UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



TESIS:

“Principios DevOps para el despliegue automatizado de un módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena - Ayacucho, 2025”

Para optar el título profesional de:
INGENIERO DE SISTEMAS

PRESENTADO POR:
Bach. Ronald Anderson MORENO HINOSTROZA

ASESOR:
Dr. Manuel Avelino LAGOS BARZOLA

AYACUCHO - PERÚ

2026

DEDICATORIA

A mis padres, por su amor, esfuerzo y apoyo incondicional, pilares fundamentales en mi formación personal y profesional. A mis hermanas, por su constante aliento y compañía durante este camino. Y, especialmente, a mi pareja y a mi hijo, por ser mi inspiración, mi fortaleza y la razón de cada uno de mis logros.

AGRADECIMIENTOS

Expreso mi más profundo agradecimiento a mi alma máter, la Universidad Nacional de San Cristóbal de Huamanga, por haber sido el escenario de mi formación profesional y crecimiento personal. A cada docente que, con su exigencia y dedicación, guio mi aprendizaje en esta ilustre casa superior de estudios.

De manera especial, agradezco a mi familia, por ser el motor y soporte incondicional en cada etapa de este proceso. Gracias por su confianza y por motivarme constantemente a alcanzar esta meta profesional.

RESUMEN

El presente trabajo aborda la baja eficiencia, los errores frecuentes y las demoras operativas ocasionadas por el despliegue manual del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena. Ante esta problemática, el objetivo principal fue describir la implementación de principios DevOps para automatizar este proceso mediante el diseño de un pipeline de integración continua y entrega continua (CI/CD) con aprobación manual para el despliegue en producción.

Para ello, se empleó una metodología de tipo aplicada y nivel descriptivo, con un diseño no experimental y transversal. El estudio se centró en el módulo de gestión de papeletas, analizando la infraestructura tecnológica y recopilando información mediante las técnicas de análisis documental y observación directa.

Tras implementar la arquitectura DevOps y evaluar el proceso bajo el modelo CALMS, los resultados evidenciaron mejoras en los indicadores operativos del proceso de despliegue. Se observó una reducción del tiempo promedio de despliegue de 168 a 37 minutos en el escenario promedio y de 276 a 55 minutos en el escenario de máxima carga operativa, además de la incorporación de mecanismos automatizados para el control de calidad, la trazabilidad y la gestión del proceso de despliegue.

Como conclusión técnica, se redujo la intervención manual, se fortaleció la trazabilidad del proceso y se incorporaron controles automatizados de calidad, consolidando un proceso de despliegue más ágil, estandarizado y replicable para la Cooperativa Santa María Magdalena.

Palabras clave: DevOps, automatización, CI/CD, microservicios, marco CALMS, transformación digital, cooperativas.

ABSTRACT

This work addresses the low efficiency, frequent errors, and operational delays caused by the manual deployment of the microservices-based ballot management module at the Santa María Magdalena Cooperative. Given this issue, the main objective was to describe the implementation of DevOps principles to automate this process thru the design of a continuous integration and continuous delivery (CI/CD) pipeline with manual approval for production deployment.

For this purpose, an applied methodology with a descriptive level was employed, with a non-experimental and cross-sectional design. The study focused on the ticket management module, analyzing the technological infrastructure and gathering information thru documentary analysis and direct observation techniques.

After implementing the DevOps architecture and evaluating the process under the CALMS model, the results showed improvements in the operational indicators of the deployment process. A reduction in the average deployment time from 168 to 37 minutes in the average scenario and from 276 to 55 minutes in the maximum operational load scenario was observed, in addition to the incorporation of automated mechanisms for quality control, traceability, and deployment process management.

As a technical conclusion, manual intervention was reduced, process traceability was strengthened, and automated quality controls were incorporated, consolidating a more agile, standardized, and replicable deployment process for the Santa María Magdalena Cooperative.

Keywords: DevOps, automation, CI/CD, microservices, CALMS framework, digital transformation, cooperatives.

CONTENIDO

DEDICATORIA ii
AGRADECIMIENTOS iii
RESUMEN iv
ABSTRACT v
LISTA DE FIGURAS xi
LISTA DE TABLAS xii
INTRODUCCIÓN xiii
CAPÍTULO I 15
PLANTEAMIENTO DEL PROBLEMA 15
1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA 15
1.2. FORMULACIÓN DEL PROBLEMA 16
1.2.1. Problema Principal 16
1.3. OBJETIVOS DE LA INVESTIGACIÓN 17
1.3.1. Objetivo General 17
1.3.2. Objetivos Específicos 17
1.4. HIPÓTESIS DE LA INVESTIGACIÓN 17
1.5. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN 18
1.5.1. Importancia 18
1.5.2. Justificación 18
1.5.2.1. Justificación Social 18
1.5.2.2. Justificación Económica 18
1.5.2.3. Justificación Técnica 19
1.5.3. Limitaciones de la Investigación 19
CAPÍTULO II 21
MARCO TEÓRICO 21
2.1. ANTECEDENTES DE LA INVESTIGACIÓN 21

2.1.1. A Nivel Internacional 21
2.1.2. A Nivel Nacional 23
2.2. MARCO TEÓRICO 26
2.2.1. DevOps 26
2.2.1.1. Definición de DevOps. 26
2.2.1.2. Objetivos de DevOps. 27
2.2.1.3. Relación entre desarrollo y operaciones. 27
2.2.1.4. Ciclo de vida DevOps. 28
2.2.2. Modelo CALMS 30
2.2.2.1. Definición del modelo CALMS 30
2.2.2.2. Ventajas de la metodología CALMS 33
2.2.2.3. Desventajas y Retos 33
2.2.2.4. Implementación 34
2.2.3. Integración y Entrega Continua (CI/CD) 35
2.2.3.1. Definición de CI/CD 35
2.2.3.2. Integración Continua (Continuous Integration) 36
2.2.3.3. Entrega continua (Continuous Delivery) 36
2.2.3.4. Pipelines 37
2.2.4. Arquitectura basada en Microservicios 38
2.2.4.1. Definición de Microservicio 38
2.2.4.2. Características de la arquitectura de microservicios 39
2.2.4.3. Relación entre microservicios y DevOps 39
2.2.5. Infraestructura como Código (IaC) y Automatización 42
2.2.5.1. Definición de Infraestructura como Código (IaC) 42
2.2.5.2. Automatización en entornos DevOps 43
2.2.5.3. Beneficios de IaC en los despliegues 43
2.2.5.4. Relación entre IaC y Ansible 44

2.2.5.5. Aplicación de IaC en la arquitectura de microservicios de la Cooperativa Santa María Magdalena 45
2.2.6. Herramientas utilizadas en la solución 45
2.2.6.1. Jenkins 45
2.2.6.2. SonarQube 47
2.2.6.3. GitLab 48
2.2.6.4. Ansible 48
2.2.7. Métricas operativas de despliegue 49
2.2.7.1. Tiempo de despliegue 49
2.2.7.2. Nivel de automatización 50
2.2.7.3. Trazabilidad del proceso de despliegue 50
2.2.7.4. Control de calidad del software 51
CAPÍTULO III 52
MATERIAL Y MÉTODOS 52
3.1. TIPO DE INVESTIGACIÓN 52
3.2. NIVEL DE INVESTIGACIÓN 52
3.3. DISEÑO DE LA INVESTIGACIÓN 53
3.4. POBLACIÓN Y MUESTRA 53
3.5. VARIABLES E INDICADORES 54
3.5.1. Definición Conceptual de las Variables 54
3.5.2. Definición Operacional de las Variables 55
3.6. TÉCNICAS E INSTRUMENTOS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN 60
3.6.1. Técnicas para Recolectar Información 60
3.6.2. Instrumentos para Recolectar Información 60
3.6.3. Herramientas para el Tratamiento de Datos e Información 61
3.6.4. Diseño Estadístico 61
3.6.5. Análisis e Interpretación de Datos 62

3.6.6. Técnicas para Aplicar el Marco de Trabajo CALMS 63
3.7. PROCEDIMIENTO METODOLÓGICO PARA LA IMPLEMENTACIÓN DE LA SOLUCIÓN DEVOPS 64
3.7.1. Diagnóstico del proceso actual de despliegue 64
3.7.2. Levantamiento de la línea base del despliegue manual 65
3.7.3. Diseño de la arquitectura DevOps propuesta 65
3.7.4. Configuración de seguridad y acceso entre servidores 68
3.7.5. Configuración de GitLab y estrategia de ramas 70
3.7.6. Configuración del pipeline CI/CD con Jenkins 72
3.7.7. Integración de control de calidad con SonarQube 74
3.7.8. Automatización del despliegue con Ansible 75
3.7.9. Aplicación del modelo CALMS como marco de análisis 78
3.7.10. Procedimiento de medición y análisis descriptivo 78
CAPÍTULO IV 81
RESULTADOS Y DISCUSIÓN 81
4.1. RESULTADOS DE LA INVESTIGACION 81
4.1.1. Resultados de la línea base del despliegue manual 81
4.1.2. Resultados de la implementación de la solución DevOps 86
4.1.3. Resultados de la Implementación por Dimensiones (Modelo CALMS) 90
4.1.4. Resultados comparativos del despliegue manual y automatizado 96
4.1.5. Resultados respecto al objetivo general 99
4.2. DISCUSIÓN DE RESULTADOS 100
4.2.1. Discusión de los resultados operativos observados 100
4.2.2. Discusión de los controles de calidad y seguridad incorporados 101
4.2.3. Contraste de resultados con la literatura científica 101
CAPÍTULO V 104
CONCLUSIONES Y RECOMENDACIONES 104

5.1. CONCLUSIONES 104
5.1.1. Conclusión general 104
5.1.2. Conclusiones específicas 104
5.2. RECOMENDACIONES 105
BIBLIOGRAFÍA 107
ANEXOS 111
ANEXO A: MATRIZ DE CONSISTENCIA 111
ANEXO B: MATRIZ DE OPERACIONALIZACIÓN DE VARIABLES 113
ANEXO C: FICHA DE OBSERVACIÓN DIRECTA 115
ANEXO C.1 REGISTRO DE OBSERVACIÓN DEL PIPELINE 116
ANEXO D.1: APLICACIÓN DE LA FICHA DE ANALISIS DOCUMENTAL 120
ANEXO E: FICHA DE REGISTRO DE EJECUCIÓN DEL PIPELINE CI/CD 122
ANEXO E.1: APLICACIÓN DE LA FICHA DE REGISTRO DEL PIPELINE 123
ANEXO F. EVIDENCIAS DE IMPLEMENTACIÓN DE PRINCIPIOS DEVOPS .. 124
ANEXO G. REGISTROS DE LÍNEA BASE DEL DESPLIEGUE MANUAL 132
ANEXO H. RESULTADOS DEL DESPLIEGUE AUTOMATIZADO 135

LISTA DE FIGURAS

Figura 1 *Ciclo de vida DevOps* *29*
Figura 2 *Dimensiones del modelo CALMS* *31*
Figura 3 *Esquema conceptual principios DevOps, herramientas y microservicios* *34*
Figura 4 *Arquitectura basada en microservicios* *40*
Figura 5 *Arquitectura del ecosistema DevOps para la gestión de papeletas* *66*
Figura 6 *Infraestructura del entorno de la organización* *68*
Figura 7 *Configuración ssh y token* *69*
Figura 8 *Repositorio de GitLab de los proyectos* *70*
Figura 9 *Flujo de proceso de rama de desarrollo a la rama Calidad* *70*
Figura 10 *Flujo de proceso de rama calidad a la rama master* *71*
Figura 11 *Flujo de despliegue a producción* *71*
Figura 12 *Configuración de webhook por microservicio* *73*
Figura 13 *Configuración de la tarea Multibranch Pipeline en Jenkins* *73*
Figura 14 *Quality Gate en SonarQube* *75*
Figura 15 *Verificación de conexión de Ansible* *76*
Figura 16 *Configuración de Playbook de Ansible* *77*
Figura 17 *Resultado de validación de calidad mediante Quality Gate en SonarQube* *87*
Figura 18 *Panel principal de pipelines en Jenkins* *88*
Figura 19 *Ejecución de pipeline Multibranch Pipeline para el microservicio de configuración* *88*
Figura 20 *Estado de ejecución de ramas dentro del pipeline Jenkins blue* *89*
Figura 21 *Gráfico de barras de Tiempo de Despliegue Manual vs Tiempo de Despliegue DevOps* *98*

LISTA DE TABLAS

Tabla 1 *Definición Operacional de las Variables* *58*
Tabla 2 *Especificación de la Infraestructura Implementada* *67*
Tabla 3 *Tiempo de despliegue en un microservicio* *82*
Tabla 4 *Tiempo promedio de despliegue por proyecto (entorno pruebas)* *83*
Tabla 5 *Tiempo promedio de despliegue por proyecto (entorno de producción)* *83*
Tabla 6 *Tiempo máximo promedio de despliegue en entorno de pruebas.* *84*
Tabla 7 *Tiempo máximo promedio de despliegue en entorno de producción* *85*
Tabla 8 *Tiempo de Despliegue Manual vs Tiempo de Despliegue DevOps* *96*

INTRODUCCIÓN

En la actualidad, la transformación digital ha impulsado a las instituciones financieras a optimizar sus procesos de entrega de software para mantener la continuidad operativa y la competitividad. En este escenario, la adopción de la cultura DevOps ha emergido como un estándar industrial, permitiendo la integración de los equipos de desarrollo y operaciones mediante la automatización, con el fin de ofrecer productos de software de alta calidad de manera ágil y resiliente.

En el contexto de la Cooperativa de Ahorro y Crédito Santa María Magdalena (CACSMM) en Ayacucho, el proceso de despliegue del ecosistema de microservicios para la gestión de papeletas se ha ejecutado bajo un esquema tradicional y artesanal. Esta metodología manual demanda una elevada inversión de tiempo y esfuerzo técnico, lo que genera cuellos de botella operativos y riesgos críticos de error humano. En escenarios de alta complejidad técnica, los tiempos de despliegue acumulados pueden alcanzar los 276 minutos (4.6 horas) de atención técnica dedicada, limitando la agilidad del área de TI. La ausencia de un pipeline automatizado y de mecanismos de control de calidad estandarizados no solo compromete la eficiencia institucional, sino que pone en riesgo la estabilidad de los servicios financieros críticos y la continuidad operativa de la cooperativa.

La presente investigación propone la implementación de principios DevOps para automatizar el ciclo de vida del software en la cooperativa. Mediante el diseño de una arquitectura de microservicios y la configuración de un pipeline de Integración y Entrega Continua (CI/CD) utilizando herramientas como Jenkins, GitLab, SonarQube y Ansible se busca transformar un proceso manual en un flujo automatizado. Esta transición permite reducir significativamente los tiempos de puesta en producción, garantizar la consistencia entre los entornos de pruebas y producción, y elevar los estándares de seguridad mediante

el uso de infraestructura como código (IaC).

El estudio se fundamenta en una metodología de tipo aplicada, con un nivel descriptivo y un diseño no experimental y transversal. A través de la aplicación del modelo CALMS (Cultura, Automatización, Lean, Medición y Compartir), se describen los beneficios operativos observados y las mejoras registradas, estableciendo una comparación técnica entre la línea base manual y el escenario automatizado.

Finalmente, esta tesis no solo busca resolver un problema técnico de eficiencia operativa, sino que también pretende servir como una guía metodológica para otras instituciones locales que deseen modernizar su infraestructura tecnológica bajo principios de DevOps, contribuyendo así al fortalecimiento de la ingeniería de sistemas en la región de Ayacucho.

CAPÍTULO I
PLANTEAMIENTO DEL PROBLEMA

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

Aunque la adopción de nuevas filosofías tecnológicas y arquitecturas de microservicios se plantea como una solución global para agilizar el desarrollo de software, el verdadero desafío organizativo se encuentra al analizar la infraestructura a nivel local. En el caso específico de la Cooperativa Santa María Magdalena - Ayacucho, el problema central radica en el proceso de despliegue de su módulo de gestión de papeletas, el cual se ejecuta actualmente bajo un esquema 100% manual.

Esta metodología de procedimiento manual no estandarizado presenta deficiencias críticas que bloquean la agilidad del área de Tecnologías de la Información (TI). Al carecer de automatización, la liberación de nuevas versiones exige la intervención directa y obligatoria del personal técnico para tareas sumamente mecánicas. Esto incluye desde la compilación local de los proyectos, pasando por la transferencia manual de archivos a los servidores a través de protocolos SFTP (empleando herramientas como WinSCP), hasta la ejecución de comandos mediante acceso remoto para detener e iniciar los servicios.

El levantamiento de información de la línea base evidencia la gravedad de esta problemática operativa con datos muy concretos:

- **Tiempos de despliegue excesivos:** En un escenario promedio que requiere actualizar solo tres microservicios y el *frontend*, el tiempo operativo acumulado entre los entornos de prueba y producción asciende a 168 minutos (2.8 horas). En

situaciones de actualización integral del ecosistema, este proceso consume hasta 276 minutos (4.6 horas) de atención técnica exclusiva.

- **Falta de trazabilidad:** El proceso carece de un registro sistematizado que permita auditar de forma automática qué versión se ha desplegado, quién lo hizo o cuándo, complicando enormemente las labores de soporte técnico y auditoría.
- **Alta dependencia y riesgo continuo:** Cualquier fallo detectado durante la puesta en marcha obliga al equipo a realizar un *rollback* (retorno a la versión anterior) de forma totalmente manual, lo que prolonga las caídas del sistema y afecta directamente la atención a los socios de la institución.

Esta acumulación de ineficiencias genera constantes cuellos de botella, forzando a que las actualizaciones deban programarse en horarios de baja actividad (madrugadas o fines de semana) e incrementando paulatinamente la deuda técnica del sistema.

Frente a este panorama, se hace indispensable la implementación de principios DevOps para automatizar este flujo de despliegue. Al integrar *pipelines* de entrega continua y despliegues orquestados, la cooperativa podría reducir significativamente la intervención manual, reducir sus tiempos de entrega y garantizar un servicio financiero estable y rastreable.

1.2. FORMULACIÓN DEL PROBLEMA

1.2.1. Problema Principal

¿Cómo se implementan los principios DevOps para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025?

Problemas Específicos

A. ¿Cuál es el diseño de una arquitectura DevOps adecuada para el despliegue del módulo de gestión de papeletas?

B. ¿Cuáles son los componentes necesarios para un proceso de integración y entrega

continua (CI/CD) para el despliegue del módulo de gestión de papeletas?

C. ¿Cuáles son los mecanismos de automatización empleados en el despliegue del módulo de gestión de papeletas?

D. ¿Qué beneficios operativos se observan en el proceso de despliegue del módulo tras la implementación de principios DevOps?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. Objetivo General

Describir la implementación de principios DevOps en el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena- Ayacucho, 2025.

1.3.2. Objetivos Específicos

A. Diseñar una arquitectura DevOps adecuada para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios.

B. Definir los componentes del pipeline CI/CD para automatizar el despliegue del módulo de gestión de papeletas.

C. Caracterizar los mecanismos de automatización empleados en el despliegue del módulo de gestión de papeletas.

D. Describir los beneficios operativos observados en el proceso de despliegue del módulo de gestión de papeletas tras la implementación de principios DevOps.

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

"Las investigaciones cuantitativas que formulan hipótesis son aquellas cuyo planteamiento define que su alcance será correlacional o explicativo, o las que tienen un alcance descriptivo, pero que intentan pronosticar una cifra o un hecho" (Hernández et al., 2014).

Se formulan hipótesis en aquellas investigaciones donde se busca probar una suposición, el impacto que causa una variable sobre otra o entre sí, en estudios en los que

se busca evidenciar la relación causa-efecto (Bernal, 2016).

La investigación no formula hipótesis porque su propósito es describir la implementación técnica y los indicadores operativos asociados al despliegue automatizado, sin probar relaciones causales entre variables.

1.5. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.5.1. Importancia

La investigación es de gran importancia porque aborda una necesidad crítica en la Cooperativa Santa María Magdalena: la optimización y modernización de sus procesos de entrega de software. Actualmente, el despliegue del módulo de gestión de papeletas es manual, lo que genera demoras significativas, errores frecuentes y baja eficiencia operativa. Al implementar principios DevOps, el estudio busca sentar un precedente práctico y un modelo replicable para futuras aplicaciones de estas metodologías en el sector cooperativo financiero. Su contribución radica en impulsar la transformación digital y mejorar la calidad de los servicios tecnológicos de la institución.

1.5.2. Justificación

1.5.2.1.Justificación Social

El estudio busca mejorar la calidad y rapidez de los servicios tecnológicos para el personal y los socios de la cooperativa. Al reducir los fallos en producción y el tiempo de despliegue, se espera favorecer la continuidad operativa y la eficiencia del área de TI, beneficiando por extensión la fiabilidad del servicio al usuario final.

1.5.2.2.Justificación Económica

La automatización del despliegue permite reducir el tiempo invertido por el personal técnico en actividades operativas repetitivas, optimizando el uso de los recursos institucionales. La disminución de errores durante la puesta en producción también contribuye a evitar costos asociados a interrupciones del servicio, correcciones de emergencia y retrabajos, generando una utilización más eficiente de la infraestructura tecnológica existente.

1.5.2.3.Justificación Técnica

Técnicamente, la investigación es crucial para superar la ausencia de pipelines de integración y entrega continua (CI/CD) y la dependencia del despliegue manual. Se propone el diseño de una arquitectura DevOps adaptada, la caracterización de los mecanismos de automatización (como la Infraestructura como Código, IaC, a través de Ansible) y la definición de los componentes CI/CD necesarios para el despliegue del módulo basado en microservicios. Esto permitirá la estandarización y trazabilidad de los procesos.

1.5.3. Limitaciones de la Investigación

La investigación está delimitada en su alcance para garantizar su viabilidad y precisión:

- **Tipo y Nivel:** El estudio es de tipo aplicada y de nivel descriptivo. Esto implica que su alcance se limita a describir la implementación de los principios DevOps y sus beneficios operativos observados, sin pretender probar efectos, relaciones causales o establecer pronósticos (por lo cual, no se formula hipótesis).
- **Alcance Funcional:** Se centra exclusivamente en la implementación y el despliegue automatizado de un solo componente: el módulo de gestión de papeletas.
- **Población y Muestra:** La población de estudio se limita a la infraestructura tecnológica y el Área de TI de la cooperativa, y la muestra se restringe al módulo de papeletas, 3 desarrolladores y 2 responsables del área de TI.
- **Límites de Evaluación:** El estudio se enfoca netamente en la mejora técnica y operativa del proceso de despliegue por parte del área de TI. Debido a esto, no evalúa el impacto financiero que supone la adopción de estas tecnologías, ni mide la satisfacción de los socios externos o clientes finales de la cooperativa.
- **Límites Técnicos:** Aunque se describe la implementación de un pipeline CI/CD

robusto, la investigación no valida el rendimiento del sistema bajo pruebas de carga o de estrés, centrándose primordialmente en la eficiencia de la entrega continua y la estandarización del código.

CAPÍTULO II
MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

2.1.1. A Nivel Internacional

Giménez Marco (2016) investigó la aplicación de principios DevOps para automatizar la gestión de incidencias en el ciclo de vida del software, utilizando una metodología de estudio de caso y análisis documental apoyado en herramientas CI/CD como Jenkins y Docker. Los resultados demostraron una reducción considerable del tiempo necesario para resolver incidencias y una mejora en la integración entre los equipos de desarrollo y operaciones. Este estudio aporta a la presente investigación evidencia temprana y práctica sobre cómo la adopción de herramientas de orquestación reduce drásticamente los tiempos operativos y errores manuales en entornos corporativos.

Forsgren et al. (2022), en el informe ACCELERATE, analizaron el nivel de maduración DevOps evaluando a más de 1,200 empresas mediante encuestas cuantitativas enfocadas en tiempos de despliegue y automatización. Sus hallazgos concluyeron que las organizaciones con alta madurez DevOps y pipelines automatizados reducen el tiempo de entrega en un 75% y presentan un 50% menos de incidentes en producción. Su aporte directo a la tesis radica en establecer la justificación de que integrar automatización y cultura colaborativa es el estándar global para maximizar la eficiencia y mitigar riesgos técnicos.

López Peña (2021) y Cusco Mejía (2022) desarrollaron estudios enfocados en

infraestructuras distribuidas, planteando el diseño de arquitecturas basadas en microservicios, contenedores y pipelines automatizados. Mediante pruebas de despliegue en escenarios reales e infraestructuras como código, ambos autores validaron que la combinación de DevOps y microservicios mejora sustancialmente la escalabilidad, la confiabilidad y el tiempo de integración de los sistemas. Estos trabajos sustentan metodológicamente el diseño de la arquitectura distribuida y el uso de automatización para gestionar múltiples servicios de forma independiente, aplicable al módulo de papeletas.

Armando-Muñoz et al. (2020) formularon lineamientos para implementar el modelo CALMS de DevOps en empresas desarrolladoras de software, aplicando un estudio de campo centrado en Cultura, Automatización, Lean, Medición y Compartir. Los resultados reflejaron que estas prácticas permitieron evolucionar de despliegues semanales a diarios con un 80% de éxito. Esta investigación provee el marco teórico y evaluativo principal para estructurar y medir las dimensiones operativas (CALMS) tras la intervención en la cooperativa.

Cusco Mejía, B. F. (2022), en su tesis “Desarrollo e implementación de una arquitectura DevOps para un sistema web basado en microservicios en infraestructuras basadas en código”, tuvo como propósito demostrar cómo la combinación de DevOps y microservicios mejora la escalabilidad y confiabilidad en sistemas web. El origen de la investigación se basó en la necesidad de gestionar infraestructuras distribuidas y crecientes demandas de los usuarios. La metodología incluyó el diseño de una arquitectura basada en contenedores Docker, orquestadores Kubernetes y pipelines automatizados para el despliegue. Los hallazgos confirmaron que la infraestructura como código permite garantizar escalabilidad, confiabilidad y eficiencia en la entrega de

software, validando la relevancia de este enfoque para entornos distribuidos.

De esta forma, el estudio resalta la importancia de integrar DevOps con microservicios e infraestructura como código, aportando evidencia sólida de su capacidad para asegurar estabilidad y escalabilidad en sistemas modernos y ofreciendo un marco técnico que puede servir de referencia en contextos donde la eficiencia y la disponibilidad son factores determinantes.

2.1.2. A Nivel Nacional

Salazar Guevara, L. J. (2022), en su investigación "Implementación de DevOps para mejorar la integración y despliegue de software", tuvo como finalidad aplicar los principios y herramientas de DevOps en el desarrollo de una aplicación en una universidad peruana, enfocándose en eliminar los cuellos de botella de los métodos tradicionales. Motivado por la necesidad de optimizar el proceso de integración y entrega continua, su metodología incluyó análisis documental, diseño de un pipeline CI/CD con Jenkins y Docker, y monitoreo automatizado en ambientes de prueba y producción. Los resultados reflejaron una reducción de fallos y mejoras en los tiempos de despliegue, además del fortalecimiento de la colaboración entre los equipos involucrados, demostrando la aplicabilidad y beneficio de DevOps en instituciones educativas.

En consecuencia, podemos decir que el DevOps puede implementarse con éxito incluso en entornos académicos, lo que resalta su potencial como una solución transversal y adaptable a distintos sectores organizacionales que buscan modernizar sus procesos tecnológicos.

Castañeda García, P. A. (2019), en su tesis "Prácticas DevOps de entrega continua de software para la implementación automatizada del proceso de despliegue", planteó como objetivo optimizar el proceso de despliegue de software en una empresa peruana de tecnología. El estudio surgió por la creciente necesidad de acelerar la entrega de nuevas funcionalidades y reducir los errores manuales. La metodología combinó el diseño de

pipelines CI/CD, la integración de contenedores Docker, y el monitoreo continuo de métricas de eficiencia y calidad. Los resultados mostraron un incremento en la velocidad de entrega y una reducción significativa en los errores de despliegue, validando la pertinencia de adoptar el enfoque DevOps en contextos empresariales peruanos.

De esta manera, el estudio demuestra que DevOps fortalece la entrega continua en empresas tecnológicas nacionales, aportando evidencia de su capacidad para mejorar tanto la eficiencia operativa como la calidad del software, y consolidándose como un referente para su aplicación en diferentes sectores organizacionales.

Espinoza Arnesquito, J. A. (2019), en su trabajo "Implementación del primer equipo DevOps para el proyecto 'Portal Web Ferreyros' en la empresa MDP", propuso crear el primer equipo DevOps formal en una empresa peruana del sector servicios para mejorar la gestión y entrega de software. La investigación fue motivada por las limitaciones del modelo tradicional, con problemas de integración y baja velocidad de respuesta frente a incidentes. La metodología incluyó el diseño e implementación de pipelines automatizados, despliegues con Docker y la adopción de buenas prácticas documentadas de colaboración. Los resultados alcanzados manifestaron mayor estabilidad del portal web, una reducción notoria en el número de incidentes críticos y una aceleración en entregas continuas, reafirmando el impacto positivo de DevOps en el contexto nacional.

En tal sentido, el estudio subraya la importancia de la conformación de equipos DevOps en empresas locales, destacando las mejoras que generan en la estabilidad, continuidad y calidad de los servicios digitales, lo que evidencia su valor como modelo aplicable a otros entornos organizacionales.

Antón Sebastián, M. E. (2019), en su tesis "Proceso de desarrollo de software basado en DevOps para las aplicaciones web de una institución financiera", planteó como

propósito mejorar los procesos de desarrollo y despliegue de aplicaciones en un banco peruano. El origen del estudio estuvo en las limitaciones de los métodos tradicionales, que generaban retrasos en la entrega de software financiero. La metodología consistió en el diseño de un pipeline DevOps adaptado a los procesos de la institución, con etapas de integración y pruebas automatizadas. Los hallazgos mostraron una mejora en la calidad del software y una reducción en los tiempos de entrega, validando que DevOps es una alternativa eficaz para entornos financieros críticos.

Por tanto, la investigación aporta evidencia directa sobre el impacto de DevOps en el sector financiero nacional, resaltando su pertinencia para procesos críticos vinculados con la calidad y la rapidez en la entrega de aplicaciones, y ofreciendo un marco de referencia útil para estudios en instituciones de naturaleza similar.

