su implementación virtual mediante un formulario en Google Drive aseguró la calidad de los datos y facilitó el análisis posterior para obtener conclusiones relevantes y orientar las mejoras necesarias en la empresa.

3.3.2. Registro de fichas de observación

Para lograr una línea base precisa, se consideró de gran importancia medir el estado actual de la empresa utilizando los indicadores propuestos en la presente investigación. Con este propósito, se elaboraron 6 fichas de observación pretest, cuyos resultados se presentan

en el “Anexo 09: Resultados de la observación pretest a la empresa”. El llenado de estas fichas de observación se llevó a cabo de forma virtual, dado que la empresa no cuenta con una sede física. En este sentido, fue fundamental contar con el apoyo, compromiso y seriedad de los colaboradores para asegurar la calidad de los datos recopilados. Las fichas de observación pretest se diseñaron para capturar información relevante sobre los indicadores propuestos en la investigación. A través de la observación, se recopilaron datos objetivos y detallados sobre el estado actual de la empresa en relación a estos indicadores específicos. La elección de una metodología de observación virtual permitió superar la limitación de la falta de una sede física y garantizó la participación de los colaboradores sin importar su ubicación geográfica. Además, facilitó la recolección y el registro de los datos de manera eficiente y organizada.

Estado actual DevOps-cultura.

El equipo de desarrollo y operaciones demuestra una actitud receptiva para adaptarse, adquirir nuevos conocimientos tecnológicos y fomentar cambios culturales. Lo más destacado es su plena conciencia de la importancia de adoptar DevOps como un componente fundamental de la transformación tecnológica tanto a nivel del mercado como de la empresa.

Estado actual DevOps-automatización.

En el proceso actual de desarrollo de software, no se ha logrado una automatización completa de tareas críticas, como la compilación, las pruebas unitarias y funcionales, la revisión de código estático y el despliegue. Cada fase del proceso de llevar el software a producción se efectúa manualmente. Los resultados de la encuesta señalan áreas de mejora en los dos pilares fundamentales de DevOps: la cultura y la automatización. Por lo tanto, el enfoque del estudio de caso se dirigirá hacia el pilar identificado como más débil, que es la automatización en todo el ciclo de desarrollo de software. Nuestro objetivo será implementar soluciones y prácticas que permitan automatizar y optimizar estas tareas, lo que resultará en un proceso de desarrollo más eficiente y ágil.

3.3.3. Delimitación del Modelo Implementado

Considerando el estado actual del proceso de desarrollo de software en la empresa, se propuso la implementación de DevOps como enfoque para optimizar dicho proceso. El objetivo principal fue reducir las brechas de interacción y comunicación entre los equipos, promoviendo la estandarización, automatización y organización del flujo de trabajo.

Con el fin de alcanzar este propósito, se empleó los lineamientos propuestos por el modelo CALMS, para alcanzar mejoras significativas en la eficiencia y agilidad en el proceso de desarrollo de software, además, se buscó elevar este sistema a un nivel superior de excelencia y éxito, aprovechando al máximo las ventajas inherentes a su implementación. Esta iniciativa no solo implicó automatizar el rendimiento y la productividad de los equipos, sino también impulsó una adopción de cultura colaborativa, con aprendizaje y mejoras en toda la organización.

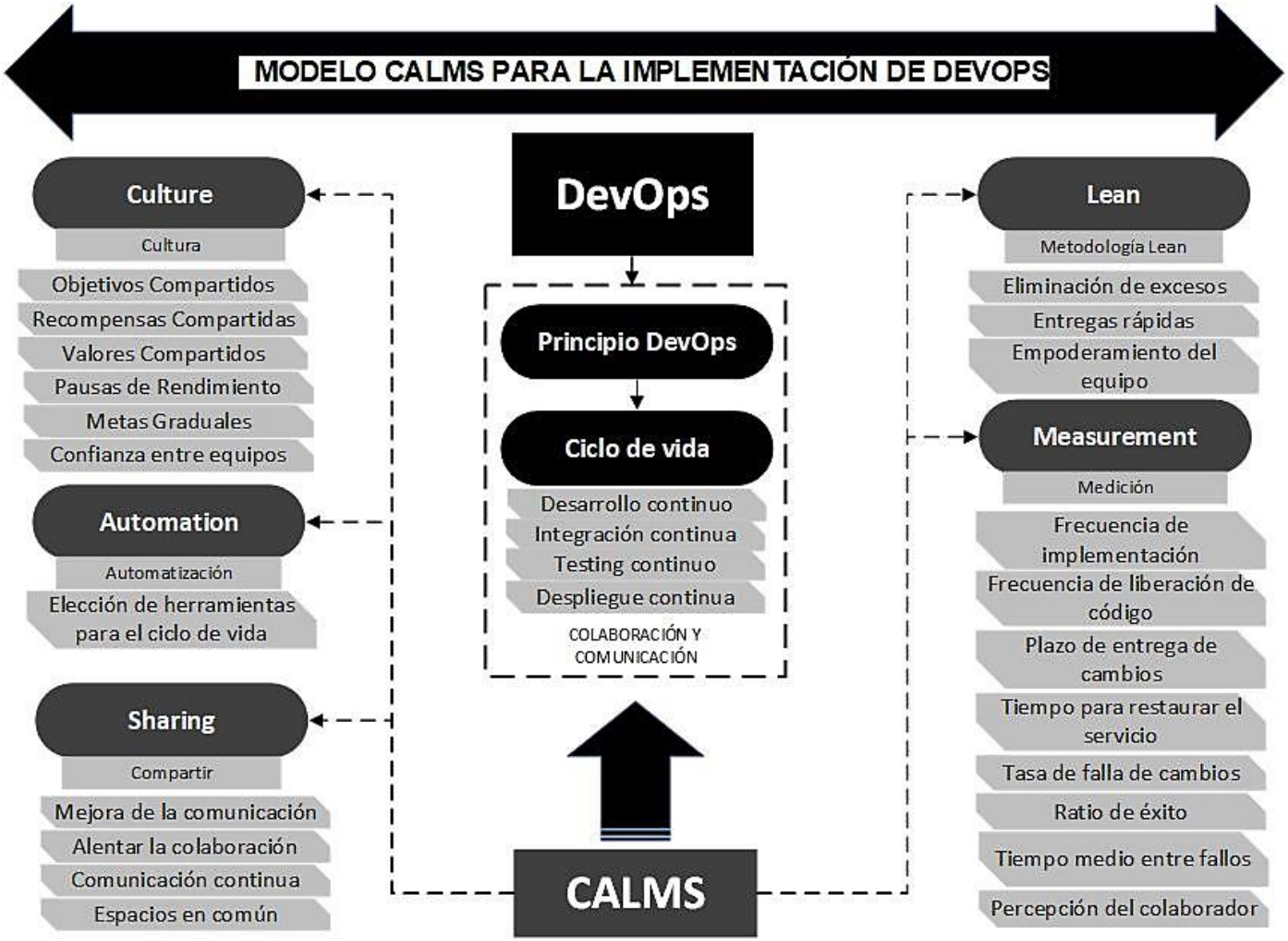


Fig. 2 Esquema del modelo Propuesto a Implementar
Fuente: Elaboración propia

A continuación, se muestra las fases y las actividades propuestas para implementación de DevOps:

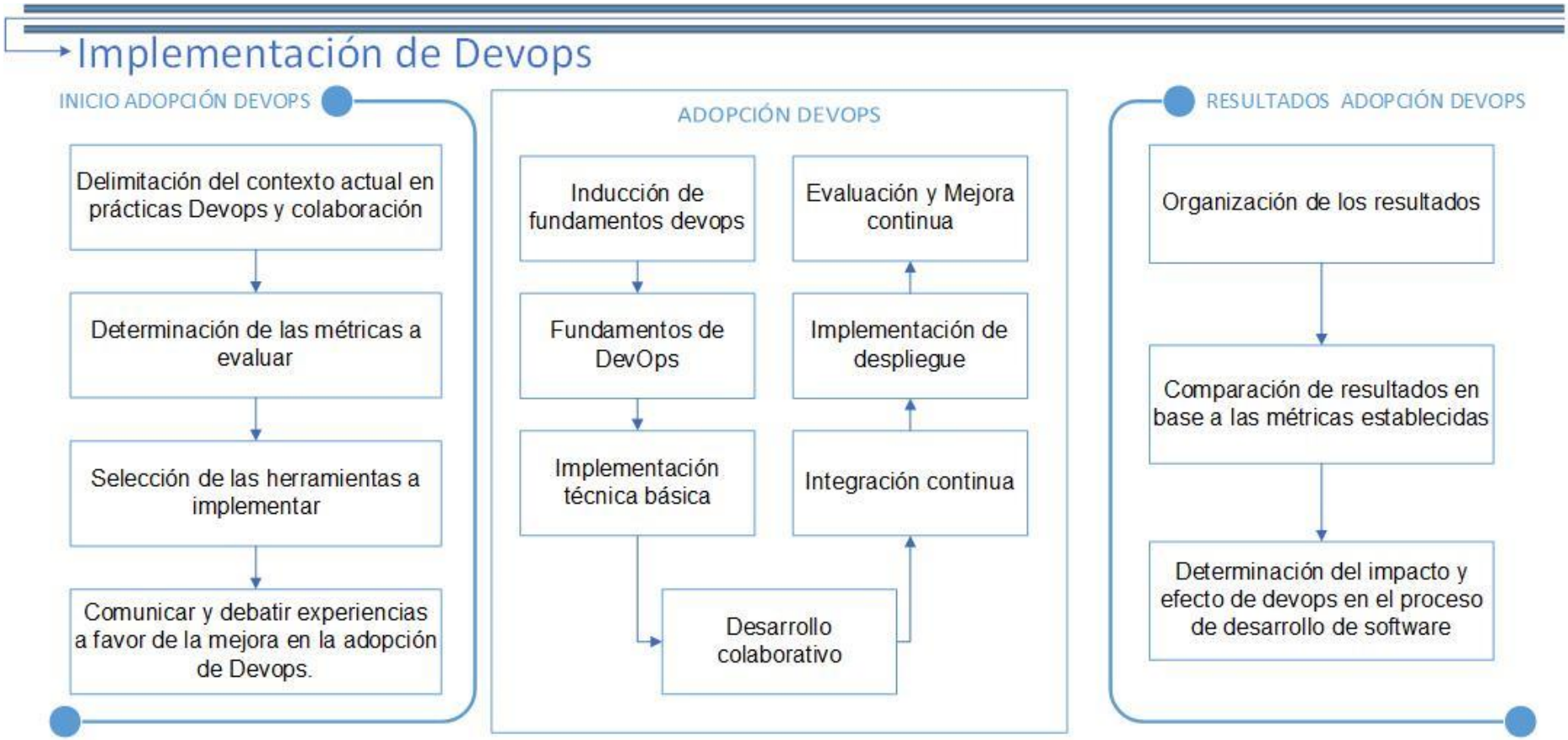


Fig. 3 Fases y actividades para implementación de DevOps
Fuente: Elaboración propia

A continuación, se muestra el flujo del proceso de desarrollo de Software mediante la configuración de herramientas

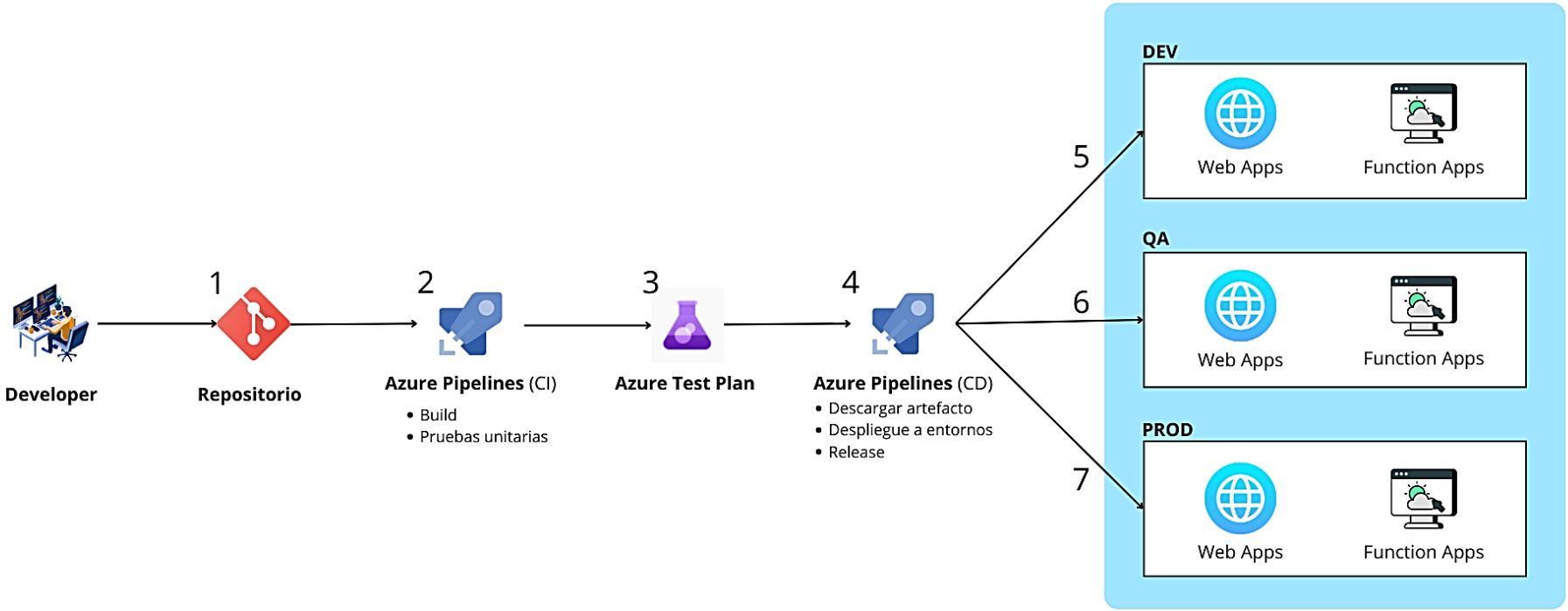


Fig. 4 Flujo de desarrollo aplicado
Fuente: Elaboración propia

La figura anteriormente mostrada representa gráficamente la integración y el flujo del proceso propuesto, el cual se enfoca principalmente en el desarrollo, la integración, las pruebas y el despliegue continuos. A continuación, se detalla cada uno de los pasos del proceso:

a) Gestión de Pull Requests: Las ramas del código se envían mediante comandos Git y se almacenan en Azure Repos. Al aceptar un Pull Request, el pipeline inicia automáticamente la recuperación de archivos, componentes y bibliotecas añadidas al proyecto, preparando el entorno para la compilación y las pruebas unitarias.

b) Integración Continua (CI): El pipeline de CI se dispara automáticamente con cada push o pull request, compilando el código y ejecutando pruebas automáticas.

c) Ejecución de Planes de Pruebas: Los planes de pruebas se ejecutan automáticamente para validar las pruebas funcionales diseñadas por el equipo de calidad. Si hay fallos, el proceso se detiene y se notifica a los desarrolladores; si las pruebas son exitosas, se procede al despliegue continuo (CD). Se realizan pruebas manuales y/o automatizadas para verificar la funcionalidad. Los resultados se registran y se crean bugs o tareas si se encuentran problemas.

d) Despliegue Continuo (CD): El despliegue se realiza automáticamente en diversos entornos, especialmente en desarrollo y pruebas, para asegurar que todo funcione correctamente antes de pasar a producción. Una vez que el código ha pasado todas las pruebas, se dispara el pipeline de release y se despliega en entornos de Desarrollo, Staging y Producción de manera controlada. El despliegue final en el entorno de producción se realiza manualmente.

e) Entorno de Desarrollo: Espacio destinado para que el equipo de desarrollo realice las pruebas necesarias.

f) Entorno de Calidad: Espacio utilizado para llevar a cabo pruebas funcionales y de aceptación. Este entorno replica el entorno de producción para asegurar que las

pruebas sean precisas.

g) Entorno de Producción: Ambiente utilizado por el usuario final.

h) Monitoreo Continuo: Este proceso se encarga de evaluar métricas de ejecución, tiempos de respuesta y errores de carga o peticiones al servidor, permitiendo mejoras continuas en la calidad del software.

Para implementar la secuencialidad de las fases presentadas, es necesario contar con un plan bien estructurado y alineado con los objetivos de la empresa. Este plan de capacitación, detallado en el anexo N°09, describe los pasos secuenciales propuestos y ejecutados para la adopción de la cultura DevOps. Además, incluye apartados que delimitan objetivos, alcance, definiciones, responsables y las sesiones aplicadas a los colaboradores involucrados en el proceso de desarrollo de software. Este flujo asegura que el código sea continuamente integrado, probado y desplegado de manera eficiente y controlada, facilitando la entrega continua y confiable de software.

3.3.3.1. Implementación del Modelo de optimización del proceso de desarrollo de software

La adopción de un nuevo enfoque generalmente implica la necesidad de contar con un plan que esté perfectamente alineado con los objetivos de la empresa, por tal motivo que el modelo comprende cuatro ítems del ciclo de vida: Desarrollo continuo, integración continua, testing continuo y despliegue continuo; para los cuales se establecen lineamientos según el modelo CALMS de DevOps:

A) Cultura

Desde una perspectiva empresarial, se propone una transformación cultural que abarca aspectos personales, procesuales y tecnológicos. El propósito de esta iniciativa fue lograr una integración armoniosa y colaborativa entre los procesos y las personas, impulsando la eficacia del trabajo y la comunicación. La cultura desempeña un papel fundamental en el modelo propuesto, reconociendo a las personas como actores clave a lo largo de todo el proceso. A

pesar de que una empresa pueda contar con procesos excepcionales o herramientas automatizadas altamente eficientes, estas carecen de valor si no se dispone de un equipo adecuado para colocar en práctica los procesos y utilizar las herramientas. Por lo tanto, se destaca que las personas son el recurso principal en la adopción del modelo propuesto. Otro aspecto destacado en el modelo es la promoción de la colaboración constante entre los diversos equipos, posiciones y roles. El enfoque se orienta hacia la empresa en su conjunto, en lugar de verla como una competencia de objetivos departamentales, de áreas o de equipos. Esto fomenta la construcción de confianza, la cooperación y el aprendizaje a través de la experiencia.

A continuación, se presenta una lista de factores a considerar durante el proceso de mejora cultural en la siguiente tabla.