Aguilar Riera, E. G., & Naula Sanisaca, J. G. (2025), en su investigación "Aplicación de principios DevOps y metodologías ágiles para el desarrollo del módulo del historial financiero de la Cooperativa de Ahorro y Crédito Jardín Azuayo", propusieron como propósito integrar DevOps y prácticas ágiles en un contexto cooperativo. El origen de la investigación fue la necesidad de responder a errores frecuentes y demoras en la gestión de servicios financieros de la cooperativa. La metodología combinó Scrum, Lean Inception y pipelines de integración continua. Los hallazgos evidenciaron una reducción de fallos en producción y una mejora en la capacidad de respuesta frente a cambios, validando la aplicabilidad de DevOps en instituciones cooperativas.

De esta manera, la investigación demuestra la viabilidad de aplicar DevOps en el sector cooperativo, aportando un referente cercano y pertinente que contribuye a consolidar la importancia de estas prácticas en instituciones financieras y ofreciendo una base comparativa útil para futuras implementaciones en contextos similares.

2.2. MARCO TEÓRICO

2.2.1. DevOps

DevOps constituye un enfoque orientado a integrar las actividades de desarrollo de software y operaciones de tecnología de la información mediante prácticas de colaboración, automatización y mejora continua. Su aplicación busca optimizar el ciclo de vida del software, permitiendo entregas más rápidas, confiables y alineadas con las necesidades de la organización. Para comprender este enfoque, a continuación, se presentan su definición, objetivos, relación entre desarrollo y operaciones, así como una introducción a su ciclo de vida.

2.2.1.1.Definición de DevOps.

DevOps es un enfoque de trabajo que integra las actividades de desarrollo de software (Development) y operaciones de tecnología de la información (Operations), con el propósito de mejorar la colaboración entre equipos, automatizar procesos y acelerar la entrega de software de manera confiable y continua. Este enfoque promueve la comunicación constante, la automatización de tareas y la mejora continua durante todo el ciclo de vida del software.

Según Forsgren, Humble y Kim (2018), DevOps permite incrementar la eficiencia operativa mediante la integración de prácticas orientadas a la automatización, la entrega continua y la medición permanente del rendimiento de los procesos. Su adopción contribuye a reducir tiempos de despliegue, mejorar la calidad del software y fortalecer la capacidad de respuesta ante cambios o incidentes.

En la presente investigación, DevOps constituye el enfoque utilizado para automatizar el proceso de despliegue del módulo de gestión de papeletas mediante la integración de herramientas como GitLab, Jenkins, SonarQube y Ansible.

2.2.1.2.Objetivos de DevOps.

El objetivo principal de DevOps es optimizar la entrega de software mediante la integración de personas, procesos y herramientas tecnológicas. Para ello, promueve la automatización de actividades repetitivas, la colaboración entre desarrollo y operaciones, así como la incorporación de mecanismos de control y retroalimentación continua.

Entre los principales objetivos de DevOps se encuentran la reducción de los tiempos de despliegue, la mejora de la calidad del software, la disminución de errores operativos, el fortalecimiento de la trazabilidad de los cambios y la capacidad de responder de manera más rápida a las necesidades del negocio.

Asimismo, DevOps busca establecer procesos repetibles y estandarizados que permitan realizar despliegues de forma consistente, minimizando riesgos asociados a la intervención manual durante las actividades operativas.

2.2.1.3.Relación entre desarrollo y operaciones.

Tradicionalmente, las áreas de desarrollo y operaciones trabajaban de manera independiente, generando diferencias en los objetivos y procedimientos utilizados para la gestión del software. Mientras el equipo de desarrollo se enfocaba en la creación de nuevas funcionalidades, el equipo de operaciones priorizaba la estabilidad y disponibilidad de los sistemas en producción.

DevOps surge como una estrategia orientada a reducir esta separación mediante la integración de ambas actividades dentro de un mismo flujo de trabajo. A través de la colaboración continua, la automatización de procesos y el uso compartido de herramientas, se busca mejorar la coordinación entre quienes desarrollan el software y quienes administran los entornos donde este opera.

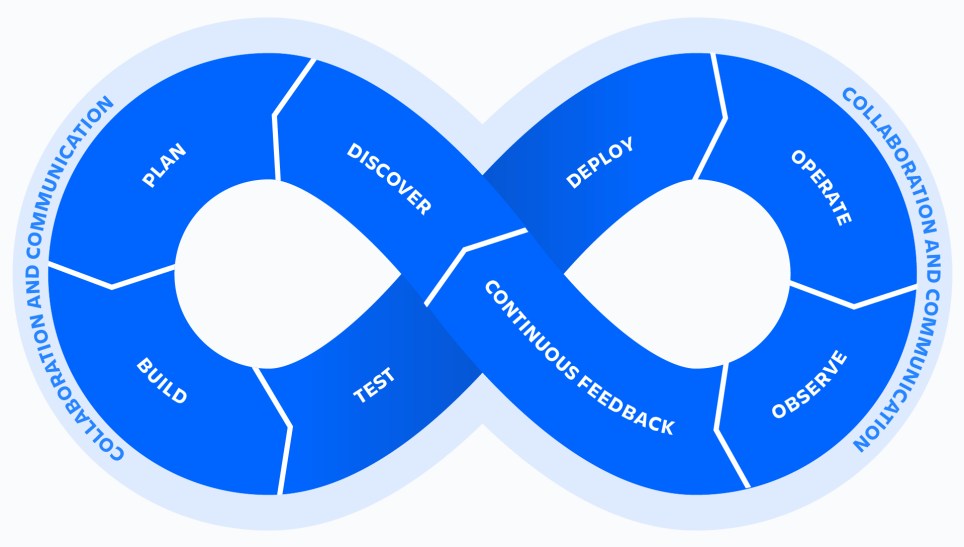
Esta integración favorece una mayor trazabilidad de los cambios, una respuesta más rápida ante incidentes y una gestión más eficiente del ciclo de vida del software.

2.2.1.4.Ciclo de vida DevOps.

Según Unity (2025), el ciclo de vida DevOps consta de siete fases principales: desarrollo y entrega continuos, integración continua, pruebas continuas, entrega/despliegue continuo, supervisión continua, retroalimentación continua y operaciones continuas.

- ✓ **Desarrollo y entrega continuos (Continuous Development & Delivery):** Incluye la planificación y la codificación de forma constante y colaborativa.
- ✓ **Integración continua (Continuous Integration):** Automatización de compilaciones, pruebas y validaciones cada vez que hay cambios en el código.
- ✓ **Pruebas continuas (Continuous Testing):** Verificación constante, tanto automática como manual, para garantizar la calidad antes de avanzar a producción.
- ✓ **Entrega/despliegue continuo (Continuous Deployment):** Automatización del lanzamiento de nuevas versiones en ambientes previos y en producción.
- ✓ **Supervisión continua (Continuous Monitoring):** Observabilidad centralizada para identificar y responder a incidentes rápidamente.
- ✓ **Retroalimentación continua (Continuous Feedback):** Recopilación de comentarios de usuarios y equipos para generar mejoras de manera continua.
- ✓ **Operaciones continuas (Continuous Operations):** Gestión, mantenimiento y ajuste de la infraestructura y aplicaciones en ejecución, basándose en métricas y aprendizajes obtenidos

Figura 1
Ciclo de vida DevOps



Nota. Adaptado de "Introducción a DevOps para la mejora de los procesos de desarrollo con herramientas Open Source" (p. 24), por A. Élez Villamarín, 2019, Universidad Oberta de Catalunya. https://openaccess.uoc.edu/bitstream/10609/97626/6/aelezTFG0619memoria.pdf

La Figura 1 presenta una representación general del ciclo de vida DevOps, el cual integra de manera continua las actividades de desarrollo, validación, despliegue y operación del software. Este modelo promueve la colaboración entre los equipos de desarrollo y operaciones mediante procesos automatizados que permiten acelerar la entrega de nuevas funcionalidades sin comprometer la calidad y estabilidad de los sistemas.

Las fases que conforman el ciclo de vida DevOps se encuentran interrelacionadas y operan de forma continua, permitiendo que la información generada en cada etapa sirva como retroalimentación para mejorar los procesos posteriores. De esta manera, las actividades de desarrollo, integración, pruebas, despliegue, monitoreo y operación forman un flujo permanente orientado a la mejora continua del software.

En el contexto de la presente investigación, este ciclo de vida sirvió como referencia para la implementación de prácticas de integración continua, control de calidad y despliegue automatizado mediante GitLab, Jenkins, SonarQube y Ansible, herramientas que permitieron optimizar el proceso de liberación del módulo de gestión de papeletas basado en microservicios.

2.2.2. Modelo CALMS

Según Albarran Salazar y Delgado Farro (2024), la implementación del modelo CALMS en una organización dedicada al desarrollo de software impulsa la optimización integral del ciclo de vida y la entrega de productos tecnológicos. CALMS acrónimo de Cultura, Automatización, Lean, Medición y Compartir se plantea como un marco que transforma la gestión de equipos y procesos mediante la estandarización, automatización e iteración constante en la construcción de software. Diversos estudios en ámbitos empresariales y académicos coinciden en que aplicar CALMS no solo mejora la eficiencia, sino también la colaboración interáreas, la calidad de los entregables y la satisfacción de los empleados (Muñoz et al., 2020; Atlassian, 2024).

Guerrero (2020) enfatiza que adoptar CALMS asegura una transición organizada y evita fallas típicas de implantaciones superficiales.

Por tanto, CALMS es un marco conceptual usado para evaluar y guiar la adopción de DevOps, describiendo los principales pilares culturales, organizativos y técnicos que habilitan la transformación digital y ágil en las organizaciones. La sigla CALMS fue introducida por Jez Humble.

2.2.2.1.Definición del modelo CALMS

El modelo CALMS es un marco conceptual utilizado para evaluar y comprender la adopción de prácticas DevOps dentro de las organizaciones. Fue propuesto por Jez

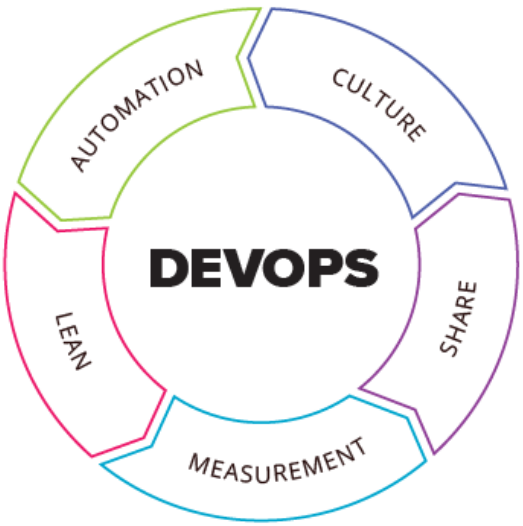
Humble y posteriormente difundido como una referencia para analizar los factores que contribuyen al éxito de las iniciativas DevOps.

El término CALMS corresponde a las iniciales de cinco dimensiones fundamentales: Cultura (Culture), Automatización (Automation), Procesos Esbeltos (Lean), Medición (Measurement) y Compartir (Sharing). Estas dimensiones permiten evaluar tanto los aspectos técnicos como organizacionales involucrados en la implementación de DevOps.

A diferencia de enfoques centrados únicamente en herramientas tecnológicas, CALMS considera elementos relacionados con la colaboración entre equipos, la mejora continua, la automatización de procesos, la utilización de métricas y la gestión del conocimiento. Por esta razón, el modelo es ampliamente utilizado como referencia para analizar el grado de madurez de las prácticas DevOps dentro de una organización.

En la presente investigación, el modelo CALMS fue utilizado como marco de análisis para describir los resultados observados tras la implementación de la solución DevOps en el proceso de despliegue del módulo de gestión de papeletas.

Figura 2
Dimensiones del modelo CALMS



Nota. Adaptado de "Introducción a DevOps para la mejora de los procesos de desarrollo con herramientas Open Source" (p. 24), por A. Élez Villamarín, 2019, Universidad Oberta de Catalunya. https://openaccess.uoc.edu/bitstream/10609/97626/6/aelezTFG0619memoria.pdf

La Figura 2 presenta las cinco dimensiones que conforman el modelo CALMS. Cada una de ellas aborda aspectos específicos relacionados con la adopción de prácticas DevOps, permitiendo analizar de forma integral los cambios organizacionales y tecnológicos asociados a la automatización de procesos y mejora continua del ciclo de vida del software.

De acuerdo con Kim et al. (2016), el modelo CALMS (Culture, Automation, Lean, Measurement, Sharing) representa los cinco principios clave de DevOps:

1. **Cultura**: Fomenta la colaboración transversal entre áreas, la responsabilidad compartida, el aprendizaje desde el error y la comunicación abierta. Esta cultura es crucial para romper silos y alinear objetivos entre equipos de desarrollo, operaciones y negocio (TechTarget, 2024).
2. **Automatización**: Busca reducir tareas manuales mediante herramientas de integración, pruebas y despliegue automatizado. Esto permite liberar tiempo, disminuir errores y lograr consistencia en las entregas y operaciones (Atlassian, 2024).
3. **Lean**: Busca reducir el desperdicio optimizando flujos de trabajo y priorizando el valor entregado al cliente. Cada ciclo es más corto, enfocado y ajustado a retroalimentación rápida (Muñoz et al., 2020)

4. **Medición**: Establecer métricas clave, como tasa de fallos, frecuencia de despliegue, velocidad de recuperación y satisfacción del usuario, permite orientar la mejora continua basada en datos objetivos y reales.
5. **Sharing(Compartir):** Promueve la apertura, transparencia, manejo colaborativo del conocimiento y la documentación accesible. La filosofía "You build it, you run it" (Tú lo creas, tú lo operas) refuerza la responsabilidad colectiva (ManageEngine, 2024; Container8, 2024).

2.2.2.2.Ventajas de la metodología CALMS

- ✓ Mejora la colaboración y el clima organizacional al fomentar la responsabilidad y objetivos compartidos.
- ✓ Permite automatización efectiva, liberando a los equipos de tareas repetitivas y minimizando errores humanos.
- ✓ Impulsa la mejora continua gracias a la medición objetiva y visibilidad de los procesos.
- ✓ Reduce tiempos de entrega y acelera la llegada de innovación al cliente.
- ✓ Favorece la innovación y el aprendizaje a partir del intercambio constante de conocimiento.
- ✓ Aporta un marco flexible que puede adaptarse a startups, pymes y grandes empresas, guiando la madurez DevOps.

2.2.2.3.Desventajas y Retos

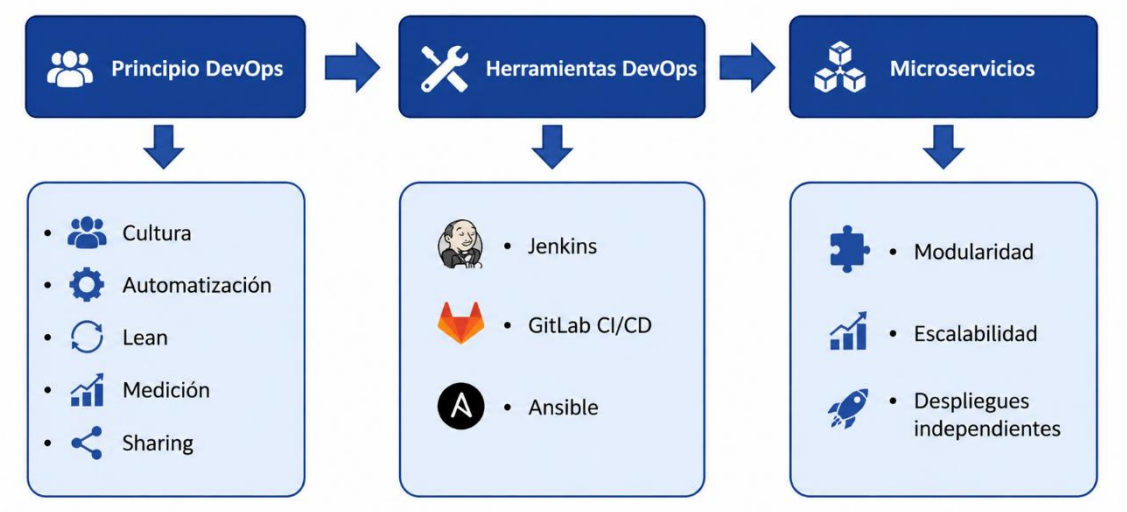
- ✓ Requiere cambio cultural profundo, lo que puede encontrar resistencia en entornos tradicionales.
- ✓ La automatización inicial demanda inversión en herramientas y capacitación.
- ✓ Complejidad al medir lo relevante: identificar métricas útiles y evitar "vanity metrics" puede ser difícil.

- ✓ Riesgo de implementación parcial: Si solo se aplican algunos principios (por ejemplo, sólo automatización sin fomentar cultura), se limitan los beneficios.
- ✓ Desafíos al compartir conocimiento en organizaciones con barreras históricas al intercambio de información (Atlassian, 2024; Container8, 2024).

2.2.2.4.Implementación

Un despliegue eficaz de CALMS inicia con un diagnóstico organizacional, seguido de la selección progresiva de herramientas y métodos, la definición iterativa y revisión de métricas y la promoción constante de una cultura de transparencia y aprendizaje compartido. La literatura recomienda acciones como campañas de capacitación, creación de espacios colaborativos y desarrollo de documentación ligera y accesible.

Figura 3
Esquema conceptual principios DevOps, herramientas y microservicios



FUENTE: *Elaboración propia.*

En conjunto, las dimensiones del modelo CALMS proporcionan un marco de referencia para analizar la adopción de prácticas DevOps desde una perspectiva integral.

Debido a que considera aspectos culturales, técnicos y organizacionales, este modelo fue utilizado en la presente investigación para evaluar los resultados observados tras la implementación de la solución propuesta.

2.2.3. Integración y Entrega Continua (CI/CD)

2.2.3.1.Definición de CI/CD

La Integración Continua y la Entrega Continua, conocidas por sus siglas CI/CD (Continuous Integration / Continuous Delivery), constituyen un conjunto de prácticas orientadas a automatizar las actividades de integración, construcción, validación y preparación del software para su liberación. De acuerdo con Humble y Farley (2010), la entrega continua busca mantener el software en un estado que permita su liberación de manera confiable mediante procesos automatizados y repetibles.

La Integración Continua se orienta a incorporar frecuentemente los cambios realizados por los desarrolladores en un repositorio compartido, donde se ejecutan procesos automatizados de compilación y validación que permiten identificar problemas en etapas tempranas del ciclo de desarrollo. Por su parte, la Entrega Continua extiende este proceso mediante la automatización de las etapas necesarias para mantener el software preparado para su liberación hacia los diferentes entornos.

Es importante diferenciar la Entrega Continua (Continuous Delivery) del Despliegue Continuo (Continuous Deployment). En la entrega continua, el software se mantiene preparado para ser liberado, pero el paso hacia el entorno de producción puede estar condicionado a una aprobación manual; mientras que, en el despliegue continuo, los cambios que superan las validaciones establecidas pueden desplegarse automáticamente en producción.

En el contexto de la presente investigación, el pipeline implementado corresponde a un modelo de Integración Continua y Entrega Continua (CI/CD), debido a que automatiza las etapas de construcción, análisis de calidad y despliegue en el entorno de

pruebas, mientras que el despliegue hacia producción requiere una aprobación manual por parte del responsable técnico. Esta configuración permite mantener un proceso automatizado y controlado, acorde con las necesidades operativas de la Cooperativa Santa María Magdalena.

2.2.3.2.Integración Continua (Continuous Integration)

González (2018), menciona que la Integración Continua (CI) es la práctica de fusionar cambios frecuentemente en un repositorio principal, acompañado de compilación y pruebas automatizadas para detectar errores tempranamente.

Según Muñoz (2021), una correcta configuración de pipelines con herramientas como Jenkins, GitLab CI/CD, Docker y Kubernetes no solo acelera el delivery, sino que establece puntos de control de calidad y retroalimentación instantánea, facilitando la mejora continua.

La Integración Continua consiste en la práctica de integrar frecuentemente los cambios realizados por los desarrolladores dentro de un repositorio centralizado. Cada modificación incorporada desencadena procesos automáticos de compilación, validación y análisis del código fuente, permitiendo identificar errores en etapas tempranas del desarrollo.

Esta práctica contribuye a reducir conflictos entre versiones, mejorar la calidad del software y mantener una visión actualizada del estado de los proyectos. Asimismo, facilita la detección temprana de problemas antes de que estos alcancen entornos de pruebas o producción.

2.2.3.3.Entrega continua (Continuous Delivery)

La entrega continua (*Continuous Delivery*) se define como una práctica metodológica en el desarrollo de software cuyo propósito central es garantizar que una aplicación permanezca en un estado constante de liberación. Bajo este enfoque, cualquier

modificación en el código fuente se somete a un flujo automatizado que incluye la compilación, la ejecución de pruebas y la validación de calidad, asegurando que el artefacto quede óptimo para su despliegue en diversos entornos mediante un procedimiento controlado. Es fundamental diferenciar este concepto del despliegue continuo (*Continuous Deployment*), puesto que, en la entrega continua, la transferencia final al entorno de producción no es automática, sino que suele depender de una validación o aprobación manual estratégica (IBM, 2024; ServiceNow, 2025).

En esta investigación, el pipeline diseñado y ejecutado responde estrictamente a un modelo de integración y entrega continua (CI/CD). Tras superar las etapas automatizadas de compilación, la evaluación de calidad de código con SonarQube y el despliegue automático en el entorno de pruebas, el paso final hacia el servidor de producción quedó condicionado a una autorización explícita por parte del responsable técnico en Jenkins. Por lo tanto, la solución tecnológica adoptada descarta el despliegue continuo automatizado, consolidándose como un flujo de entrega continua gobernado por una aprobación manual para su puesta en producción.

2.2.3.4.Pipelines

Un pipeline (tubería) en el contexto DevOps es la secuencia automatizada de etapas o tareas por las que atraviesa el código desde que se desarrolla hasta que se despliega en producción. Los pipelines estructuran y automatizan pasos como la integración, las pruebas y el despliegue, permitiendo una entrega continua eficiente y ordenada (Muñoz, 2023).

En integración y entrega continua, los pipelines representan los flujos de trabajo automatizados que gestionan las distintas etapas del ciclo de vida del software: desde compilar, probar, integrar, hasta desplegar el producto final. Su objetivo es reducir el trabajo manual, minimizar errores y facilitar lanzamientos frecuentes y confiables (Mercedes Brito, 2022).

En general la Integración Continua y la Entrega Continua constituyen componentes fundamentales dentro de la filosofía DevOps. Su aplicación permite automatizar procesos relacionados con la construcción, validación y despliegue del software, contribuyendo a mejorar la eficiencia operativa y la calidad de las liberaciones. En la presente investigación, estas prácticas fueron implementadas mediante la integración de GitLab, Jenkins, SonarQube y Ansible dentro de un flujo automatizado de despliegue.

2.2.4. Arquitectura basada en Microservicios

2.2.4.1.Definición de Microservicio

De acuerdo con Richardson (s.f.), la arquitectura de microservicios es un enfoque de desarrollo de software en el que una aplicación se construye como un conjunto de servicios pequeños e independientes, cada uno orientado a una funcionalidad específica del negocio. Estos servicios se ejecutan de manera autónoma y se comunican entre sí mediante mecanismos ligeros, generalmente a través de interfaces de programación de aplicaciones (API).

A diferencia de las arquitecturas monolíticas tradicionales, donde todos los componentes se encuentran integrados en una única aplicación, los microservicios permiten dividir el sistema en módulos independientes que pueden ser desarrollados, desplegados, mantenidos y escalados de forma individual. Esta característica proporciona una mayor flexibilidad para la evolución de los sistemas, facilita la implementación de cambios específicos y reduce el impacto de las actualizaciones sobre el resto de la aplicación.

Asimismo, la independencia funcional de cada servicio favorece la adopción de prácticas DevOps, permitiendo automatizar procesos de integración continua, validación y despliegue mediante pipelines CI/CD, lo que contribuye a una gestión más eficiente del ciclo de vida del software.

2.2.4.2.Características de la arquitectura de microservicios

La arquitectura basada en microservicios presenta diversas características que favorecen el desarrollo y mantenimiento de aplicaciones modernas.

Entre las principales características destacan:

- ✓ Independencia funcional de cada servicio.
- ✓ Despliegue individual de componentes.
- ✓ Escalabilidad selectiva según la demanda de cada servicio.
- ✓ Aislamiento de fallos entre módulos.
- ✓ Facilidad para realizar actualizaciones sin afectar todo el sistema.

Estas características permiten una mayor flexibilidad en la administración de aplicaciones empresariales y favorecen la adopción de prácticas de automatización y despliegue y/o entrega continua.

2.2.4.3.Relación entre microservicios y DevOps

Muñoz (2021) indica que adoptar una arquitectura de microservicios permite a las organizaciones descomponer aplicaciones complejas en servicios independientes, que pueden ser desarrollados, probados y desplegados de forma autónoma. En este sentido, la combinación de DevOps y microservicios fomenta la escalabilidad, el despliegue incremental y la recuperación rápida, mejorando la eficiencia y capacidad de respuesta ante los cambios del negocio.

Por su parte Guerrero (2020) enfatiza que cada microservicio puede tener su propio pipeline CI/CD, lo que habilita la entrega continua y la automatización total.

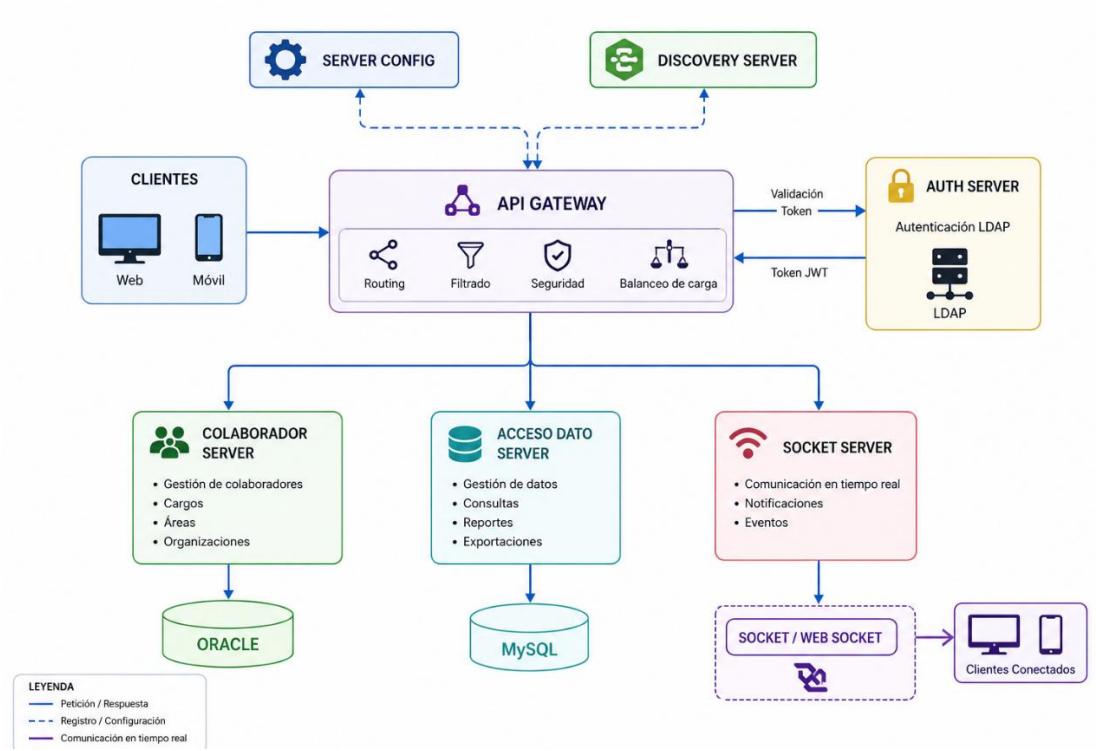
Este enfoque facilita la evolución progresiva de los sistemas, permitiendo realizar modificaciones sobre componentes específicos sin afectar el funcionamiento general de la aplicación.

Las arquitecturas basadas en microservicios mantienen una estrecha relación con los principios DevOps debido a que ambos enfoques buscan mejorar la velocidad, flexibilidad y confiabilidad en la entrega de software.

La independencia de los microservicios permite que cada componente pueda ser construido, validado y desplegado de manera individual, facilitando la automatización de procesos mediante pipelines CI/CD. De esta manera, los equipos pueden liberar nuevas versiones de un servicio específico sin necesidad de desplegar nuevamente la totalidad de la aplicación.

En este contexto, la combinación de DevOps y microservicios favorece la escalabilidad, el despliegue incremental, la recuperación rápida ante fallos y la incorporación de mecanismos de integración continua y control de calidad automatizado. Como resultado, las organizaciones pueden optimizar la gestión del ciclo de vida del software y responder de manera más eficiente a los cambios y necesidades del negocio.

Figura 4
Arquitectura basada en microservicios



Nota. Representación general de una aplicación estructurada mediante microservicios independientes que interactúan para proporcionar funcionalidades específicas del negocio. Elaboración propia, 2025.

2.2.4.4 Beneficios de los microservicios en el despliegue de software

La adopción de una arquitectura basada en microservicios proporciona diversas ventajas para los procesos de despliegue y mantenimiento de aplicaciones empresariales. Según Richardson (s.f.), la independencia funcional de cada servicio permite realizar actualizaciones específicas sin necesidad de desplegar nuevamente la totalidad del sistema, reduciendo el impacto de los cambios y facilitando el mantenimiento de las aplicaciones.

Muñoz (2021) señala que una de las principales ventajas de los microservicios es la posibilidad de ejecutar despliegues incrementales, permitiendo incorporar nuevas funcionalidades o correcciones únicamente en los componentes afectados. Esta característica contribuye a disminuir los tiempos requeridos para la liberación de versiones y favorece una respuesta más ágil ante las necesidades del negocio.

Asimismo, Guerrero (2020) destaca que la independencia de los microservicios facilita la implementación de prácticas DevOps y procesos de integración y entrega continua (CI/CD), ya que cada componente puede disponer de su propio pipeline automatizado para construcción, validación y despliegue. Esto permite incrementar la flexibilidad operativa y mejorar la gestión del ciclo de vida del software.

En la presente investigación, estas ventajas permitieron automatizar el despliegue del módulo de gestión de papeletas mediante mecanismos que facilitan la actualización independiente de los microservicios involucrados, contribuyendo a la reducción de tiempos operativos y al fortalecimiento de la trazabilidad del proceso de despliegue.

La arquitectura basada en microservicios constituye un elemento complementario a las prácticas DevOps, ya que facilita la automatización, escalabilidad y gestión independiente de los componentes de software. Debido a estas ventajas, este enfoque fue utilizado como base para la implementación del módulo de gestión de papeletas considerado en la presente investigación.

2.2.5. Infraestructura como Código (IaC) y Automatización

2.2.5.1. Definición de Infraestructura como Código (IaC)

La Infraestructura como Código (Infrastructure as Code – IaC) es una práctica que permite administrar y aprovisionar recursos tecnológicos mediante archivos de configuración y scripts automatizados, evitando la ejecución manual de tareas administrativas. Este enfoque facilita la definición, implementación y mantenimiento de infraestructuras de manera consistente y reproducible.

Según Saxena et al. (2025), IaC constituye uno de los pilares fundamentales de la automatización en entornos DevOps, debido a que permite gestionar servidores, configuraciones y servicios mediante código versionable, favoreciendo la estandarización y reduciendo errores asociados a intervenciones manuales.

La adopción de Infraestructura como Código contribuye a mejorar la trazabilidad de los cambios, simplificar la administración de entornos y fortalecer la capacidad de replicar configuraciones en diferentes escenarios operativos.

2.2.5.2.Automatización en entornos DevOps

La automatización representa uno de los principios fundamentales de DevOps y busca reducir la ejecución manual de actividades repetitivas relacionadas con la construcción, validación, configuración y despliegue de software. Mediante la utilización de herramientas especializadas, las organizaciones pueden ejecutar procesos de manera más rápida, consistente y controlada.

Kim et al. (2016) sostienen que la automatización permite incrementar la eficiencia operativa y mejorar la confiabilidad de los procesos tecnológicos, reduciendo la posibilidad de errores humanos durante la liberación de nuevas versiones de software.

En este contexto, la automatización se convierte en un elemento clave para la implementación de prácticas de integración continua, entrega continua e infraestructura como código dentro de un ecosistema DevOps.

2.2.5.3.Beneficios de IaC en los despliegues

La utilización de Infraestructura como Código aporta diversos beneficios a los procesos de despliegue y administración de sistemas. Entre ellos destacan la estandarización de configuraciones, la reducción de errores operativos, la mejora de la trazabilidad y la posibilidad de reproducir entornos de manera consistente.

Saxena et al. (2025) señalan que IaC facilita la automatización de tareas relacionadas con configuración, actualización y mantenimiento de infraestructura, permitiendo reducir tiempos operativos y mejorar la confiabilidad de los despliegues.

Asimismo, la utilización de configuraciones definidas mediante código favorece el control de versiones y la documentación de cambios, fortaleciendo la gestión del conocimiento dentro de las organizaciones.

La adopción de IaC ofrece múltiples beneficios en entornos organizacionales:

- ✓ **Reproducibilidad**: los entornos pueden replicarse de forma consistente en desarrollo, pruebas y producción.
- ✓ **Automatización**: se reducen los procesos manuales propensos a errores.
- ✓ **Escalabilidad**: la infraestructura puede crecer o disminuir según la demanda del negocio.
- ✓ **Auditoría y versionamiento**: al tratarse de código, es posible rastrear cambios y mantener un historial claro de configuraciones. Según Red Hat (2025), IaC contribuye significativamente a mejorar la eficiencia operativa y reducir costos asociados a errores humanos.

2.2.5.4.Relación entre IaC y Ansible

Ansible es una herramienta ampliamente utilizada para implementar prácticas de Infraestructura como Código y automatización de despliegues. Su funcionamiento se basa en la utilización de archivos denominados playbooks, los cuales permiten definir procedimientos automatizados para la configuración y administración de servidores.

Según Sharma (2017), herramientas como Ansible facilitan la automatización de tareas operativas mediante configuraciones declarativas, permitiendo ejecutar procedimientos repetibles y consistentes sobre múltiples entornos tecnológicos.

La IaC constituye un pilar esencial en DevOps porque asegura coherencia entre entornos, acelera la entrega de nuevas versiones, facilita la integración con pipelines automatizados y disminuye riesgos asociados a la intervención manual (Microsoft, 2024).

En su parte Vasileiou et al. (2025) menciona que la adopción de IaC incrementa la capacidad de las organizaciones para responder de manera flexible a los cambios en entornos complejos

En general en la presente investigación, Ansible fue utilizado para automatizar actividades relacionadas con la transferencia de artefactos, actualización de aplicaciones

y reinicio de servicios en los entornos de Calidad y Producción, contribuyendo a la estandarización del proceso de despliegue.

2.2.5.5.Aplicación de IaC en la arquitectura de microservicios de la Cooperativa Santa María Magdalena

En el contexto de la Cooperativa de Ahorro y Crédito Santa María Magdalena (CACSMM), la adopción de Infraestructura como Código (IaC) representa una estrategia esencial para estandarizar la creación y configuración de entornos destinados al ecosistema de microservicios institucionales. Mediante el uso de herramientas como Ansible, sería posible automatizar el aprovisionamiento y despliegue de servicios clave como *acceso-dato-service*, *auth-service*, *colaborador-service*, *config-server*, *discovery-server*, *gateway-server*, *gestión-documental-service* y *web-socket-service*, sobre servidores Ubuntu y MySQL, garantizando uniformidad entre los entornos de desarrollo, calidad y producción. Esta práctica reduciría de forma significativa los tiempos de despliegue, incrementaría la confiabilidad de las operaciones y aseguraría la alineación con los principios de DevOps, especialmente en lo referente a la automatización, la entrega continua y la reducción de errores humanos.

2.2.6. Herramientas utilizadas en la solución

2.2.6.1.Jenkins

Jenkins es una herramienta de automatización ampliamente utilizada para implementar procesos de Integración Continua y Entrega Continua (CI/CD). Su principal función consiste en coordinar y ejecutar de forma automática actividades relacionadas con compilación, validación, pruebas y despliegue de software.

Humble y Farley (2010) destacan que las herramientas de automatización permiten construir procesos repetibles y controlados, reduciendo la intervención manual y mejorando la consistencia de las liberaciones de software.

Por su parte Morales Vélez (2022), menciona que Jenkins es una de las herramientas open source más usadas para la automatización CI/CD. Permite definir pipelines gráficos y como código (Jenkinsfile), integrar pruebas automatizadas, análisis estático, despliegues en entornos virtualizados y la administración programada (cron jobs). Jenkins soporta plugins que facilitan la integración con Docker, GitHub/GitLab, Kubernetes, sistemas de notificación y gestión de artefactos.

Jenkins es una plataforma de automatización de código abierto, muy utilizada en los procesos de integración y entrega continua (CI/CD) dentro del desarrollo de software moderno. Su objetivo principal es facilitar y agilizar la creación, prueba e implementación de aplicaciones, automatizando las tareas repetitivas y complejas del ciclo de vida del software (Universitat Oberta de Catalunya, 2021).

Ventajas de usar Jenkins

Jenkins es una herramienta respaldada por una comunidad muy activa y abierta. Todos los meses se realizan reuniones públicas donde cualquier persona puede participar y aportar ideas para mejorar el proyecto.

- ✓ Hasta ahora, se han resuelto alrededor de 280 solicitudes y el proyecto lanza actualizaciones estables aproximadamente cada tres meses, lo que da confianza sobre su desarrollo continuo.
- ✓ Jenkins sigue creciendo al ritmo de la tecnología. Actualmente, cuenta con unos 320 complementos disponibles en su base de datos, haciendo que la herramienta sea mucho más versátil y poderosa.
- ✓ Se puede usar Jenkins en la nube, lo que permite implementarlo en diferentes plataformas y aprovechar la flexibilidad que esto ofrece.
- ✓ Una de las razones por las que Jenkins se popularizó es porque fue creado por un desarrollador pensando en facilitar la vida a otros desarrolladores.

Desventajas de usar Jenkins

Aunque Jenkins ofrece muchas posibilidades, también tiene sus puntos débiles.

- ✓ Su interfaz visual está un poco anticuada y no es tan intuitiva ni moderna comparada con otras herramientas recientes.
- ✓ A pesar de ser querida por muchos desarrolladores, administrar Jenkins puede no ser sencillo; se ejecuta sobre un servidor, por lo que hace falta saber cómo manejar y monitorear servidores.
- ✓ Muchas personas no se animan a usar Jenkins porque instalarla y configurarla puede ser complicado al principio.
- ✓ Las integraciones continuas que Jenkins habilita a veces dejan de funcionar por pequeños cambios en la configuración. Cuando esto pasa, se detienen los procesos automáticos y hay que intervenir para solucionarlo.

En esta investigación, Jenkins fue utilizado como motor de orquestación del pipeline CI/CD, permitiendo automatizar la compilación de los microservicios, ejecutar validaciones de calidad y coordinar los despliegues hacia los entornos de Calidad y Producción.

2.2.6.2.SonarQube

SonarQube es una plataforma especializada en análisis estático de código que permite evaluar aspectos relacionados con calidad, mantenibilidad, confiabilidad y seguridad del software. La herramienta analiza automáticamente el código fuente y genera indicadores que facilitan la identificación temprana de errores y oportunidades de mejora.

Dentro de los entornos DevOps, SonarQube suele integrarse a los pipelines CI/CD mediante mecanismos denominados Quality Gate, los cuales establecen criterios mínimos que deben cumplirse antes de continuar con el proceso de despliegue.

En la presente investigación, SonarQube fue integrado al pipeline automatizado para realizar validaciones de calidad sobre los microservicios antes de su despliegue, contribuyendo a fortalecer los mecanismos de control implementados durante el ciclo de liberación del software.

2.2.6.3.GitLab

GitLab es una plataforma de gestión de repositorios basada en Git que permite administrar el código fuente, controlar versiones y coordinar el trabajo colaborativo dentro de los proyectos de software. Además de las funcionalidades relacionadas con el control de versiones, GitLab incorpora herramientas para la gestión de incidencias, revisión de cambios y automatización de procesos dentro del ciclo de desarrollo.

Según Atlassian (2024), las plataformas de control de versiones constituyen un componente fundamental dentro de los entornos DevOps, debido a que permiten centralizar el código fuente y mantener la trazabilidad de las modificaciones realizadas por los equipos de desarrollo.

En la presente investigación, GitLab fue utilizado como repositorio central para la administración del código fuente de los microservicios que conforman el módulo de gestión de papeletas, permitiendo gestionar ramas, registrar cambios y activar los procesos automatizados mediante Webhooks.

2.2.6.4.Ansible

Ansible es una herramienta de automatización de TI que permite el aprovisionamiento, la administración de configuración y el despliegue de aplicaciones, eliminando tareas manuales repetitivas y facilitando la orquestación de sistemas de forma sencilla y sin agentes. (Awati & Courtemanche, 2023).

Ansible es una plataforma de automatización de infraestructura como código desarrollado por Red Hat, que utiliza archivos YAML llamados playbooks para definir

tareas de configuración y despliegue de manera declarativa, garantizando entornos reproducibles y consistentes tanto on-premise como en la nube. (Red Hat, 2025).

Sharma (2017) señala que Ansible facilita la automatización de tareas operativas mediante configuraciones declarativas, permitiendo ejecutar procedimientos consistentes y repetibles sobre diferentes entornos tecnológicos.

En esta investigación, Ansible fue utilizado para automatizar actividades relacionadas con la transferencia de artefactos, actualización de aplicaciones y reinicio de servicios en los servidores de Calidad y Producción, contribuyendo a reducir la intervención manual durante los despliegues.

2.2.7. Métricas operativas de despliegue

Las métricas operativas constituyen indicadores utilizados para evaluar el desempeño de los procesos relacionados con el desarrollo, despliegue y mantenimiento de software. Su aplicación permite obtener información objetiva sobre la eficiencia de las actividades ejecutadas y facilita la identificación de oportunidades de mejora dentro de los procesos tecnológicos.

De acuerdo con Forsgren, Humble y Kim (2018), la medición continua es un componente fundamental dentro de los entornos DevOps, debido a que proporciona evidencia cuantificable para evaluar el impacto de las prácticas implementadas y apoyar la toma de decisiones basada en datos.

En la presente investigación, las métricas operativas fueron utilizadas para describir el comportamiento del proceso de despliegue antes y después de la implementación de principios DevOps, permitiendo realizar comparaciones entre el escenario manual y el escenario automatizado.

2.2.7.1.Tiempo de despliegue

El tiempo de despliegue representa el período total requerido para completar la liberación de una nueva versión de software desde el inicio de las actividades de

preparación hasta la validación final en el entorno correspondiente. Este indicador permite evaluar la eficiencia operativa de los procedimientos utilizados para la distribución de aplicaciones.

La reducción del tiempo de despliegue constituye uno de los principales objetivos de las prácticas DevOps, debido a que favorece una entrega más rápida de nuevas funcionalidades y correcciones de software (Forsgren et al., 2018).

En la presente investigación, esta métrica fue utilizada para comparar los tiempos registrados durante el proceso manual de despliegue y los obtenidos después de la implementación de la solución automatizada.

2.2.7.2.Nivel de automatización

El nivel de automatización hace referencia al grado en que las actividades relacionadas con compilación, validación y despliegue son ejecutadas mediante herramientas automatizadas en lugar de procedimientos manuales. Un mayor nivel de automatización contribuye a reducir errores operativos, incrementar la consistencia de los procesos y optimizar el uso de recursos técnicos.

Kim et al. (2016) señalan que la automatización constituye uno de los pilares fundamentales de DevOps, debido a su capacidad para mejorar la confiabilidad y repetibilidad de los procesos tecnológicos.

En esta investigación, el nivel de automatización fue evaluado mediante la identificación de actividades que anteriormente requerían intervención manual y que posteriormente fueron incorporadas dentro de pipelines CI/CD y playbooks automatizados.

2.2.7.3.Trazabilidad del proceso de despliegue

La trazabilidad corresponde a la capacidad de registrar y dar seguimiento a las actividades ejecutadas durante el ciclo de vida del software. Este concepto permite

identificar el origen de los cambios, los responsables de las modificaciones y las acciones realizadas durante cada despliegue.

La utilización de repositorios de código, sistemas de control de versiones y herramientas de automatización contribuye a fortalecer la trazabilidad de los procesos tecnológicos, facilitando auditorías y actividades de seguimiento.

En la presente investigación, la trazabilidad fue fortalecida mediante la integración de GitLab, Jenkins y Ansible, permitiendo registrar los cambios realizados sobre los microservicios y las actividades ejecutadas durante cada despliegue.

2.2.7.4.Control de calidad del software

El control de calidad comprende las actividades orientadas a verificar que el software cumpla con criterios definidos de funcionamiento, mantenibilidad y confiabilidad antes de su liberación hacia los entornos operativos.

Según Humble y Farley (2010), la incorporación de mecanismos automatizados de validación dentro de los procesos de integración y entrega continua permite detectar errores de manera temprana y mejorar la calidad de las liberaciones de software.

En esta investigación, el control de calidad fue implementado mediante SonarQube y la utilización de Quality Gate dentro del pipeline CI/CD, permitiendo evaluar automáticamente el código fuente antes de su despliegue.

Las métricas operativas descritas proporcionan una base para evaluar el comportamiento del proceso de despliegue y analizar los resultados obtenidos tras la implementación de principios DevOps. Estos indicadores fueron utilizados como referencia para describir las mejoras observadas en términos de eficiencia operativa, automatización, trazabilidad y control de calidad dentro de la Cooperativa Santa María Magdalena.

CAPÍTULO III
MATERIAL Y MÉTODOS

3.1. TIPO DE INVESTIGACIÓN

"La investigación aplicada se enfoca en la resolución de problemas prácticos con un margen de generalización limitado. cuyo objetivo es de realizar aportes al conocimiento científico" (Cecilia salgado,2018,p.64).

Según (Carrasco,2006), la investigación aplicada tiene propósitos prácticos inmediatos bien definidos, es decir, se investiga para actuar, transformar, modificar o producir cambios en un determinado sector de la realidad.

Teniendo en cuenta lo expuesto en los párrafos anteriores, la presente investigación es de tipo aplicada, ya que su propósito principal es porque desarrolla e implementa una solución tecnológica orientada a optimizar un proceso real de despliegue de software en la Cooperativa Santa María Magdalena, Ayacucho 2025. El estudio busca actuar directamente sobre los procesos tecnológicos internos de la organización, proporcionando mejoras reales y transferibles al sector, en línea con la orientación transformadora y de impacto inmediato que caracteriza a la investigación aplicada.

3.2. NIVEL DE INVESTIGACIÓN

El nivel descriptivo tiene como objetivo describir fenómenos. Corresponde al nivel más básico de conocimiento científico. Existe una gran variedad de estudios descriptivos. La observación es elemento básico. Algunos investigadores utilizan metodología cuantitativa y otros cualitativas (Cecilia salgado,2018, p.67).

La investigación descriptiva nos ayuda a poder medir o recoger información de

manera independiente o conjunta sobre las variables de interés, su objetivo no es analizar el impacto que tiene una variable sobre otra o entre sí (Hernández et al., 2014).

En tal sentido, la presente investigación se enfoca en el módulo de gestión de papeletas desarrollado bajo una arquitectura de microservicios en la Cooperativa Santa María Magdalena, Ayacucho, durante el año 2025. Para ello, el estudio describe la arquitectura implementada, los componentes del pipeline, los mecanismos de automatización y los indicadores operativos observados antes y después de la automatización.

3.3. DISEÑO DE LA INVESTIGACIÓN

Hernández et al. (2014), indican que la investigación no experimental es el estudio donde no se realiza la manipulación deliberada de variables, dado que las variables independientes ya sucedieron no se puede influir en ellas, se observan y analizan las variables en su contexto natural.

En el presente trabajo, la investigación corresponde a un diseño no experimental de tipo descriptivo. El diseño es no experimental porque no se manipulan variables con fines de experimentación ni se trabaja con grupo de control; se describe la implementación técnica realizada en un contexto institucional real y se analizan descriptivamente sus indicadores operativos.

Aunque la investigación compara indicadores operativos antes y después de la implementación de la solución DevOps, dicha comparación tiene carácter descriptivo y técnico. No se manipulan variables con fines experimentales, no se emplea grupo de control ni se aplican pruebas inferenciales; únicamente se contrastan los registros de la línea base del despliegue manual con los registros generados luego de la implementación del pipeline CI/CD, a fin de describir las diferencias operativas observadas.

3.4. POBLACIÓN Y MUESTRA

Para asegurar el rigor metodológico de la investigación y evitar la mezcla de objetos técnicos con personas, se han definido los siguientes componentes:

Unidad de análisis: Proceso de despliegue del módulo de gestión de papeletas basado en microservicios.

Población técnica: Procesos de despliegue de software desarrollados en la Cooperativa Santa María Magdalena – Ayacucho.

Muestra técnica: Proceso de despliegue del módulo de papeletas basado en microservicios.

Informantes: 3 desarrolladores y 2 responsables del área de Tecnologías de la Información (TI), vinculados al desarrollo, despliegue y supervisión del módulo de gestión de papeletas.

3.5. VARIABLES E INDICADORES

3.5.1. Definición Conceptual de las Variables

VARIABLE DE ESTUDIO

Implementación de principios DevOps:

Se refiere al grado de adopción y aplicación de prácticas y componentes fundamentales del enfoque DevOps para mejorar la calidad, eficiencia y agilidad en el desarrollo y entrega de software, (Albarran Salazar & Delgado Farro, 2024).

DIMENSIONES

A. **Diseño de la arquitectura DevOps utilizada:** Se refiere a la organización e integración de las herramientas, servicios y flujos de trabajo que conforman la arquitectura DevOps implementada para soportar el proceso de despliegue. Comprende elementos relacionados con el control de versiones, integración y entrega continua, automatización de infraestructura y mecanismos de seguimiento del proceso (Muñoz et al., 2020; Wiesche et al., 2022).
B. **Componentes del pipeline CI/CD implementado:** Comprende los componentes

y etapas que conforman el flujo de integración y entrega continua, incluyendo la gestión del código fuente, construcción, validación de calidad, generación de artefactos y despliegue hacia los entornos definidos. Estas etapas permiten estructurar un proceso automatizado y trazable para la entrega de software (Albarran Salazar & Delgado Farro, 2024; Atlassian, 2024).

C. **Mecanismos de automatización empleados en el despliegue:** Se refiere al conjunto de herramientas, configuraciones y scripts utilizados para automatizar las actividades asociadas al proceso de despliegue de software, reduciendo la necesidad de intervención manual en tareas repetitivas y permitiendo la ejecución estandarizada de los procedimientos definidos (Muñoz et al., 2020; Caridade, 2023).

D. **Beneficios operativos observados en el proceso de despliegue:** Se refiere a los resultados operativos observados en el proceso de despliegue tras la implementación de principios DevOps, considerando aspectos como el tiempo de despliegue, el nivel de intervención manual, la incorporación de controles automatizados de calidad, la seguridad de acceso y la trazabilidad del proceso.

3.5.2. Definición Operacional de las Variables

VARIABLE DE ESTUDIO

Implementación de principios DevOps

La variable de estudio se operacionaliza mediante cuatro dimensiones que permiten analizar la implementación de principios DevOps en el proceso de despliegue del módulo de gestión de papeletas: arquitectura DevOps utilizada, componentes del pipeline CI/CD implementado, mecanismos de automatización empleados y beneficios operativos observados en el proceso de despliegue.

DIMENSIONES

A. **Diseño de la arquitectura DevOps utilizada:** Se analiza mediante la identificación de las herramientas, servicios y flujos de trabajo que conforman la arquitectura DevOps implementada, considerando la integración entre el control de versiones, integración y entrega continua, análisis de calidad y automatización del despliegue.

B. **Componentes del pipeline CI/CD implementado:** Se analiza mediante la identificación de las etapas y herramientas que conforman el pipeline de integración y entrega continua, considerando los procesos de construcción, validación de calidad, generación de artefactos, despliegue automatizado en el entorno de pruebas y aprobación manual para el despliegue en producción.

C. **Mecanismos de automatización empleados en el despliegue:** Se analiza mediante la identificación de las herramientas, configuraciones y scripts utilizados para reducir la intervención manual y estandarizar las actividades del proceso de despliegue, considerando mecanismos como webhooks, pipelines, playbooks de Ansible, control de versiones y autenticación mediante llaves SSH.

D. **Beneficios operativos observados en el proceso de despliegue:** Se analizan mediante la comparación descriptiva entre la línea base del proceso manual y los registros obtenidos tras la implementación de principios DevOps, considerando indicadores como el tiempo de despliegue, la intervención manual, el control de calidad automatizado, la seguridad de acceso y la trazabilidad del proceso.

La variable de estudio "Implementación de principios DevOps" se analiza mediante cuatro dimensiones que corresponden directamente con los problemas y objetivos específicos establecidos en la investigación: arquitectura DevOps utilizada, componentes del pipeline CI/CD implementado, mecanismos de automatización

empleados y beneficios operativos observados en el proceso de despliegue. Cada dimensión fue evaluada mediante indicadores verificables a través de observación directa y análisis documental, asegurando la correspondencia metodológica entre los objetivos específicos, las dimensiones y los resultados presentados en la investigación.

Los indicadores fueron seleccionados considerando su correspondencia con los objetivos específicos, su pertinencia respecto a la arquitectura DevOps implementada y su verificabilidad mediante los registros de GitLab, Jenkins, SonarQube y Ansible, la ejecución del pipeline, la configuración de los servidores y la documentación técnica. Esta selección permite mantener la coherencia entre las dimensiones de la variable, los instrumentos aplicados, las evidencias recopiladas y los resultados presentados.

Tabla 1

Definición Operacional de las Variables

Variables	Dimensión	Indicador	Forma de medición	Escala	Fuente de datos	Técnica	Instrumento
Implementación de principios DevOps	**Arquitectura DevOps**	Número de herramientas integradas	Conteo de herramientas	Razón	Entorno TI (Servidores)	Observación directa	Ficha de observación
		Nivel de estandarización del proceso	Porcentaje (%)	Razón	Manuales y procesos de TI	Análisis documental	Ficha documental
		Existencia de documentación técnica	Presencia / Ausencia	Nominal	Repositorio institucional	Observación directa	Ficha de observación
	Pipeline CI/CD	Etapas automatizadas vs. manuales	Conteo de etapas	Razón	Configuración del Jenkinsfile	Observación directa	Ficha de observación
		Tiempo de integración y pruebas	Minutos	Razón	Logs de Jenkins y SonarQube	Observación directa	Ficha de observación
		Trazabilidad del despliegue	Presencia/Ausencia	Nominal	GitLab y Jenkins	Análisis documental	Ficha documental
	Mecanismos de automatización	Tareas repetitivas automatizadas	Porcentaje (%)	Razón	Playbooks de Ansible / Scripts	Observación directa	Ficha de observación
		Uso de playbooks y scripts automatizados	Presencia/Ausencia	Nominal	Playbooks de Ansible y Jenkinsfile	Análisis documental	Ficha documental
		Nivel de intervención humana	Porcentaje (%)	Razón	Registros del proceso de despliegue	Observación directa	Ficha de observación
		Tiempo de despliegue	Diferencia porcentual entre línea base y despliegue automatizado.	Razón	Logs de Jenkins	Observación directa	Ficha de observación

	beneficios operativos observados	Control de calidad automatizado	Presencia/Ausencia de Quality Gate	Nominal	SonarQube y registros de calidad	Análisis documental	Ficha documental
		Seguridad de acceso mediante SSH/Tokens	Presencia/Ausencia	Nominal	Configuración de servidores y llaves SSH	Análisis documental	Ficha documental
		Mejora de la trazabilidad	Presencia/Ausencia de registros automáticos	Nominal	GitLab y Jenkins	Análisis documental	Ficha documental

Fuente: elaboración propia

3.6. TÉCNICAS E INSTRUMENTOS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

3.6.1. Técnicas para Recolectar Información

- ✓ **Análisis Documental:** Técnica mediante la cual se extrae información pertinente sobre la configuración y los despliegues. Esta técnica incluye explícitamente la revisión de logs de Jenkins, la revisión de reportes SonarQube, la revisión de repositorios GitLab y la revisión de logs Ansible.
- ✓ **Observación Directa:** Técnica utilizada para registrar in situ el desempeño del proceso de entrega de software en los servidores de la cooperativa, aplicable tanto al esquema tradicional como al automatizado.

3.6.2. Instrumentos para Recolectar Información

Los instrumentos diseñados para registrar y sistematizar la información recopilada durante la investigación son los siguientes:

- ✓ **Ficha de observación directa:** Instrumento utilizado para registrar los tiempos, actividades y procedimientos ejecutados durante el proceso de despliegue manual del módulo de gestión de papeletas, permitiendo establecer la línea base operativa previa a la implementación de la solución DevOps.
- ✓ **Ficha de análisis documental:** Empleada para recopilar y registrar información proveniente de documentos técnicos, configuraciones de infraestructura, registros de ejecución (logs), reportes generados por Jenkins, resultados de análisis de calidad de código en SonarQube y demás evidencias técnicas relacionadas con el proceso de despliegue.
- ✓ **Ficha de registro de ejecución del pipeline CI/CD**: Utilizada para documentar los resultados obtenidos durante la ejecución de los pipelines automatizados, incluyendo los tiempos de construcción, validación, análisis de calidad y

despliegue, así como el estado de éxito o fallo de cada etapa del proceso automatizado.

3.6.3. Herramientas para el Tratamiento de Datos e Información

Para el procesamiento de la información recolectada durante la implementación del ecosistema DevOps, se utilizarán herramientas que permitan transformar datos brutos en indicadores de rendimiento (KPIs):

- ✓ **Microsoft Excel / Google Sheets:** Para la consolidación de las fichas de observación y el cálculo de los tiempos de despliegue (Línea Base vs. Post-Intervención). Se empleará para generar las tablas comparativas de eficiencia porcentual.
- ✓ **Analíticas de Jenkins:** Se utilizarán los complementos de visualización de Jenkins (Build History y Build Stats) para extraer datos precisos sobre la duración de cada etapa del pipeline (Build, Test, Deploy).
- ✓ **Dashboards de SonarQube:** Herramienta principal para el tratamiento de datos de calidad de software, permitiendo cuantificar la densidad de errores, vulnerabilidades y el porcentaje de deuda técnica.
- ✓ **Logs de Ansible:** Para verificar la consistencia de las tareas de automatización y la tasa de éxito de los despliegues en los nodos pruebas y producción.

3.6.4. Diseño Estadístico

Dado que la investigación es de tipo aplicada y de nivel descriptivo, con un diseño no experimental y transversal, no se aplican pruebas de hipótesis ni procedimientos estadísticos inferenciales. En su lugar, el análisis estadístico se desarrollará mediante técnicas descriptivas orientadas a caracterizar los indicadores operativos observados durante la implementación de la solución DevOps.

- ✓ **Estadística descriptiva:** Se utilizarán medidas de tendencia central, principalmente la media aritmética (promedio), para determinar los tiempos promedio de despliegue antes y después de la implementación de la solución.
- ✓ **Análisis comparativo:** Se empleará el cálculo de variación porcentual para describir las diferencias observadas entre el proceso de despliegue manual y el proceso automatizado mediante DevOps.
- ✓ **Tabulación de resultados:** Los datos obtenidos mediante las fichas de observación y análisis documental serán organizados en tablas de frecuencias, tablas comparativas y cuadros resumen para facilitar su interpretación.
- ✓ **Representación gráfica**: Los resultados se presentarán mediante gráficos de barras y tablas comparativas que permitan visualizar las diferencias observadas en los indicadores de tiempo de despliegue, automatización, trazabilidad y control de calidad.

3.6.5. Análisis e Interpretación de Datos

El análisis de la información se realizará mediante un enfoque descriptivo, orientado a interpretar los resultados obtenidos durante la implementación de principios DevOps en el proceso de despliegue automatizado del módulo de gestión de papeletas.

- ✓ **Organización de la información:** Los datos recopilados serán clasificados de acuerdo con las dimensiones definidas en la matriz de operacionalización: arquitectura DevOps, pipeline CI/CD, mecanismos de automatización y beneficios operativos.
- ✓ **Integración de fuentes:** Los resultados obtenidos mediante la ficha de observación directa serán contrastados con la información recopilada mediante la ficha de análisis documental, permitiendo verificar la consistencia de los datos registrados durante la implementación.