Tabla XXV Lineamientos culturales del modelo

LINEAMIENTO	RESULTADO ESPERADO	ACTIVIDADES PROPUESTAS
Objetivos Compartidos	Los equipos de desarrollo y operaciones colaboran para mejora del proceso de desarrollo de software.	Se identifica y elimina cuellos de botella, automatiza tareas repetitivas y mejora la eficiencia en la gestión y operación de sistemas.
Recompensas Compartidas	Reconocimiento de los logros de los equipos de desarrollo y operaciones de manera pública y visible.	Reconocimiento mediante anuncios en reuniones, correos electrónicos de reconocimiento o tableros de reconocimiento en espacios comunes.
	Se establecen incentivos por el logro de metas específicas o en el reconocimiento de un trabajo excepcional.	Establecer bonificaciones y premios económicos o en forma de incentivos para aquellos equipos que demuestren una colaboración y resultados destacados en la adopción de prácticas DevOps.
	Reconocimiento y valoración del esfuerzo y los resultados en la implementación de la cultura DevOps como parte integral del proceso de evaluación.	Incluir criterios específicos relacionados con la adopción de prácticas DevOps en las evaluaciones de desempeño de los miembros del equipo.
Valores Compartidos	Eliminación de brecha entre el equipo de desarrollo con el equipo de operaciones.	Adoptar buenas prácticas, estandarizando procesos y organización respecto a los servicios generados.

Metas Graduales	Generación de conciencia sobre los principios y beneficios de DevOps entre los equipos de desarrollo y operaciones.	Capacitaciones, compartiendo información relevante para fomentar la comprensión y la adopción de la cultura DevOps.
	Establecimiento de la meta de mejorar la colaboración y la comunicación entre las áreas	Reuniones regulares, canales de comunicación efectivos y el intercambio de conocimientos y mejores prácticas.
	Establecimiento de la meta de implementar la integración continua y así poder detectar y corregir errores de manera temprana y acelerar el proceso de desarrollo.	Integrar y probar el código de forma continua a medida que se desarrolla.
	Establecimiento de la meta de implementar la entrega continua.	Implementar nuevas versiones del software de manera rápida y confiable en los entornos de producción.
	Identificación y solución de problemas de manera proactiva, así como la recopilación de información para futuras mejoras.	Implementar un sistema de monitoreo continuo para recopilar datos sobre el rendimiento del software en producción
	Establecimiento de una cultura de mejora continua.	Evaluaciones periódicas, identificación de oportunidades de mejora e implementación de cambios graduales en los procesos, herramientas y prácticas de trabajo.
Confianza entre equipos	Equipos que no operan de forma aislada, promoviendo la educación constante y erradicando el miedo a cometer errores.	La colaboración de todos los equipos se orienta hacia el beneficio principal de la empresa.

Fuente: [53]

El propósito de esta perspectiva cultural es orientar a todos los miembros del equipo para que avancen de manera consistente hacia metas compartidas en pro del bienestar de la empresa.

B) Principios y ciclo de vida

La implementación de DevOps, basada en el modelo CALMS, tiene como objetivo principal generar valor y mejorar los productos o servicios entregados a los clientes. Con esta propuesta, se busca optimizar el proceso de desarrollo de software en la empresa, tomando en cuenta el estado actual del proceso como punto de partida. A través de la integración del

ciclo de vida que ofrece DevOps, se pretende impulsar la eficiencia, colaboración y entrega continua, permitiendo así alcanzar los objetivos comerciales y cumplir con los requerimientos de los clientes efectivamente.

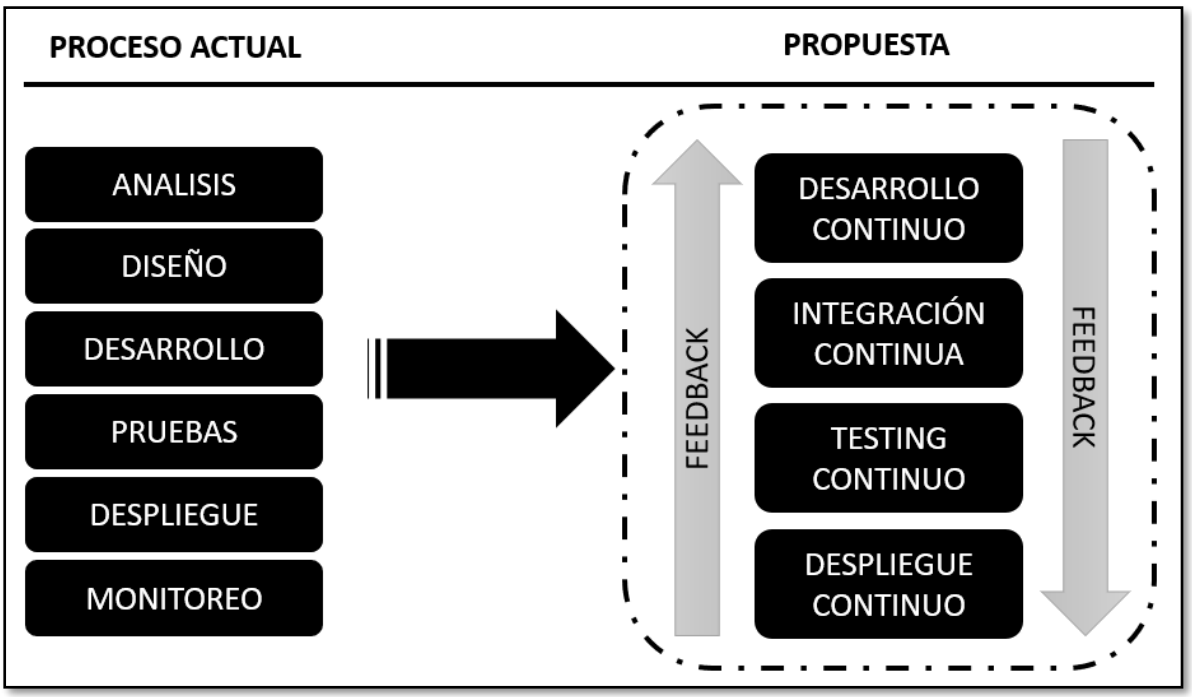


Fig. 5 Propuesta de la arquitectura del modelo
Fuente: Elaboración propia

La imagen ilustra el flujo del proceso actual, que se caracteriza por ser un procedimiento aislado entre los diversos equipos de la empresa. Este enfoque presenta múltiples desafíos a lo largo del ciclo de desarrollo. Por tanto, se propone la implementación de un enfoque que promueva una comunicación continua entre los equipos y cada una de las cuatro etapas del proceso. El propósito de esta propuesta es establecer, en cada fase, un proceso continuo que permita la realización de evaluaciones y validaciones constantes de los productos o servicios que se están desarrollando.

A continuación, se presenta el modelo técnico diseñado para integrar las diversas fases en un proceso continuo, con la finalidad de estandarizar y optimizar los procesos. Se pone un fuerte énfasis en la mejora de las operaciones y el desarrollo en la empresa.

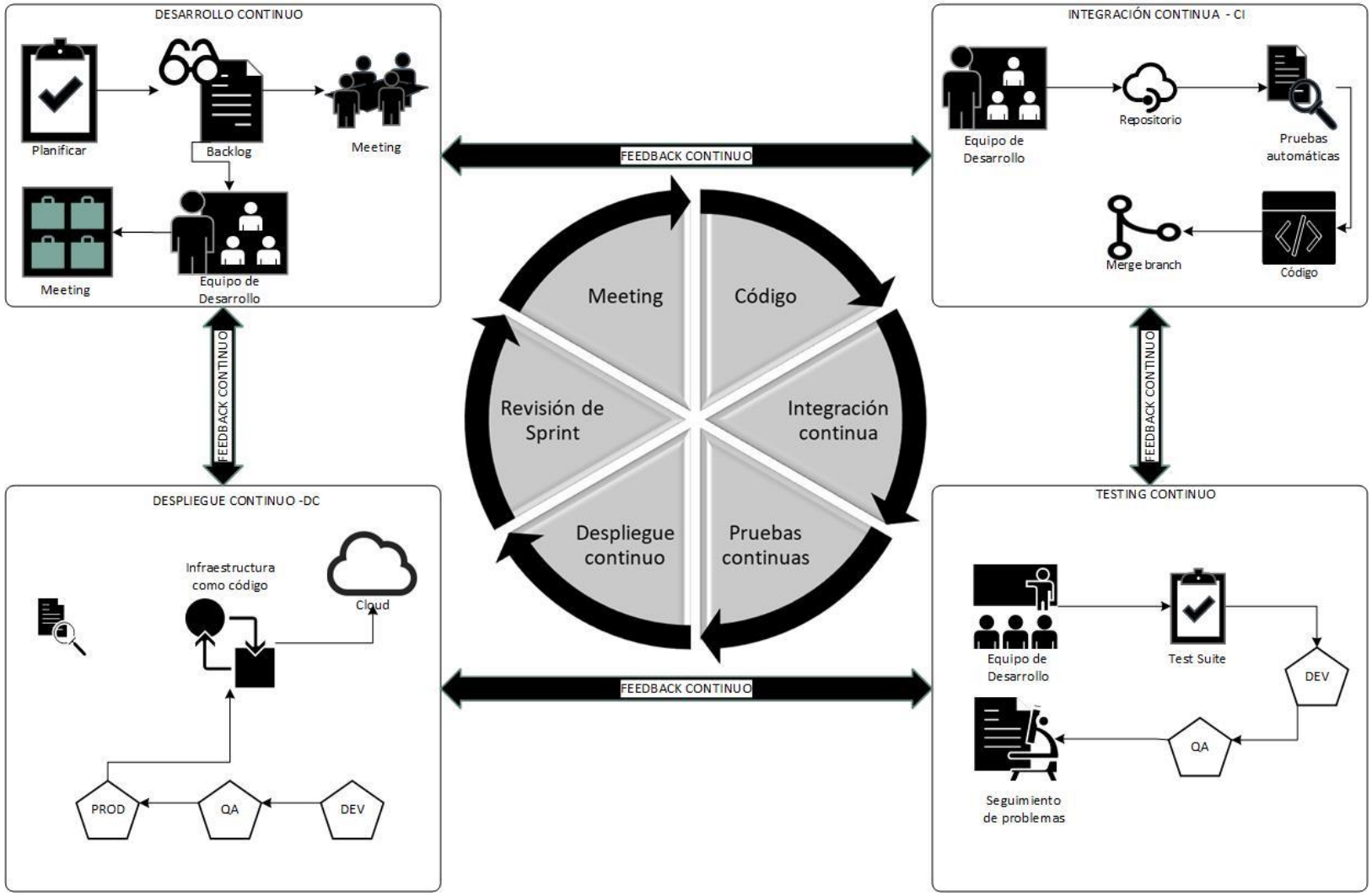


Fig. 6 Modelo a Nivel Técnico
Fuente: Elaboración propia

C) Fases de implementación

En cuanto al diagrama de la propuesta, se desglosa el proceso requerido para llevar a cabo el modelo, paso a paso. Esto establece un marco que permite la aplicación completa del proceso de desarrollo de software recomendado para la empresa.

Escenario

La empresa Ayarcode Consulting SAC se consideró como caso de estudio, donde se implementó la automatización de principio a fin en uno de sus productos de software llamado "MHO-Higiene Ocupacional", el cual tiene como objetivo el registro y seguimiento de los diversos estudios realizados por el área usuaria de Higiene Ocupacional, con el propósito de guardar la información de los estudios contemplados, incluyendo documentos adjuntos que evidencien el registro de monitoreo y el informe realizado por el área; asimismo permite guardar la información de las observaciones realizadas, respaldadas con una evidencia y así también el historial de las actualizaciones y/o modificaciones que se realicen. La plataforma permite el registro manual y masivo de observaciones, asimismo permite consultar historial de observaciones registradas por estudio, contando con módulos para realizar el mantenimiento respectivo, que incluye: búsqueda, registro, actualización y eliminación a: Empresa, Sede, Área, Función, Tipo Agente, Agente, Estado, Puesto Laboral y Usuario.

Herramientas tecnológicas

En la creación del escenario se empleó el entorno proporcionado por Microsoft Azure DevOps, que se basa en sus servicios en la nube. Es importante señalar que había la posibilidad de optar por otros entornos que también cumplen con el enfoque DevOps de referencia.

Tabla XXVI Herramientas tecnológicas

TECNOLOGÍA	DESCRIPCIÓN
Azure DevOps	Azure Boards, Azure Pipelines, Azure Repos, Azure Test Plans.
Portal Azure	Máquina virtual, SQL Azure, Azure Container, App Service, Azure DevOps, Azure Monitor
Git	Control de versiones

Fuente: Elaboración propia

CALMS como marco de trabajo

Durante todo el proceso de desarrollo del escenario propuesto, se tomó como punto de referencia el modelo CALMS, ya que este modelo proporcionó factores fundamentales que fomentaron la agilidad y la colaboración en el proceso de desarrollo de software.

Tabla XXVII Componentes ágiles para el proceso colaborativo DevOps

	COMPONENTE	ACCIÓN
Iterativo	Product Backlog	Listado de actividades y tareas pendientes.
	Sprint	Estimación de tiempo basada en el Product Backlog, que puede abarcar días, semanas o meses.
	Daily Meet	Reuniones diarias para discutir: ¿Qué hice? ¿Qué haré? ¿Hay algún impedimento?
	Sprint Review /Retrospective	Reunión general con todos los equipos del proyecto para cerrar el Sprint y entregar el producto.

Fuente: [24]

Se observan algunos de los elementos de la metodología Lean, los cuales refuerzan la colaboración entre las áreas, reduciendo los problemas entre los equipos en relación al proyecto en curso. Estos equipos se integran en el enfoque DevOps. Para llevar a cabo y verificar el escenario, se emplearán todas las etapas propuestas para proporcionar una solución eficaz en el proceso de creación de software, que se detallan en las siguientes fases.

D) Desarrollo continuo

Todos los desarrolladores, miembros del equipo de operaciones, probadores y analistas de negocios en una plataforma conjunta para la planificación y ejecución del plan inicial. El objetivo es establecer el backlog de manera apropiada, siguiendo las directrices proporcionadas por el modelo CALMS. En esta etapa, es necesario crear y delimitar los siguientes elementos:

Planificación: Etapa inicial de deliberación sobre el alcance del proyecto, donde se evalúan y analizan tanto los requisitos funcionales como los no funcionales.

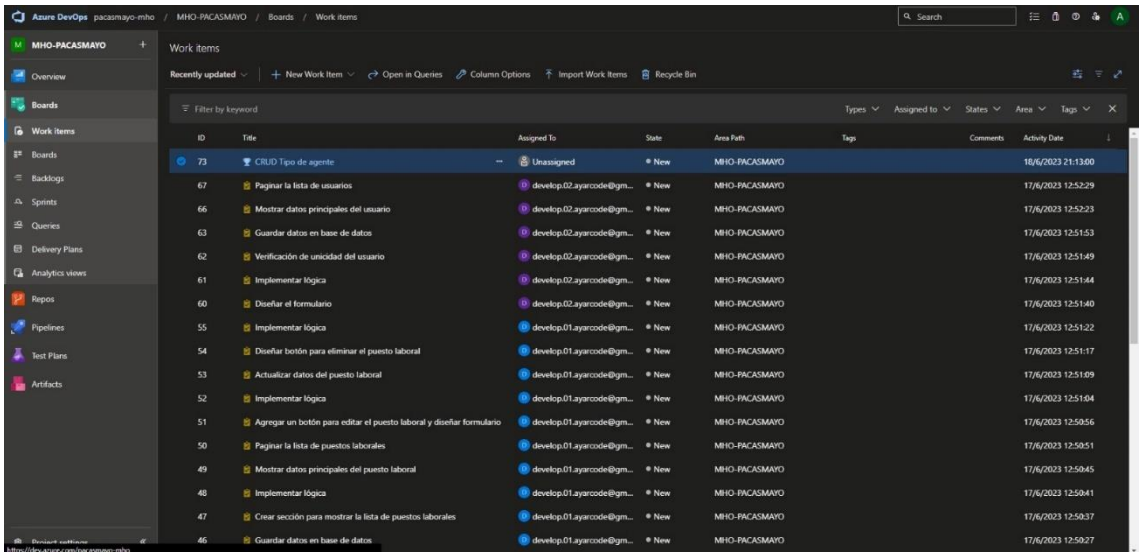


Fig. 7 Lista de requerimientos
Fuente: Elaboración propia

Backlog: Inventario de las actividades dispuestas en función de su valor o prioridad, con el propósito de identificar los resultados a abordar en las primeras iteraciones (sprints).

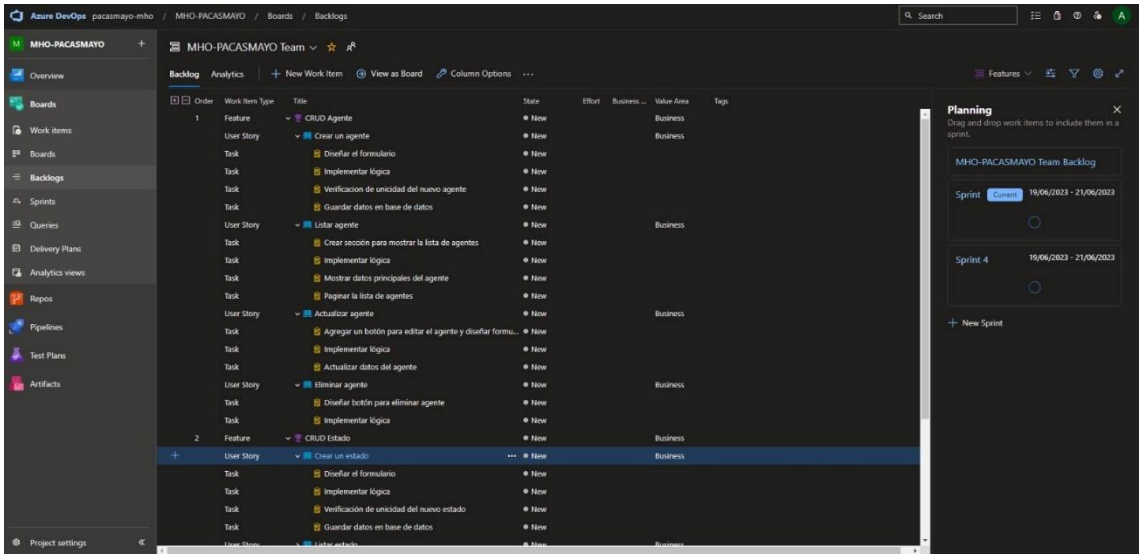


Fig. 8 Backlog de prioridades
Fuente: Elaboración propia

Sprint: Fraccionar los requisitos en actividades con el propósito de completarlas en un plazo de una semana, teniendo en cuenta los estados de las actividades asignadas. Asignado, En progreso y Terminado.