- ✓ **Análisis comparativo:** Se evaluarán descriptivamente las diferencias observadas entre el proceso de despliegue manual y el proceso automatizado, considerando indicadores como tiempo de despliegue, nivel de automatización, trazabilidad y control de calidad.
- ✓ **Interpretación de resultados:** Los hallazgos serán interpretados en función de los objetivos de investigación y del marco teórico relacionado con DevOps, CI/CD, automatización e infraestructura como código, identificando los beneficios operativos observados tras la implementación de la solución.

3.6.6. Técnicas para Aplicar el Marco de Trabajo CALMS

Para asegurar que la implementación cumpla con los estándares internacionales de DevOps, se aplicarán técnicas específicas para cada pilar del modelo CALMS:

- ✓ **Culture (Cultura):** Técnica de Gestión del Cambio, mediante la involucración de los desarrolladores en la definición de las reglas de SonarQube, promoviendo la responsabilidad compartida entre desarrollo y operaciones.
- ✓ **Automation (Automatización):** Técnica de Infraestructura como Código (IaC) utilizando Ansible. Se automatizará el ciclo de vida completo del microservicio, eliminando procesos manuales y garantizando la repetibilidad en los servidores de la cooperativa.
- ✓ **Lean (Esbelto):** Esta técnica se aplicó como estrategia fundamental para la optimización del flujo de valor, centrándose en la identificación y eliminación sistemática de desperdicios presentes en el proceso tradicional de la institución. Mediante un análisis de los cuellos de botella operativos, se priorizó la supresión de tareas manuales que no aportaban valor técnico como la transferencia de archivos por SFTP, la detención manual de procesos y la configuración empírica de servidores, las cuales generaban tiempos de espera excesivos y riesgos de error humano. Al implementar principios de entregas en lotes pequeños y flujo

continuo, la metodología Lean permitió transformar un ciclo de despliegue fragmentado en un proceso fluido y automatizado, reduciendo drásticamente los tiempos muertos y asegurando que cada etapa del pipeline contribuya directamente a la agilidad y calidad de la entrega final.

- ✓ **Measurement (Medición):** Técnica de Establecimiento de Telemetría, integrando logs y métricas de éxito en cada etapa del pipeline de Jenkins para obtener visibilidad total del estado de los despliegues.
- ✓ **Sharing (Compartir):** Técnica de Documentación Técnica Colaborativa, mediante el uso de repositorios en GitLab y manuales de configuración que permiten que el conocimiento técnico sea accesible para todo el equipo de la Cooperativa.

3.7. PROCEDIMIENTO METODOLÓGICO PARA LA IMPLEMENTACIÓN DE LA SOLUCIÓN DEVOPS

El procedimiento metodológico permitió organizar las etapas técnicas seguidas para implementar la solución DevOps y describir los indicadores operativos observados antes y después de la automatización, sin establecer relaciones causales ni aplicar un diseño experimental.

3.7.1. Diagnóstico del proceso actual de despliegue

Para comprender la situación inicial del proceso de despliegue del módulo de gestión de papeletas, se realizó un diagnóstico basado en la observación directa de las actividades operativas y en el análisis documental de registros técnicos generados durante los despliegues. Para ello, se revisaron logs de ejecución, registros históricos de incidencias, procedimientos operativos y evidencias de despliegues manuales utilizados por el área de Tecnologías de la Información.

Este diagnóstico permitió identificar y documentar las tareas manuales que generaban mayores tiempos operativos y riesgos de error humano, incluyendo la

compilación local de los proyectos, la transferencia de archivos mediante SFTP, la ejecución manual de comandos en los servidores y las actividades de validación posteriores al despliegue. Asimismo, permitió establecer la línea base de los tiempos de despliegue y caracterizar las principales limitaciones del proceso tradicional antes de la implementación de los principios DevOps.

3.7.2. Levantamiento de la línea base del despliegue manual

La línea base operativa se construyó cronometrando el método manual tradicional para tener un punto de comparación exacto. Para asegurar la precisión de los datos, la medición se apoyó en fichas de observación estructuradas, abarcando tres escenarios específicos de la rutina del equipo: la actualización de un solo microservicio, un despliegue promedio estándar y un escenario de carga máxima.

3.7.3. Diseño de la arquitectura DevOps propuesta

La nueva arquitectura técnica se estructuró en torno a un ecosistema integrado que cubre todo el ciclo de vida del software. Se definió a GitLab como el repositorio central y a Jenkins como el motor orquestador (CI/CD), apoyados por SonarQube para auditar la calidad del código y Ansible para automatizar la infraestructura. Todo esto operando sobre servidores dedicados de forma exclusiva para separar los entornos de Calidad (QA) y Producción.

La arquitectura está conformada por los siguientes componentes:

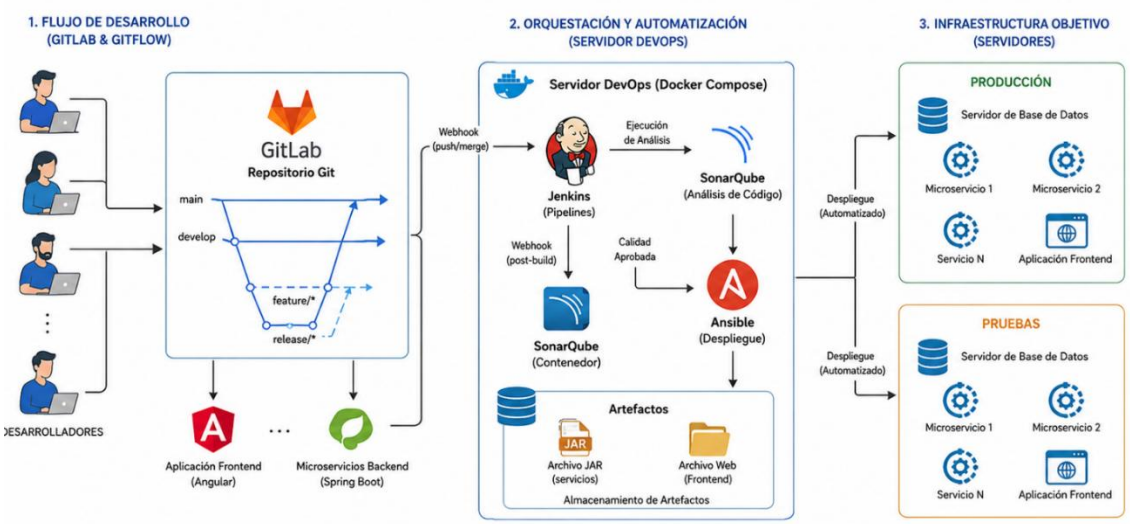
- ✓ **Servidor DevOps:** Nodo principal encargado de centralizar los servicios de automatización y coordinación de los procesos DevOps.
- ✓ **Servidor GitLab:** Responsable del almacenamiento y control de versiones del código fuente de los microservicios. Asimismo, permite la gestión de ramas, seguimiento de cambios y colaboración entre los desarrolladores.
- ✓ **Servidor Jenkins:** Encargado de la ejecución automática de los pipelines CI/CD. Este servidor realiza las actividades de compilación, validación, integración continua

y coordinación de despliegues.

- ✓ **Servidor SonarQube:** Utilizado para el análisis estático del código fuente y la validación de estándares de calidad mediante la aplicación de Quality Gate dentro del pipeline.
- ✓ **Servidor de Calidad (QA):** Entorno destinado a la validación funcional y técnica de los microservicios antes de su liberación al entorno productivo.
- ✓ **Servidor de Producción:** Entorno final donde se ejecutan los servicios utilizados por los usuarios de la Cooperativa Santa María Magdalena.
- ✓ La distribución de nodos implementada permitió separar adecuadamente los entornos de desarrollo, validación y producción, garantizando un mayor control sobre el proceso de despliegue y reduciendo riesgos asociados a cambios directos en producción.

Para visualizar la interconexión de las herramientas que sustentan estas dimensiones, en la Figura N°5 se presenta la arquitectura lógica del ecosistema implementado, la cual integra GitLab, Jenkins, SonarQube y Ansible.

Figura 5
Arquitectura del ecosistema DevOps para la gestión de papeletas



***Nota*:** *Representación lógica del flujo de integración continua y entrega continua implementado. La arquitectura comprende la gestión del código fuente en GitLab, la*

activación automática mediante Webhooks, la orquestación del pipeline con Jenkins, la compilación mediante Maven, el análisis de calidad con SonarQube y el despliegue automatizado mediante Ansible hacia los entornos de pruebas y producción. Jenkins y SonarQube se ejecutan en el servidor DevOps mediante Docker Compose. Elaboración propia, 2025. Elaboración propia.

El flujo general de la arquitectura implementada se inicia con la gestión del código fuente en GitLab y la activación automática del pipeline mediante un Webhook. Jenkins orquesta la compilación con Maven, la generación del artefacto, el análisis de calidad mediante SonarQube y el despliegue automatizado con Ansible hacia el entorno de pruebas. Después de la validación funcional en QA, el paso hacia producción requiere una aprobación manual en Jenkins; una vez otorgada, Ansible ejecuta el despliegue automatizado en el entorno productivo.

La arquitectura implementada utiliza Docker Compose para ejecutar Jenkins y SonarQube dentro del servidor DevOps. Jenkins orquesta las etapas del pipeline, mientras que SonarQube realiza el análisis estático del código y valida el cumplimiento del Quality Gate. Por su parte, Ansible automatiza la transferencia y el despliegue de los artefactos generados hacia los entornos de pruebas y producción.

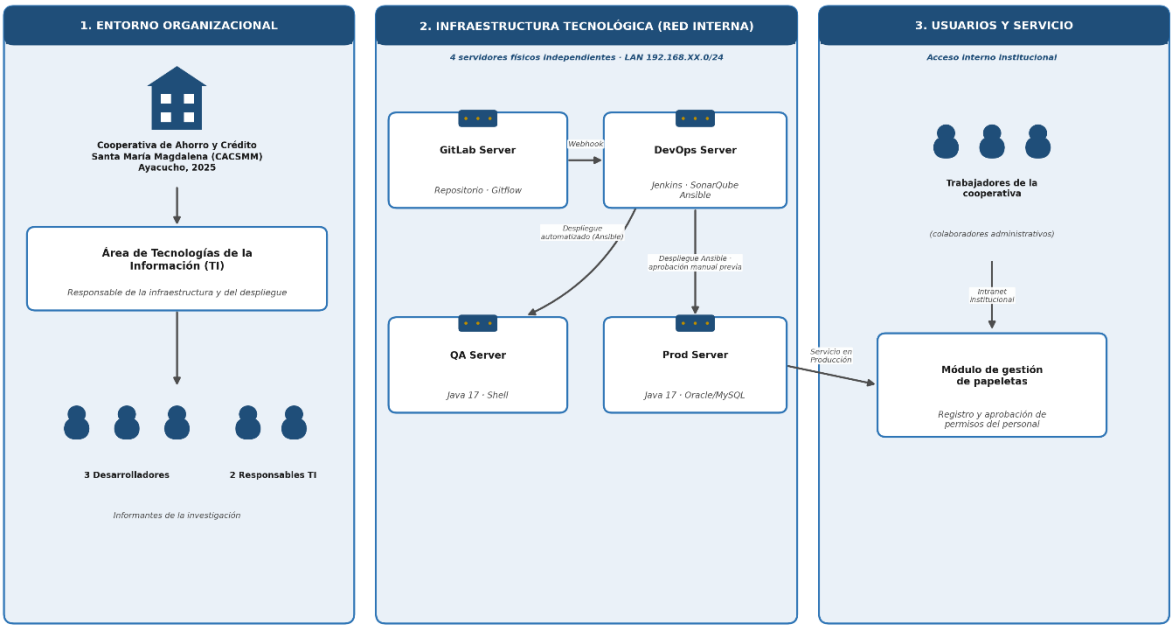
Los microservicios backend se distribuyen como archivos ejecutables .jar, mientras que la aplicación frontend se entrega como un artefacto web. En consecuencia, dentro del alcance de la investigación, Docker se utiliza para alojar los servicios de soporte del ecosistema DevOps, pero no para contenerizar las aplicaciones desplegadas en los servidores de pruebas y producción.

Tabla 2
Especificación de la Infraestructura Implementada

Entorno	Dirección IP	Rol / Función	Tecnologías Clave
GitLab	192.168.XX.XX	Repositorio, master, calidad, desarrollo, feature/*	GitLab, Gitflow
DevOps Server	192.168.XX.XX	Orquestador central (CI/CD)	Jenkins, SonarQube, Ansible
QA Server	192.168.XX.XX	Pruebas y Calidad	Java 17, Shell Scripts
Prod Server	192.168.XX.XX	Producción (Entorno Estable)	Java 17, Oracle DB,MySql

***Fuente**: Elaboración propia.*

Figura 6 *Infraestructura del entorno de la organización*



Nota: Representación del entorno organizacional en el que se implementó la solución DevOps. GitLab, DevOps, QA y Producción operan en cuatro servidores físicos independientes dentro de la red interna de la Cooperativa de Ahorro y Crédito Santa María Magdalena (CACSMM). El módulo de gestión de papeletas es de uso interno: permite a los trabajadores de la cooperativa registrar y gestionar sus permisos administrativos. Elaboración propia, 2025.

La Figura 6 muestra el entorno organizacional en el que se desplegó la solución, evidenciando que los servidores GitLab, DevOps, QA y Producción operan en cuatro nodos físicos independientes dentro de la red interna de la cooperativa, y que el módulo de gestión de papeletas es utilizado por los trabajadores de la institución a través de la intranet institucional.

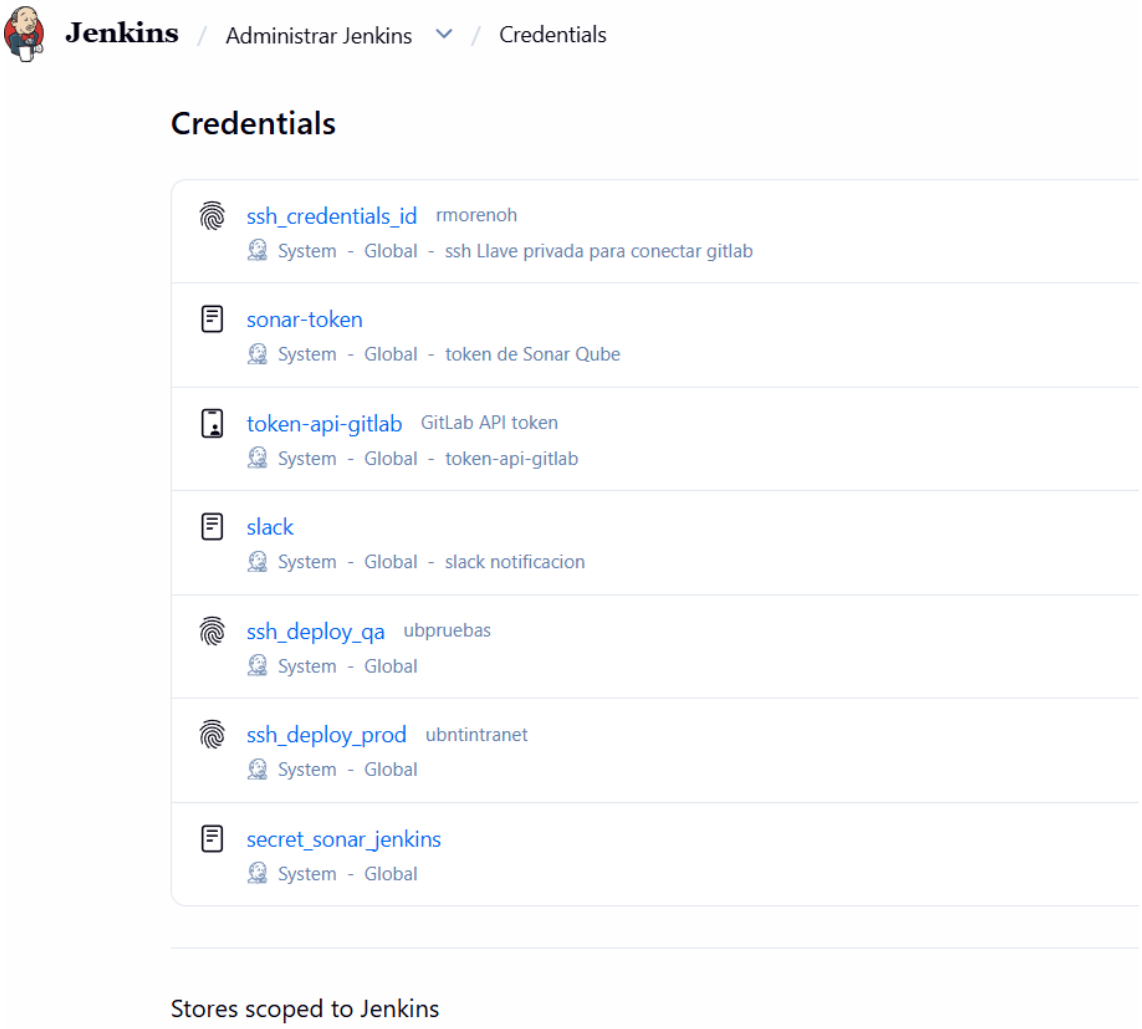
3.7.4. Configuración de seguridad y acceso entre servidores

Con el fin de erradicar el uso de contraseñas vulnerables en texto plano y permitir operaciones desatendidas, se configuró una red de confianza entre las herramientas y los nodos de destino. Esto se logró fortaleciendo la seguridad de la comunicación mediante claves SSH, tokens cifrados y políticas estrictas de permisos, asegurando un tránsito de

datos robusto entre GitLab, Jenkins, SonarQube y Ansible.

La utilización de claves RSA eliminó la necesidad de autenticación manual mediante contraseñas durante la ejecución de los procesos automatizados, mientras que los tokens permitieron la integración segura entre las herramientas involucradas en el pipeline CI/CD.

Figura 7
Configuración ssh y token

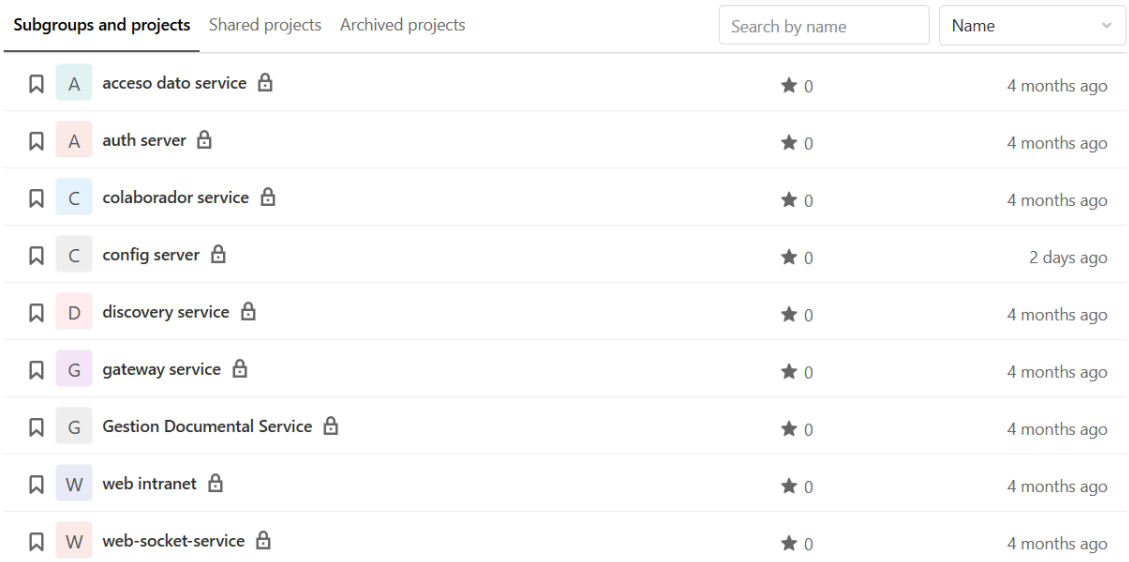


Nota: Configuración de credenciales globales en Jenkins. Se gestionan de forma centralizada los tokens de SonarQube y GitLab, así como las llaves SSH privadas, garantizando una autenticación segura y desatendida entre los servidores, cumpliendo con las prácticas iniciales de seguridad.

3.7.5. Configuración de GitLab y estrategia de ramas

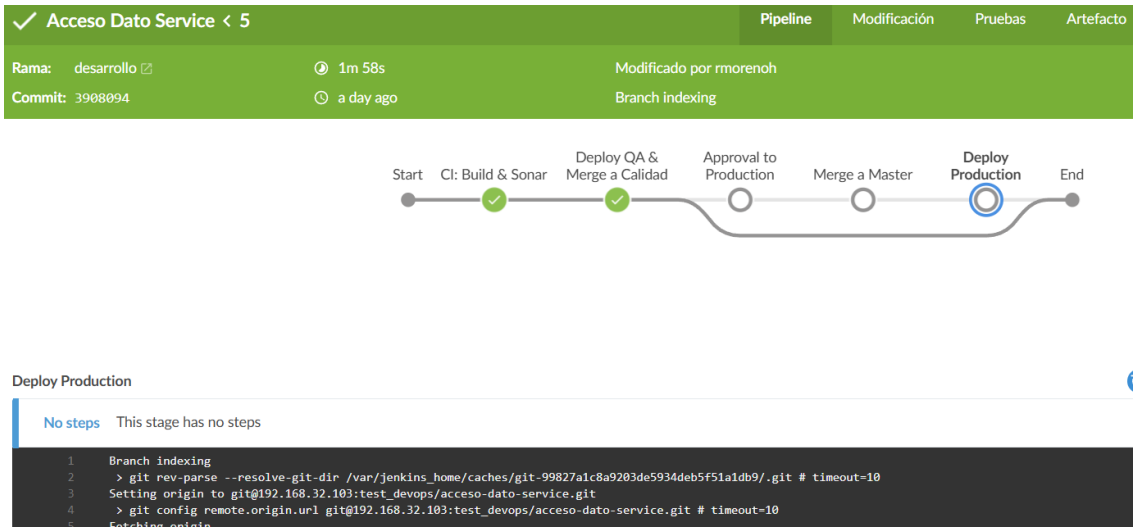
La gestión del código fuente se organizó adoptando el modelo Gitflow. Al segmentar el trabajo en ramas específicas (feature, desarrollo, calidad y master), el equipo obtuvo un control riguroso sobre cada cambio, garantizando la trazabilidad histórica y asegurando que ninguna nueva característica avance sin pasar por un proceso formal de maduración.

Figura 8
Repositorio de GitLab de los proyectos



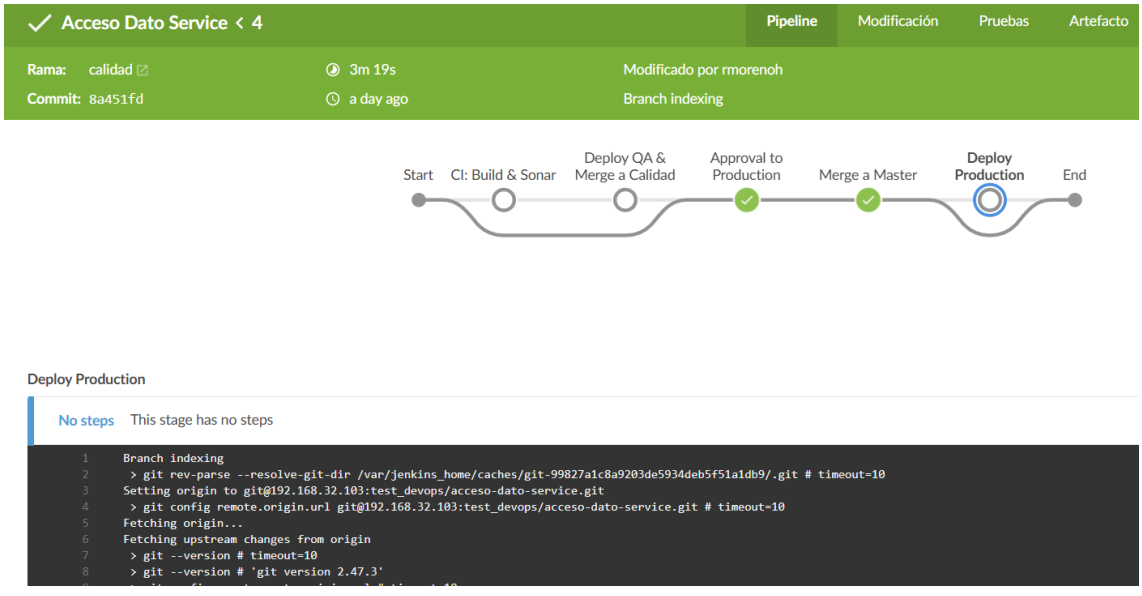
Nota: *Captura de imagen de GitLab de los proyectos para la gestión de papeletas basado en microservicios.*

Figura 9
Flujo de proceso de rama de desarrollo a la rama Calidad



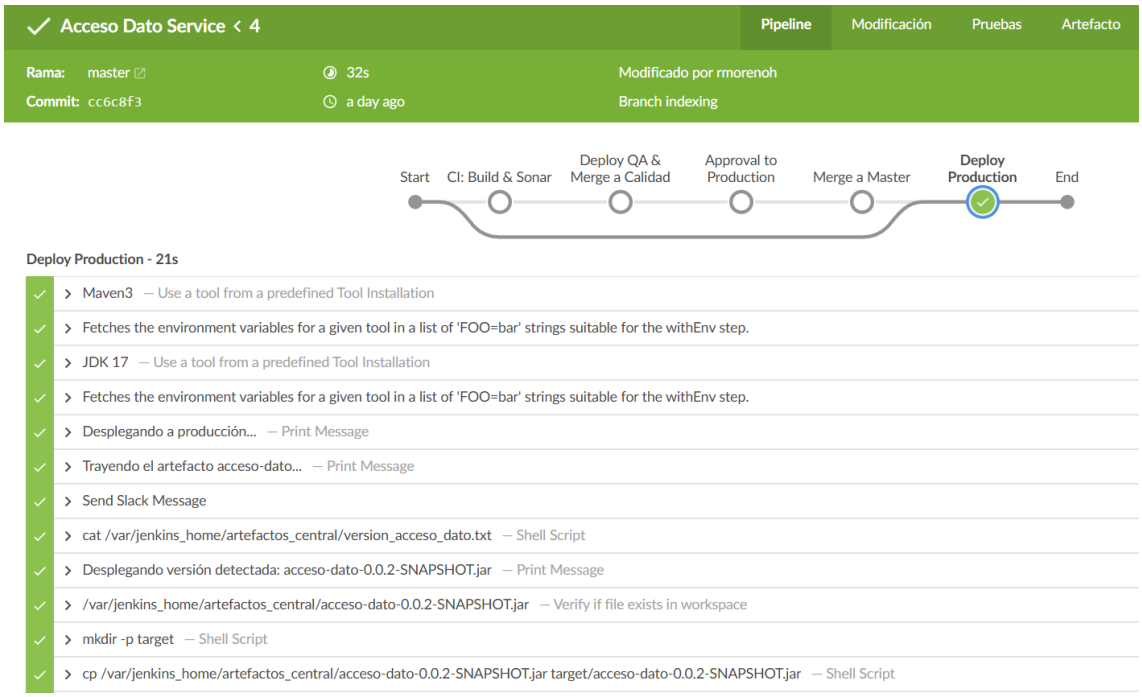
Nota: *Visualización del pipeline a través de Jenkins Blue Ocean. Se evidencia la ejecución secuencial y exitosa de las etapas del ciclo: Checkout, Build, análisis de calidad (Quality Gate) y despliegue automatizado, registrando los tiempos exactos de cada fase operativa.*

Figura 10
Flujo de proceso de rama calidad a la rama master



Nota: *Aprobación de usuario para la realización del merge automático y despliegue*

Figura 11
Flujo de despliegue a producción



Nota: *merge de la rama de calidad a la rama master y despliegue en servidor de producción. Elaboración propia.*

La estrategia de gestión de versiones se implementó utilizando el modelo Gitflow. Este enfoque permitió organizar el desarrollo mediante ramas especializadas para nuevas funcionalidades, integración, validación y liberación a producción.

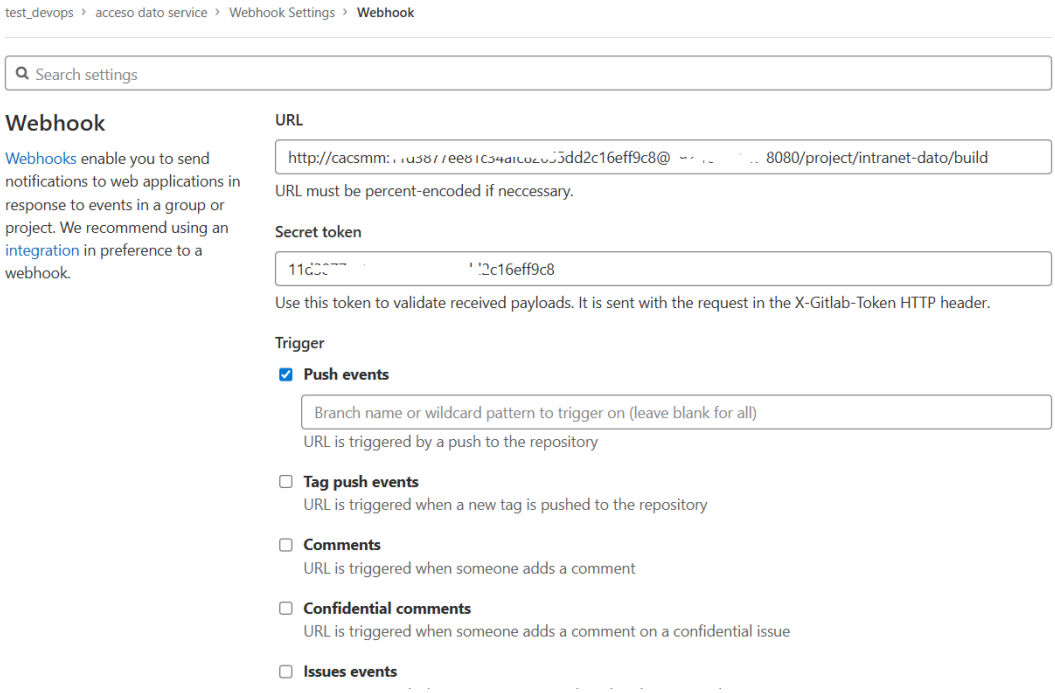
Las ramas feature fueron utilizadas para el desarrollo individual de funcionalidades. Posteriormente, los cambios fueron integrados en la rama desarrollo para pruebas internas. Una vez completadas las validaciones correspondientes, los cambios fueron promovidos hacia la rama calidad y finalmente hacia la rama master para su despliegue en producción.

3.7.6. Configuración del pipeline CI/CD con Jenkins

El núcleo de la automatización recayó en Jenkins, el cual se configuró mediante pipelines multirama diseñados para "escuchar" los eventos de GitLab a través de webhooks. De este modo, cada actualización de código en el repositorio desencadena automáticamente un flujo secuencial que incluye la compilación, el análisis estático, la

generación del artefacto inmutable y su posterior despliegue.

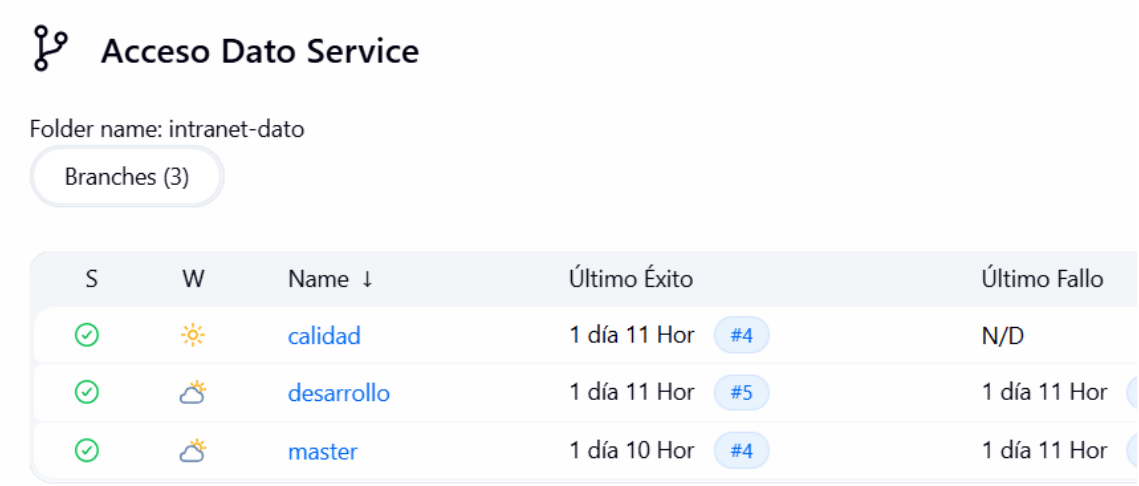
Figura 12
Configuración de webhook por microservicio



Nota: Detalle del Webhook configurado en GitLab. Este disparador está programado para enviar un payload al puerto del orquestador Jenkins ante cada evento de tipo "Push", iniciando automáticamente el ciclo de construcción sin intervención manual.

Elaboración propia.

Figura 13
Configuración de la tarea Multibranch Pipeline en Jenkins



Nota: Interfaz de Jenkins mostrando el descubrimiento automático de ramas. El orquestador identifica el estado de salud, los fallos y la duración del historial de construcción de cada rama (master, calidad, desarrollo) de forma aislada, garantizando una integración continúa estructurada. Elaboración propia.

La automatización del proceso CI/CD fue implementada mediante Jenkins utilizando Pipelines Multibranch (multirrama). Cada repositorio fue configurado para notificar automáticamente los cambios realizados mediante webhooks de GitLab.

El pipeline ejecuta de manera secuencial las etapas de compilación, análisis de calidad, generación de artefactos y despliegue automatizado. Esta configuración permitió reducir significativamente la intervención manual y asegurar la ejecución uniforme de los procedimientos de integración y entrega continua.

3.7.7. Integración de control de calidad con SonarQube

Con el propósito de incorporar mecanismos de control de calidad dentro del ciclo de integración continua, se integró SonarQube al pipeline CI/CD implementado. Esta herramienta fue configurada para realizar análisis estático del código fuente durante cada ejecución del pipeline, permitiendo identificar de manera temprana defectos potenciales,

vulnerabilidades de seguridad y problemas de mantenibilidad.

Para prevenir la incorporación de deuda técnica en los microservicios institucionales, SonarQube se estableció como una etapa obligatoria dentro del flujo automatizado. La validación se realizó mediante la utilización de Quality Gate, los cuales establecen criterios mínimos de aceptación relacionados con calidad, confiabilidad, mantenibilidad y seguridad del código.

Solo aquellos proyectos que cumplen los estándares definidos continúan con las siguientes etapas del proceso automatizado, mientras que los desarrollos que no superan dichas validaciones son detenidos automáticamente hasta la corrección de las observaciones identificadas.

Figura 14
Quality Gate en SonarQube



Nota. *La figura muestra el panel de resultados generado por SonarQube para el microservicio analizado. La herramienta permite identificar métricas relacionadas con seguridad, confiabilidad, mantenibilidad y duplicación de código, proporcionando información para la evaluación de la calidad del software antes de su despliegue. En la presente investigación, SonarQube fue integrado al pipeline CI/CD como mecanismo de control de calidad dentro del proceso de automatización.*

3.7.8. Automatización del despliegue con Ansible

La fase final de entrega se confió a Ansible bajo el paradigma de Infraestructura como Código (IaC). Mediante el uso de inventarios y playbooks especializados para los servidores de Calidad (QA) y Producción, se logró que cada paso a producción deje de

ser un proceso con alto riesgo operativo para convertirse en una operación repetible, consistente y completamente trazable.

Para la implementación se configuró un inventario centralizado que permitió administrar los servidores de destino desde el entorno DevOps. Asimismo, se desarrollaron playbooks encargados de automatizar tareas relacionadas con la validación de conectividad, transferencia de artefactos, actualización de microservicios, reinicio de servicios y verificación de la correcta ejecución de las aplicaciones desplegadas.

Previo a cada despliegue, Ansible realiza una validación automática de acceso hacia los servidores de destino mediante conexiones seguras SSH. Posteriormente, ejecuta las tareas definidas en los playbooks, garantizando que las configuraciones y procedimientos se realicen de manera uniforme en todos los entornos administrados.

Las Figuras 9, 10, 15 y 16 presentan la validación de conectividad, la ejecución de tareas automatizadas y la estructura del playbook utilizado para el despliegue de los microservicios institucionales.

Figura 15
Verificación de conexión de Ansible

```
root@ubntdevops:/home/cacsmm# docker exec -u 0 -it jenkins bash
root@82c6428c9caf:/# ansible qa -m ping
server_qa | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
root@82c6428c9caf:/# ansible prod -m ping
server_prod | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
root@82c6428c9caf:/#
```

Nota: Evidencia de instalación de Ansible. Elaboración propia.

Figura 16

Configuración de Playbook de Ansible

deploy.yml

BACKEND > acceso-dato-service > deploy.yml

```yaml
---
- name: Despliegue de Microservicio CACSMM
  hosts: "{{ target_env }}"
  become: no
  tasks:
    - name: 1. Asegurar estructura de directorios
      file:
        path: "{{ base_dir }}"
        state: directory
        mode: '0755'

    - name: 2. Sincronizar script de gestión
      copy:
        src: "{{ playbook_dir }}/scripts/ejecutar_backend_devops.sh"
        dest: "{{ base_dir }}/ejecutar_backend_devops.sh"
        mode: '0755'

    - name: 3. Transferir nuevo artefacto JAR (.new)
      copy:
        src: "{{ item }}"
        #src: "{{ playbook_dir }}/target/{{ service_name }}-*.jar"
        dest: "{{ base_dir }}/{{ service_name }}.jar.new"
        owner: "{{ ansible_user }}"
        group: "{{ ansible_user }}"
        mode: '0755'
      with_fileglob:
        - "target/*.jar"

    - name: 4. Ejecutar script de despliegue (Backup y Reinicio)
      shell: "bash {{ base_dir }}/ejecutar_backend_devops.sh {{ service_name }}"
      register: deploy_output

    - name: 5. Mostrar resumen de despliegue
      debug:
        var: deploy_output.stdout_lines
```

Nota: Estructura del Playbook de Ansible como Infraestructura como Código (IaC). El archivo YAML define las tareas secuenciales y automatizadas: creación de directorios,

transferencia del artefacto (.jar) y ejecución de scripts de reinicio de forma idempotente. Elaboración propia.

3.7.9. Aplicación del modelo CALMS como marco de análisis

Con el propósito de realizar una evaluación integral de la implementación DevOps, se adoptó el modelo CALMS como marco de referencia para el análisis de los resultados obtenidos. Este modelo permitió organizar la evaluación considerando tanto aspectos técnicos como organizacionales involucrados en el proceso de transformación.

Las dimensiones consideradas fueron Cultura (Culture), Automatización (Automation), Lean, Medición (Measurement) y Compartir (Sharing), las cuales constituyen los pilares fundamentales para la adopción de prácticas DevOps dentro de una organización.

La dimensión Cultura permitió analizar los mecanismos de colaboración y responsabilidad compartida entre los participantes del proceso. La dimensión Automatización se utilizó para describir la incorporación de herramientas y procedimientos automatizados dentro del ciclo de vida del software. La dimensión Lean permitió identificar la reducción de actividades manuales y procesos repetitivos. La dimensión Medición se orientó al análisis de indicadores operativos relacionados con los despliegues, mientras que la dimensión Compartir permitió evaluar los mecanismos utilizados para la gestión y difusión del conocimiento técnico.

La aplicación de este marco permitió estructurar de manera sistemática la presentación y análisis de los resultados obtenidos, los cuales se desarrollan en el Capítulo IV de la presente investigación.

3.7.10. Procedimiento de medición y análisis descriptivo

Con el propósito de evaluar descriptivamente la implementación de la solución DevOps, se definieron indicadores operativos relacionados con el proceso de despliegue

del módulo de gestión de papeletas. Los indicadores considerados fueron: tiempo de despliegue manual, tiempo de despliegue automatizado, variación porcentual del tiempo de despliegue, nivel de intervención humana, trazabilidad del proceso y mecanismos de control de calidad incorporados al pipeline CI/CD.

La medición de la línea base se realizó mediante observación directa y registro cronométrico de las actividades ejecutadas durante el proceso tradicional de despliegue. Para ello, se consideraron escenarios representativos que incluyeron el despliegue individual de microservicios, despliegues promedio y escenarios de máxima carga operativa.

Posteriormente, se registraron los tiempos correspondientes al proceso automatizado implementado mediante GitLab, Jenkins, SonarQube y Ansible, siguiendo los mismos criterios utilizados durante el levantamiento de la línea base, con el fin de garantizar la comparabilidad de los datos obtenidos.

La variación porcentual de los tiempos de despliegue fue calculada mediante la siguiente expresión:

Mejora (%) = ((Tiempo Manual - Tiempo Automatizado) / Tiempo Manual) × 100

Los datos obtenidos fueron procesados mediante estadística descriptiva, utilizando promedios, porcentajes y análisis comparativos entre el escenario manual y el escenario automatizado. Los resultados obtenidos permitieron describir las diferencias operativas observadas tras la implementación de principios DevOps, sin establecer relaciones causales ni aplicar pruebas de inferencia estadística.

La comparación entre la línea base del proceso manual y los registros posteriores a la implementación se realiza para responder al objetivo específico D, orientado a describir los beneficios operativos observados tras la aplicación de principios DevOps.

Esta comparación permite analizar descriptivamente indicadores como tiempo de despliegue, intervención humana, trazabilidad, control de calidad y seguridad de acceso.

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

El presente capítulo desarrolla los resultados obtenidos a partir de la implementación de los principios DevOps en el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, durante el año 2025. Los resultados se presentan de manera descriptiva y ordenada, en correspondencia directa con el objetivo general y los objetivos específicos de la investigación, considerando un análisis comparativo del estado previo y posterior a la implementación, así como una discusión fundamentada en la literatura especializada.

4.1. RESULTADOS DE LA INVESTIGACION

4.1.1. Resultados de la línea base del despliegue manual

Con el propósito de establecer una referencia para el análisis de la implementación de principios DevOps, se realizó el levantamiento de la línea base correspondiente al proceso tradicional de despliegue utilizado en la Cooperativa Santa María Magdalena. La evaluación se enfocó en registrar los tiempos operativos requeridos para la actualización de los microservicios que conforman el módulo de gestión de papeletas, considerando los entornos de pruebas (QA) y producción.

Las mediciones fueron realizadas mediante observación directa y registro cronométrico de las actividades ejecutadas por el personal del área de Tecnologías de la Información. Entre las actividades observadas se incluyeron la compilación de proyectos, transferencia manual de artefactos, validaciones operativas y reinicio de servicios en los servidores correspondientes.

Los resultados obtenidos permitieron identificar los tiempos requeridos para distintos escenarios de despliegue, constituyendo la línea base utilizada posteriormente para el análisis comparativo con el proceso automatizado implementado mediante principios DevOps.

Tabla 3
Tiempo de despliegue en un microservicio

Actividad Manual	Tiempo Estimado (min)	Riesgo Asociado
Compilación y empaquetado de microservicios	5	Error de dependencias
Transferencia de artefactos mediante SFTP	3	Interrupción de red
Detención de servicios anteriores	2	Error en el PID (Kill erróneo)
copia y reemplazo del artefacto	5	Sobreescritura errónea
Inicio y validación del servicio desplegado	10	Error de configuración, puertos o variables
Validación funcional visual	20	Omisión de errores en Producción
Tiempo Total	45.0 min	Alta Criticidad

Fuente: Ficha de observación del proceso de despliegue manual, 2025.

La Tabla 3 presenta el tiempo requerido para realizar el despliegue manual de un microservicio. Los resultados muestran que las actividades asociadas a compilación, transferencia de archivos, validación y puesta en marcha demandan una cantidad considerable de tiempo operativo, debido a la necesidad de intervención directa del personal técnico en cada una de las etapas del proceso.

Tiempo de Despliegue de una Release Completa

En condiciones normales de operación no siempre se desplegaban todos los microservicios del sistema, ya que los cambios funcionales generalmente afectaban únicamente algunos componentes específicos. Por ello, se consideró un escenario promedio de despliegue de tres microservicios y el frontend web.

Tabla 4
Tiempo promedio de despliegue por proyecto (entorno pruebas)

Componente	Cantidad	Tiempo Unitario (min)	Tiempo Total (min)
Compilación y empaquetado de microservicios	3	15	45
Compilación y empaquetado de Frontend	1	15	15
Transferencia de artefactos mediante SFTP	4	2	8
Detención de servicios anteriores	4	2	8
copia y reemplazo del artefacto	4	2	8
Inicio y validación del servicio desplegado	1	10	10
Validación funcional visual	1	20	20
Tiempo Total			114

Fuente: Ficha de observación del proceso de despliegue manual, 2025.

La Tabla 4 presenta el tiempo promedio requerido para ejecutar un despliegue completo en el entorno de pruebas. Los resultados muestran un tiempo total de 114 minutos, siendo las actividades de compilación de microservicios y validación funcional visual las que concentran la mayor cantidad de tiempo dentro del proceso. Asimismo, se observa que la ejecución manual de múltiples tareas operativas obliga al personal técnico a intervenir de forma constante durante todo el ciclo de despliegue, incrementando la carga de trabajo y el tiempo de respuesta ante nuevas liberaciones.

Tabla 5
Tiempo promedio de despliegue por proyecto (entorno de producción)

Componente	Cantidad	Tiempo Unitario (min)	Tiempo Total (min)
Transferencia de artefactos mediante SFTP	4	2	8
Detención de servicios anteriores	4	2	8
copia y reemplazo del artefacto	4	2	8
Inicio y validación del servicio desplegado	1	10	10
Validación funcional visual	1	20	20
Tiempo Total			**54 min**

Fuente: Ficha de observación del proceso de despliegue manual, 2025.

La Tabla 5 muestra el tiempo promedio requerido para realizar el despliegue en el entorno de producción, alcanzando un total de 54 minutos. Aunque en esta etapa ya no se ejecutan procesos de compilación, aún se mantienen actividades manuales relacionadas con la transferencia de artefactos, actualización de servicios y validación funcional. Esto evidencia que una parte importante del tiempo operativo continúa dependiendo de la intervención directa del personal responsable del despliegue.

Escenario máximo de despliegue (release completo)

En escenarios de actualización integral del sistema, donde era necesario desplegar los siete microservicios backend y el frontend Angular en los ambientes de pruebas y producción, el tiempo operativo podía alcanzar aproximadamente 276 minutos (4.6 horas).

Tabla 6
Tiempo máximo promedio de despliegue en entorno de pruebas.

Componente	Cantidad	Tiempo Unitario (min)	Tiempo Total (min)
Compilación y empaquetado de microservicios	7	15	105
Compilación y empaquetado de Frontend	1	15	15
Transferencia de artefactos mediante SFTP	8	2	16
Detención de servicios anteriores	8	2	16
copia y reemplazo del artefacto	8	2	16
Inicio y validación del servicio desplegado	1	10	10
Validación funcional visual	1	20	20
Tiempo Total General			198

Fuente: Ficha de observación del proceso de despliegue manual, 2025.

La Tabla 6 presenta el escenario de máxima carga operativa identificado durante el levantamiento de la línea base en el entorno de pruebas. En este caso, el tiempo total

alcanza los 198 minutos debido al incremento en la cantidad de microservicios involucrados en el proceso. Se observa que la compilación de los servicios representa la actividad con mayor demanda temporal, seguida por las tareas de validación funcional. Estos resultados reflejan cómo el crecimiento de componentes impacta directamente en la duración del despliegue cuando las actividades son ejecutadas manualmente.

Tabla 7
Tiempo máximo promedio de despliegue en entorno de producción

Componente	Cantidad	Tiempo Unitario (min)	Tiempo Total (min)
Transferencia de artefactos mediante SFTP	8	2	16
Detención de servicios anteriores	8	2	16
copia y reemplazo del artefacto	8	2	16
Inicio y validación del servicio desplegado	1	10	10
Validación funcional visual	1	20	20
Tiempo Total			**78**

Fuente: Ficha de observación del proceso de despliegue manual, 2025.

La Tabla 7 presenta el escenario de máxima carga operativa para el entorno de producción, registrando un tiempo total de 78 minutos. Los resultados muestran que las actividades de transferencia de archivos, detención de servicios, reemplazo de artefactos e inicio manual continúan ejecutándose de forma secuencial, lo que prolonga la duración del procedimiento y aumenta la exposición a posibles errores operativos. Este comportamiento evidencia las limitaciones del proceso tradicional cuando se requiere desplegar múltiples componentes en un mismo ciclo de liberación.

De manera general, los resultados obtenidos durante el levantamiento de la línea base evidencian que el proceso tradicional de despliegue presenta una elevada dependencia de actividades manuales, tiempos prolongados de ejecución y limitadas capacidades de trazabilidad. De manera general, los resultados obtenidos durante el

levantamiento de línea base, evidencian que el proceso tradicional de despliegue presenta una alta dependencia de actividades manuales, tiempos prolongados de ejecución y capacidades limitadas de trazabilidad. Estos resultados permitieron identificar oportunidades para incorporar mecanismos de automatización orientados a optimizar la eficiencia operativa y estandarizar los procedimientos de despliegue en la organización.

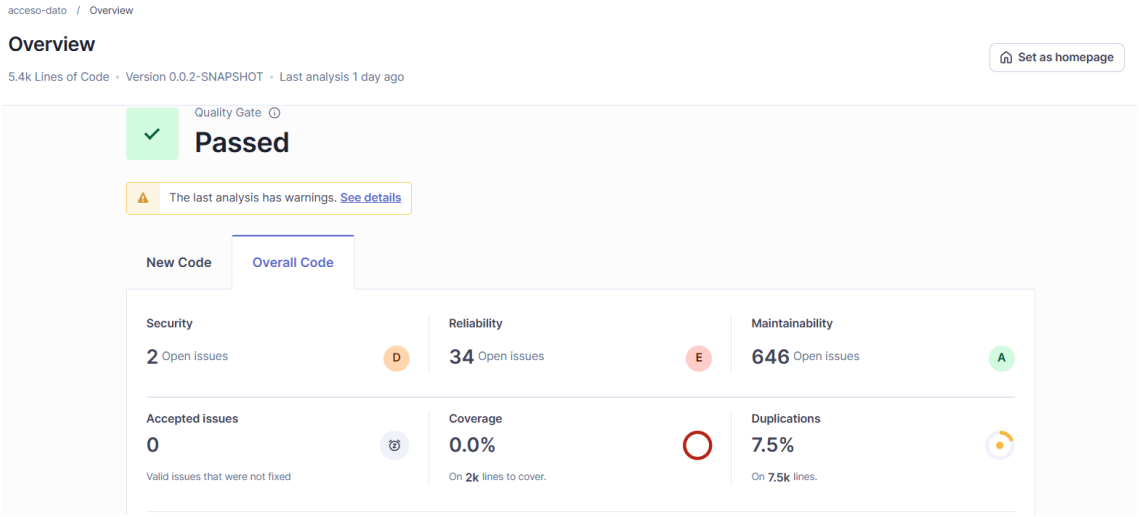
4.1.2. Resultados de la implementación de la solución DevOps

Como resultado de la implementación de principios DevOps en el módulo de gestión de papeletas de la Cooperativa Santa María Magdalena, se obtuvo un ecosistema tecnológico que automatiza las actividades relacionadas con la integración continua, validación de calidad y despliegue de software.

La solución implementada incorporo un flujo de trabajo automatizado que incluye la gestión de versiones, la construcción de artefactos, el análisis de calidad del código y la distribución controlada de nuevas versiones a los entornos de pruebas (QA) y producción. También se incorporaron mecanismos de trazabilidad que permitan registrar y dar seguimiento a cada una de las etapas ejecutadas durante el proceso de despliegue.
Asimismo, los resultados obtenidos demuestran el funcionamiento coordinado de las herramientas implementadas, observándose procesos de despliegue repetibles, consistentes y controlados a través de la arquitectura DevOps.

Figura 17

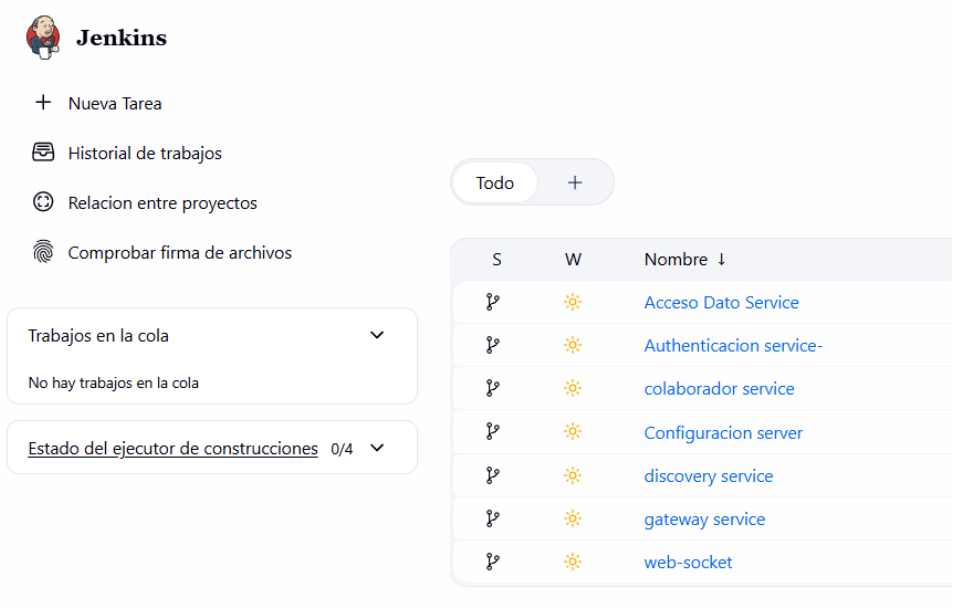
Resultado de validación de calidad mediante Quality Gate en SonarQube



Nota: Panel de resultados de SonarQube. Muestra el estado "Passed" del Quality Gate, indicando que el código del microservicio cumple con los criterios de fiabilidad y seguridad definidos en el Quality Gate antes de continuar con el proceso de despliegue.

La Figura 16 muestra el resultado del análisis de calidad realizado mediante SonarQube sobre los componentes evaluados dentro del pipeline CI/CD. Se observa que el proyecto cumple satisfactoriamente los criterios establecidos por los Quality Gate configurados, permitiendo la continuidad del proceso automatizado hacia las siguientes etapas del despliegue.

Figura 18
Panel principal de pipelines en Jenkins



Nota. Evidencia del panel de control de Jenkins donde se visualiza los Jobs configurados para cada microservicio. Elaboración propia.

La Figura 17 presenta el panel principal de Jenkins utilizado para la administración de los procesos de integración y entrega continua. Desde este entorno es posible supervisar el estado general de los pipelines configurados y verificar la ejecución de los procesos automatizados asociados a los microservicios institucionales

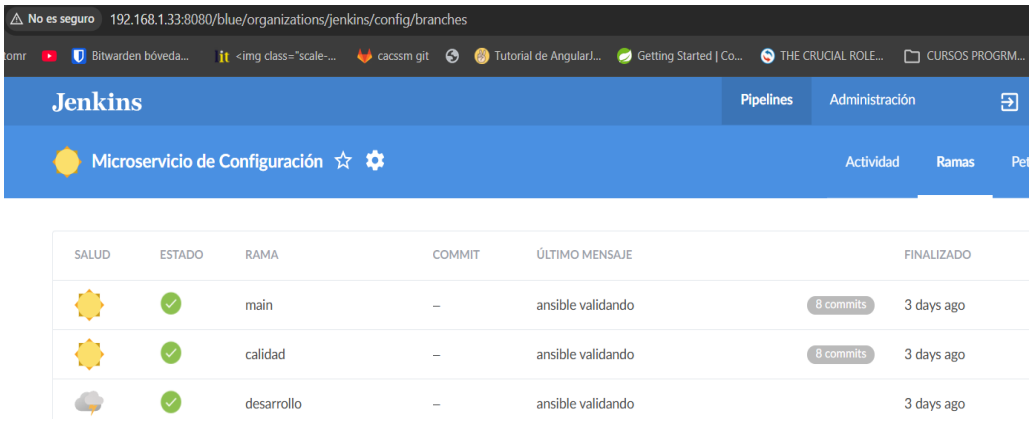
Figura 19
Ejecución de pipeline Multibranch Pipeline para el microservicio de configuración



Nota. Evidencia de la ejecución del pipeline multibranch configurado en Jenkins para la gestión automatizada de las ramas desarrollo, calidad y master. Elaboración propia.

La Figura 18 muestra la ejecución del pipeline multibranch implementado para el microservicio de configuración. Se observa la gestión independiente de las ramas desarrollo, calidad y master, permitiendo mantener un control adecuado del ciclo de vida del software y facilitar la trazabilidad de los cambios realizados.

Figura 20
Estado de ejecución de ramas dentro del pipeline Jenkins blue



***Nota**: La figura muestra el dashboard principal donde se visualiza el estado de cada rama de los Jobs. Esta visibilidad compartida permite que todos los miembros del equipo técnico conozcan en tiempo real las vulnerabilidades y la deuda técnica, eliminando silos de información y fomentando la transparencia en el desarrollo. Elaboración propia.*

La Figura 19 presenta el estado de las diferentes ramas administradas por Jenkins dentro del pipeline CI/CD. Los resultados evidencian la correcta integración entre la estrategia de ramas implementada y los procesos automatizados de construcción y validación, permitiendo un seguimiento continuo de las versiones del software

De manera general, los resultados obtenidos muestran que la implementación de la solución DevOps introdujo mecanismos de automatización, control, seguimiento y

validación que no estaban presentes en el proceso tradicional de despliegue. Estos resultados sirven de base para el análisis de los beneficios operacionales observados bajo el modelo CALMS, desarrollado en la siguiente sección.

4.1.3. Resultados de la Implementación por Dimensiones (Modelo CALMS)

La evaluación de los principios DevOps aplicados al módulo de gestión de papeletas se desarrolló utilizando el modelo CALMS, permitiendo analizar de manera integral el impacto de la automatización y mejora de procesos dentro de la Cooperativa Santa María Magdalena.

A. Cultura (Culture)

Uno de los principales cambios observados durante la implementación de la solución DevOps fue el fortalecimiento de la colaboración entre las actividades de desarrollo y operación. Antes de la implementación, los despliegues eran percibidos como procesos complejos y de alto riesgo, generando dependencia de determinados miembros del equipo y dificultades de coordinación durante las liberaciones de nuevas versiones.

La adopción de prácticas DevOps permitió incorporar procedimientos estandarizados para la gestión de cambios y despliegues, observándose una menor dependencia técnica individual durante las actividades de liberación de software, reduciendo la dependencia técnica individual y promoviendo procedimientos estandarizados para la gestión de cambios y despliegues.

- ✓ **Colaboración y estrategia Gitflow:** Se implementó una estrategia de gestión de versiones basada en Gitflow, organizando el desarrollo mediante ramas especializadas como *feature/* para nuevas funcionalidades, *desarrollo* para integración continua, *calidad* para validación y *master* para la versión estable en producción. Esta estructura permitió el desarrollo paralelo de funcionalidades sin comprometer la estabilidad del sistema.
- ✓ **Responsabilidad compartida:** La automatización de los pipelines y la

integración de mecanismos de validación favorecieron una mayor responsabilidad compartida respecto a la calidad del software. Cada cambio realizado quedó asociado a registros de commits y procesos de validación dentro de GitLab, mejorando la trazabilidad y el seguimiento de las modificaciones efectuadas en los microservicios.

Asimismo, la incorporación de procesos automatizados de revisión y validación permitió incrementar la transparencia en las actividades de despliegue, favoreciendo una mayor comunicación entre los integrantes del equipo técnico y facilitando la coordinación entre las etapas de desarrollo, validación y liberación.

B. Automatización (Automation)

La automatización representó el cambio técnico más significativo observado durante la implementación de la solución DevOps. Antes de la implementación, la mayoría de las actividades relacionadas con el despliegue se realizaban de forma manual, incluyendo la compilación de proyectos, transferencia de artefactos, detención de servicios, actualización de aplicaciones y validaciones posteriores, lo que incrementaba el riesgo de errores operativos y demandaba una elevada intervención del personal técnico.