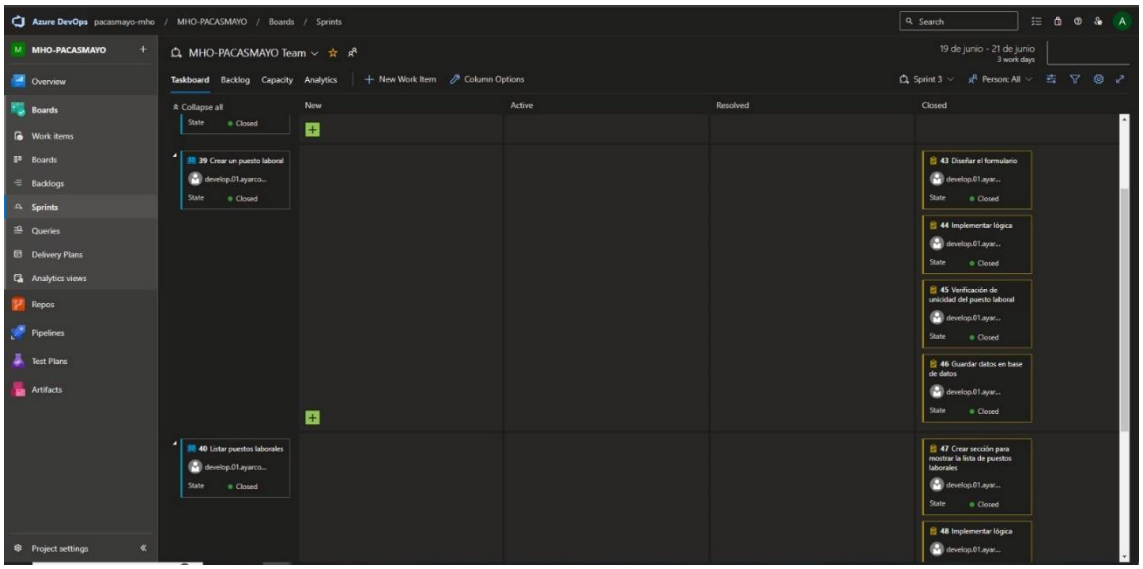


Fig. 9 Sprint con asignaciones
Fuente: Elaboración propia

Desarrollo:

Inicia con la ejecución de las actividades asignadas en el sprint. Durante esta fase, se crean branchs en el repositorio del proyecto, utilizando la branch principal 'master' o 'main' como base. Es crucial considerar la gestión de versiones, las ramas principales como 'Master', 'Main', y 'Dev', así como las ramas específicas para cada requerimiento o tarea asignada.

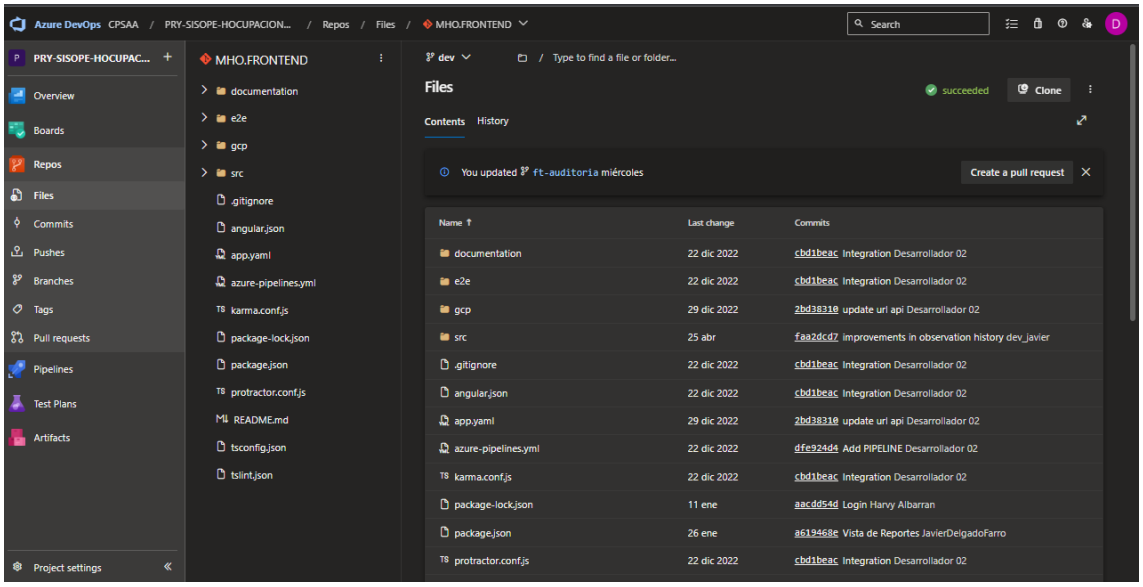


Fig. 10 Repositorio y ramas
Fuente: Elaboración propia

Daily Meeting: Encuentros breves de hasta 15 minutos en los que los equipos comparten lo que han logrado, sus próximos pasos y si enfrentan algún obstáculo. El objetivo es fomentar la colaboración entre todos los participantes.

Sprint Review /Retrospective: Exposición de las tareas completadas durante el sprint, ya sea desde una perspectiva funcional o técnica, con el propósito de recibir feedback sobre lo que se ha presentado. En la Retrospective, el equipo debe señalar oportunidades de mejora o sugerencias para perfeccionar la gestión de los procesos, tanto a nivel funcional como técnico.

Métricas: Mantener una revisión continua de la distribución de tareas, los plazos previstos y el estado de avance de las tareas, con el propósito de generar sugerencias de mejora a lo largo del proceso de desarrollo ininterrumpido.

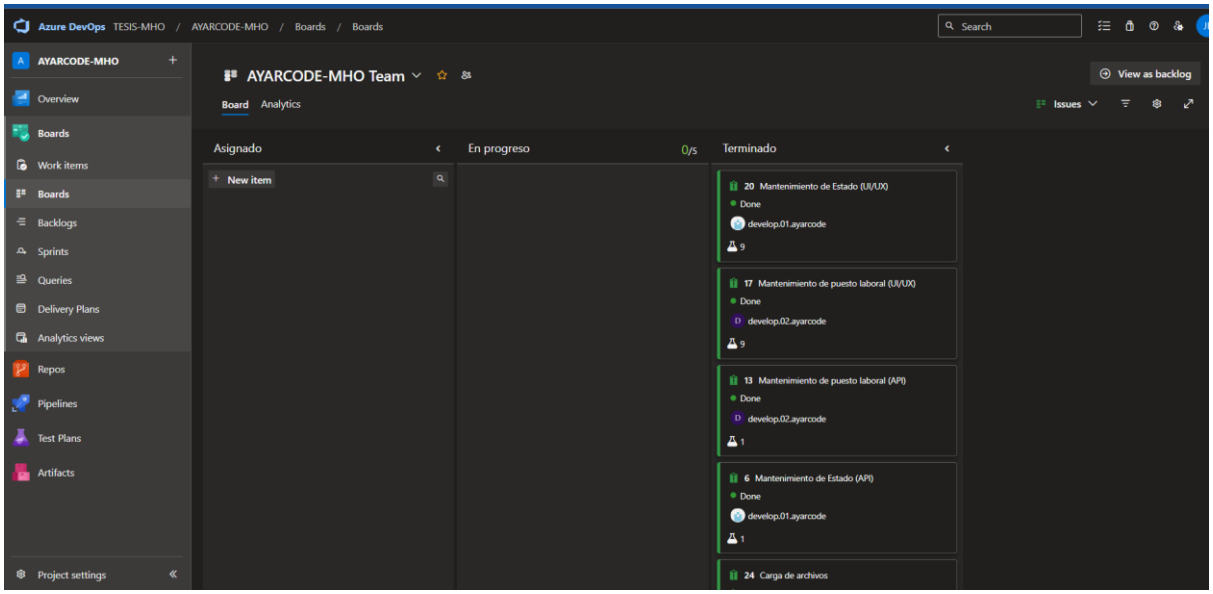


Fig. 11 Dashboard de métricas
Fuente: Elaboración propia

E) Integración continua

Se realiza la fusión de branch que contienen las funciones encargadas a los desarrolladores, se evalúa la calidad del código, se realizan testeos unitarios y se verifica que la integración no contenga errores. Al igual que en la etapa anterior, resulta fundamental mantener una

comunicación fluida para corregir cualquier problema que surja. En esta fase, se llevan a cabo las siguientes etapas:

Merge: Se integran las ramas que contienen las distintas funcionalidades, y se lleva a cabo una coordinación con el equipo para proceder con la integración y validación de las modificaciones realizadas.

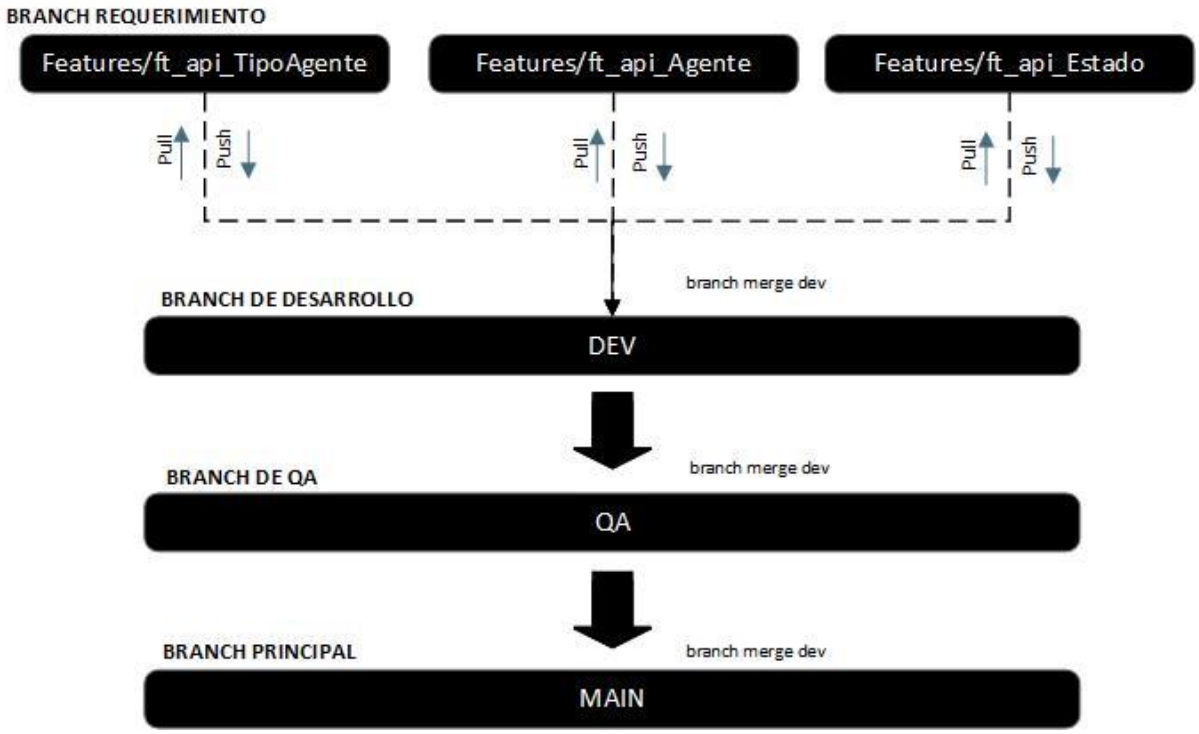


Fig. 12 Flujo de integración de Ramas
Fuente: Elaboración propia

La imagen ilustra el proceso de fusión de múltiples ramas desarrolladas con el propósito de implementar modificaciones, mejoras o nuevas características en ramas independientes, sin causar impacto en la rama principal (main). A continuación, se detalla la función prevista para cada rama:

- **Rama de requerimiento:** En esta etapa, se crean ramas según los requerimientos solicitados, generando una "copia" de la rama principal para la implementación de nuevos procesos en un entorno de producción separado.

- **Rama de desarrollo:** Esta rama engloba todos los cambios realizados por el equipo de desarrollo, trabajando exclusivamente con requerimientos validados e integrados con el equipo.
- **Rama de QA:** Aquí se llevan a cabo pruebas unitarias, funcionales y de aceptación, evaluando el cumplimiento de los criterios de aceptación definidos en las tareas del Product Backlog.
- **Branch Principal:** Esta rama contiene los cambios que aprobaron las pruebas, garantizando la calidad de las funcionalidades, y se relaciona con el entorno de producción.
- **Pipeline:** Se utiliza para automatizar el proceso de integrar código, eliminando la necesidad de realizar configuraciones manuales. Al asignar la rama a trabajar, el pipeline se ejecuta automáticamente en respuesta a cualquier Pull Request realizado en la rama.

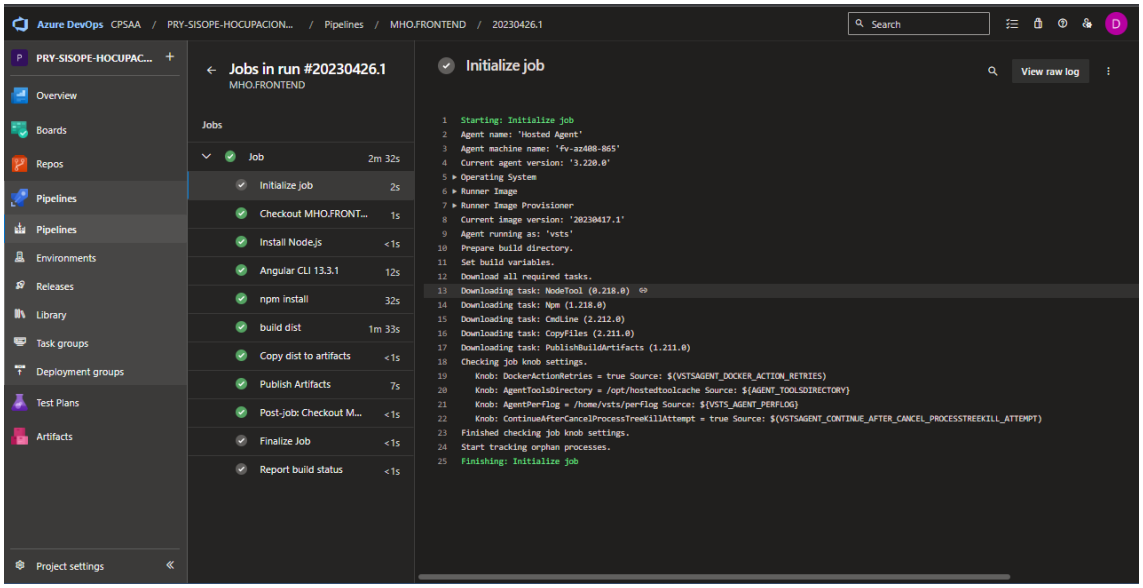


Fig. 13 Configuración de pipeline
Fuente: Elaboración propia

F) Despliegue continuo

Esta fase se concibe como una actividad de ingeniería de software que autoriza a los equipos a crear software de alta calidad en intervalos breves y a asegurar una distribución consistente en diversos entornos configurados de manera constante. En el contexto de este modelo, se destaca la priorización de la automatización en las fases de desarrollo, pruebas y

preparación para el despliegue en el entorno de producción. Para comprender este proceso, es fundamental tener en cuenta los siguientes elementos:

a) Garantizar que las variables de entorno no se guarden en la configuración de la aplicación, sino que sean de naturaleza dinámica.
b) Reservar o encriptar información confidencial, como contraseñas y credenciales de cuentas, para su posterior configuración durante el proceso de despliegue.
c) Establecer alertas y notificaciones para identificar y abordar fallos.
d) Gestionar las versiones de software a través de las herramientas de control de versiones y metodologías ágiles.
e) Automatizar el despliegue en entornos de desarrollo y pruebas.
f) Para configurar y automatizar el proceso de despliegue continuo, se crea y configura un pipeline que controla la liberación de cada entrega. Esto implica:
g) Configurar entornos: Se crean tres entornos o ambientes de servidor que replican el entorno principal para validar las funcionalidades en situaciones comunes, como se ilustra en la siguiente figura

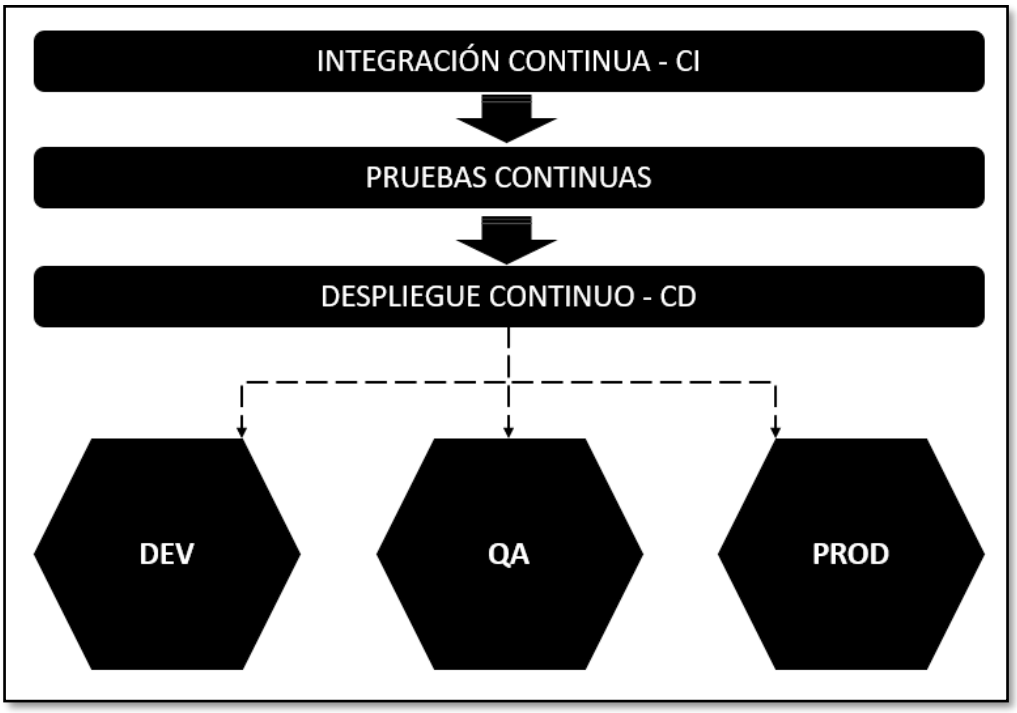


Fig. 14 Modelo de Integración

Fuente: Elaboración propia

Configuración de release

Se realiza en el portal de Azure para luego integrarlo al configurar el pipeline de liberación. Esto implica la inclusión de la rama que contiene el código necesario para el despliegue en cualquier entorno.

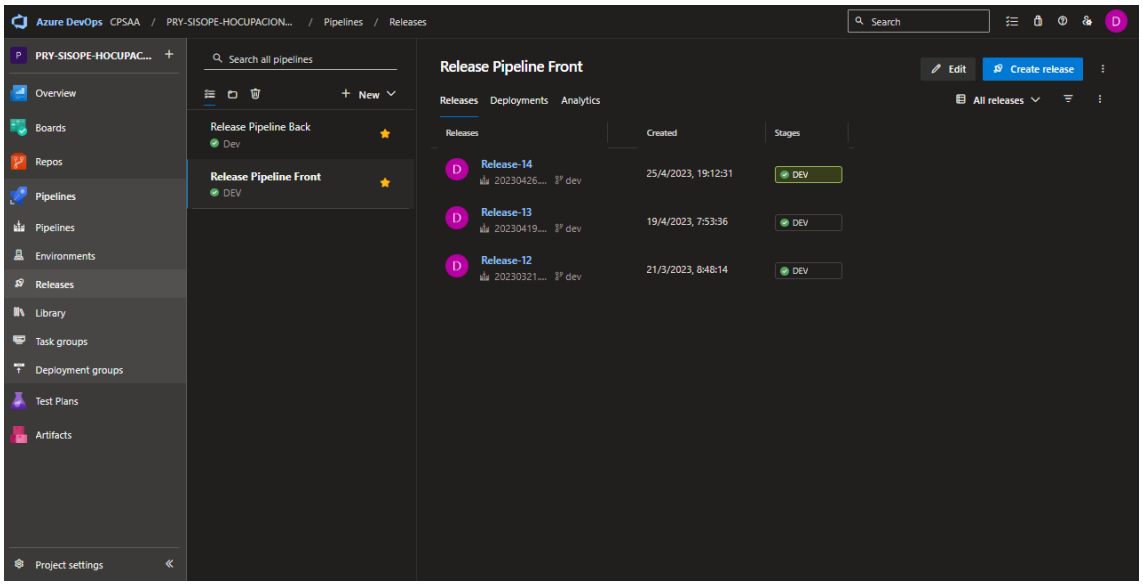


Fig. 15 Configuración de reléase
Fuente: Elaboración propia

Deploy

En la imagen a continuación, se muestra la ejecución automatizada del proceso de despliegue continuo, llevada a cabo automáticamente en los entornos de desarrollo y calidad (QA). Una vez que el entorno de calidad confirma que todas las pruebas se han completado exitosamente, se procede a realizar el despliegue manual en el entorno de producción.

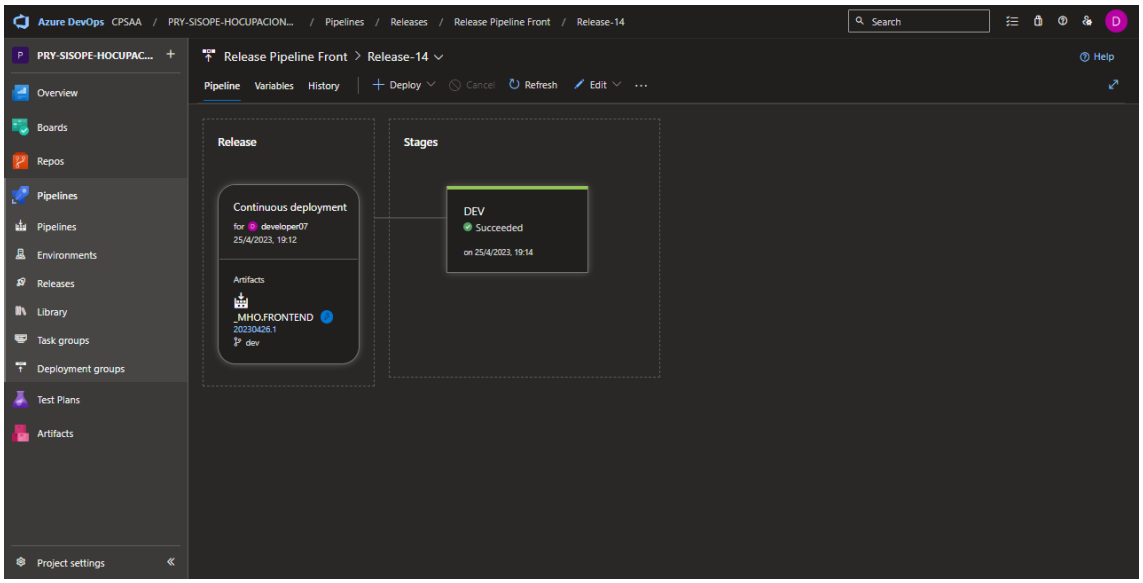


Fig. 16 Ejecución de reléase
Fuente: Elaboración propia

Monitoreo

En el enfoque propuesto, a diferencia de otros equipos donde la fase de despliegue o pase a producción marcaba el final del proceso, se entendió que esta etapa representaba una oportunidad crucial para supervisar y controlar el desarrollo posterior al despliegue. Esto permitió recopilar información valiosa sobre los resultados obtenidos, la eficiencia de las tareas asignadas, el desempeño individual de los colaboradores, la identificación de las fases críticas y la dinámica del equipo en su conjunto. Además, fue un paso esencial para continuar con el desarrollo de software.

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

Se estableció la línea base del desempeño de la empresa AYARCODE en el proceso de desarrollo de software mediante el uso de técnicas y herramientas específicas. A través de fichas de observación, se evidenció el contexto actual referente al uso y administración de los recursos empleados en la fase inicial del estudio y a través de la aplicación de encuesta se estableció el contexto inicial en cuanto a la percepción del colaborador.

Se utilizó la metodología Lean siguiendo los lineamientos del modelo CALMS de DevOps, lo que fortaleció y mejoró la colaboración y el proceso iterativo entre las áreas involucradas en el proceso de desarrollo de software. El punto de partida fue un cambio en la cultura dentro de la empresa y la adopción de este enfoque.

La implementación de DevOps ha mejorado significativamente la integración y despliegue de software en la empresa, aumentando tanto la velocidad como la calidad de estos procesos. Esto permitió alcanzar cumplir lo propuesta en la presente investigación y reafirmar la hipótesis planteada. En las etapas de desarrollo continuo, integración continua, pruebas y despliegue continuo, el modelo implementado organizó, integró y automatizó los procesos manuales, resultando en una reducción y control del tiempo necesario para el desarrollo de software.

El monitoreo continuo de las actividades se realizó con el propósito de minimizar errores comunes en las fases de implementación, integración y despliegue del software. Esto resultó en una mejora de 1.66 días en el tiempo de restauración del servicio y una reducción del 44.45% en la tasa de fallas de cambios. Este enfoque ha mejorado la eficiencia y confiabilidad del proceso, ofreciendo una mejora del 25.47% en la frecuencia de implementación, una frecuencia de liberación de código de 0.66 despliegues por sprint, y un adelanto de 1.5 días en el plazo de entrega de cambios, permitiendo entregar con mayor rapidez y calidad con un ratio de éxito del 44.45%.

4.2. Recomendaciones

Se recomienda utilizar la metodología LEAN en la implementación de DevOps debido a su enfoque de optimización de recursos. Esta metodología es especialmente adecuada para empresas pequeñas de desarrollo de software con menor complejidad, ya que su implementación es más sencilla, rápida y requiere menos recursos; de esta manera las empresas pueden obtener los beneficios de DevOps de manera más eficaz y efectiva, permitiendo una mejor integración y colaboración entre los equipos de desarrollo y operaciones. Ello se ve reflejado en una mejora en la entrega continua y eficiencia general del proceso de desarrollo de software.

Se recomienda hacer hincapié en la adopción de la cultura por parte de los colaboradores, ya que incluso si el modelo propuesto está bien organizado y estructurado, no funcionará ni dará resultados positivos si los colaboradores no se involucran, adoptan la cultura y pueden transmitirla y explicarla a los nuevos miembros del equipo, así como apoyarse mutuamente ante cualquier duda. La participación y compromiso de los colaboradores son relevantes y determinantes para el éxito de la implementación y consolidación de la cultura de DevOps. Esto implica fomentar la comunicación abierta, proporcionar capacitación y recursos adecuados, y crear un entorno de confianza y colaboración en el que los colaboradores se sientan motivados y empoderados para abrazar la cultura de DevOps. Al hacerlo, se promueve una mayor eficiencia, trabajo en equipo y éxito a largo plazo en la implementación de DevOps en la organización.

Se recomienda analizar las herramientas de gestión y desarrollo basadas en el enfoque DevOps, teniendo en cuenta el tipo de proyecto a implementar, esto permitirá elaborar un plan de acción que se ajuste a las metodologías ágiles adecuadas para el proceso de desarrollo de software, considerando el tamaño y tipo de la organización. Para optimizar el proceso de desarrollo de software, se recomienda utilizar la estrategia de repositorio Micro-repo, esta estrategia facilita el trabajo con microservicios y ayuda a diferenciar las versiones, lo que resulta en una mayor eficiencia y control en el desarrollo y despliegue de software.

REFERENCIAS

[1] S. Yaman, F. Fagerholm, M. Munezero, T. Männistö, and T. Mikkonen, “Patterns of user involvement in experiment-driven software development,” *Inf Softw Technol*, vol. 120, p. 106244, 2020, doi: 10.1016/J.INFSOF.2019.106244.

[2] INEI, “Perú: Tecnologías de Información y Comunicaciones en las Empresas,2018,” *Encuesta Económica Anual 2019*, 2019. https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1815/libro.pdf (accessed Sep. 22, 2022).

[3] A. Hermawan and L. P. Manik, “The Effect of DevOps Implementation on Teamwork Quality in Software Development,” *Journal of Information Systems Engineering and Business Intelligence*, vol. 7, no. 1, p. 84, Apr. 2021, doi: 10.20473/JISEBI.7.1.84-90.

[4] L. Manik, “Design Pattern Evaluation on A RESTful API Wrapper: A Case Study of Software Integration with An Internet Payment Gateway using Model-Driven Architecture,” *Journal of Information Technology and Computer Science*, vol. 4, no. 3, p. 222, 2019, [Online]. Available: https://www.researchgate.net/publication/338093812_Design_Pattern_Evaluation_on_A_RESTful_API_Wrapper_A_Case_Study_of_Software_Integration_with_An_Internet_Payment_Gateway_using_Model-Driven_Architecture

[5] M. A. Akbar, S. Rafi, A. A. Alsanad, S. F. Qadri, A. Alsanad, and A. Alothaim, “Toward Successful DevOps: A Decision-Making Framework,” *IEEE Access*, vol. 10, pp. 51343–51362, 2022, doi: 10.1109/ACCESS.2022.3174094.

[6] R. Stroud and E. Klavens, “DevOps Heat Map 2017 | Forrester,” 2017. https://www.forrester.com/report/DevOps-Heat-Map-2017/RES137782

[7] K. Panetta, “2017 CEO Survey Infographic,” 2017. https://www.gartner.com/smarterwithgartner/2017-ceo-survey-infographic

[8] P. Raj and P. Sinha, “Project Management In Era Of Agile And Devops Methodlogies,” *ResearchGate*, 2021. https://www.researchgate.net/publication/348338269_Project_Management_In_Era_Of_Agile_And_Devops_Methodlogies

[9] F. Almeida, J. Simões, and S. Lopes, “Exploring the Benefits of Combining DevOps and Agile,” *Future Internet 2022, Vol. 14, Page 63*, vol. 14, no. 2, p. 63, 2022, doi: 10.3390/FI14020063.

[10] S. S. Samarawickrama and I. Perera, “Continuous scrum: A framework to enhance scrum with DevOps,” *17th International Conference on Advances in ICT for Emerging Regions, ICTer 2017 - Proceedings*, vol. 2018-January, pp. 19–25, 2017, doi: 10.1109/ICTER.2017.8257808.

[11] A. Khalid, S. A. Butt, T. Jamal, and S. Gochhait, “Agile Scrum Issues at Large-Scale Distributed Projects: Scrum Project Development At Large,” *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/IJSI.2020040106*, vol. 8, no. 2, pp. 85–94, Jan. 2020, doi: 10.4018/IJSI.2020040106.

[12] M. Senapathi, J. Buchan, and H. Osman, “DevOps capabilities, practices, and challenges: Insights from a case study,” *ACM International Conference Proceeding Series*, vol. Part F137700, Jun. 2018, doi: 10.1145/3210459.3210465.

[13] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A Survey of DevOps Concepts and Challenges,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, 2019, doi: 10.1145/3359981.

[14] A. Mishra and Z. Otaiwi, “DevOps and software quality: A systematic mapping,” *Comput Sci Rev*, vol. 38, p. 100308, Nov. 2020, doi: 10.1016/J.COSREV.2020.100308.

[15] M. S. Khan, A. W. Khan, F. Khan, M. A. Khan, and T. K. Whangbo, “Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review,” *IEEE Access*, vol. 10, pp. 14339–14349, 2022, doi: 10.1109/ACCESS.2022.3145970.

[16] L. Lwakatare *et al.*, “DevOps in practice: A multiple case study of five companies,” *Inf Softw Technol*, vol. 114, pp. 217–230, 2019, doi: 10.1016/J.INFSOF.2019.06.010.

[17] N. Forsgren, J. Humble, and G. Kim, “Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations,” 2018.

[18] H. L. Truong and P. Klein, “DevOps Contract for Assuring Execution of IoT Microservices in the Edge,” *Internet of Things*, vol. 9, p. 100150, Mar. 2020, doi: 10.1016/J.IOT.2019.100150.

[19] P. Rodríguez *et al.*, “Continuous deployment of software intensive products and

services: A systematic mapping study," *Journal of Systems and Software*, vol. 123, pp. 105–113, Jan. 2016, doi: 10.1016/j.jss.2015.12.015.

[20] A. Lichtenberger, "Fünf kritische Erfolgsfaktoren für eine erfolgreiche DevOps Transformation," *HMD Praxis der Wirtschaftsinformatik*, vol. 54, no. 2, pp. 244–250, Apr. 2017, doi: 10.1365/S40702-017-0293-6.

[21] M. Waseem, P. Liang, and M. Shahin, "A Systematic Mapping Study on Microservices Architecture in DevOps," *Journal of Systems and Software*, vol. 170, p. 110798, Dec. 2020, doi: 10.1016/J.JSS.2020.110798.

[22] V. Arulkumar and R. Lathamanju, "Start to Finish Automation Achieve on Cloud with Build Channel: By DevOps Method," *Procedia Comput Sci*, vol. 165, pp. 399–405, Jan. 2019, doi: 10.1016/J.PROCS.2020.01.032.

[23] Azure, "Procedimientos recomendados de sprint y scrum," *Scrum y procedimientos recomendados*, 2023. https://learn.microsoft.com/es-es/azure/devops/boards/sprints/best-practices-scrum?view=azure-devops (accessed Apr. 25, 2023).

[24] O. H. Plant, J. van Hillegersberg, and A. Aldea, "Rethinking IT governance: Designing a framework for mitigating risk and fostering internal control in a DevOps environment," *International Journal of Accounting Information Systems*, vol. 45, p. 100560, Jun. 2022, doi: 10.1016/J.ACCINF.2022.100560.

[25] J. Nørbjerg and T. Winkler, "Closing the IT Development-Operations Gap: The DevOps Knowledge Sharing Framework," *Conference: Joint Proceedings of the BIR 2017 pre-BIR Forum, Workshops and Doctoral Consortium*, 2017, [Online]. Available: https://www.researchgate.net/publication/335272884_Closing_the_IT_Development-Operations_Gap_The_DevOps_Knowledge_Sharing_Framework

[26] N. Forsgren, D. Smith, J. Humble, and J. Frazelle, "2019 Accelerate State of DevOps Report." 2019. Accessed: May 02, 2023. [Online]. Available: https://research.google/pubs/pub48455/

[27] D. Ståhl, T. Mårtensson, and J. Bosch, "Continuous practices and Devops: Beyond the buzz, what does it all mean?," *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, vol. 2017-January, pp. 440–448, Sep. 2017, doi: 10.1109/SEAA.2017.8114695.

[28] T. Laukkarinen, K. Kuusinen, and T. Mikkonen, “DevOps in regulated software development: Case medical devices,” *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track, ICSE-NIER 2017*, pp. 15–18, Jun. 2017, doi: 10.1109/ICSE-NIER.2017.20.

[29] M. Virmani, “Understanding DevOps & bridging the gap from continuous integration to continuous delivery,” *5th International Conference on Innovative Computing Technology, INTECH 2015*, pp. 78–82, Jul. 2015, doi: 10.1109/INTECH.2015.7173368.

[30] R. Pietrantuono, A. Bertolino, G. De Angelis, B. Miranda, and S. Russo, “Towards continuous software reliability testing in DevOPs,” *Proceedings - 2019 IEEE/ACM 14th International Workshop on Automation of Software Test, AST 2019*, pp. 21–27, May 2019, doi: 10.1109/AST.2019.00009.

[31] P. Agrawal and N. Rawat, “Devops, A New Approach to Cloud Development Testing,” *IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques, ICICT 2019*, Sep. 2019, doi: 10.1109/ICICT46931.2019.8977662.

[32] ATARC, “DevOps Metrics-Performance Playbook,” 2022, Accessed: May 02, 2023. [Online]. Available: www.atarc.org|info@atarc.org

[33] P. Perera, R. Silva, and I. Perera, “Improve software quality through practicing DevOps,” *17th International Conference on Advances in ICT for Emerging Regions, ICTer 2017 - Proceedings*, pp. 13–18, 2017, doi: 10.1109/ICTER.2017.8257807.

[34] H. Baumeister, H. Lichter, and M. Riebisch, “Agile Processes in Software Engineering and Extreme Programming,” vol. 283, 2017, doi: 10.1007/978-3-319-57633-6.