Con la implementación de pipelines multibranch en Jenkins, integrados con GitLab mediante Webhooks, el proceso de despliegue pasó a ejecutarse de manera automatizada, controlada y trazable. La automatización se estructuró en tres fases principales:

- ✓ **Integración:** Compilación y empaquetado automático de los microservicios utilizando Maven, garantizando la generación estandarizada de artefactos para cada versión del software.
- ✓ **Validación:** Ejecución automática de análisis estático y controles de calidad mediante SonarQube, permitiendo verificar el cumplimiento de los criterios

establecidos antes de continuar con las siguientes etapas del pipeline.

- ✓ **Entrega:** Despliegue automatizado utilizando playbooks de Ansible para la transferencia, actualización y puesta en operación de los artefactos JAR en los entornos de Calidad y Producción.

Asimismo, Jenkins permitió orquestar de forma centralizada el flujo de despliegue hacia los diferentes entornos, reduciendo significativamente la intervención manual y mejorando la consistencia de los procedimientos ejecutados.

Actividades que anteriormente requerían ejecución manual, tales como la detención de procesos Java, transferencia de artefactos, limpieza de directorios temporales, actualización de aplicaciones e inicio de servicios, fueron incorporadas dentro de los pipelines y playbooks automatizados. Como resultado, se observaron despliegues más rápidos, repetibles y controlados, fortaleciéndose la trazabilidad y la estandarización del proceso de liberación de software.

C. Procesos Esbeltos (Lean)

La implementación de principios DevOps incorporó mecanismos orientados a optimizar el flujo de trabajo asociado al proceso de despliegue, reduciendo actividades manuales repetitivas y eliminando tareas que no aportaban valor directo a la liberación de nuevas versiones del software. Como resultado, se logró una ejecución más eficiente de los procedimientos operativos relacionados con la gestión de los microservicios institucionales.

- ✓ **Arquitectura de microservicios:** La adopción de una arquitectura basada en microservicios permitió realizar actualizaciones focalizadas sobre componentes específicos del módulo de gestión de papeletas, evitando la necesidad de desplegar la totalidad de la plataforma. Esta característica permitió reducir el alcance de los

cambios, disminuyendo los riesgos operativos y el tiempo requerido para las actividades de actualización.

- ✓ **Detección temprana de incidencias (Fail-Fast):** La integración de SonarQube dentro del pipeline CI/CD permitió identificar oportunamente problemas relacionados con calidad, mantenibilidad y posibles vulnerabilidades del código. La detección temprana de estas observaciones evitó que desarrollos con deficiencias avanzaran hacia entornos superiores, reduciendo la necesidad de retrabajo y fortaleciendo el control de calidad durante el ciclo de vida del software.
- ✓ **Reducción de actividades sin valor agregado:** La automatización de tareas operativas mediante Jenkins y Ansible permitió automatizar procesos manuales repetitivos relacionados con compilación, transferencia de artefactos, actualización de aplicaciones y validaciones técnicas. Esto contribuyó a simplificar el flujo de despliegue y reducir tiempos de espera dentro del proceso.

La aplicación de estos principios favoreció una gestión más eficiente de los recursos técnicos disponibles, permitiendo que el equipo concentre sus esfuerzos en actividades de mayor valor para la organización.

D. Medición (Measurement)

La implementación de prácticas DevOps bajo el modelo CALMS permitió incorporar mecanismos de medición orientados al seguimiento y evaluación del proceso de despliegue. Para ello, se definieron indicadores cuantitativos que permitieron comparar el comportamiento del proceso antes y después de la automatización.

El principal indicador utilizado fue el Tiempo de Ciclo de Despliegue (Deployment Cycle Time), definido como el tiempo total requerido para completar un despliegue desde la generación del artefacto hasta su disponibilidad en los entornos

operativos.

Línea base del proceso manual

La medición inicial se realizó mediante el registro cronométrico de las actividades ejecutadas durante el despliegue manual en los entornos de Pruebas y Producción.

Escenario promedio: Consideró el despliegue de tres microservicios y un componente frontend.

- ✓ Entorno de Pruebas: 114 minutos.
- ✓ Entorno de Producción: 54 minutos.
- ✓ Tiempo total de despliegue: 168 minutos (2.8 horas).

Escenario de máxima carga: Consideró el despliegue simultáneo de ocho microservicios y un componente frontend.

- ✓ Entorno de Pruebas: 198 minutos.
- ✓ Entorno de Producción: 78 minutos.
- ✓ Tiempo total de despliegue: 276 minutos (4.6 horas).

Escenario automatizado mediante DevOps

Posteriormente, se registraron los tiempos obtenidos tras la implementación de la solución DevOps basada en GitLab, Jenkins, SonarQube y Ansible. La automatización incorporó la ejecución automática de diversas actividades previamente realizadas de forma manual y centralizar el proceso de despliegue mediante pipelines CI/CD.

- ✓ Ejecución automatizada del pipeline: entre 8 y 12 minutos.
- ✓ Validación funcional y aprobación operativa: 25 minutos.
- ✓ Tiempo total promedio de despliegue: 37 minutos.

Evaluación de la eficiencia operativa

La mejora observada fue calculada mediante la siguiente expresión:

$$\%\ Mejora = \frac{Tiempo\ Manual(TM) - Tiempo\ DevOps(TD)}{Tiempo\ Manual(TM)} * 100$$

Aplicando esta fórmula a los escenarios evaluados, se obtuvieron los siguientes resultados:

- ✓ Escenario promedio: reducción del tiempo de despliegue de 168 a 37 minutos, equivalente a una mejora del 77.98%.
- ✓ Escenario de máxima carga: reducción del tiempo de despliegue de 276 a 55 minutos, equivalente a una mejora del 80.07%.

Los resultados obtenidos evidencian una disminución considerable de los tiempos operativos requeridos para la ejecución de despliegues, así como una mayor capacidad de seguimiento y control del proceso mediante el uso de herramientas de automatización y monitoreo.

E. Compartir (Sharing)

La dimensión Compartir (Sharing) se evidenció mediante la centralización de la información técnica, configuraciones y procedimientos utilizados durante el proceso de despliegue. Antes de la implementación, gran parte del conocimiento operativo dependía de la experiencia individual de determinados miembros del equipo, dificultando la transferencia de conocimientos y la continuidad de las actividades ante cambios de personal o ausencia de los responsables.

La adopción de GitLab como plataforma centralizada para la gestión de versiones permitió almacenar y compartir de manera organizada el código fuente, historial de cambios y estructura de ramas utilizadas durante el desarrollo. Asimismo, cada modificación realizada quedó registrada mediante commits y solicitudes de integración, facilitando la trazabilidad y consulta de información técnica por parte de los integrantes del equipo.

De manera complementaria, los pipelines definidos en Jenkins y los playbooks desarrollados en Ansible permitieron documentar y estandarizar los procedimientos de

integración, validación y despliegue. Esto favoreció la reutilización de configuraciones, reduciendo la dependencia de procedimientos ejecutados de forma manual y facilitando la replicación de tareas en distintos entornos.

La centralización de repositorios, scripts y configuraciones favoreció a la gestión del conocimiento dentro del área de Tecnologías de la Información, permitiendo que los procedimientos implementados puedan ser consultados, mantenidos y mejorados de manera continua por los diferentes integrantes del equipo.

De manera general, los resultados obtenidos en las cinco dimensiones del modelo CALMS evidencian la incorporación de prácticas orientadas a la automatización, estandarización y mejora continua del proceso de despliegue del módulo de gestión de papeletas. La adopción de mecanismos de colaboración, automatización del flujo, operativo, medición de indicadores y centralización del conocimiento evidencio en la gestión del proceso de despliegues dentro de la Cooperativa Santa María Magdalena.

4.1.4. Resultados comparativos del despliegue manual y automatizado

Con la finalidad de analizar las diferencias operativas observadas tras la implementación de principios DevOps, se realizó una comparación entre los tiempos registrados durante el proceso tradicional de despliegue y los obtenidos mediante el proceso automatizado. El análisis consideró escenarios representativos de despliegue promedio y escenarios de máxima carga operativa, permitiendo evaluar descriptivamente las variaciones observadas en los tiempos de ejecución y en las actividades involucradas durante el proceso.

Tabla 8
Tiempo de Despliegue Manual vs Tiempo de Despliegue DevOps

Indicador de Éxito	**Escenario**	**Estado Manual (Línea Base)**	**Estado DevOps (Propuesta)**	**Mejora (%)**
Tiempo de	Promedio (3	168 min (2.8 h)	37 minutos	77.98%

Despliegue	Microservicios + Frontend)			
Tiempo de Despliegue	Máximo (8 Microservicios + Frontend)	276 min (4.6 h)	55 minutos	80.07%
Intervención Humana	Todos	Constante y crítica	Supervisión y Aprobación	Reducción alta (orquestación automatizada)
Control de Calidad	Todos	Discrecional / Manual	Automatizado (SonarQube)	Control automatizado mediante Quality Gate
Seguridad de Acceso	Todos	Contraseñas texto plano	Llaves RSA / Tokens SSH	Mejora de acceso por llaves RSA/tokens

***Fuente**: Ficha de observación Línea base y (Logs) de Jenkins de proceso automatizado. Elaboración propia 2025.*

La Tabla 8 presenta la comparación entre el proceso tradicional de despliegue y el proceso automatizado implementado mediante principios DevOps.

En el escenario promedio, el tiempo total requerido para completar el despliegue pasó de 168 minutos a 37 minutos, representando una reducción del 77.98%. Por otro lado, en el escenario de máxima carga operativa, el tiempo de despliegue se redujo de 276 minutos a 55 minutos, equivalente a una mejora del 80.07%.

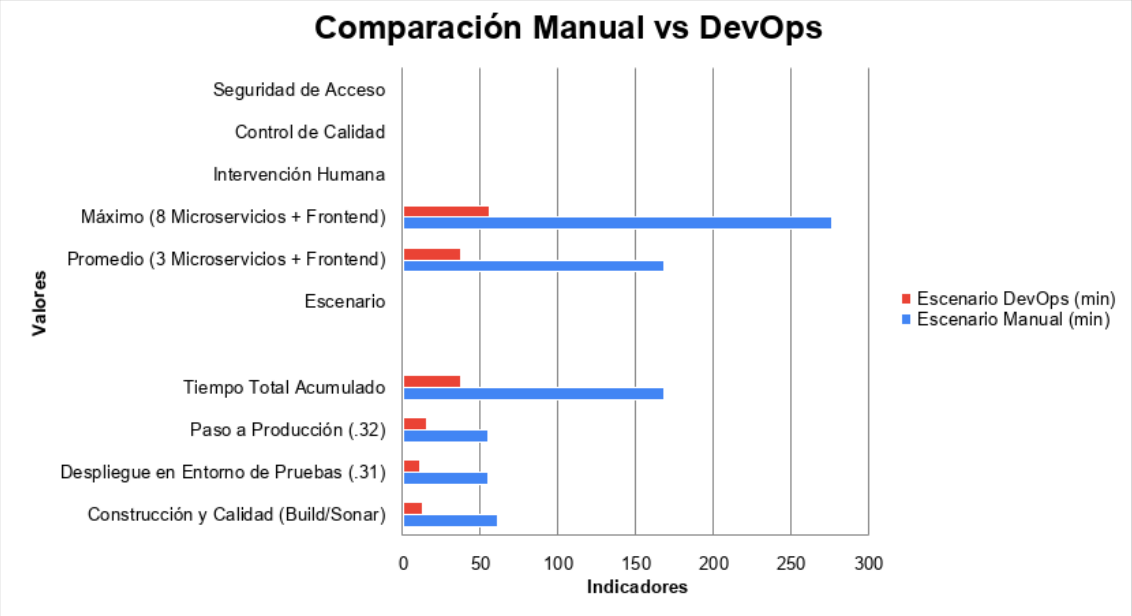
Asimismo, se observa una disminución de la intervención manual requerida durante la ejecución de los despliegues. Mientras que en el proceso tradicional las actividades de compilación, transferencia de artefactos, actualización de servicios y validación dependían de la ejecución directa por parte del personal técnico, en el escenario automatizado estas tareas fueron integradas dentro de pipelines y playbooks ejecutados de manera controlada.

De igual forma, se incorporaron mecanismos automatizados de control de calidad mediante SonarQube y esquemas de autenticación basados en llaves SSH y tokens de

acceso, incorporándose mecanismos de trazabilidad y seguridad durante el proceso de despliegue.

Respecto al mecanismo de recuperación ante fallos, el rollback automático no se contempló dentro del alcance de la implementación desarrollada en la presente investigación. Sin embargo, gracias a la integración entre GitLab y Jenkins, pudimos volver a obtener una versión estable ejecutando de nuevo el pipeline de forma controlada sobre un commit que ya habíamos validado anteriormente. Por lo tanto, este proceso se corresponde con un mecanismo de recuperación semiautomatizada, puesto que requiere la intervención del operador para seleccionar la versión a desplegar y autorizar su ejecución. Se recomienda incluir en futuras versiones del pipeline CI/CD mecanismos de rollback totalmente automatizados.

Figura 21
Gráfico de barras de Tiempo de Despliegue Manual vs Tiempo de Despliegue DevOps



***Fuente**: Elaboración propia*

La Figura 20 permite visualizar de manera comparativa las diferencias observadas entre los tiempos de despliegue manual y los tiempos obtenidos mediante la solución DevOps implementada.

Se aprecia una reducción considerable del tiempo requerido para completar los despliegues en ambos escenarios evaluados. Esta diferencia muestra una reducción en los tiempos requeridos para completar los despliegues en ambos escenarios evaluados. Asimismo, se observó una disminución de las actividades manuales repetitivas y de los tiempos de espera durante el proceso de liberación de nuevas versiones del software, reduciendo tiempos de espera y minimizando la ejecución de tareas manuales repetitivas. Los resultados obtenidos muestran diferencias operativas a favor del proceso automatizado con relación al procedimiento tradicional utilizado como línea base.

4.1.5. Resultados respecto al objetivo general

Los resultados obtenidos durante la investigación permiten describir la implementación de principios DevOps para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena.

La solución implementada integró herramientas especializadas para la gestión de versiones, automatización de procesos CI/CD, control de calidad del código y despliegue automatizado, permitiendo establecer un flujo de trabajo más estructurado, trazable y consistente respecto al proceso tradicional utilizado anteriormente.

Asimismo, los resultados evidencian mejoras operativas observadas en la reducción de los tiempos de despliegue, la disminución de actividades manuales, la incorporación de mecanismos automatizados de control de calidad y el fortalecimiento de la trazabilidad durante las diferentes etapas del ciclo de liberación del software.

Desde la perspectiva del modelo CALMS, la implementación permitió incorporar prácticas relacionadas con colaboración, automatización, optimización de procesos,

medición de indicadores y gestión del conocimiento, contribuyendo a una gestión más eficiente del proceso de despliegue institucional.

En consecuencia, los resultados obtenidos permiten afirmar que el objetivo general de la investigación fue alcanzado, al haberse implementado una solución basada en principios DevOps que permitió automatizar y estandarizar el proceso de despliegue del módulo de gestión de papeletas en la organización.

4.2. DISCUSIÓN DE RESULTADOS

La discusión de resultados se realizó considerando los hallazgos obtenidos durante la implementación de principios DevOps en el despliegue automatizado del módulo de gestión de papeletas de la Cooperativa Santa María Magdalena. Para ello, los resultados fueron analizados desde una perspectiva operativa, de control de calidad y seguridad, así como en relación con los planteamientos teóricos y estudios previos relacionados con DevOps, integración continua y automatización de despliegues.

El análisis permitió identificar los principales beneficios operativos observados tras la implementación, así como establecer relaciones entre los resultados obtenidos y los fundamentos teóricos presentados en la investigación.

4.2.1. Discusión de los resultados operativos observados

Los resultados obtenidos muestran una reducción en los tiempos requeridos para la ejecución de despliegues dentro de la Cooperativa Santa María Magdalena. Mientras que en el escenario manual de máxima carga operativa el proceso podía demandar hasta 276 minutos, en el escenario automatizado dicho tiempo se redujo a aproximadamente 55 minutos. De manera similar, el escenario promedio pasó de 168 minutos a 37 minutos.

Estas diferencias indican una reducción en las tareas manuales requeridas para finalizar el proceso de despliegue. La integración de Jenkins con Ansible incluyó la ejecución automática de tareas de compilación, validación, transferencia de artefactos y

actualización de servicios, lo cual se reflejó en menor intervención directa por parte del personal técnico.

Asimismo, la estandarización de los procedimientos favoreció a una mayor consistencia de los despliegues y una mejor la trazabilidad de las actividades ejecutadas durante cada liberación de software. Los resultados obtenidos demuestran una gestión más eficiente del proceso de despliegue institucional.

4.2.2. Discusión de los controles de calidad y seguridad incorporados

La incorporación de mecanismos automatizados de control de calidad constituyó uno de los aspectos relevantes observados durante la implementación. Antes de la automatización, gran parte de las validaciones dependían de procedimientos manuales realizados por el equipo técnico, lo que aumenta la probabilidad de omisiones involuntarias durante el proceso de despliegue.

La integración de SonarQube dentro del pipeline CI/CD permitió incorporar análisis estático del código fuente antes de la liberación de nuevas versiones, facilitando la identificación temprana de observaciones relacionadas con calidad, mantenibilidad y posibles vulnerabilidades.

Asimismo, la implementación de Quality Gate permitió establecer criterios mínimos de aceptación para los proyectos analizados, mientras que la utilización de llaves RSA y tokens SSH fortaleció los mecanismos de autenticación utilizados durante los despliegues automatizados.

Los resultados obtenidos muestran la incorporación de controles orientados a mejorar la confiabilidad y seguridad operativa del proceso de despliegue, sin requerir la implementación de una arquitectura DevSecOps completa.

4.2.3. Contraste de resultados con la literatura científica

Los resultados obtenidos muestran concordancia con los planteamientos de Forsgren, Humble y Kim (2018), quienes señalan que la adopción de prácticas DevOps favorece la reducción de los tiempos de entrega de software y mejora la estabilidad operativa. En la presente investigación, el tiempo promedio de despliegue se redujo de 168 a 37 minutos y el tiempo máximo de 276 a 55 minutos, evidenciando una reducción de los tiempos requeridos para el proceso de liberación de software.

Asimismo, los hallazgos coinciden con Kim et al. (2016) y Humble y Farley (2010), quienes destacan la integración continua, la automatización y la estandarización de procesos como factores clave para lograr despliegues más consistentes y trazables. La integración de GitLab, Jenkins, SonarQube y Ansible incorporo un flujo automatizado que redujo la intervención manual y fortaleció los mecanismos de control durante el ciclo de despliegue.

Desde la perspectiva del modelo CALMS, se identificaron mejoras en las dimensiones de Cultura, mediante una mayor colaboración entre desarrollo y operaciones; Automatización, a través de la implementación de pipelines CI/CD; Lean, por la reducción de actividades repetitivas; Medición, mediante el seguimiento de indicadores operativos; y Compartir, gracias a la centralización de configuraciones y repositorios.

Finalmente, los resultados guardan relación con investigaciones previas desarrolladas por López Peña (2021), Cusco Mejía (2022) y Guerrero (2020), las cuales resaltan la importancia de la automatización, los microservicios y la infraestructura como código para optimizar los procesos de despliegue. En ese sentido, los resultados obtenidos permiten describir la factibilidad de implementar principios DevOps en una infraestructura on-premise dentro de una institución financiera cooperativa, observándose

beneficios operativos observados relacionados con la automatización del proceso de despliegue, sin depender de servicios cloud.

De manera general, la discusión realizada permite establecer que los resultados observados durante la implementación de principios DevOps presentan correspondencia con los fundamentos teóricos y antecedentes revisados en la investigación. La automatización de procesos, la incorporación de mecanismos de control de calidad y la adopción de prácticas alineadas con el modelo CALMS que permitieron incorporar mecanismos orientados a optimizar el proceso de despliegue del módulo de gestión de papeletas, fortaleciendo la eficiencia operativa y la trazabilidad de las actividades desarrolladas dentro de la Cooperativa Santa María Magdalena.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

5.1.1. Conclusión general

- ✓ Se describió la implementación de principios DevOps para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena. La implementación integró GitLab, Webhooks, Jenkins, Maven, SonarQube y Ansible en un flujo que comprendió la gestión del código fuente, la compilación, el análisis de calidad, el despliegue automatizado en el entorno de pruebas, la aprobación manual y el posterior despliegue en producción. Durante la implementación se observó una reducción de la intervención manual, mayor trazabilidad del proceso y la incorporación de controles automatizados de calidad.

5.1.2. Conclusiones específicas

- ✓ **Primera.** Se diseñó una arquitectura DevOps adaptada a la infraestructura tecnológica de la Cooperativa Santa María Magdalena. La arquitectura integró GitLab como repositorio centralizado; Jenkins y SonarQube ejecutados mediante Docker Compose en el servidor DevOps; Maven para la compilación y generación de artefactos; y Ansible para automatizar los despliegues en los entornos diferenciados de pruebas y producción. Esta organización permitió centralizar el flujo de integración, validación y entrega del módulo de gestión de papeletas.

- ✓ **Segunda.** Se definieron los componentes del pipeline de integración continua y entrega continua, incorporando la activación automática mediante Webhooks, la compilación con Maven, la generación de artefactos, el análisis estático del código y la validación del Quality Gate mediante SonarQube. Asimismo, se incluyeron el despliegue automatizado en QA, la validación funcional, la aprobación manual en Jenkins y el posterior despliegue automatizado en producción.
- ✓ **Tercera.** Se caracterizaron los mecanismos de automatización empleados en el proceso de despliegue, entre ellos los Webhooks, los archivos Jenkinsfile, los playbooks de Ansible, el control de versiones mediante GitFlow y la autenticación mediante llaves SSH y tokens. Estos mecanismos permitieron automatizar actividades de compilación, generación, transferencia y despliegue de artefactos, manteniendo la intervención del responsable técnico para autorizar el paso hacia producción.
- ✓ **Cuarta.** Se describieron los beneficios operativos observados mediante la comparación entre la línea base manual y los registros obtenidos después de la implementación. El tiempo promedio de despliegue disminuyó de 168 a 37 minutos, lo que representa una reducción del 77,98 %. En el escenario de máxima carga operativa, el tiempo pasó de 276 a 55 minutos, equivalente a una reducción del 80,07 %. También se observó el fortalecimiento de la trazabilidad mediante GitLab y Jenkins, la incorporación del control automatizado de calidad con SonarQube y el uso de llaves SSH y tokens para el acceso entre los servidores.

5.2. RECOMENDACIONES

- ✓ Monitoreo avanzado: Implementar herramientas de observabilidad continua, como Prometheus y Grafana, para monitorear en tiempo real el consumo de recursos (CPU, memoria, red) de los servidores on-premise tras cada despliegue.

- ✓ Pruebas automatizadas: Incorporar la ejecución automatizada de pruebas unitarias y pruebas de integración dentro de las etapas del pipeline para asegurar el comportamiento funcional del código antes del análisis de calidad.
- ✓ Gestión segura de secretos: Adoptar un gestor de secretos especializado (como HashiCorp Vault) para almacenar y rotar dinámicamente credenciales, bases de datos y tokens, superando el manejo de variables estáticas.
- ✓ Medición de métricas DORA: Establecer la medición continua y formal de las métricas DORA (Frecuencia de Despliegue, Tiempo de Entrega, Tiempo de Restauración y Tasa de Fallo) para evaluar la madurez DevOps del equipo a mediano plazo.
- ✓ Extensión progresiva: Escalar este diseño de despliegue automatizado hacia los demás módulos y sistemas core de la institución, utilizando el artefacto actual como plantilla base de arquitectura.

BIBLIOGRAFÍA

Akbar, M. A. (2024). DevOps project management success factors: A decision-making framework. Software: Practice and Experience, 54(2), 257–280. https://doi.org/10.1002/spe.3294

Albarran Salazar, H. J., & Delgado Farro, J. M. (2024). Implementación del modelo CALMS de DevOps para la optimización del proceso de desarrollo de software: Caso de estudio empresa Ayarcode [Tesis de pregrado, Universidad Señor de Sipán]. Repositorio USS. https://repositorio.uss.edu.pe/

Amitkumar, V., & Jha, R. T. (2023). From theory to practice: Understanding DevOps culture and mindset. Cogent Engineering, 10(1), 2251758. https://doi.org/10.1080/23311916.2023.2251758

Atlassian. (2024). CALMS framework – DevOps. https://www.atlassian.com/es/devops/frameworks/calms-framework

Atlassian. (2024). DevOps. https://www.atlassian.com/es/devops/what-is-devops

Bernal, C. A. (2016). Metodología de la investigación (4.ª ed.). Pearson Educación.

Bucena, I., & Kirikova, M. (2017). El puente entre ISO/IEC 29110 y DevOps. Revista Ibérica de Sistemas e Tecnologias de Informação, (53), 5–18. https://scielo.pt/pdf/rist/n53/1646-9895-rist-53-5.pdf

Castañeda García, P. A. (2019). Prácticas DevOps de entrega continua de software para la implementación automatizada del proceso de despliegue [Tesis de posgrado, Universidad EAFIT]. Repositorio EAFIT. https://repository.eafit.edu.co/handle/10784/13842

Debois, P. (2008). Agile infrastructure and operations: How infra-gile are you? Agile Conference. IEEE. https://doi.org/10.1109/Agile.2008.42

Forsgren, N., Humble, J., Kim, G., & Kersten, M. (2022). Informe ACCELERATE – State of DevOps 2022. DORA. https://dora.dev/research/2022

Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps. IT Revolution Press.

Gandomani, T. J., & Nafchi, M. Z. (2016). DevOps adoption challenges: A systematic review. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1). https://ojs.aaai.org/index.php/AAAI/article/view/10186

Giménez Marco, G. (2016). DevOps, la nueva tendencia en el desarrollo de sistemas TI: Un caso práctico en el análisis de incidencias de software [Trabajo de grado, Universitat Politècnica de Catalunya]. Repositorio UPC. https://upcommons.upc.edu/handle/2117/85074

González, A. (2018). Entrega continua y automatización: Fundamentos de integración y entrega continua (CI/CD) en DevOps [Trabajo final de máster, Universitat Politècnica de València]. RiuNet. https://riunet.upv.es/handle/10251/110200

Guerrero, D. F. (2020). Modelo de referencia para la adopción de DevOps en empresas de desarrollo de software [Tesis]. Universidad del Cauca. http://repositorio.unicauca.edu.co/handle/123456789/8557

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). Metodología de la investigación (6.ª ed.). McGraw-Hill Education.

Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley.

IBM. (2024). Continuous deployment. IBM. https://www.ibm.com/topics/continuous-deployment

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. IT Revolution Press.

ManageEngine. (2024). DevOps culture and collaboration. https://www.manageengine.com/

Morales Vélez, H. (2022). Implementación de una cultura DevOps en empresas de TI [Tesis]. Universidad Técnica de Cotopaxi. https://repositorio.utc.edu.ec/handle/27000/9855

Morales Vélez, S. (2022). DevOps: Automatización y despliegue con Jenkins y Docker [Tesis de maestría, Universitat Oberta de Catalunya]. UOC. https://openaccess.uoc.edu/handle/10609/132367

Muñoz, D. C. (2021). Integración y entrega continua modernizada con DevOps y microservicios [Tesis de maestría]. Universidad Militar Nueva Granada. https://repository.unimilitar.edu.co/handle/10654/40265

Peche Sánchez, C. W. (2025). Diseño de arquitectura de microservicios para optimizar procesos y automatizar integraciones en la administración pública local apoyado por DevOps [Tesis, Universidad Señor de Sipán]. Repositorio USS. https://repositorio.uss.edu.pe/handle/20.500.12802/14898

Richardson, C. (s.f.). Microservices architecture. Microservices.io. https://microservices.io/

Rodríguez, Z. M. (2020). Arquitectura basada en microservicios y DevOps para una ingeniería de software continua. Redalyc. https://www.redalyc.org/journal/816/81665362014

Salazar Guevara, L. J. (2022). Implementación de DevOps para mejorar la integración y despliegue de software [Tesis]. Universidad Nacional Pedro Ruiz Gallo. https://repositorio.unprg.edu.pe/handle/20.500.12893/11869

Saxena, A., Singh, S., Prakash, S., Yang, T., & Rathore, R. S. (2025). DevOps automation pipeline deployment with IaC (Infrastructure as Code). IEEE SILCON 2024. https://doi.org/10.1109/SILCON63976.2024.10910699

ServiceNow. (2025). What is continuous deployment? ServiceNow. https://www.servicenow.com/

Sharma, S. (2017). The DevOps adoption playbook: A guide to adopting DevOps in a multi-speed IT enterprise. Wiley.