[35] G. Kim, J. Humble, P. Debois, and Willis Jhon, “The DevOps Handbook: How to Create World-Class Agility, Reliability, and ... - Gene Kim, Jez Humble, Patrick Debois, John Willis - Google Libros,” 2016. https://books.google.com.pe/books?id=ui8hDgAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false (accessed May 02, 2023).

[36] L. Bass, I. Weber, L. Zhu “DevOps: A Software Architect’s Perspective” *ACM Comput Surv*, vol. 55, no. 9, Jan. 2015, doi: 10.1145/3555803.

[37] N. Govil and A. Sharma, “Validation of agile methodology as ideal software development process using Fuzzy-TOPSIS method,” *Advances in Engineering Software*, vol. 168, p.

103125, 2022, doi: 10.1016/J.ADVENGSOFT.2022.103125.

[38] C. Gonzalez, L. Pufahl, I. Weber, and J. Mendling, “Uses of business process modeling in agile software development projects,” *Inf Softw Technol*, vol. 152, p. 107028, Dec. 2022, doi: 10.1016/J.INFSOF.2022.107028.

[39] A. Alami, O. Krancher, and M. Paasivaara, “The journey to technical excellence in agile software development,” *Inf Softw Technol*, vol. 150, p. 106959, Oct. 2022, doi: 10.1016/J.INFSOF.2022.106959.

[40] C. P. Godoy, A. F. Cruz, E. P. Silva, L. M. Santos, R. S. Zerbini, and C. A. L. Pahins, “Blueprint Model: A new Approach to Scrum Agile Methodology,” *Proceedings - 2019 ACM/IEEE 14th International Conference on Global Software Engineering, ICGSE 2019*, pp. 95–99, 2019, doi: 10.1109/ICGSE.2019.00014.

[41] A. Bryant, F. García, F. Almeida, and E. Espinheira, “Adoption of Large-Scale Scrum Practices through the Use of Management 3.0,” *Informatics 2022, Vol. 9, Page 20*, vol. 9, no. 1, p. 20, 2022, doi: 10.3390/INFORMATICS9010020.

[42] C. Matthies, J. Huegle, T. Dürschmid, and R. Teusner, “Attitudes, beliefs, and development data concerning agile software development practices,” *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET 2019*, pp. 158–169, 2019, doi: 10.1109/ICSE-SEET.2019.00025.

[43] F. Soto, P. Quezada, L. Condolo, D. Moreno, and I. Rey, “Development of a methodological framework for educational innovation in the context of the tutorial action based on agile methodologies and software engineering knowledge standards,” *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, pp. 233–242, 2018, [Online]. Available: https://www.researchgate.net/publication/326029854_Development_of_a_methodological_framework_for_educational_innovation_in_the_context_of_the_tutorial_action_based_on_agile_methodologies_and_software_engineering_knowledge_standards

[44] A. Yacelga and M. Cabrera, “Use of kanban boards as support for the development of agile methodologies,” *Universidad y Sociedad*, vol. 14, no. S2, pp. 208–214, 2022, [Online]. Available: https://rus.ucf.edu.cu/index.php/rus/article/view/2760

[45] M. Hron and N. Obwegeser, “Why and how is Scrum being adapted in practice: A

systematic review," *Journal of Systems and Software*, vol. 183, p. 111110, Jan. 2022, doi: 10.1016/J.JSS.2021.111110.

[46] R. Yarlagadda, "DevOps and Its Practices," 2021. https://www.researchgate.net/publication/350006289_DevOps_and_Its_Practices

[47] B. Napoleão, É. de Souza, G. Ruiz, K. Felizardo, G. Meinerz, and N. Vijaykumar, "Synthesizing researches on Knowledge Management and Agile Software Development using the Meta-ethnography method," *Journal of Systems and Software*, vol. 178, p. 110973, 2021, doi: 10.1016/J.JSS.2021.110973.

[48] K. Curcio, R. Santana, S. Reinehr, and A. Malucelli, "Usability in agile software development: A tertiary study," *Comput Stand Interfaces*, vol. 64, pp. 61–77, 2019, doi: 10.1016/J.CSI.2018.12.003.

[49] R. Hernández, C. Fernández, and M. Baptista, "Metodología de la investigación," México, 2014. Accessed: Apr. 25, 2023. [Online]. Available: https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf

[50] D. Curbeira, M. Bravo, and C. Morales, "Diseño cuasi experimental para la formación de habilidades profesionales," *Universidad y Sociedad*, vol. 9, no. 5, pp. 24–34, Nov. 2017, Accessed: Sep. 22, 2022. [Online]. Available: https://rus.ucf.edu.cu/index.php/rus/article/view/707

[51] S. Carrasco, "Metodologia de la Investigacion Cientifica: Pautas metodologicas para diseñar y elaborar el proyecto de investigación," p. 476, 2015.

[53] D. A. Muñoz, H. A. Ordóñez, y V. Buchelli, "Lineamientos para la implementación del modelo CALMS de DevOps en mipymes desarrolladoras de software en el contexto surcolombiano", Rev. Guillermo Ockham, vol. 18, núm. 1, pp. 81–91, 2021.

[54] Atlassian, "DevOps", Atlassian. Disponible en: https://www.atlassian.com/es/devops.

[55] P. Agrawal y N. Rawat, "Devops, A new approach to cloud development & testing", en 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2019.

[56] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery", en Fifth International Conference on the Innovative Computing Technology (INTECH 2018), 2018.

[57] G. Alegre y J. Janine, “Implementación del enfoque ágil y la mejora en los proyectos de desarrollo de software en una empresa de Telecomunicaciones”, Universidad Ricardo Palma, 2019.

[58] P. A. Bejarano De la Hoz y A. M. Heredia Guerrero. “DevOps como estrategia de aporte a la calidad de productos de software en MIPYMES desarrolladoras en el contexto colombiano”. Biblioteca Digital USB - Universidad de San Buenaventura, 2019

[59] J. H. Retamal Valenzuela. “Plataforma de desarrollo de aplicaciones en el DCC basada en técnicas de DevOps”. Universidad de Chile

[60] J. P. Carvalho Faustino. “DevOps Practices in Incident Management Process”. Dissertation submitted as partial fulfilment of the requirements for the degree of Master in Computer Engineering, 2018

[61] J. Bonhomme y E. Camejo, “Plataforma de Integración basada en Microservicios”, Inst. Comput. - Fac. Ing. Univ. Republica Montev., Urug., p. 132, 2019.

[62] C. E. Orozco Garcés. “Modelo de métricas para apoyar la evaluación de DevOps en empresas de desarrollo de software”. Universidad del Cauca. 2022

ANEXOS

ANEXO 01: ACTA DE REVISIÓN DE SIMILITUD DE LA INVESTIGACIÓN

Universidad Señor de Sipán

Yo **Mejía Cabrera Heber Ivan** docente del curso de **Investigación II** del Programa de Estudios de **Ingeniería de Sistemas** y revisor de la investigación del (los) estudiante(s), Albarran Salazar Harvy Jefferson, Delgado Farro Javier Manuel, titulada:

IMPLEMENTACIÓN DEL MODELO CALMS DE DEVOPS PARA LA OPTIMIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE: CASO DE ESTUDIO EMPRESA AYARCODE

Se deja constancia que la investigación antes indicada tiene un índice de similitud del **16%**, verificable en el reporte final del análisis de originalidad mediante el software de similitud TURNITIN. Por lo que se concluye que cada una de las coincidencias detectadas no constituyen plagio y cumple con lo establecido en la Directiva sobre índice de similitud de los productos académicos y de investigación en la Universidad Señor de Sipán S.A.C., aprobada mediante Resolución de Directorio N° 145-2022/PD-USS.

En virtud de lo antes mencionado, firma:

Mejía Cabrera Heber Ivan	DNI: 41639565	

Pimentel, 19 de julio del 2023.

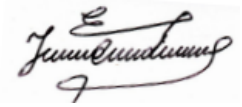
ANEXO 02: ACTA DE APROBACIÓN DEL ASESOR

Universidad Señor de Sipán

ACTA DE APROBACIÓN DEL ASESOR

Yo **Juan Carlos Arcila Diaz**, quien suscribe como asesor designado mediante Resolución de Facultad N° **0366-2023/FIAU-USS**, del proyecto de investigación titulado **IMPLEMENTACIÓN DEL MODELO CALMS DE DEVOPS PARA LA OPTIMIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE: CASO DE ESTUDIO EMPRESA AYARCODE**, desarrollado por los estudiantes: **Harvy Jefferson Albarrán Salazar, Javier Manuel Delgado Farro**, del programa de estudios de **Ingeniería de Sistemas**, acredito haber revisado, ydeclaro expedito para que continúe con los trámites pertinentes.

En virtud de lo antes mencionado, firman:

Arcila Diaz Juan Carlos (**Asesor**)	DNI: 47715777	

Pimentel, 14 de mayo de 2024

ANEXO 03: ENCUESTA REALIZADA A LOS EQUIPOS DE DESARROLLO Y OPERACIONES EN LA EMPRESA

Con el propósito de Conocer el estado actual del proceso de desarrollo de software. El éxito del proceso depende de su objetividad y colaboración.

Dimensión	Pregunta	SI	NO
Integración continua	¿Se han definido políticas, procedimientos o estrategias explícitas para llevar a cabo el control de versiones en el proyecto?		
	¿Se han definido repositorios que permitan automatizar el control de versiones en los artefactos involucrados durante el proceso de desarrollo?		
	¿El proyecto utiliza herramientas que garanticen la integridad del código fuente antes de realizar actualizaciones de cualquier tipo?		
	¿Existen procedimientos claros para la implementación y documentación de pruebas unitarias?		
	¿La compilación, revisión y despliegue del código fuente está automatizada?		
	¿Existen mecanismos que permitan identificar de manera clara cualquier error durante la integración de nuevo código fuente durante el proceso de desarrollo?		
	¿Existen mecanismos, políticas o procedimientos que permitan la recuperación de un estado estable del sistema en caso de un fallo durante el proceso de integración de nuevo código?		
	¿Los equipos de desarrollo y operaciones tienen acceso al sistema de control de versiones dispuesto para el proyecto?		
Entrega continua	¿El proyecto cuenta con uno o más ambientes de preproducción?		
	¿El proyecto cuenta con uno o más ambientes productivos?		
	¿Se han definido políticas, mecanismos o procedimientos claros que permitan identificar cuando y como se puede realizar una entrega a stage, preproducción o producción?		
	¿El proyecto utiliza herramientas que automaticen la entrega de nuevos cambios a preproducción o producción?		
	¿Los ambientes de preproducción y/o producción cuentan con un conjunto de Criterios para verificar la integridad de nuevas funcionalidades a liberar?		
Pruebas continuas	¿El proyecto cuenta con uno o más ambientes de pruebas (stage)?		
	¿Se llevan a cabo pruebas de integración que garanticen la consistencia del sistema como conjunto?		

	¿El proceso de pruebas incluye escenarios claramente definidos para garantizar la integridad de nuevos cambios aplicados al sistema?
	¿Se realizan pruebas para verificar la integridad de los requerimientos no funcionales presentes en el proyecto?
	¿Las pruebas funcionales son llevadas a cabo siguiendo un plan claramente definido?
	¿Existen políticas, mecanismos o procedimientos claros para la solución de incidentes identificados durante la etapa de pruebas funcionales y/o no funcionales?
	¿Se ha definido un espacio común que permita realizar seguimiento al proceso de pruebas?
	¿Al menos uno de los ambientes de stage dispuestos en el proyecto es un espejo de ambientes productivos?
Monitoreo y observabilidad continua	¿Se han definido procesos claros para llevar a cabo el seguimiento a los logs implementados en el proyecto?
	¿Existen mecanismos para realizar el monitoreo de la infraestructura implementada en el proyecto?
	¿Existen mecanismos para monitorear el cumplimiento de aspectos no funcionales en entornos productivos?
	¿Existen mecanismos para la detección de problemas en ambiente productivo?
Educación en torno a DevOps	¿Se han definido espacios para capacitar a las partes interesadas en DevOps?
	¿Se realiza seguimiento a la forma en la cual las partes interesadas aplican DevOps?
	¿Todas las partes interesadas saben aplicar DevOps?
Realimentación continua e innovación	¿Se han definido espacios para compartir conocimiento?
	¿Existe apertura a nuevas ideas para mejorar los procesos existentes?
	¿Se buscan nuevos mecanismos para garantizar que los procesos existentes no se quedan obsoletos?
	¿Se han definido espacios comunes (wikis) para compartir información de interés común?
	¿Se han definido espacios de retrospección para identificar aspectos de mejora de manera constante?
Medición de la cultura	¿Se han definido mecanismos para medir la cultura organizacional?
	¿Se realiza seguimiento para garantizar que existe una cultura organizacional adecuada?
	¿Las partes involucradas comprenden y aceptan la cultura presente en la organización?
	¿Se han definido procedimientos claros para llevar a cabo planes de acción en caso de ser necesarios?
Despliegue continuo	¿Se han definido procedimientos claros que garanticen la integridad de un artefacto a desplegar?

	¿Se han definido herramientas para automatizar el despliegue de artefactos?
	¿Se ha definido un proceso de orquestación que garantice el despliegue de nuevos artefactos?
	¿El proceso de despliegue requiere que una aprobación?
Satisfacción laboral	¿Existen mecanismos claros que permitan identificar el nivel de satisfacción laboral en los miembros del equipo de manera objetiva?
	¿La organización ofrece incentivos que permitan mejorar el nivel de satisfacción laboral?
	¿El ambiente de trabajo insentiva la comunicación, colaboración y coordinación asertiva?
	¿Se han definido espacios de realimentación que permitan identificar el grado de satisfacción laboral en el equipo?

ANEXO 04: FICHA TÉCNICA DE INSTRUMENTO

Nombre:

Modelo de métricas para apoyar la evaluación de DevOps en empresas de desarrollo de software

Autor:

- Carlos Eduardo Orozco

- César Jesús Pardo Calvache

Objetivo:

Proponer un modelo de métricas para apoyar la evaluación del grado de implementación de DevOps basado en un conjunto de prácticas propuestas en la literatura, y que apoye a los profesionales y empresas de software a identificar oportunidades de mejora en sus procesos de DevOps.

Metodología:

Se utilizó el método Investigación-Acción con múltiples ciclos de forma lineal. La evaluación de la propuesta fue llevada a cabo a través de un grupo focal (focus group) y un estudio de caso. A continuación, se describen los ciclos y las actividades que se llevaron a cabo para el desarrollo del proyecto

Modo de Evaluación:

Para responder las preguntas, se definió un instrumento de evaluación tipo cuestionario con dos posibles respuestas (“SI”, “NO”). La respuesta a cada pregunta se debe dar de acuerdo con los criterios descritos en la siguiente tabla

Respuesta	Descripción
SI	Se presentan las evidencias suficientes y necesarias para garantizar que el conjunto de actividades asociadas a la pregunta es llevado a cabo de

	manera correcta. Las evidencias pueden ser tomadas a partir de diferentes mecanismos, así como: (i) evidencia directa tomada mediante observación de una práctica, (ii) recopilación de opiniones por cada uno de los roles involucrados en la práctica o (iii) registros históricos consistentes que permitan evidencias el cumplimiento de la práctica.
NO	Esta respuesta se da en dos escenarios: (i) la empresa no cuenta con algún tipo de evidencia o (ii) se observa que el cumplimiento parcial de la práctica, es decir, no se cumplen todos los aspectos necesarios para garantizar que la pregunta se responde de manera completa.

Componentes:

El modelo sigue una estructura jerárquica en la cual: (i) los valores comprenden el nivel más alto de abstracción, representando el conjunto de aspectos que se deben considerar para garantizar que la cultura propuesta por DevOps se lleva a cabo adecuadamente, (ii) el segundo nivel de abstracción comprende las dimensiones que describen cada una de las actividades, roles, prácticas y herramientas requeridas para la implementación efectiva de los valores propuestos para DevOps y (iii) el tercer nivel de abstracción describe el conjunto de prácticas que se deben aplicar y/o adoptar para cumplir con cada una de las dimensiones propuestas para DevOps. El protocolo para la definición del modelo de métricas siguió el paradigma Goal, Question, Metric – GQM, el cual propone: (i) un nivel conceptual (Goal), (ii) un nivel operacional (Question) y (iii) un nivel cuantitativo (Metric). En el nivel conceptual; se identificaron las dimensiones, prácticas y valores propuestos por DevOps. En el nivel operacional; se definieron las preguntas asociadas a cada práctica de DevOps, las cuales fueron caracterizadas con base en un conjunto de objetivos asociados a cada práctica. Finalmente, en el nivel operacional; se definió un conjunto de métricas que, a partir de las respuestas a las preguntas descritas en el nivel operacional, permiten conocer el grado de implementación de las prácticas, dimensiones y valores de DevOps. Después de aplicar cada uno de los niveles propuestos por GQM, fue posible establecer un conjunto de objetivos y preguntas asociadas para las prácticas fundamentales y complementarias. Se presenta la cantidad de objetivos y preguntas resultantes después de definir el modelo de métricas

ANEXO 05: CARTA DE AUTORIZACIÓN PARA LA RECOLECCIÓN DE LA INFORMACIÓN

AyarCode

Chiclayo 16 de junio del 2023

.

CARTA DE ACEPTACIÓN

Señor:
Ing. MEJIA CABRERA HEBER IVAN
Escuela Profesional de Ingenierías de Sistemas
Universidad Señor de Sipán S.A.C.
ASUNTO: Aceptación de proyecto de investigación

Es grato dirigirme a usted mediante la presente dejo constancia que **AYARCODE CONSULTING S.A.C**. con RUC **20610186255,** debidamente representada por su Gerente **Diego Alonso Carbajal Vásquez** identificado con DNI **41575804**, acepta que los Señores: HARVY **JEFFERSON ALBARRÁN SALAZAR** con código universitario N°2181801786, identificado con el DNI 72121734 y **JAVIER MANUEL DELGADO FARRO** con código universitario N°2181801868, identificado con el DNI 70922936, estudiantes del X ciclo de la Escuela Profesional de Ingeniería de Sistemas, realicen su proyecto de investigación "Implementación del modelo CALMS de DevOps para la optimización del proceso de desarrollo de software: caso de estudio empresa Ayarcode" en la empresa **AYARCODE CONSULTING S.A.C,** de acuerdo a los recursos y el asesoramiento requerido para el cumplimiento de las actividades que le sean asignadas.

Sin otro particular me suscribo de Usted, reiterando las muestras de mi especial consideración y estima.

Ing. Diego Carbajal Vásquez
Gerente

ANEXO 06: MATRIZ DE CONSISTENCIA LÓGICA

IMPLEMENTACIÓN DEL MODELO CALMS DE DEVOPS PARA LA OPTIMIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE: CASO DE ESTUDIO EMPRESA AYARCODE

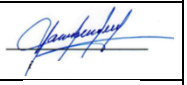
Problema	Variables	Indicadores	Población	Muestra	Método de recolección de Datos	Técnicas de procesamiento de datos
¿De qué manera se puede mejorar el proceso de desarrollo de software en la empresa AYARCODE?	**Variable Dependiente:** Optimización del proceso de desarrollo de software **Variable independiente:** Implementación del modelo CALMS de DevOps	**Optimización del proceso de desarrollo de software:** - Percepción del colaborador **Implementación de Devops:** - Frecuencia de implementación: (Número total de despliegues exitosos / Duración del período de tiempo) - Frecuencia de liberación de código: Cantidad de Despliegues - Plazo de entrega de cambios: *(Fecha de entrega acordada de un cambio específico - Fecha real de su entrega)* - Tiempo para restaurar el servicio: *(Hora de resolución - Hora de inicio del incidente)* - Tasa de fallas en los cambios: *(Número total de cambios que causaron un servicio degradado o que posteriormente requirieron reparación / Número total de*	(04) cuatro proyectos de desarrollo de software en la empresa Ayarcode	Método no probabilístico de elección por conveniencia de la investigación, considerando como muestra a (01) un proyecto de desarrollo de software	Mediante la aplicación de entrevista a los miembros de la organización y a través del registro realizado por medio de las técnicas de observación y análisis documental.	Técnicas: - Observación - Entrevista Instrumentos: -Ficha de observación - Ficha de registro - Encuesta

		cambios implementados) x 100				
Hipótesis	**Objetivo General**	**Objetivos específicos**	**Método propuesto y desarrollado**		**Resultados preliminares**	
Mediante la implementación del modelo CALMS de Devops se mejorará el proceso de desarrollo de software en la empresa AYARCODE.	Implementar el modelo CALMS de DevOps en el proceso de desarrollo de Software en la empresa AYARCODE.	1. Establecer la línea base del desempeño de la empresa AYARCODE en el proceso de desarrollo de software. 2. Seleccionar la metodología ágil y las herramientas como parte de las buenas prácticas de DevOps. 3. Implementar la metodología ágil y las herramientas como parte de las buenas prácticas de DevOps. 4. Aplicar los instrumentos para la medición del desempeño de la empresa AYARCODE en el proceso de desarrollo de software.	1.1. Selección del proyecto a evaluar. 1.2. Delimitación de las métricas a considerar. 1.3. Selección de técnicas y herramientas para realizar medición. 1.4. Aplicación del Pretest 1.5. Consolidación de resultados de línea base. 2.1. Selección del enfoque y modelo agile. 2.2. Selección de herramientas y configuraciones para implementación. 2.3. Elaboración del plan de capacitación. 3.1. Ejecución del plan de capacitación 3.2. Implementación de los lineamientos CALMS de DevOps. 3.3. Configuraciones de herramientas y coordinaciones constantes. 3.4. Organización del proyecto por tareas, tiempos y responsables. 3.5. Monitoreo del proceso. 4.1. Aplicación del Postest considerando los mismos instrumentos y métricas. 4.2. Análisis y organización de resultados. 4.3. Comparación de resultados y análisis de efectos generados.		Identificación de los cuellos de botella en el proceso de desarrollo de software existente mediante la realización de entrevistas a los miembros del equipo de desarrollo y análisis de los datos obtenidos. Selección de las herramientas DevOps adecuadas para la organización mediante la realización de una evaluación técnica y financiera de diferentes opciones. Implementación de DevOps en un proyecto piloto y medición de los resultados mediante la recopilación de métricas como el tiempo de entrega, la tasa de fallas y el tiempo de respuesta. Análisis de los resultados obtenidos y evaluación de la efectividad de la implementación de DevOps en la organización. Implementación exitosa de DevOps a favor de la mejora del proceso de desarrollo de software	

ANEXO 07: RESULTADOS DE LA ENCUESTA APLICADA A LOS EQUIPOS DE DESARROLLO Y OPERACIONES EN LA EMPRESA

A continuación, se presentan los resultados obtenidos al culminar de aplicar la encuesta (Anexo 03).

Colaboradores encuestados:

N°	Nombre y Apellido	Cargo	Firma
1	Oscar García Fuentes	Ingeniero de Software II	
2	Manuel Llontop Soplapuco	FullStack Developer	
3	Lenin Salazar Guevara	Líder de Aplicaciones	
4	Nicolaz Prado Bobadilla	FullStack Developer	
5	Diego Carbajal Vasquez	Gerente	
6	Cinthia Zufardi Patiño Delgado	Analista de QA	

Escala
a) Si (1) b) No (0)

Antes

Preguntas	Encuestado 1	Encuestado 2	Encuestado 3	Encuestado 4	Encuestado 5	Encuestado 6
P01	0	0	1	1	0	0
P02	0	0	0	0	0	0
P03	1	0	1	0	0	1
P04	0	0	0	0	0	0
P05	0	0	0	0	0	0
P06	0	0	0	0	0	0
P07	1	0	1	1	0	0
P08	1	1	0	0	1	1
P09	0	0	1	0	1	0
P10	0	0	0	0	0	0
P11	0	0	0	0	0	0

P12	0	0	0	0	0	0
P13	0	0	0	0	0	0
P14	0	0	0	0	0	0
P15	1	0	1	0	1	1
P16	0	0	0	0	0	0
P17	0	1	0	1	0	1
P18	0	0	0	0	0	0
P19	0	0	0	0	0	0
P20	0	0	0	0	0	0
P21	0	1	0	0	0	1
P22	0	0	0	0	0	0
P23	0	0	0	0	0	0
P24	0	0	0	0	0	0
P25	0	0	0	0	0	0
P26	0	0	0	0	0	0
P27	0	0	0	0	0	0
P28	0	0	0	0	0	0
P29	0	0	0	0	0	0
P30	1	1	0	1	1	1
P31	0	0	0	0	0	0
P32	0	0	0	0	0	0
P33	0	0	0	0	0	0
P34	0	0	0	0	0	0
P35	0	0	0	0	0	0
P36	0	0	0	0	0	0
P37	0	0	0	0	0	0
P38	0	0	0	0	0	0
P39	0	1	0	1	1	0
P40	0	0	0	0	0	0
P41	0	0	1	0	1	0
P42	0	0	0	0	0	0
P43	0	0	0	0	0	0
P44	0	0	1	0	0	1
P45	1	0	0	0	1	0

Después

Preguntas	Encuestado 1	Encuestado 2	Encuestado 3	Encuestado 4	Encuestado 5	Encuestado 6
P01	1	1	0	0	1	1
P02	1	1	1	1	1	1
P03	1	1	0	0	1	1
P04	1	1	1	1	1	1
P05	1	1	1	1	1	1
P06	1	1	1	1	1	1

P07	0	1	0	0	1	1
P08	0	0	1	1	0	0
P09	1	1	0	1	0	1
P10	1	1	1	1	1	1
P11	1	1	1	1	1	1
P12	1	1	1	1	1	1
P13	1	1	1	1	1	1
P14	1	1	1	1	1	1
P15	0	1	0	1	0	0
P16	1	1	1	1	1	1
P17	1	0	1	0	1	0
P18	1	1	1	1	1	1
P19	1	1	1	1	1	1
P20	1	1	1	1	1	1
P21	1	0	1	1	1	0
P22	1	1	1	1	1	1
P23	1	1	1	1	1	1
P24	1	1	1	1	1	1
P25	0	0	0	1	0	0
P26	1	0	1	0	1	0
P27	1	1	1	0	1	0
P28	1	1	1	1	1	1
P29	1	1	1	1	1	1
P30	1	1	1	1	1	1
P31	1	1	1	1	1	1
P32	1	1	1	1	1	1
P33	1	1	1	1	1	1
P34	1	1	1	1	1	1
P35	1	1	1	1	1	1
P36	0	0	1	0	0	0
P37	1	1	1	1	1	1
P38	1	1	1	1	1	1
P39	1	1	1	1	1	1
P40	1	1	1	1	1	1
P41	1	1	1	1	1	1
P42	1	1	1	1	1	1
P43	1	1	1	1	1	1
P44	1	1	1	1	1	1
P45	1	0	1	0	0	1

ANEXO 08: RESULTADOS DE LA OBSERVACIÓN A LA EMPRESA

PRETEST

FICHA DE OBSERVACIÓN - PRETEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**02-06-2023**	**INDICADOR**	Frecuencia de implementación
DESCRIPCIÓN	Número de despliegues exitosos de software realizados en un periodo de tiempo específico		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Porcentaje	FI= Frecuencia de Implementación DE= Despliegue exitoso Cycletime=DD+DT+DP DD= Días de Desarrollo DT= Días de Testing DP= Días de Despliegue	
INSTRUMENTO			
Ficha de observación			

SPRINT	TAREAS	DE	DD	DT	DP	FI
1	Mantenimiento de Usuario Login	1	4.5	2	2	11.18%
2	Mantenimiento de Empresa Mantenimiento de Sede	1	3.5	1	1	18.18%
3	Mantenimiento de Área Mantenimiento de Función	1	3	1	1	20.00%
TOTAL						16.65%

FICHA DE OBSERVACIÓN - PRETEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**02-06-2023**	**INDICADOR**	Frecuencia de liberación de Código
DESCRIPCIÓN	Evalúa la frecuencia y velocidad de los despliegues de software realizados durante un sprint		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Numérico	CD = Cantidad de Despliegues por sprint	
INSTRUMENTO			
Ficha de observación			

SPRINT	TAREAS	CD
1	Mantenimiento de Usuario Login	2
2	Mantenimiento de Empresa Mantenimiento de Sede	2
3	Mantenimiento de Área Mantenimiento de Función	1
TOTAL		1.67

FICHA DE OBSERVACIÓN - PRETEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**02-06-2023**	**INDICADOR**	Plazo de entrega de cambios
DESCRIPCIÓN	Tiempo que transcurre desde la finalización del desarrollo o la corrección hasta su implementación y disponibilidad.		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Numérico	$PEC = DAE - DEE$	
INSTRUMENTO		**PEC**= Plazo de entrega de cambios **DAE**= Días acordados para la entrega **DEE**= Días empleados para la entrega	
Ficha de observación			

ITEM	TAREAS	DAE	DEE	PEC
1	Mantenimiento de Usuario Login	6	8.5	2.5
2	Mantenimiento de Empresa Mantenimiento de Sede	5	5.5	0.5
3	Mantenimiento de Área Mantenimiento de Función	5	5	0
TOTAL				1

FICHA DE OBSERVACIÓN - PRETEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**02-06-2023**	**INDICADOR**	Tiempo para restaurar el servicio
DESCRIPCIÓN	Tiempo que el equipo necesita para reparar un servicio que ha dejado de funcionar adecuadamente		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Numérico	TR= Tiempo para restaurar el servicio	
INSTRUMENTO			
Ficha de observación			

ITEM	SPRINT	TR
1	Mantenimiento de Usuario Login	3
2	Mantenimiento de Empresa Mantenimiento de Sede	1
3	Mantenimiento de Área Mantenimiento de Función	1.5
TOTAL		1.83

FICHA DE OBSERVACIÓN - PRETEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**02-06-2023**	**INDICADOR**	Tasa de fallas en los cambios
DESCRIPCIÓN	Porcentaje de cambios realizados en una aplicación que terminan afectando su desempeño o funcionamiento		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Porcentaje	$TF = \frac{DF}{TD}$	
INSTRUMENTO		**TF**= Tasa de fallos en los cambios **TD** = DF + DE **DF**=Despliegues Fallidos **DE**=Despliegues Exitosos	
Ficha de observación			

ITEM	SPRINT	DF	DE	TD	TF
1	Mantenimiento de Usuario Login	2	1	3	66.67%
2	Mantenimiento de Empresa Mantenimiento de Sede	1	1	2	50%
3	Mantenimiento de Área Mantenimiento de Función	1	1	2	50%
TOTAL					55.56%

FICHA DE OBSERVACIÓN - PRETEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**02-06-2023**	**INDICADOR**	Ratio de Éxito
DESCRIPCIÓN	Evalúa la cantidad de código desplegado exitosamente por cada sprint		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Porcentaje	$RE = \frac{DE}{TD}$	
INSTRUMENTO		**RE**= Ratio de Éxito **DE**=Despliegues Exitosos **DF**=Despliegues Fallidos **TD** = DF + DE	
Ficha de observación			

ITEM	SPRINT	DE	DF	TD	TF
1	Mantenimiento de Usuario Login	1	2	3	33.33%
2	Mantenimiento de Empresa Mantenimiento de Sede	1	1	2	50%
3	Mantenimiento de Área Mantenimiento de Función	1	1	2	50%
TOTAL					44.44%

POSTEST

FICHA DE OBSERVACIÓN - POSTEST						
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier					
Empresa	Ayarcode SAC					
Proyecto	Higiene Ocupacional					
Fecha	**23-06-2023**	**INDICADOR**	Frecuencia de implementación			
DESCRIPCIÓN	Número de despliegues exitosos de software realizados en un periodo de tiempo específico					
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**				
Fichaje	Porcentaje	FI= Frecuencia de Implementación DE= Despliegue exitoso Cycletime=DD+DT+DP DD= Días de Desarrollo DT= Días de Testing DP= Días de Despliegue				
INSTRUMENTO						
Ficha de observación						
SPRINT	**TAREAS**	**DE**	**DD**	**DT**	**DP**	**FI**
4	Mantenimiento de Agente	2	2	1	2	40.00%
	Mantenimiento tipo de Agente					
5	Mantenimiento de Estado	2	2	1	1	50.00%
	Mantenimiento de Puesto Laboral					
6	Estudio y carga masiva	2	3.5	1	1	36.36%
	Historial de Observaciones					
TOTAL						42.12%

FICHA DE OBSERVACIÓN - POSTEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**23-06-2023**	**INDICADOR**	Frecuencia de liberación de Código
DESCRIPCIÓN	Evalúa la frecuencia y velocidad de los despliegues de software realizados durante un sprint		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Numérico	CD = Cantidad de Despliegues por sprint	
INSTRUMENTO			
Ficha de observación			
SPRINT	**TAREAS**	**CD**	
4	Mantenimiento de Agente	2	
	Mantenimiento tipo de Agente		
5	Mantenimiento de Estado	2	
	Mantenimiento de Puesto Laboral		
6	Estudio y carga masiva	3	
	Historial de Observaciones		
TOTAL		2.33	

FICHA DE OBSERVACIÓN - POSTEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**23-06-2023**	**INDICADOR**	Plazo de entrega de cambios
DESCRIPCIÓN	Tiempo que transcurre desde la finalización del desarrollo o la corrección hasta su implementación y disponibilidad.		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Numérico	$PEC = DAE - DEE$	
INSTRUMENTO		**PEC**= Plazo de entrega de cambios **DAE**= Días acordados para la entrega **DEE**= Días empleados para la entrega	
Ficha de observación			

SPRINT	TAREAS	DAE	DEE	PEC
4	Mantenimiento de Agente Mantenimiento tipo de Agente	5	4	-1
5	Mantenimiento de Estado Mantenimiento de Puesto Laboral	5	4	-1
6	Estudio y carga masiva Historial de Observaciones	5	5.5	0.5
	TOTAL			-0.5

FICHA DE OBSERVACIÓN - POSTEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**23-06-2023**	**INDICADOR**	Tiempo para restaurar el servicio
DESCRIPCIÓN	Tiempo que el equipo necesita para reparar un servicio que ha dejado de funcionar adecuadamente		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Numérico	TR= Tiempo para restaurar el servicio	
INSTRUMENTO			
Ficha de observación			

SPRINT	TAREAS	TR
4	Mantenimiento de Agente Mantenimiento tipo de Agente	0
5	Mantenimiento de Estado Mantenimiento de Puesto Laboral	0
6	Estudio y carga masiva Historial de Observaciones	0.5
	TOTAL	0.17

FICHA DE OBSERVACIÓN - POSTEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**23-06-2023**	**INDICADOR**	Tasa de fallas en los cambios
DESCRIPCIÓN	Porcentaje de cambios realizados en una aplicación que terminan afectando su desempeño o funcionamiento		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Porcentaje	$TF = \frac{DF}{TD}$	
INSTRUMENTO		**TF**= Tasa de fallos en los cambios **TD** = DF + DE **DF**=Despliegues Fallidos **DE**=Despliegues Exitosos	
Ficha de observación			

ITEM	SPRINT	DF	DE	TD	TF
4	Mantenimiento de Agente Mantenimiento tipo de Agente	0	2	2	0.0%
5	Mantenimiento de Estado Mantenimiento de Puesto Laboral	0	2	2	0.00%
6	Estudio y carga masiva Historial de Observaciones	1	2	3	33.33%
	TOTAL				11.11%

FICHA DE OBSERVACIÓN - POSTEST			
Investigadores	Albarrán Salazar Harvy – Delgado Farro Javier		
Empresa	Ayarcode SAC		
Proyecto	Higiene Ocupacional		
Fecha	**23-06-2023**	**INDICADOR**	Ratio de Éxito
DESCRIPCIÓN	Evalúa la cantidad de código desplegado exitosamente por cada sprint		
TÉCNICA	**UNIDAD DE MEDIDA**	**FÓRMULA**	
Fichaje	Porcentaje	$RE = \frac{DE}{TD}$	
INSTRUMENTO		**RE**= Ratio de Éxito **DE**=Despliegues Exitosos **DF**=Despliegues Fallidos **TD** = DF + DE	
Ficha de observación			

ITEM	SPRINT	DE	DF	TD	TF
4	Mantenimiento de Agente Mantenimiento tipo de Agente	2	0	2	100.00%
5	Mantenimiento de Estado Mantenimiento de Puesto Laboral	2	0	2	100.00%
6	Estudio y carga masiva Historial de Observaciones	2	1	3	66.67%
	TOTAL				88.89%

ANEXO 09: PLAN DE CAPACITACIÓN

PLAN DE CAPACITACIÓN PARA LA ADOPCIÓN DE LA CULTURA DEVOPS

AyarCode Consulting SAC

HOJA DE REVISIÓN Y ENTREGA

Registro de Control

Fecha	Autor	Versión	Referencia de Control
01/06/2023	Albarrán Salazár Harvy – Delgado Farro Javier	1.0	Versión vigente

Revisado y aprobado por:

Representante	Cargo	Versión	Fecha
Diego Carbajal	Gerente de AyarCode Consulting SAC	1.0	02/06/2023

INTRODUCCIÓN

La empresa Ayarcode ha sido reconocida por su capacidad de adaptación y su actitud proactiva para considerar nuevos paradigmas tecnológicos. En un mundo en constante cambio, donde la competitividad es cada vez más feroz, la empresa ha demostrado una disposición única para tomar en cuenta la innovación y la mejora continua. Esta actitud ha sido especialmente evidente en el equipo de desarrollo y operaciones, donde los profesionales han demostrado una apertura y una receptividad excepcionales para aprender y aplicar nuevas tecnologías y prácticas.Una de las tendencias más relevantes en el ámbito del desarrollo de software es DevOps. DevOps no es solo una metodología o un conjunto de herramientas, sino una filosofía que promueve la colaboración y la comunicación entre los equipos de desarrollo y operaciones. En un mundo cada vez más rápido y complejo, DevOps se ha convertido en un elemento esencial para la eficiencia y la competitividad. La empresa AyarCode ha reconocido la importancia estratégica de DevOps y ha decidido implementarlo como parte de su estrategia tecnológica.