TechTarget. (2024). DevOps culture. TechTarget. https://www.techtarget.com/

Unity. (2025). What is DevOps? Unity DevOps. https://unity.com/solutions/devops

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019). The DevOps phenomenon. Queue, 17(2), 93-112. https://doi.org/10.1145/3329781.3338532

ANEXOS

ANEXO A: MATRIZ DE CONSISTENCIA

TITULO: "Principios DevOps para el despliegue automatizado de un módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025"

<table>
<tr><th>PROBLEMA</th><th>OBJETIVO</th><th colspan="2">HIPÓTESIS</th><th>METODOLOGÍA</th></tr>
<tr>
<td rowspan="2">PROBLEMA PRINCIPAL
¿Cómo se implementan los principios DevOps para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025?
PROBLEMAS SECUNDARIOS
a) ¿Cuál es el diseño de una arquitectura DevOps adecuada para el despliegue del módulo de gestión de papeletas?
b) ¿Cuáles son los componentes necesarios para un proceso de integración y</td>
<td rowspan="2">OBJETIVO GENERAL
Describir la implementación de principios DevOps en el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025.
OBJETIVOS ESPECÍFICOS
a) Diseñar una arquitectura DevOps adecuada para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios.
b) Definir los componentes del pipeline CI/CD para automatizar el despliegue del módulo de gestión de papeletas.
c) Caracterizar los mecanismos de automatización empleados</td>
<td colspan="2">HIPÓTESIS GENERAL
No se plantea ninguna hipótesis por ser un estudio de nivel descriptivo.</td>
<td rowspan="2">TIPO DE INVESTIGACIÓN
Aplicada

NIVEL DE INVESTIGACIÓN
Descriptiva

DISEÑO

No experimental y Transversal

POBLACIÓN
Procesos de despliegue de software de la Cooperativa Santa María Magdalena – Ayacucho.
MUESTRA
-Módulo de gestión de papeletas y el proceso de despliegue automatizado con DevOps
INFORMANTES
-3 desarrolladores y 2 responsables del área de TI

TÉCNICAS
Análisis documental
Observación, Observación directa</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>INDICADORES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VARIABLE DE ESTUDIO
X. Implementación de principios DevOps

Variables descriptivas

a) Diseño de la arquitectura DevOps utilizada

b) Componentes del pipeline CI/CD implementado</td>
<td>-Número de herramientas integradas
-Nivel de estandarización del proceso
-Existencia de documentación técnica

-Etapas automatizadas vs. manuales
-Tiempo de integración y pruebas
-Trazabilidad del despliegue</td>
<td></td>
</tr>
</table>

entrega continua (CI/CD) para el despliegue del módulo de gestión de papeletas? c) ¿Cuáles son los mecanismos de automatización empleados en el despliegue del módulo de gestión de papeletas? d) ¿Qué beneficios operativos se observan en el proceso de despliegue del módulo tras la implementación de principios DevOps?	en el despliegue del módulo de gestión de papeletas. d) Describir los beneficios operativos observados en el proceso de despliegue del módulo de gestión de papeletas tras la implementación de principios DevOps.	c) Mecanismos de automatización empleados en el despliegue d) Beneficios operativos observados en el proceso de despliegue	-Tareas repetitivas automatizadas -Uso de playbooks y scripts automatizados -Nivel de intervención humana. -Tiempo de despliegue -Control de calidad automatizado -Seguridad de acceso mediante SSH/Tokens -Mejora de la trazabilidad	**INSTRUMENTOS** -Ficha de análisis documental -Ficha de observación

ANEXO B: MATRIZ DE OPERACIONALIZACIÓN DE VARIABLES

Variables	Dimensión	Indicador	Forma de medición	Escala	Fuente de datos	Técnica	Instrumento
Implementación de principios DevOps	**Arquitectura DevOps**	Número de herramientas integradas	Conteo de herramientas	Razón	Entorno TI (Servidores)	Observación directa	Ficha de observación
		Nivel de estandarización del proceso	Porcentaje (%)	Razón	Manuales y procesos de TI	Análisis documental	Ficha documental
		Existencia de documentación técnica	Presencia / Ausencia	Nominal	Repositorio institucional	Observación directa	Ficha de observación
	Pipeline CI/CD	Etapas automatizadas vs. manuales	Conteo de etapas	Razón	Configuración del Jenkinsfile	Observación directa	Ficha de observación
		Tiempo de integración y pruebas	Minutos	Razón	Logs de Jenkins y SonarQube	Observación directa	Ficha de observación
		Trazabilidad del despliegue	Presencia/Ausencia	Nominal	GitLab y Jenkins	Análisis documental	Ficha documental
	Mecanismos de automatización	Tareas repetitivas automatizadas	Porcentaje (%)	Razón	Playbooks de Ansible / Scripts	Observación directa	Ficha de observación
		Uso de playbooks y scripts automatizados	Presencia/Ausencia	Nominal	Playbooks de Ansible y Jenkinsfile	Análisis documental	Ficha documental
		Nivel de intervención humana	Porcentaje (%)	Razón	Registros del proceso de despliegue	Observación directa	Ficha de observación
	Beneficios operativos observados en el	Tiempo de despliegue	Diferencia porcentual entre línea base y despliegue automatizado.	Razón	Logs de Jenkins	Observación directa	Ficha de observación

	proceso de despliegue	Control de calidad automatizado	Presencia/Ausencia de Quality Gate	Nominal	SonarQube y registros de calidad	Análisis documental	Ficha documental
		Seguridad de acceso mediante SSH/Tokens	Presencia/Ausencia	Nominal	Configuración de servidores y llaves SSH	Análisis documental	Ficha documental
		Mejora de la trazabilidad	Presencia/Ausencia de registros automáticos	Nominal	GitLab y Jenkins	Análisis documental	Ficha documental

Fuente: elaboración propia

ANEXO C: FICHA DE OBSERVACIÓN DIRECTA

Técnica: Observación Directa.

Instrumento: Ficha de registro de ejecución del pipeline CI/CD

Objetivo: Registrar y verificar la ejecución del proceso de despliegue automatizado mediante la implementación de principios DevOps para el módulo de gestión de papeletas.

Sujeto de observación: Pipeline CI/CD del módulo de gestión de papeletas.

Fecha: _____

Observador: _____

Entorno: QA ( ) Producción ( )

Fecha de Observación: _____ N° de Despliegue: _____

Fase del Pipeline	Etapa del Proceso CI/CD	Estado (Sí / No)	Métrica / Tiempo	Evidencia Técnica (Logs / Herramientas)
1. Orquestación Inicial	El pipeline se activa automáticamente tras un evento (commit/merge) en el repositorio.			
2. Construcción (Build)	Generación exitosa de un artefacto inmutable (.jar) reutilizable en múltiples entornos.			
	Tiempo exacto que tarda el proceso de compilación y empaquetado.			
3. Calidad y Seguridad	Ejecución automática de pruebas o análisis estático integrados.			
	El "Quality Gate" aprueba o rechaza el código antes de avanzar.			
4. Despliegue en Pruebas (QA)	La transferencia del artefacto y el reinicio de servicios en QA se realiza sin intervención manual.			
	Ejecución de la configuración mediante Infraestructura como Código (IaC).			
	Tiempo destinado a las verificaciones funcionales visuales en el entorno QA.			

5. Transición a Producción	El avance hacia Producción requiere una aprobación explícita manual (Gated Deployment).			
	Uso del mismo artefacto y scripts de configuración (IaC) empleados en la fase de QA.			
6. Despliegue en Producción	El despliegue en Producción se ejecuta de forma automatizada tras otorgar la aprobación.			
	Tiempo exacto de orquestación en el servidor de Producción.			
7. Monitoreo y Resiliencia	Verificación de métricas de observabilidad, disponibilidad de puertos o revisión de logs post-despliegue.			
	Generación de logs centralizados que identifican la fase exacta de éxito o fallo del pipeline.			
	Capacidad de ejecutar un rollback de forma automatizada o con un solo comando ante simulaciones de fallo.			
8. Consolidado	Tiempo Total de Despliegue (End-to-End, incluyendo tiempos de máquina y validación humana).			

ANEXO C.1 REGISTRO DE OBSERVACIÓN DEL PIPELINE

Técnica: Observación Directa

Objetivo: Verificar la implementación práctica, la automatización y la estandarización del proceso de despliegue CI/CD para el módulo de gestión de papeletas.

Sujeto de Observación: Proceso de Despliegue del Módulo de Gestión de Papeletas.

Fecha/Hora de Observación:

Observador:

Nº	Aspecto a Observar	Resultado	Observaciones (Tiempo, Fallos, Herramienta)
1	Inicio Automatizado: El pipeline se activa automáticamente tras un commit o merge al repositorio principal.	SÍ	Activación exitosa vía Webhook desde GitLab hacia Jenkins tras el push en la rama correspondiente.
2	Tiempo de Compilación: Medir el tiempo exacto que tarda el proceso de build o compilación.	SÍ	Compilación y empaquetado ejecutado con Maven en 6 min 39 seg (según log de Jenkins).
3	Análisis de Calidad y Seguridad: Se observa la ejecución automática de análisis estático de código y validación mediante Quality Gate dentro del pipeline.	SÍ	Ejecución de análisis estático con **SonarQube**; el Quality Gate arroja estado "Passed" en ~1.5 min.
4	Creación de Artefacto: El proceso genera un artefacto inmutable que se reutiliza en diferentes entornos.	SÍ	Generación exitosa de archivo .jar inmutable. Se compila una sola vez para asegurar la consistencia.
5	Despliegue QA: El despliegue al entorno de Control de Calidad (QA) se realiza sin intervención manual.	SÍ	Transferencia del artefacto y reinicio de servicios en el servidor pruebas ejecutado automáticamente.
6	Uso de IaC: Se ejecuta un Playbook (ej: Ansible) o script de configuración para el entorno de QA.	SÍ	Ejecución exitosa de playbook de Ansible. Proceso idempotente, sin necesidad de SSH manual.
7	Verificación del Entorno QA: El equipo realiza verificaciones funcionales manuales después del despliegue en QA.	SÍ	Intervención humana de 25 minutos para validación visual y funcional en el entorno de pruebas.
8	Aprobación de Producción: El paso de QA a Producción requiere una aprobación explícita (Gated Deployment).	SÍ	El pipeline se pausa en Jenkins esperando el clic manual de validación ("Input Requested").
9	Despliegue Producción: El despliegue en el entorno de Producción se realiza automáticamente después de la aprobación.	SÍ	Aprobación otorgada; Ansible despliega el mismo .jar en el servidor Producción en 4 minutos.
10	Consistencia de Entornos: Los pasos de despliegue en QA y Producción utilizan el mismo artefacto y los mismos scripts de configuración (IaC).	SÍ	Verificado mediante Jenkins y Ansible. El mismo archivo .jar generado durante la fase Build fue promovido desde QA hacia Producción sin recopilación adicional.
11	Monitoreo Post-Despliegue: El equipo de TI verifica logs o métricas de observabilidad tras el despliegue en Producción.	SÍ	El desarrollador verifica disponibilidad de puertos y revisa logs en servidor durante ~5 minutos.
12	Tiempo Total de Despliegue: Medir el tiempo total transcurrido desde la activación hasta la disponibilidad en Producción.	SÍ	Tiempo total End-to-End validado: 37 minutos (incluyendo tiempos de máquina y validación humana). Evidenciado mediante los registros históricos del pipeline en Jenkins.
13	Rollback (Simulación/Fallo): Si ocurre un fallo, la función de reversión es accionada con un solo comando o de forma automática.	SÍ	Ante un fallo simulado, se re-ejecuta el pipeline apuntando al commit anterior estable en GitLab.

14	Logs Claros: El pipeline genera un resumen o logs claros que permiten identificar rápidamente la fase del fallo.	SÍ	Interfaz visual de Jenkins (Console Output) muestra el éxito o fallo detallado por cada "Stage".

ANEXO D: FICHA DE ANALISIS DOCUMENTAL

Técnica: Análisis Documental

Instrumento: Ficha de Análisis Documental

Objetivo:

Recopilar y registrar información relacionada con la arquitectura DevOps, los componentes del pipeline CI/CD, los mecanismos de automatización y los beneficios operativos observados a partir de la documentación técnica institucional.

Fuente Documental:

( ) Diagramas de Arquitectura

( ) Jenkinsfile

( ) Logs de Jenkins

( ) Reportes SonarQube

( ) Repositorios GitLab

( ) Playbooks Ansible

( ) Manuales Técnicos

( ) Reportes de Incidencias

( ) Otros: _____

Documento Analizado: _____

Fecha de Análisis: ____ / ____ / _____

Analista: _____

	Ítem a Analizar	**Cumplimiento (Sí/No)**	**Observaciones / Valor Encontrado**
1	Existencia de documentación formal y actualizada sobre la Arquitectura DevOps (Diagramas).		
2	Documento que defina la topología de la arquitectura de microservicios del		

	módulo de gestión de papeletas.		
3	Listado oficial de las herramientas DevOps implementadas (GitLab, Jenkins, SonarQube, Maven y Ansible).		
4	Existencia de un manual o guía de uso de la Infraestructura como Código (IaC), como Ansible Playbooks.		
5	Documentación formal que defina las etapas del Pipeline CI/CD (Build, Test, Deploy).		
6	Registro de logs o evidencia de ejecución automática de análisis de calidad y seguridad del código mediante SonarQube		
7	Procedimiento documentado para la reversión o rollback ante un fallo en producción.		
8	Registro de métricas que muestren el Tiempo Promedio de Despliegue (antes de DevOps).		
9	Registro de métricas que muestren el Tiempo Promedio de Despliegue (después de DevOps).		
10	Reporte de incidencias críticas relacionadas con el despliegue manual (periodo anterior a DevOps).		
11	Reporte de incidencias críticas relacionadas con el despliegue automatizado (periodo posterior a DevOps).		
12	Evidencia documental de prácticas colaborativas alineadas a los principios DevOps y al modelo CALMS.		

13	Evidencia de automatización de tareas de configuración y despliegue mediante Infraestructura como Código (IaC).		
14	Métrica registrada sobre la Frecuencia de Despliegues a producción (antes de DevOps: por ejemplo, semanal/mensual).		
15	Métrica registrada sobre la Frecuencia de Despliegues a producción (después de DevOps: por ejemplo, diaria/bajo demanda).		

ANEXO D.1: APLICACIÓN DE LA FICHA DE ANALISIS DOCUMENTAL

Técnica: Análisis Documental

Instrumento: Ficha de Análisis Documental

Objetivo:

Recopilar y registrar información relacionada con la arquitectura DevOps, los componentes del pipeline CI/CD, los mecanismos de automatización y los beneficios operativos observados a partir de la documentación técnica institucional.

- ✓ Fuente Documental:
- ✓ Diagramas de Arquitectura DevOps
- ✓ Jenkinsfile
- ✓ Logs de Jenkins
- ✓ Reportes SonarQube
- ✓ Repositorios GitLab
- ✓ Playbooks Ansible
- ✓ Registros históricos de despliegues
- ✓ Reportes de incidencias

Documento Analizado: _____

Fecha de Análisis: ____ / ____ / _____

Analista: _____

N°	Ítem a Analizar	Cumplimiento (Sí/No)	Observaciones / Valor Encontrado
1	Existencia de documentación formal y actualizada sobre la Arquitectura DevOps (Diagramas).	SÍ	Diagrama de arquitectura CI/CD actualizado (GitLab -> Jenkins -> SonarQube -> Ansible -> Servidores .31/.32).
2	Documento que defina la topología de la arquitectura de microservicios del módulo de gestión de papeletas.	SÍ	Topología documentada: 8 microservicios integrados (Spring Boot) y 1 frontend (Angular) alojados en nodos Linux.
3	Listado oficial de las herramientas DevOps implementadas (GitLab, Jenkins, SonarQube, Maven y Ansible).	SÍ	Stack tecnológico oficializado: GitLab (Repositorio), Jenkins (Orquestador), SonarQube (Quality Gate), Maven, Ansible (Despliegue).
4	Existencia de un manual o guía de uso de la Infraestructura como Código (IaC), como Ansible Playbooks.	SÍ	Archivos YAML (Playbooks) documentados para la transferencia idempotente de artefactos .jar y reinicio de servicios.
5	Documentación formal que defina las etapas del Pipeline CI/CD (Build, Test, Deploy).	SÍ	Archivo Jenkinsfile estructurado en etapas: Build, Análisis SonarQube, Deploy QA (.31) y Deploy Prod (.32).
6	Registro de logs o evidencia de ejecución automática de análisis de calidad y seguridad del código mediante SonarQube	SÍ	Logs extraídos de Jenkins y métricas del dashboard de SonarQube validando el estado "Passed" (80% de cumplimiento).
7	Procedimiento documentado para la reversión o rollback ante un fallo en producción.	SÍ	Procedimiento definido para re-ejecutar el pipeline apuntando al último commit estable o artefacto en el repositorio.
8	Registro de métricas que muestren el Tiempo Promedio de Despliegue (antes de DevOps).	SÍ	Registros históricos de soporte indican un tiempo promedio de 168 minutos y un máximo de 276 minutos (4.6 h).
9	Registro de métricas que muestren el Tiempo Promedio de Despliegue (después de DevOps).	SÍ	Logs de Jenkins certifican un tiempo de ciclo optimizado de 37 minutos (promedio) y 55 minutos (máximo).
10	Reporte de incidencias críticas relacionadas con el despliegue manual (periodo anterior a DevOps).	SÍ	Historial de fallos por errores de digitación en puertos, credenciales y omisión de transferencia de archivos por SFTP.

11	Reporte de incidencias críticas relacionadas con el despliegue automatizado (periodo posterior a DevOps).	SÍ	0 incidencias críticas por error humano durante el pase a producción gracias a la inmutabilidad del artefacto y Ansible.
12	Evidencia documental de prácticas colaborativas alineadas a los principios DevOps y al modelo CALMS.	SÍ	Adopción de la validación automatizada y flujo de aprobación de pases a producción entre Desarrollo y Operaciones.
13	Evidencia de automatización de tareas de configuración y despliegue mediante Infraestructura como Código (IaC).	SÍ	Uso de llaves RSA/SSH y playbooks de Ansible para automatizar las actividades de configuración y despliegue, reduciendo significativamente la intervención manual.
14	Métrica registrada sobre la Frecuencia de Despliegues a producción (antes de DevOps: por ejemplo, semanal/mensual).	SÍ	Frecuencia Mensual/Quincenal, ejecutada exclusivamente fuera del horario laboral o fines de semana debido al alto riesgo.
15	Métrica registrada sobre la Frecuencia de Despliegues a producción (después de DevOps: por ejemplo, diaria/bajo demanda).	SÍ	Frecuencia de despliegue bajo demanda, con capacidad de realizar múltiples despliegues durante la jornada laboral gracias a la automatización del proceso y a las validaciones de calidad integradas.

ANEXO E: FICHA DE REGISTRO DE EJECUCIÓN DEL PIPELINE CI/CD

Técnica: Observación Directa

Instrumento: Ficha de Registro de Ejecución del Pipeline CI/CD

Objetivo: Registrar las métricas técnicas obtenidas durante la ejecución del pipeline CI/CD para el despliegue automatizado del módulo de gestión de papeletas.

Fecha de Ejecución: _____

N° de Pipeline: _____

Proyecto / Microservicio: _____

Versión Desplegada: _____

Commit ID: _____

Rama: Desarrollo ( ) Calidad ( ) Producción ( )

Responsable: _____

Etapa del Pipeline	Estado	Tiempo	Evidencia
Activación por Webhook			
Build (Compilación)			
Generación de Artefacto			
Análisis SonarQube			
Quality Gate			
Deploy QA			
Validación QA			
Aprobación Producción			
Deploy Producción			
Verificación Final			

ANEXO E.1: APLICACIÓN DE LA FICHA DE REGISTRO DEL PIPELINE

Técnica: Observación Directa

Instrumento: Ficha de Registro de Ejecución del Pipeline CI/CD

Objetivo: Registrar las métricas técnicas obtenidas durante la ejecución del pipeline CI/CD para el despliegue automatizado del módulo de gestión de papeletas.

Fecha de Ejecución: _____

Pipeline: Pipeline CI/CD Jenkins

N° de Pipeline: Despliegue Automatizado QA – Producción

Proyecto / Microservicio: Módulo de Gestión de Papeletas

Versión Desplegada: _____

Commit ID: _____

Rama: Desarrollo (X) Calidad (X) Producción (X)

Responsable: Ronald Moreno Hinostroza

Etapa	Estado	Tiempo
Activación Webhook	Exitoso	No registrado
Build	Exitoso	6 min 39 seg
Análisis SonarQube	Exitoso	1 min 30 seg
Deploy QA	Exitoso	Según registro Jenkins
Validación QA	Exitoso	25 min
Deploy Producción	Exitoso	Según registro Jenkins
Verificación Final	Exitoso	No registrado

Tiempo Total Pipeline: 37 minutos.

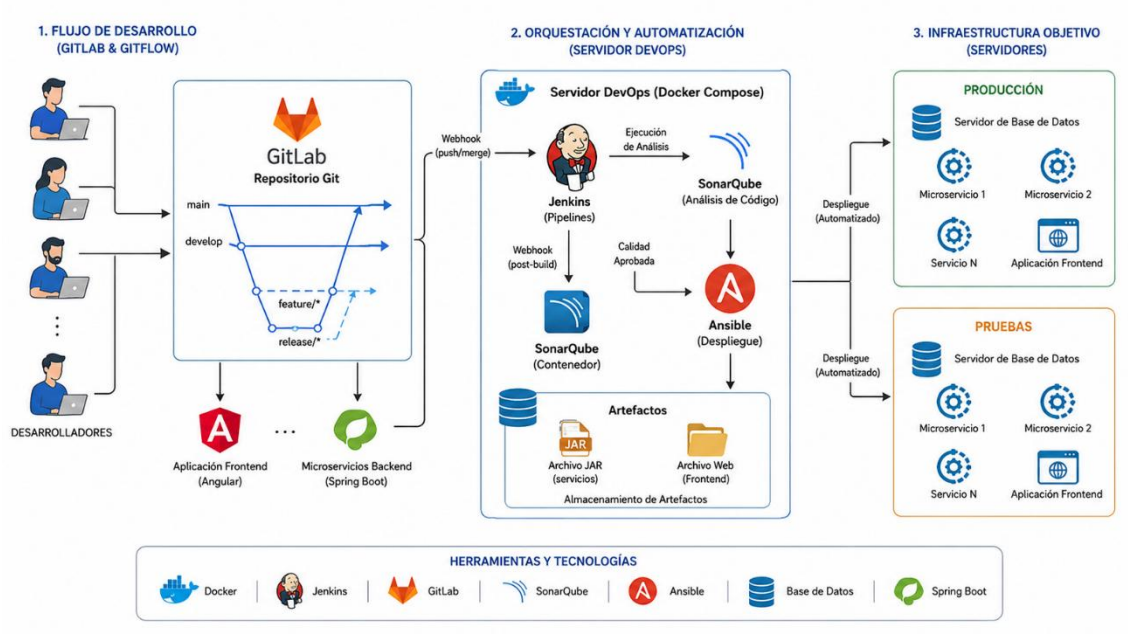
ANEXO F. EVIDENCIAS DE IMPLEMENTACIÓN DE PRINCIPIOS DEVOPS

El presente anexo contiene las evidencias técnicas que respaldan la implementación de los principios DevOps para el despliegue automatizado del módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena.

F.1 Arquitectura Devops

Diagrama de la arquitectura DevOps implementada para el módulo de gestión de papeletas, integrando GitLab, Jenkins, SonarQube, Ansible y los entornos de pruebas y producción.

Figura 1
Diagrama de la arquitectura DevOps

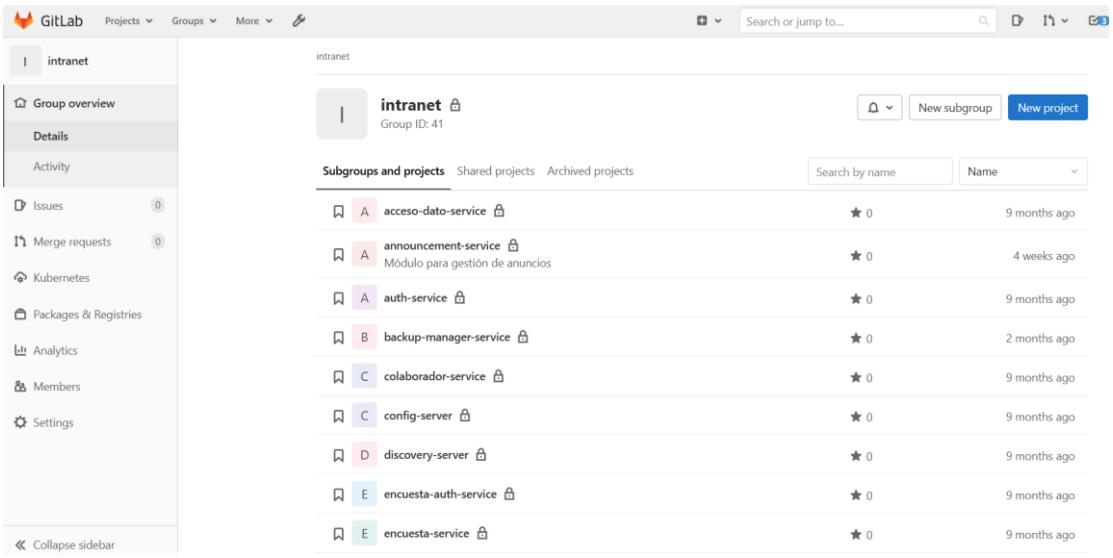


Fuente: Elaboración propia (2025).

F.2 Repositorio GitLab

Repositorio utilizado para la gestión de versiones, control de cambios y administración de ramas del módulo de gestión de papeletas.

Figura 2
Captura del Repositorio GitLab

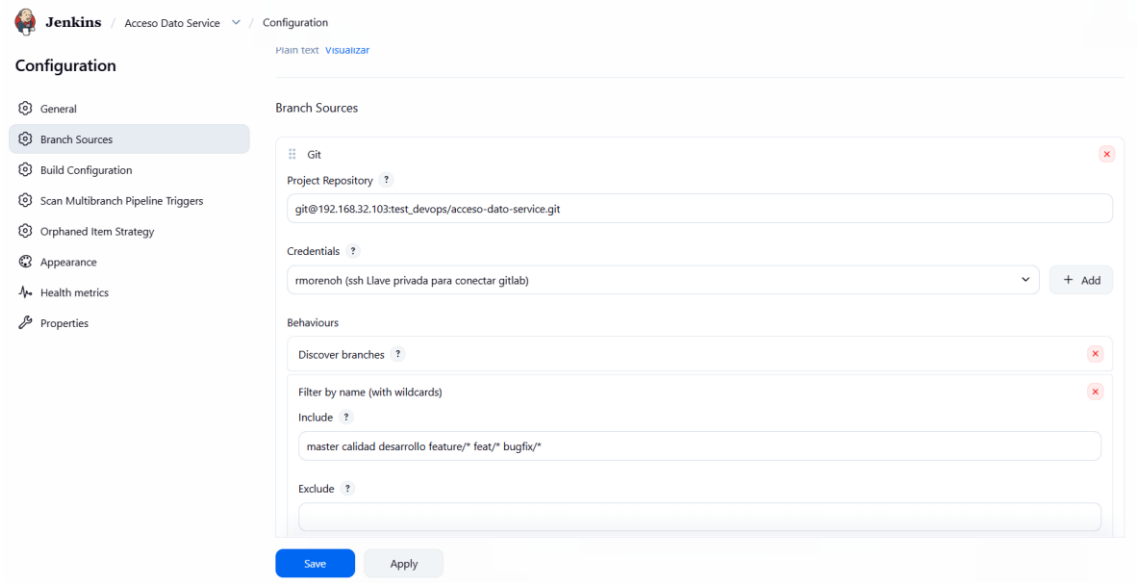


Fuente: Elaboración propia (2025).

F.3.Configuración Jenkins

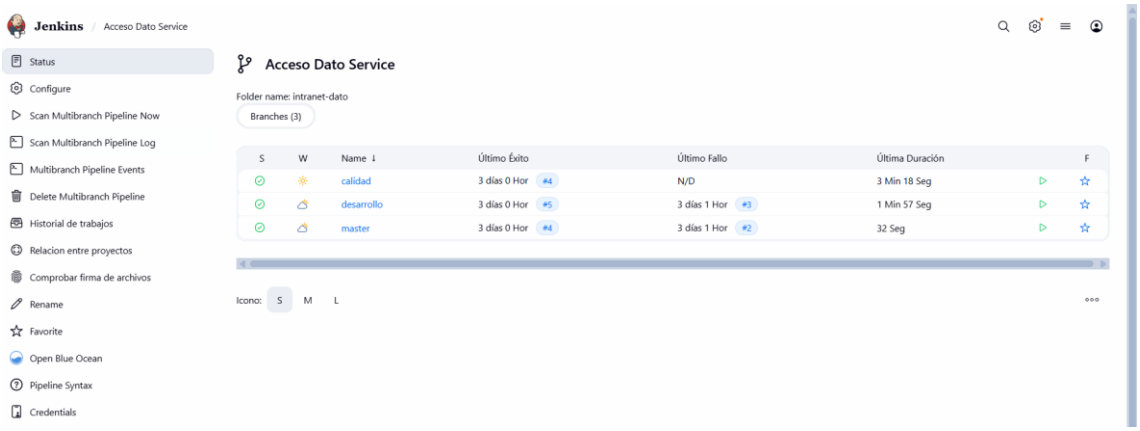
Configuración del servidor Jenkins empleado para la orquestación y ejecución automática del pipeline CI/CD.

Figura 3.
Configuración de Jenkins para Acceso a Gitlab



Fuente: Elaboración propia (2025).

Figura 4.
Sincronizando las ramas del repositorio.

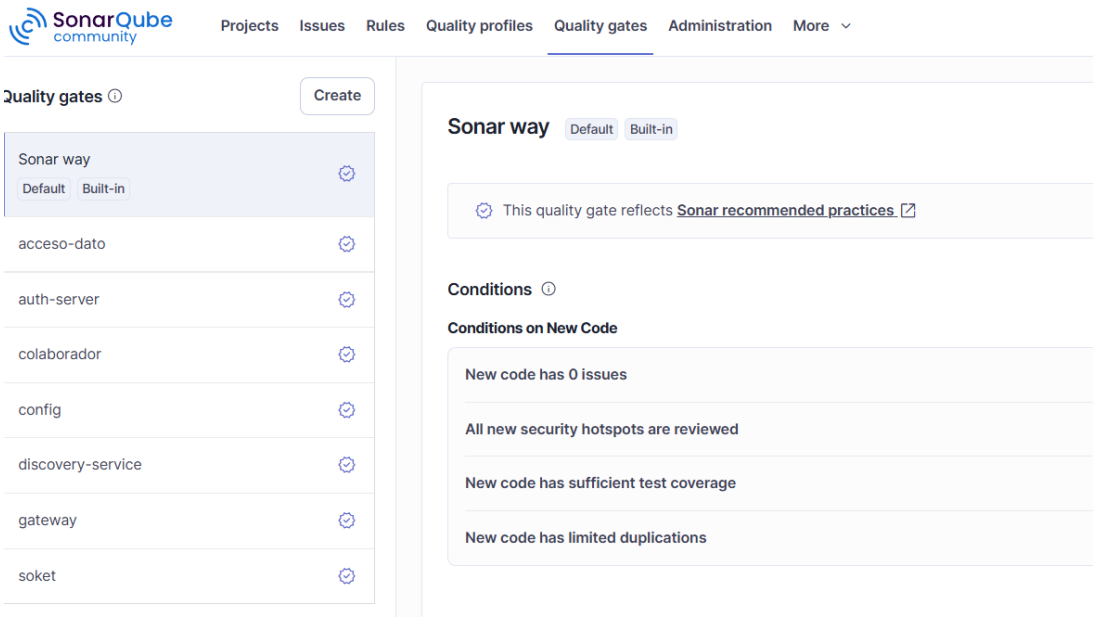


Fuente: Elaboración propia (2025).

F.4. Configuración SonarQube

Configuración de SonarQube para el análisis continuo de calidad, seguridad y deuda técnica del código fuente.