NECESIDAD DEL PRESENTE

A pesar del compromiso y la adaptabilidad demostrados por nuestro equipo, la situación actual en el proceso de desarrollo de software revela una carencia significativa en el pilar de automatización de DevOps. Actualmente, no se implementa una automatización completa en tareas cruciales como la compilación, pruebas unitarias y funcionales, revisión de código estático y despliegue. De acuerdo con los resultados de una encuesta reciente, la falta de automatización se ha identificado como un área clave de mejora en los pilares fundamentales de DevOps, que incluyen tanto la cultura como la automatización.

Este plan de capacitación se presenta como una respuesta directa a la necesidad de abordar y fortalecer el pilar más débil identificado: la automatización en todo el ciclo de desarrollo del software. A través de la implementación de soluciones y prácticas específicas, buscamos transformar el actual proceso manual en un proceso más eficiente y ágil. Consolidando la mejora en la productividad y la calidad del software.

PLAN DE CAPACITACIÓN EN CULTURA DEVOPS

1. Objetivos

1.1. Objetivo General

Introducir y fomentar la cultura DevOps en el proceso de desarrollo de software para mejorar la colaboración, eficiencia y calidad del software entregado.

1.2. Objetivos Específicos

- Concientizar al equipo sobre los principios y valores de DevOps.
- Comprender la importancia y beneficios de la implementación de DevOps.
- Analizar el contexto y desafíos actuales en la empresa.
- Extraer lecciones aprendidas y mejores prácticas de casos exitosos de implementación de DevOps.
- Fomentar una mentalidad de mejora continua y adaptación a los cambios.
- Promover la colaboración y comunicación entre los equipos de desarrollo y operaciones.
- Identificar y abordar posibles obstáculos y resistencias a la implementación de DevOps.
- Establecer métricas para medir el éxito y el progreso de la implementación de DevOps.
- Crear un plan de acción para implementar y seguir la cultura DevOps en la empresa.

2. Alcance

El alcance de este plan de capacitación se centra en la formación y adopción de la cultura DevOps en el equipo responsable del desarrollo de software de la empresa. Este enfoque aborda tanto los aspectos teóricos como prácticos de DevOps, incluyendo la comprensión de los principios y valores, la importancia y beneficios de la implementación de DevOps, y la identificación de desafíos y obstáculos actuales en

el proceso de desarrollo de software. Además, se analizan casos de éxito de implementación de DevOps para extraer lecciones aprendidas y mejores prácticas.

3. Definiciones

- **DevOps:** Metodología que busca integrar el desarrollo de software (Dev) y las operaciones (Ops) para mejorar la colaboración y eficiencia en el ciclo de vida del desarrollo de software.
- **Principios de DevOps:** Los principios de DevOps incluyen la automatización, la colaboración, la comunicación, la transparencia, la responsabilidad y la mejora continua. Estos principios guían la implementación y adopción de DevOps en una organización.
- **Valores de DevOps:** Considera a la colaboración, la eficiencia, la calidad, la innovación, la agilidad y la mejora continua. Estos valores son fundamentales para crear una cultura DevOps en una organización y guiar las prácticas y decisiones relacionadas con el desarrollo de software.
- Componentes de DevOps**:** Incluyen la automatización de la infraestructura, la integración continua, la entrega continua, la monitorización y la retroalimentación continua. Estos componentes son esenciales para implementar y mantener una cultura DevOps en una organización.
- **Ventajas de DevOps:** Permiten una mayor eficiencia, una entrega más rápida de software, una mayor calidad del software, una mayor colaboración entre equipos, una mayor transparencia y responsabilidad, y una mejora continua en el proceso de desarrollo de software. Estas ventajas son el resultado de la implementación y adopción de DevOps en una organización.
- **Desafíos de DevOps:** Los desafíos de DevOps incluyen la resistencia al cambio, la falta de habilidades y experiencia en DevOps, la falta de herramientas y tecnologías adecuadas, la falta de colaboración y comunicación entre equipos, y la falta de métricas y KPIs para medir el éxito de DevOps. Estos desafíos son

comunes en la implementación y adopción de DevOps en una organización y deben ser abordados para lograr el éxito de DevOps.

4. Responsabilidades

- **Responsable del Proyecto DevOps:** [Evert Sanchez] - Se encarga de liderar y coordinar el proyecto DevOps, asegurando que se cumplan los objetivos y se alcance el éxito deseado. El responsable también brinda el contexto actual y alcance del proyecto a los miembros del equipo.
- **Facilitador de la Sesión de Presentación**: [Javier Delgado Farro] – Encargado de facilitar la sesión de presentación sobre la cultura DevOps, introduciendo a los miembros del equipo a los principios y valores de DevOps, así como a la importancia y beneficios de su implementación.
- **Facilitador del Estudio de Casos Exitosos:** [Harvy Albarrán Salazar] - Encargado de facilitar el estudio de casos exitosos de implementación de DevOps. Esto incluirá la revisión de casos de éxito de implementación de DevOps en otras organizaciones, con el objetivo de extraer lecciones aprendidas y mejores prácticas para aplicar en el proyecto.

5. Requisitos Generales:

Disponibilidad de los miembros del equipo durante las semanas del 22/05/23 – 02/06/23 para participar en las sesiones de concientización e implementación.

6. Descripción Textual de Capacitación: (2 semanas)

Fase 1: Concientización (Día 1 de capacitación)

- Sesión de Presentación:

Actividad a Realizar: Presentación sobre los principios y valores de DevOps.

Responsable: [Javier Delgado Farro]

Descripción Detallada:

- **Definición de DevOps:** Explicar qué es DevOps, cuál es su origen y evolución, y cómo se relaciona con la cultura organizacional y la transformación digital.
- **Principios de DevOps:** Presentar los principios fundamentales de DevOps, como la colaboración, la automatización, la medición y la mejora continua, y cómo se aplican en el contexto específico del equipo y la empresa.
- **Beneficios de DevOps:** Destacar los beneficios potenciales de la implementación de DevOps, como una mayor eficiencia, calidad del software, tiempo de entrega reducido y mejor experiencia del cliente.
- **Desafíos actuales en la empresa:** Identificar los desafíos específicos que enfrenta la empresa en su proceso de desarrollo de software y cómo DevOps puede ayudar a abordarlos.
- **Sesión interactiva:** Proporcionar un espacio para que los miembros del equipo puedan hacer preguntas, compartir sus experiencias y participar en discusiones para aclarar dudas y fomentar la participación.

➢ Estudio de Casos Exitosos:

Actividad a Realizar: Análisis de casos de éxito de empresas que han implementado DevOps.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Selección de casos relevantes:** El facilitador del estudio de casos exitosos realizará una investigación exhaustiva para identificar casos de éxito que sean relevantes y aplicables a la empresa, teniendo en cuenta su tamaño, industria y desafíos actuales.
- **Entrevistas y entrevistas en profundidad:** El facilitador del estudio de casos exitosos realizará entrevistas con representantes de las empresas seleccionadas para obtener una comprensión más profunda de los desafíos previos, las estrategias utilizadas y los resultados obtenidos en la implementación de DevOps.

- **Análisis y resumen:** El facilitador del estudio de casos exitosos analizará la información recopilada de las entrevistas y la presentará de manera clara y concisa, resaltando los principales desafíos, estrategias y resultados de cada caso.
- **•Sesión de discusión:** Después de presentar los casos de éxito, se llevará a cabo una sesión de discusión en la que los miembros del equipo podrán compartir sus opiniones, hacer preguntas y debatir sobre las lecciones aprendidas y las mejores prácticas identificadas en cada caso.
- **Documentación de conclusiones:** Finalmente, el facilitador del estudio de casos exitosos documentará todas las conclusiones y lecciones aprendidas de la actividad, para que puedan ser utilizadas como referencia futura en la implementación de DevOps en la empresa.

Fase 2: Fundamentos de DevOps (Día 2 de capacitación)

- Principios y Valores de DevOps:

Actividad a Realizar: Sesión teórica sobre los principios fundamentales de DevOps.

Responsable: [Javier Delgado Farro]

Descripción Detallada:

- **Presentación detallada de los principios clave de DevOps, incluyendo colaboración, automatización, medición y mejora continua:** El facilitador proporcionará una presentación que abarque los principios fundamentales de DevOps y cómo se relacionan con la colaboración, la automatización, la medición y la mejora continua en el desarrollo de software. Se hará énfasis en la importancia de cada uno de estos principios y cómo pueden aplicarse en el contexto específico del equipo y la empresa.
- **Discusión sobre cómo estos principios pueden aplicarse en el contexto específico del equipo y la empresa:** Después de la presentación, se facilitará

una discusión en la que los miembros del equipo podrán reflexionar sobre cómo los principios de DevOps pueden aplicarse a sus propias prácticas de desarrollo de software y cómo pueden beneficiar a la empresa en general. Se alentará a los participantes a compartir sus ideas y experiencias relacionadas con la aplicación de estos principios.

- **Ejemplos prácticos de cómo los principios de DevOps han sido implementados con éxito en otras organizaciones:** Para ilustrar la aplicación práctica de los principios de DevOps, el facilitador presentará ejemplos reales de organizaciones que han implementado con éxito estos principios en sus procesos de desarrollo de software. Se discutirán los desafíos que enfrentaron estas organizaciones y cómo superaron estos desafíos para lograr una implementación exitosa de DevOps.

Actividad a Realizar: Sesión de discusión sobre los valores centrales de DevOps.

Responsable: [Javier Delgado Farro]

Descripción Detallada:

- **Exploración de los valores culturales fundamentales de DevOps, como la colaboración, la transparencia y la responsabilidad compartida:** Durante esta sesión, se explorarán en profundidad los valores culturales fundamentales de DevOps. Se discutirán los conceptos de colaboración, transparencia y responsabilidad compartida, y se analizará cómo estos valores pueden impactar la dinámica del equipo de desarrollo y la entrega de software.
- **Discusión abierta sobre cómo estos valores pueden transformar la dinámica del equipo y mejorar la entrega de software:** Se fomentará una discusión abierta y participativa sobre cómo los valores culturales de DevOps pueden transformar la dinámica del equipo y mejorar la entrega de software. Se alentará a los

participantes a compartir sus experiencias y perspectivas sobre cómo estos valores pueden influir en su trabajo diario.

- **Ejemplos de empresas que han experimentado cambios positivos al adoptar estos valores:** Se presentarán ejemplos de empresas que han experimentado cambios positivos al adoptar los valores culturales de DevOps. Se analizarán los casos de estudio y se discutirán las lecciones aprendidas y las mejores prácticas que pueden aplicarse en el contexto de la empresa.

Esta sesión de discusión proporcionará una oportunidad para que los miembros del equipo profundicen su comprensión de los valores culturales de DevOps y reflexionen sobre cómo pueden aplicarse en su propio trabajo.

➢ Automatización y Herramientas:

Actividad a Realizar: Introducción a herramientas de automatización.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Presentación general de herramientas clave de automatización en el contexto de DevOps (CI/CD, integración continua, despliegue continuo, etc.):** El responsable de esta actividad proporcionará una visión general de las herramientas de automatización más importantes en el contexto de DevOps. Se discutirán los conceptos básicos de cada herramienta y cómo se relacionan con los principios de DevOps. Se destacarán las herramientas más relevantes para el entorno específico de la empresa.
- **Descripción detallada de cómo estas herramientas pueden mejorar la eficiencia y la calidad del proceso de desarrollo:** Se profundizará en cada herramienta, explicando cómo puede mejorar la eficiencia y la calidad del proceso de desarrollo de software. Se proporcionarán ejemplos de casos de uso y se

discutirá cómo estas herramientas pueden integrarse en el flujo de trabajo existente.

- **Identificación de las herramientas más relevantes para el entorno específico de la empresa:** Se identificarán las herramientas que son más relevantes para el entorno específico de la empresa, considerando los desafíos y las necesidades específicas del equipo de desarrollo.

Actividad a Realizar: Demostraciones prácticas de herramientas de automatización.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Demostraciones paso a paso de cómo configurar y utilizar herramientas seleccionadas:** El especialista en herramientas de automatización proporcionará demostraciones prácticas de cómo configurar y utilizar las herramientas seleccionadas. Se mostrarán ejemplos de configuraciones comunes y se guiará a los miembros del equipo a través de los procesos de configuración.
- **Ejemplos prácticos de cómo estas herramientas pueden integrarse en el flujo de trabajo existente:** Se proporcionarán ejemplos prácticos de cómo estas herramientas pueden integrarse en el flujo de trabajo existente del equipo de desarrollo. Se discutirán las mejores prácticas y se ofrecerán sugerencias para una integración efectiva.
- **Sesiones interactivas para que los miembros del equipo practiquen con las herramientas bajo la guía del especialista:** Se dedicará tiempo para que los miembros del equipo practiquen con las herramientas bajo la guía del especialista. Se alentará a los participantes a hacer preguntas y a experimentar con las herramientas en un entorno de práctica.

Fase 3: Implementación Técnica Básica: (Día 3 de capacitación)

- Configuración de Entornos de Desarrollo y Pruebas:

Actividad a Realizar: Sesión práctica de configuración de entornos de desarrollo y pruebas.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Definición de requisitos de entorno y configuración de máquinas virtuales o contenedores:** Se realizará una definición detallada de los requisitos del entorno de desarrollo y pruebas, incluyendo los recursos necesarios como máquinas virtuales o contenedores. Se configurarán estos entornos según las necesidades específicas del equipo y el proyecto.
- **Instalación y configuración de dependencias y herramientas necesarias para el desarrollo y las pruebas:** Se instalarán y configurarán las dependencias y herramientas necesarias para el desarrollo y las pruebas de software. Esto incluirá software de desarrollo, bibliotecas, frameworks y herramientas de pruebas.
- **Verificación de la conectividad y funcionalidad del entorno:** Se verificará la conectividad y funcionalidad del entorno de desarrollo y pruebas para asegurar que todo esté funcionando correctamente y listo para su uso por parte del equipo.

➢ Introducción a Herramientas de Automatización:

Actividad a Realizar: Sesión teórica y práctica sobre herramientas de automatización.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Introducción a herramientas de automatización relevantes para el desarrollo de software (CI/CD, integración continua, despliegue continuo, etc.):** Se presentará una introducción detallada a las herramientas de automatización relevantes para el desarrollo de software, incluyendo conceptos como CI/CD (Integración Continua/Despliegue Continuo) y otras herramientas de automatización de pruebas y despliegue.

- **Demostraciones prácticas de cómo configurar y utilizar estas herramientas:** Se realizarán demostraciones prácticas de cómo configurar y utilizar estas herramientas de automatización en entornos de desarrollo y pruebas. Esto incluirá la configuración de pipelines de CI/CD, integración de pruebas automatizadas, y despliegue continuo.
- **Asistencia para la instalación y configuración de las herramientas en los entornos de desarrollo:** Se brindará asistencia para la instalación y configuración de estas herramientas en los entornos de desarrollo y pruebas, asegurando que estén correctamente instaladas y configuradas para su uso por parte del equipo.

Fase 4: Desarrollo Colaborativo (Día 4 de capacitación)

➢ Implementación de Control de Versiones Colaborativo:

Entrenamiento en el Uso Efectivo de Sistemas de Control de Versiones:

Actividad a Realizar: Sesión práctica de entrenamiento en el uso de sistemas de control de versiones.

Responsable: [Javier Delgado Farro]

Descripción Detallada:

- **Introducción a los conceptos de control de versiones (Git u otro sistema utilizado):** Se realizará una introducción a los conceptos básicos de control de versiones, incluyendo cómo funciona Git y otros sistemas utilizados para el control de versiones. Se explicará cómo se gestionan los cambios en el código a lo largo del tiempo, cómo se almacenan las versiones anteriores y cómo se gestionan las ramas.
- **Creación de repositorios, clonación y manejo de ramas:** Se enseñará cómo crear repositorios en Git, cómo clonar repositorios existentes y cómo gestionar ramas para trabajar en paralelo en diferentes características o versiones del

código. Se explicará cómo crear nuevas ramas, cambiar entre ramas y fusionar ramas.

- **Resolución de conflictos y prácticas de colaboración en equipo:** Se enseñará cómo resolver conflictos que puedan surgir al trabajar en equipo en un mismo repositorio, incluyendo cómo manejar cambios concurrentes y cómo solucionar conflictos de fusión. Se explicarán las prácticas recomendadas para trabajar en equipo en un repositorio compartido, incluyendo cómo comunicar los cambios, cómo revisar el código de otros miembros del equipo y cómo gestionar las contribuciones al código compartido.

Fase 5: Integración Continua (Día 5 de capacitación)

➢ Configuración de Pipelines de CI:

Automatización de Procesos de Integración Continua:

Actividad a Realizar: Sesión teórica y práctica sobre la configuración de pipelines de CI.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Definición de pasos y flujos de trabajo para la integración continua:** Se explicará cómo definir los pasos necesarios para automatizar el proceso de integración continua, incluyendo cómo configurar la compilación del código, las pruebas unitarias, la ejecución de análisis estático de código, y otras verificaciones necesarias.
- **Configuración de pipelines utilizando herramientas específicas:** Se enseñará cómo configurar pipelines de CI utilizando herramientas específicas como Azure Pipeline. Se explicará cómo definir los pasos del pipeline, cómo configurar la integración con el repositorio de código, y cómo definir los triggers para la ejecución automática del pipeline.

- • **Automatización de pruebas unitarias, análisis estático de código y otras verificaciones:** Se demostrará cómo automatizar la ejecución de pruebas unitarias, análisis estático de código, y otras verificaciones necesarias para garantizar la calidad del código. Se explicará cómo configurar las pruebas y las verificaciones dentro del pipeline, cómo interpretar los resultados, y cómo actuar en caso de errores.

Fase 6: Despliegue Continuo (Día 6 de capacitación)

- Automatización de Despliegues:

Implementación de Prácticas de Despliegue Continuo:

Actividad a Realizar: Sesión práctica sobre la automatización de despliegues.

Responsable: [Harvy Albarrán Salazar]

Descripción Detallada:

- **Diseño e implementación de scripts de despliegue:** Se enseñará cómo diseñar e implementar scripts de despliegue automatizados que permitan desplegar la aplicación en entornos de preproducción y producción de forma rápida y segura. Se explicará cómo definir los pasos necesarios para el despliegue, cómo configurar la automatización del despliegue dentro del pipeline de CI, y cómo gestionar las variables de entorno necesarias para cada entorno.
- **Configuración de entornos de preproducción y producción:** Se explicará cómo configurar los entornos de preproducción y producción para el despliegue continuo, incluyendo cómo definir las infraestructuras necesarias, cómo configurar los servicios y las bases de datos, y cómo gestionar las variables de entorno específicas de cada entorno.
- **Pruebas de despliegue y rollbacks automáticos en caso de fallos:** Se demostrará cómo automatizar las pruebas de despliegue para garantizar que el despliegue se realice correctamente en cada entorno. Se explicará cómo

configurar las pruebas dentro del pipeline de CI, cómo interpretar los resultados, y cómo gestionar los fallos. Se explicará cómo configurar los rollbacks automáticos en caso de fallos, incluyendo cómo revertir los cambios realizados durante el despliegue y cómo notificar a los responsables.

Fase 7: Evaluación y Mejora Continua

- Monitoreo y Retroalimentación:

Implementación de Herramientas de Monitoreo y Retroalimentación Continua:

Actividad a Realizar: Sesión teórica y práctica sobre herramientas de monitoreo.

Responsable: [Javier Delgado Farro]

Descripción Detallada:

- **Introducción a herramientas de monitoreo de aplicaciones y servidores:** Se proporcionará una visión general de las herramientas disponibles para monitorear el rendimiento de las aplicaciones y servidores. Esto incluirá una descripción de las funciones y características de las herramientas más comunes utilizadas en el monitoreo, como Prometheus, Grafana, Nagios, entre otras.
- **Configuración de alertas y notificaciones:** Se explicará cómo configurar alertas y notificaciones en las herramientas de monitoreo para recibir notificaciones en tiempo real sobre problemas o eventos importantes. Esto incluirá la definición de umbrales, la selección de métricas a monitorear y la configuración de canales de notificación, como correo electrónico, SMS o Slack.
- **Análisis de métricas clave y establecimiento de KPIs:** Se analizarán las métricas clave que deben monitorearse para evaluar el rendimiento y la salud del sistema. Esto incluirá métricas relacionadas con la disponibilidad, el tiempo de respuesta, la utilización de recursos, entre otros. También se discutirá cómo establecer KPIs basados en estas métricas para medir el rendimiento y establecer objetivos de mejora.

EVIDENCIA DE LOS OBJETIVOS PLANTEADOS EN LA IMPLEMENTACIÓN DEL MODELO CALMS DE DEVOPS EN LA EMPRESA AYARCODE

1. Concientizar al equipo sobre los principios y valores de DevOps:

Evidencia:

Se realizaron sesiones de capacitación: Organizamos sesiones de capacitación donde se presentaron y discutieron los principios y valores fundamentales de DevOps. Estas sesiones incluyeron presentaciones interactivas, estudios de casos relevantes y ejercicios de grupo para fomentar la comprensión y participación del equipo.

2. Comprender la importancia y beneficios de la implementación de DevOps:

Evidencia:

Se llevaron a cabo talleres prácticos: Realizamos talleres prácticos donde el equipo pudo experimentar directamente los beneficios de la implementación de DevOps. Mediante la realización de actividades prácticas y ejercicios de simulación, los miembros del equipo pudieron comprender de manera concreta cómo DevOps mejora la eficiencia y la calidad del proceso de desarrollo de software.

3. Analizar el contexto y desafíos actuales en la empresa:

Evidencia:

Se realizaron entrevistas y encuestas: Realizamos entrevistas individuales y encuestas para recopilar información detallada sobre los desafíos actuales en la empresa en relación con el desarrollo de software. Estas entrevistas y encuestas proporcionaron una comprensión profunda de los obstáculos que enfrenta el equipo y los puntos de mejora potenciales.

4. Extraer lecciones aprendidas y mejores prácticas de casos exitosos de implementación de DevOps:

Evidencia:

Se estudiaron casos de éxito: Realizamos un análisis de casos exitosos de implementación de DevOps en empresas similares. Estos estudios de casos nos proporcionaron lecciones aprendidas y mejores prácticas que pudimos aplicar en nuestro propio contexto empresarial.

5. Fomentar una mentalidad de mejora continua y adaptación a los cambios:

Evidencia:

Se implementaron procesos de retroalimentación continua: Establecimos procesos estructurados de retroalimentación continua donde el equipo pudo proporcionar comentarios y sugerencias sobre el proceso de implementación de DevOps. Estos procesos nos permitieron identificar áreas de mejora y realizar ajustes en tiempo real.

6. Promover la colaboración y comunicación entre los equipos de desarrollo y operaciones:

Evidencia:

Se organizaron reuniones interdepartamentales: Organizamos reuniones regulares entre los equipos de desarrollo y operaciones para promover la colaboración y la comunicación. Estas reuniones proporcionaron un espacio para discutir problemas, compartir conocimientos y coordinar esfuerzos de manera efectiva.

7. Identificar y abordar posibles obstáculos y resistencias a la implementación de DevOps:

Evidencia:

Se realizaron sesiones de trabajo en equipo: Organizamos sesiones de trabajo en equipo dedicadas a identificar posibles obstáculos y resistencias a la implementación de DevOps. Estas sesiones nos permitieron abordar de manera proactiva las preocupaciones del equipo y desarrollar estrategias para superar los obstáculos identificados.

8. Establecer métricas para medir el éxito y el progreso de la implementación de DevOps:

Evidencia:

Se definieron indicadores clave de rendimiento: Definimos indicadores clave de rendimiento específicos para evaluar el éxito y el progreso de la implementación de DevOps. Incluyeron métricas relacionadas con la eficiencia, la calidad y el tiempo de entrega del software.

9. Crear un plan de acción para implementar y seguir la cultura DevOps en la empresa:

Evidencia:

Se elaboró un plan detallado de implementación: Desarrollamos un plan detallado que incluía objetivos claros, actividades específicas y plazos para la implementación de DevOps en la empresa. Este plan se basó en las necesidades y desafíos identificados durante el proceso de diagnóstico y se ajustó según sea necesario a lo largo del tiempo.

ANEXO DE DIAGNÓSTICO DE NECESIDAD DE CAPACITACIÓN

I. Evaluación del Estado Actual de DevOps en la Empresa

Cultura DevOps:

El equipo de desarrollo y operaciones ha demostrado una actitud receptiva y proactiva para adaptarse a los cambios, adquirir nuevos conocimientos tecnológicos y fomentar transformaciones culturales significativas. Esta disposición es destacable, ya que refleja la plena conciencia de la importancia de DevOps como un componente fundamental en la transformación tecnológica, tanto a nivel del mercado como de la empresa. Esta conciencia es crucial en el establecimiento de una cultura colaborativa y ágil que siente las bases para la evolución continua en el desarrollo de software.

Automatización en el Proceso de Desarrollo:

Contrariamente, en el proceso actual de desarrollo de software, se ha identificado una falta sustancial de automatización en tareas críticas como la compilación, pruebas unitarias y funcionales, revisión de código estático y despliegue. Sorprendentemente, todo el proceso de llevar el software a producción se lleva a cabo manualmente. Los resultados de la encuesta realizada confirman que los pilares fundamentales de DevOps, tanto la cultura como la automatización, presentan áreas específicas de mejora. El pilar más débil identificado es la automatización en todo el ciclo de desarrollo del software.

II. Necesidad de Desarrollar un Plan de Capacitación Específico

Razones para el Enfoque en Automatización:

La falta de automatización en el ciclo de desarrollo de software emerge como un desafío crítico que afecta la eficiencia y agilidad de nuestro equipo. Dada la importancia de DevOps en la mejora continua y la entrega rápida de software, es esencial abordar este aspecto clave para avanzar hacia nuestras metas empresariales.

Objetivos del Plan de Capacitación:

El enfoque del estudio de caso se centrará en abordar específicamente la falta de automatización identificada en todo el ciclo de desarrollo del software. El objetivo principal es implementar soluciones y prácticas que permitan automatizar y optimizar las tareas críticas, como la compilación, pruebas y despliegue. Se busca transformar un proceso actualmente manual en un sistema más eficiente y ágil, aprovechando al máximo los principios y herramientas fundamentales de DevOps.

Impacto Esperado:

Este plan de capacitación no solo busca corregir las deficiencias identificadas, sino también cultivar una cultura de colaboración y mejora continua. Se anticipa que la implementación efectiva de la automatización fortalecerá la eficiencia operativa, mejorará la calidad del software y permitirá una entrega más rápida y confiable.

Conclusión del Diagnóstico:

El diagnóstico de necesidades de capacitación destaca la urgencia de abordar la falta de automatización en el proceso de desarrollo de software. La implementación de un plan de capacitación específico en este aspecto fortalecerá no solo nuestras capacidades técnicas, sino también la cohesión y agilidad de nuestro equipo, impulsando así el éxito continuo de la empresa en un entorno tecnológico competitivo.

ANEXO 10: PLAN DE TEST

ID	Work Item Type	Title	Test Step	Step Action	Step Expected	Area Path	Assigned To	State
1	Test Case	Validación de título, descripción, filtro, servidor	1	Identificar si el título cumple con los requisitos generales de Ortografía (Mayúsculas, Minúsculas, Tíldes)	El título cumple con los requisitos generales de ortografía	Dev-Ayarcode/Pacasmayo		Design
			2	Validar que la primera letra del Título sea mayúscula	Cumple que la primera letra del título sea mayúscula			
			3	Identificar si el título está en el Margen Izquierdo	El título se encuentra al margen izquierdo			
			4	Validar tipo de filtro por descripción tipo agente	Filtro cumple la búsqueda por la descripción			
			5	Validad color de los botones editar (negro) y eliminar (rojo)	Botones cumplen con colores establecidos			
			6	Validar texto botón guardar en mayúsculas	Texto del botón guardar está en mayúsculas			
			7	Validar funcionamiento de botón guardar, editar y eliminar	Botones de guardar, editar y eliminar cumplen con su funcionamiento			
			8	Validar notificación de guardado, editado y eliminado, asi como el color del mismo conforme a lo establecido	Notificaciones y colores de los mismos cumplen con lo establecido			
			9	Validación de caída del servidor	El servidor validado que no sufra alguna caída ante el uso del software			
			10	Validación de registros vacíos	Registros validados que no se permiten NULL			

2	Test Case	Validación de título, descripción, filtro, servidor	1	Identificar si el título cumple con los requisitos generales de Ortografía (Mayúsculas, Minúsculas, Tíldes)	El título cumple con los requisitos generales de ortografía	Dev-Ayarcode/Pacasmayo		Design
			2	Validar que la primera letra del Título sea mayúscula	Cumple que la primera letra del título sea mayúscula			
			3	Identificar si el título está en el Margen Izquierdo	El título se encuentra al margen izquierdo			
			4	Validar tipo de filtro por descripción agente	Filtro cumple la búsqueda por la descripción			
			5	Validad color de los botones editar (negro) y eliminar (rojo)	Botones cumplen con colores establecidos			
			6	Validar texto botón guardar en mayúsculas	Texto del botón guardar está en mayúsculas			
			7	Validar funcionamiento de botón guardar, editar y eliminar	Botones de guardar, editar y eliminar cumplen con su funcionamiento			
			8	Validar notificación de guardado, editado y eliminado, asi como el color del mismo conforme a lo establecido	Notificaciones y colores de los mismos cumplen con lo establecido			
			9	Validar seleccionar tipo de agente	Selector de tipo agente está conforme su funcionalidad			
			10	Validación de caída del servidor	El servidor validado que no sufra alguna caída ante el uso del software			
			11	Validación de registros vacios	Registros validados que no se permiten NULL			

3	Test Case	Validación de título, descripción, filtro, servidor	1	Identificar si el título cumple con los requisitos generales de Ortografía (Mayúsculas, Minúsculas, Tíldes)	El título cumple con los requisitos generales de ortografía	Dev-Ayarcode/Pacasmayo		Design
			2	Validar que la primera letra del Título sea mayúscula	Cumple que la primera letra del título sea mayúscula			
			3	Identificar si el título está en el Margen Izquierdo	El título se encuentra al margen izquierdo			
			4	Validar tipo de filtro por descripción estado	Filtro cumple la búsqueda por la descripción			
			5	Validad color de los botones editar (negro) y eliminar (rojo)	Botones cumplen con colores establecidos			
			6	Validar texto botón guardar en mayúsculas	Texto del botón guardar está en mayúsculas			
			7	Validar funcionamiento de botón guardar, editar y eliminar	Botones de guardar, editar y eliminar cumplen con su funcionamiento			
			8	Validar notificación de guardado, editado y eliminado, asi como el color del mismo conforme a lo establecido	Notificaciones y colores de los mismos cumplen con lo establecido			
			9	Validación de caída del servidor	El servidor validado que no sufra alguna caída ante el uso del software			
			10	Validación de registros vacíos	Registros validados que no se permiten NULL			
4	Test Case	Validación de título, descripción, filtro, servidor	1	Identificar si el título cumple con los requisitos generales de Ortografía (Mayúsculas, Minúsculas, Tíldes)	El título cumple con los requisitos generales de ortografía			
			2	Validar que la primera letra del Título sea mayúscula	Cumple que la primera letra del título sea mayúscula			

			3	Identificar si el título está en el Margen Izquierdo	El título se encuentra al margen izquierdo	Dev-Ayarcode/Pacasmayo		Design
			4	Validar tipo de filtro por descripción puesto laboral	Filtro cumple la búsqueda por la descripción			
			5	Validad color de los botones editar (negro) y eliminar (rojo)	Botones cumplen con colores establecidos			
			6	Validar texto botón guardar en mayúsculas	Texto del botón guardar está en mayúsculas			
			7	Validar funcionamiento de botón guardar, editar y eliminar	Botones de guardar, editar y eliminar cumplen con su funcionamiento			
			8	Validar notificación de guardado, editado y eliminado, asi como el color del mismo conforme a lo establecido	Notificaciones y colores de los mismos cumplen con lo establecido			
			9	Validación de caída del servidor	El servidor validado que no sufra alguna caída ante el uso del software			
			10	Validación de registros vacíos	Registros validados que no se permiten NULL			
5	Test Case	Validación de título, descripción, filtro, funcionalidades, servidor	1	Identificar si el título cumple con los requisitos generales de Ortografía (Mayúsculas, Minúsculas, Tíldes)	El título cumple con los requisitos generales de ortografía	Dev-Ayarcode/Pacasmayo		Design
			2	Validar que la primera letra del Título sea mayúscula	Cumple que la primera letra del título sea mayúscula			
			3	Identificar si el título está en el Margen Izquierdo	El título se encuentra al margen izquierdo			
			4	Validar tipo de filtro por estudio	Filtro cumple la búsqueda por la descripción			

			5	Validad color de los botones editar (negro) y eliminar (rojo)	Botones cumplen con colores establecidos			
			6	Validar texto botón guardar en mayúsculas	Texto del botón guardar está en mayúsculas			
			7	Validar funcionamiento de botón editar y eliminar	Botones de guardar, editar y eliminar cumplen con su funcionamiento			
			8	Validar notificación de guardado, editado y eliminado, asi como el color del mismo conforme a lo establecido	Notificaciones y colores de los mismos cumplen con lo establecido			
			9	Validar botón de seguimiento	Botón de seguimiento validado			
			10	Validar funcionamiento de agregar nuevo estudio	Funcionalidad validada			
			11	Validar filtros por empresa y sede	Filtro validado por empresa y sede			
			12	Validar limpieza de filtros	Botón de "limpieza" de filtros validado			
			13	Validar funcionalidad de carga masiva de observaciones	Funcionalidad validada			
			14	Validar agregar nueva observación	Funcionalidad validada			
			15	Validación de caída del servidor	El servidor validado que no sufra alguna caída ante el uso del software			
			16	Validación de registros vacios	Registros validados que no se permiten NULL			
6	Test Case	Validación de título, descripción, filtro, funcionalidades, servidor	1	Identificar si el título cumple con los requisitos generales de Ortografía (Mayúsculas, Minúsculas, Tíldes)	El título cumple con los requisitos generales de ortografía	Dev-Ayarcode/Pacasmayo		Design

			2	Validar que la primera letra del Título sea mayúscula	Cumple que la primera letra del título sea mayúscula			
			3	Identificar si el título está en el Margen Izquierdo	El título se encuentra al margen izquierdo			
			4	Validar tipo de filtro por empresa, estudio, estado y fecha	Filtro cumple la búsqueda			
			5	Validar limpieza de filtros	Botón de "limpieza" de filtros validado			
			6	Validar funcionalidad de desglosar el historial de las observaciones	Funcionalidad validada			
			7	Validación de caída del servidor	El servidor validado que no sufra alguna caída ante el uso del software			
			8	Validación de registros vacíos	Registros validados que no se permiten NULL			

Reporte de similitud

NOMBRE DEL TRABAJO

ALBARRAN_SALAZAR_HARVY-DELGADO_FARRO_JAVIER_TURNITIN.docx

AUTOR

Albarragan Salazar

RECUENTO DE PALABRAS

16384 Words

RECUENTO DE CARACTERES

92663 Characters

RECUENTO DE PÁGINAS

72 Pages

TAMAÑO DEL ARCHIVO

2.2MB

FECHA DE ENTREGA

Jun 26, 2024 10:10 AM GMT-5

FECHA DEL INFORME

Jun 26, 2024 10:11 AM GMT-5

12% de similitud general

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

- 12% Base de datos de Internet
- 1% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 3% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Coincidencia baja (menos de 8 palabras)