Figura 5.
Creación de Quality Gate en SonarQube

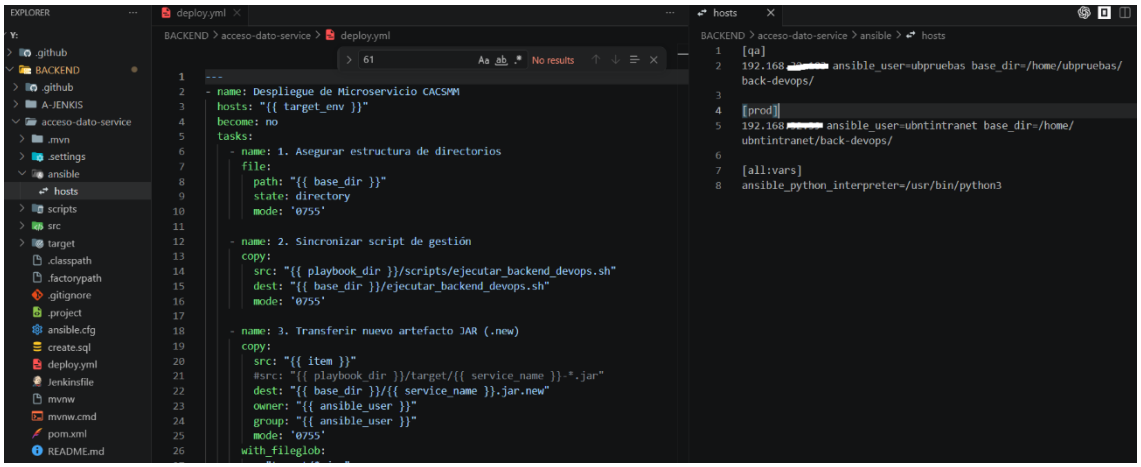


Fuente: Elaboración propia (2025).

F.5. Implementación de IaC con Ansible

Playbooks e inventarios utilizados para automatizar el despliegue y configuración de los entornos de pruebas y producción.

Figura 6.
Vista del Playbooks para el despliegue de los artefactos.

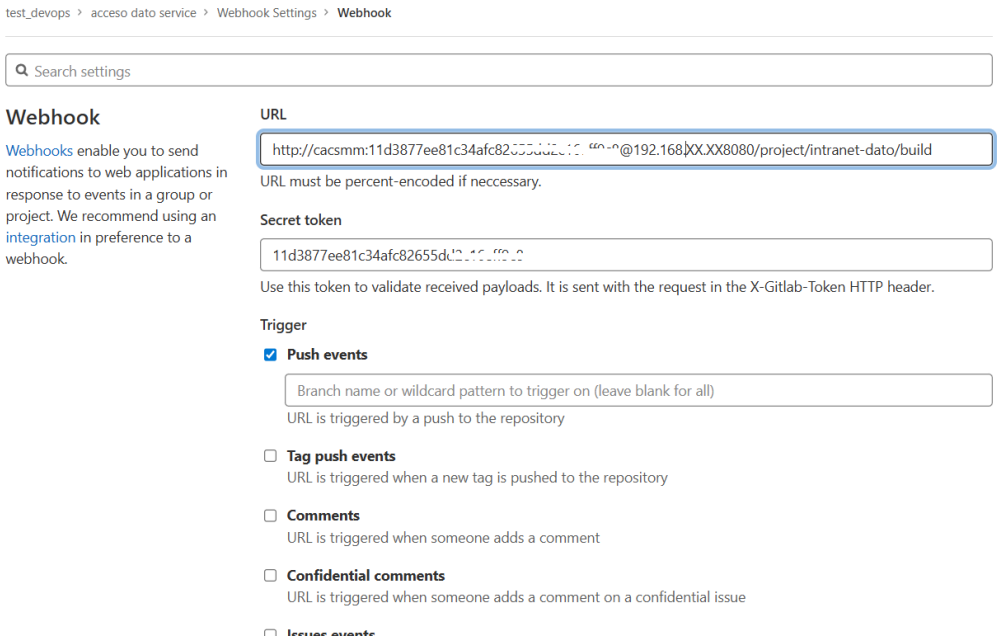


Fuente: Elaboración propia (2025).

F.6. Webhooks GitLab - Jenkins

Integración entre GitLab y Jenkins mediante Webhooks para la activación automática del pipeline CI/CD.

Figura 7.
Configuración webhook en Gitlab

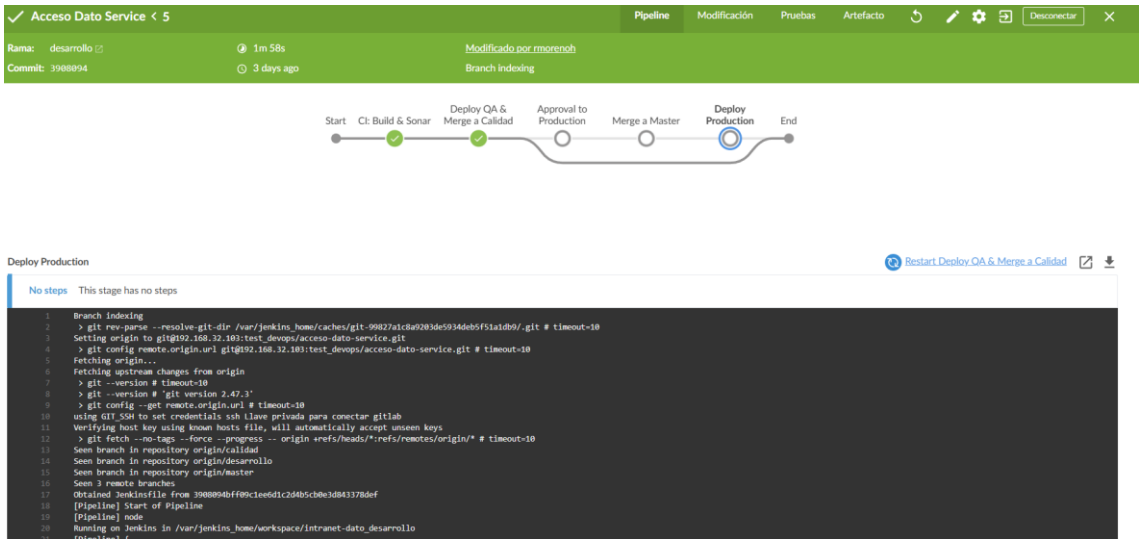


Fuente: Elaboración propia (2025).

F.7. Orquestación del Pipeline

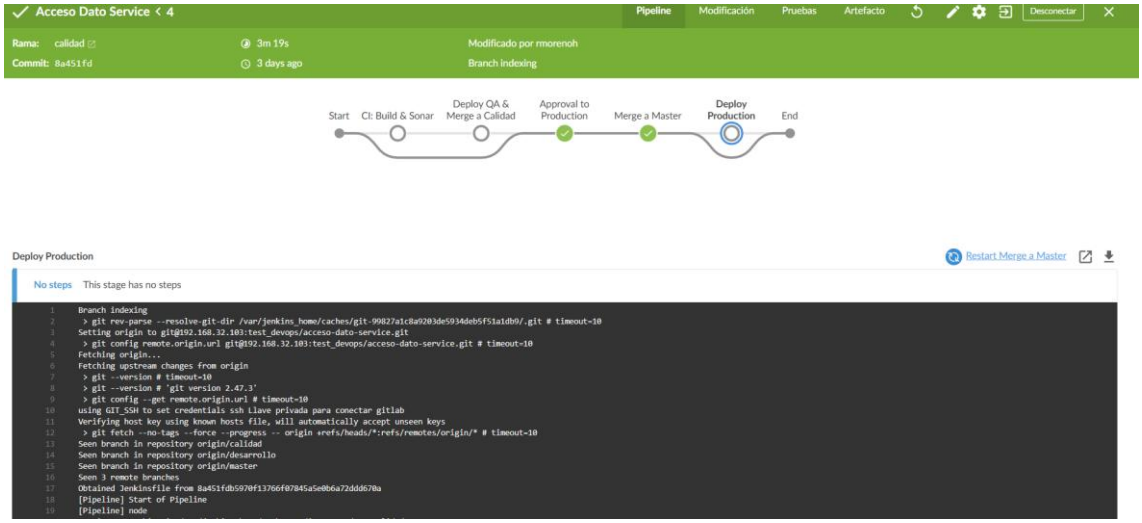
Visualización de las etapas ejecutadas automáticamente por Jenkins durante el proceso de integración continua y despliegue automatizado.

Figura 8.
Build, Sonar, deploy a pruebas y merge automático del repositorio.



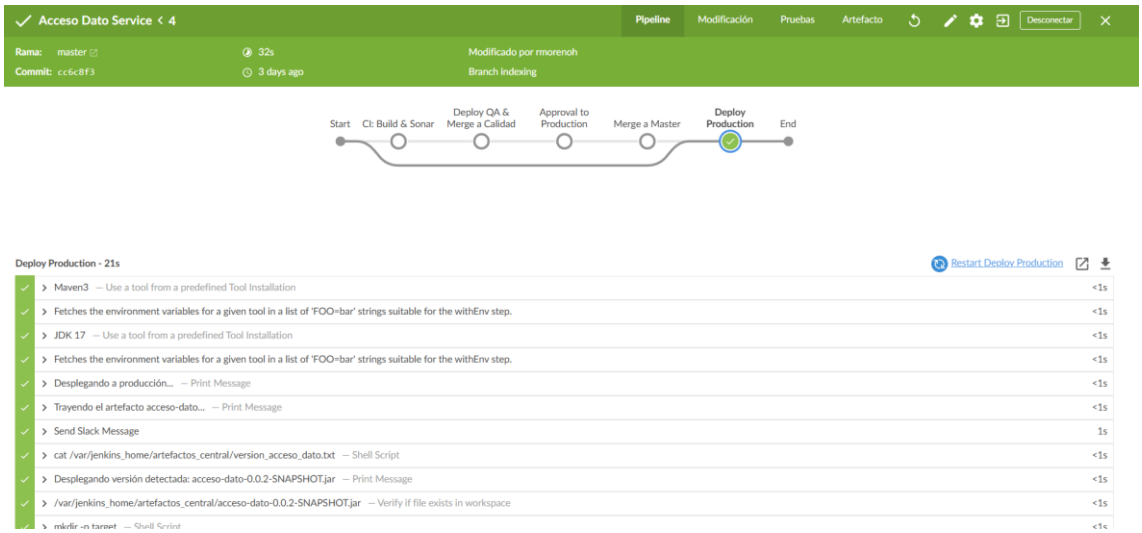
Fuente: Elaboración propia (2025).

Figura 9.
Aprobación y merge automático para el despliegue a producción



Fuente: Elaboración propia (2025).

Figura F.10.
Deploy a producción

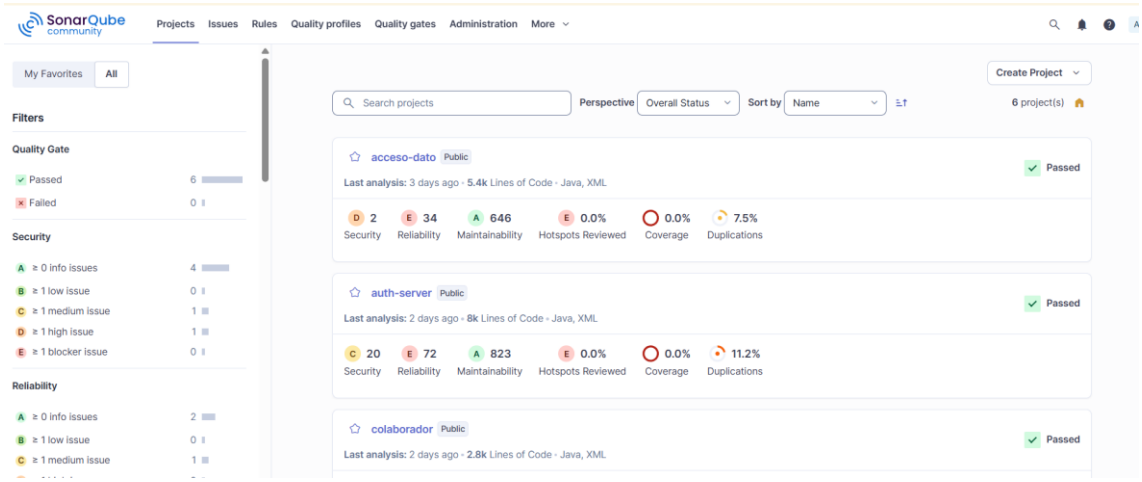


Fuente: Elaboración propia (2025).

F.8. Validación del Quality Gate

Resultado de la validación automática de calidad realizada por SonarQube antes de permitir el despliegue del código.

Figura 11.
Resultado de Quality Gate SonarQube

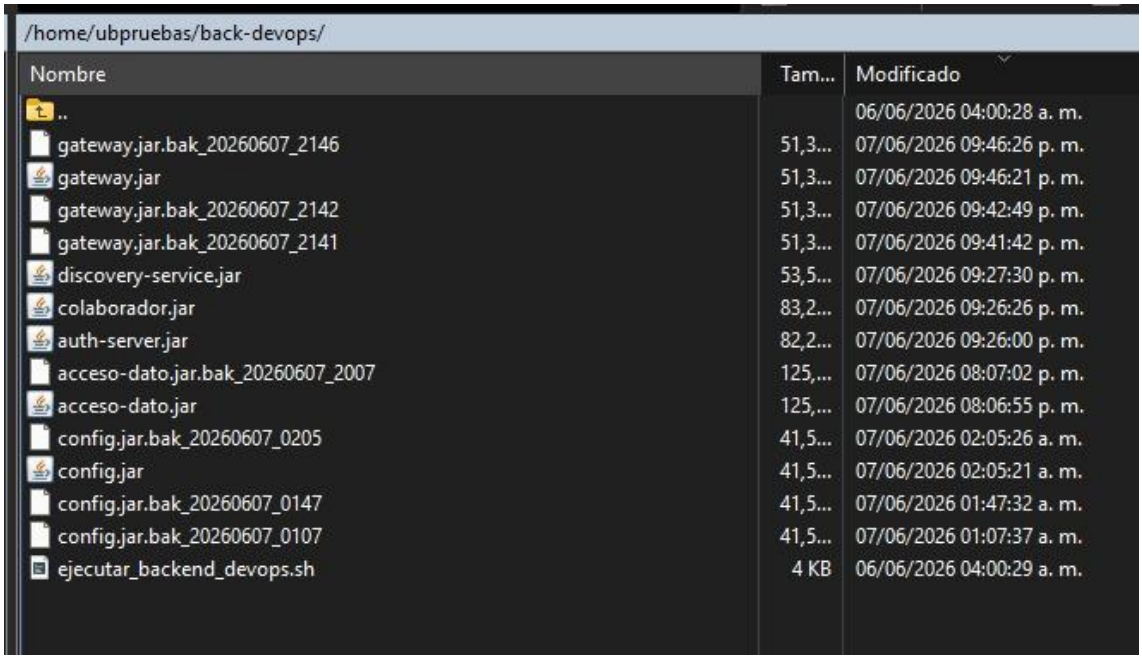


Fuente: Elaboración propia (2025).

F.9. Despliegue Automatizado (QA) y Producción

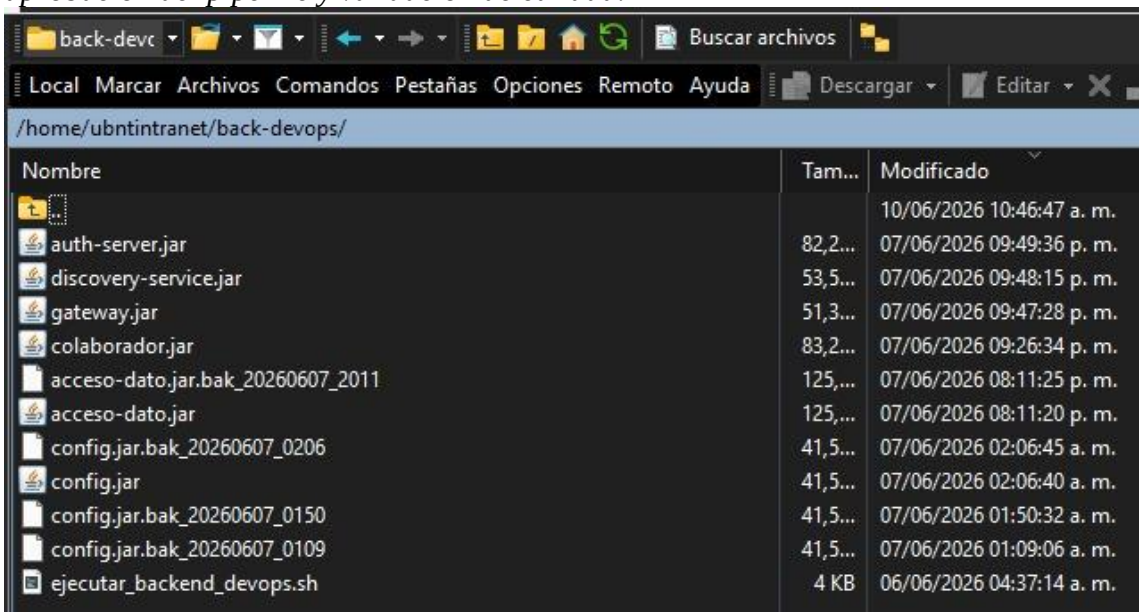
Evidencia del despliegue automatizado del artefacto en el entorno de pruebas mediante Jenkins y Ansible.

Figura 12.
Evidencia de despliegue automatizado al entorno de pruebas



Fuente: Elaboración propia (2025).

Figura 13.
Evidencia de despliegue automatizado al entorno de producción posterior a la aprobación del pipeline y validación de calidad.



Fuente: Elaboración propia (2025).

ANEXO G. REGISTROS DE LÍNEA BASE DEL DESPLIEGUE MANUAL

El presente anexo contiene las evidencias recopiladas durante el diagnóstico inicial del proceso de despliegue manual del módulo de gestión de papeletas, utilizadas para establecer la línea base de comparación previa a la implementación de los principios DevOps.

G.1 Cronometraje del Proceso de Despliegue Manual

Tabla 1.
Registro de tiempos observados durante el despliegue manual del módulo de gestión de papeletas.

Actividad	Tiempo (min)
Compilación local	
Empaquetado	
Transferencia SFTP	
Copia de archivos en servidor	
Reinicio de servicios	
Validación funcional	
Total	

Fuente: Elaboración propia (2025).

G.2 Registro de Actividades Manuales

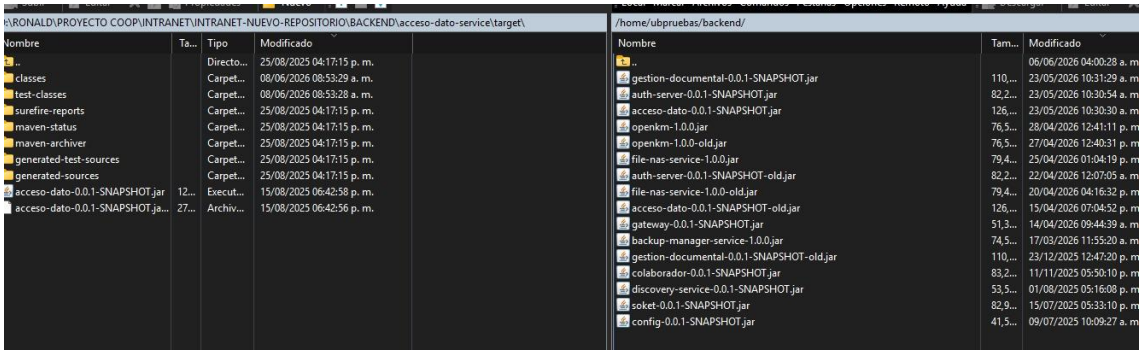
Tabla 2.
Actividades ejecutadas manualmente durante el proceso tradicional de despliegue.

N°	Actividad
1	Compilación local del proyecto
2	Generación manual del artefacto
3	Transferencia mediante SFTP
4	Acceso remoto al servidor
5	Detención de servicios
6	Copia manual de archivos
7	Reinicio de servicios
8	Verificación manual de logs
9	Validación funcional

Fuente: Elaboración propia (2025).

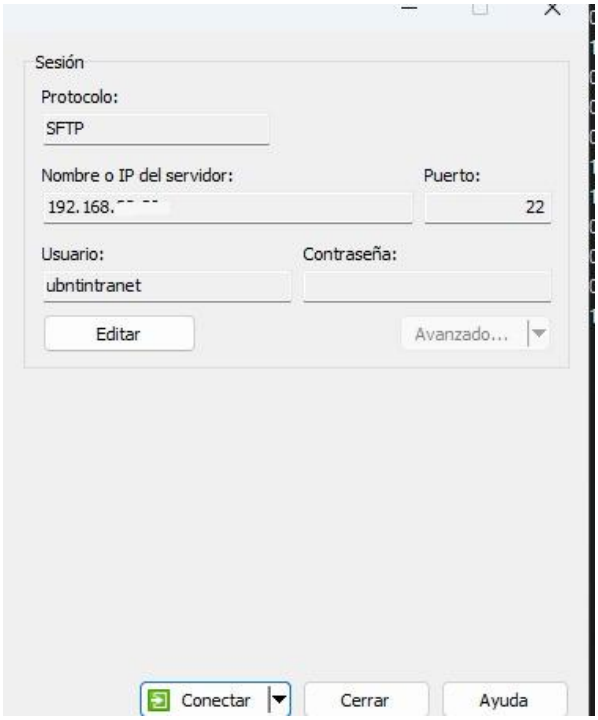
G.3 Evidencia de Transferencia Manual mediante SFTP

Figura 14.
Transferencia manual de artefactos hacia el servidor mediante herramienta SFTP.



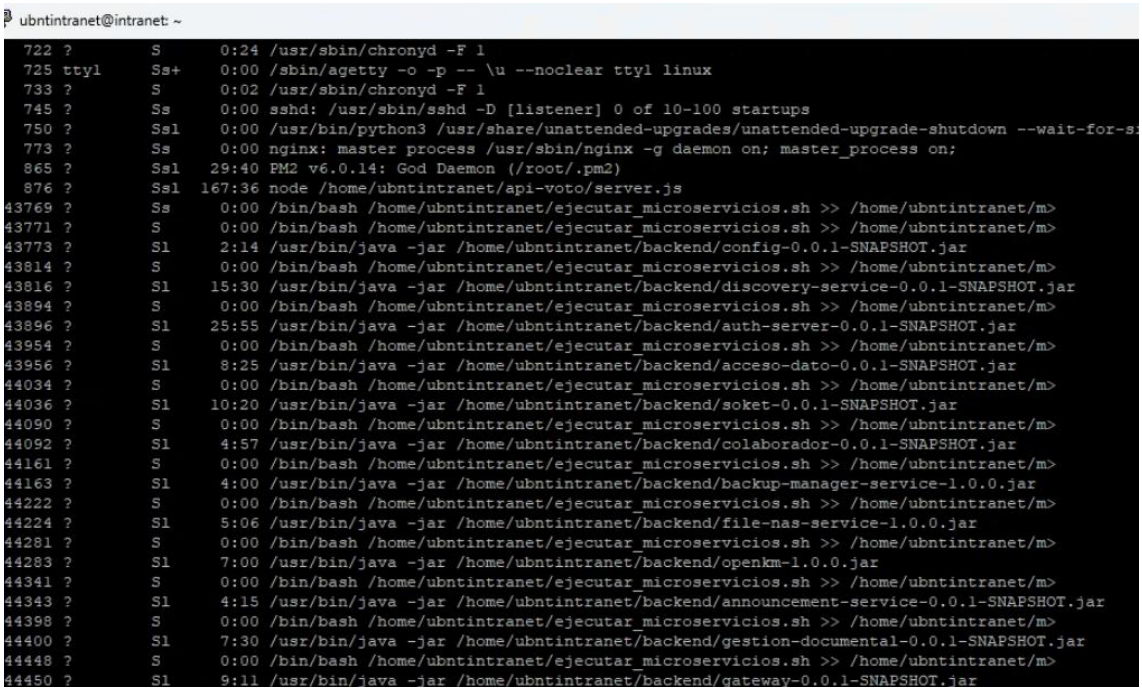
Fuente: Elaboración propia (2025).

Figura 15.
Evidencia de Acceso remoto Manual al Servidor



Fuente: Elaboración propia (2025).

Figura 16.
Evidencia de Validación Manual Posterior al Despliegue



Fuente: Elaboración propia (2025).

G.6 Resumen de la Línea Base Operativa

Tabla 3
Indicadores observados antes de la implementación de DevOps.

Indicador	Valor
Tiempo promedio de despliegue	168 min
Tiempo máximo registrado	276 min
Frecuencia de despliegue	Mensual / Quincenal
Transferencia manual de archivos	Sí
Uso de CI/CD	No
Uso de IaC	No
Rollback automatizado	No
Quality Gate	No

Fuente: Elaboración propia (2025).

ANEXO H. RESULTADOS DEL DESPLIEGUE AUTOMATIZADO

El presente anexo contiene las evidencias técnicas obtenidas durante la ejecución del pipeline CI/CD implementado mediante principios DevOps para el despliegue automatizado del módulo de gestión de papeletas.

H.1 Registro de Tiempos de Ejecución del Pipeline

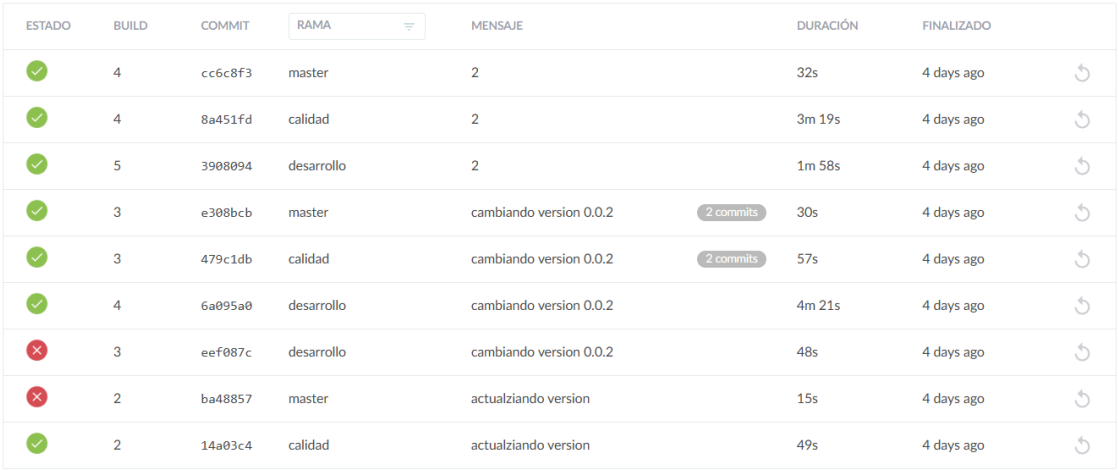
Tabla 4
Tiempos registrados durante la ejecución del pipeline automatizado.

Etapa del Pipeline	Tiempo
Build (Compilación)	6 min 39 seg
Análisis SonarQube	1 min 30 seg
Validación QA	25 min
Despliegue Producción	Según registro Jenkins
Tiempo Total End-to-End	37 min

Fuente: Jenkins (2025).

H.2 Historial de Builds en Jenkins

Figura 17
Historial de Ejecuciones Exitosas del Pipeline en Jenkins

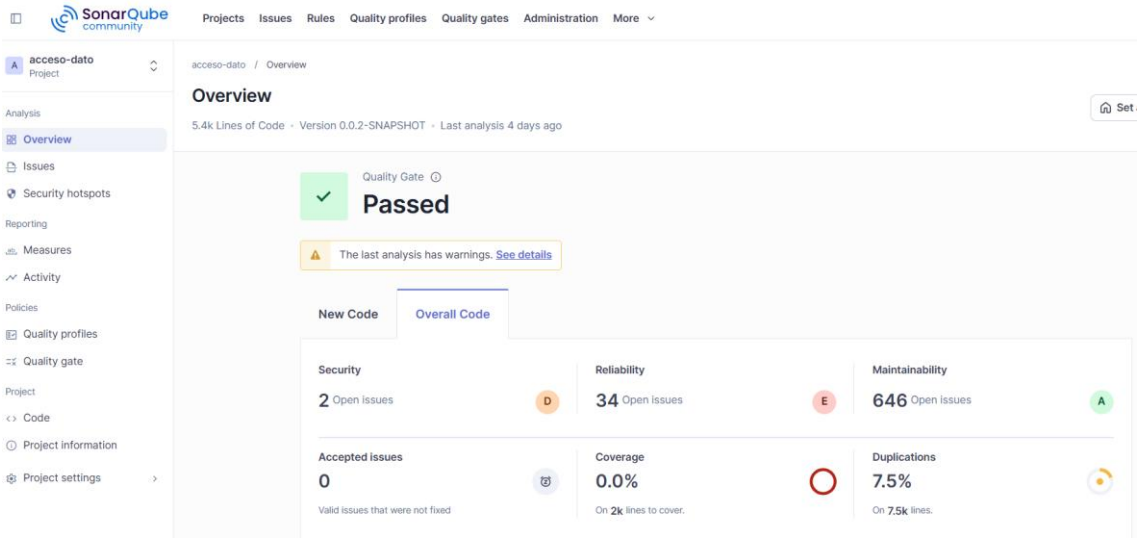


ESTADO	BUILD	COMMIT	RAMA	MENSAJE		DURACIÓN	FINALIZADO
✓	4	cc6c8f3	master	2		32s	4 days ago
✓	4	8a451fd	calidad	2		3m 19s	4 days ago
✓	5	3908094	desarrollo	2		1m 58s	4 days ago
✓	3	e308bcb	master	cambiando version 0.0.2	2 commits	30s	4 days ago
✓	3	479c1db	calidad	cambiando version 0.0.2	2 commits	57s	4 days ago
✓	4	6a095a0	desarrollo	cambiando version 0.0.2		4m 21s	4 days ago
✗	3	eef087c	desarrollo	cambiando version 0.0.2		48s	4 days ago
✗	2	ba48857	master	actualziando version		15s	4 days ago
✓	2	14a03c4	calidad	actualziando version		49s	4 days ago

Fuente: Jenkins (2025).

Figura 18

Validación del Quality Gate y Resultados obtenidos durante el análisis estático de código realizada por SonarQube.

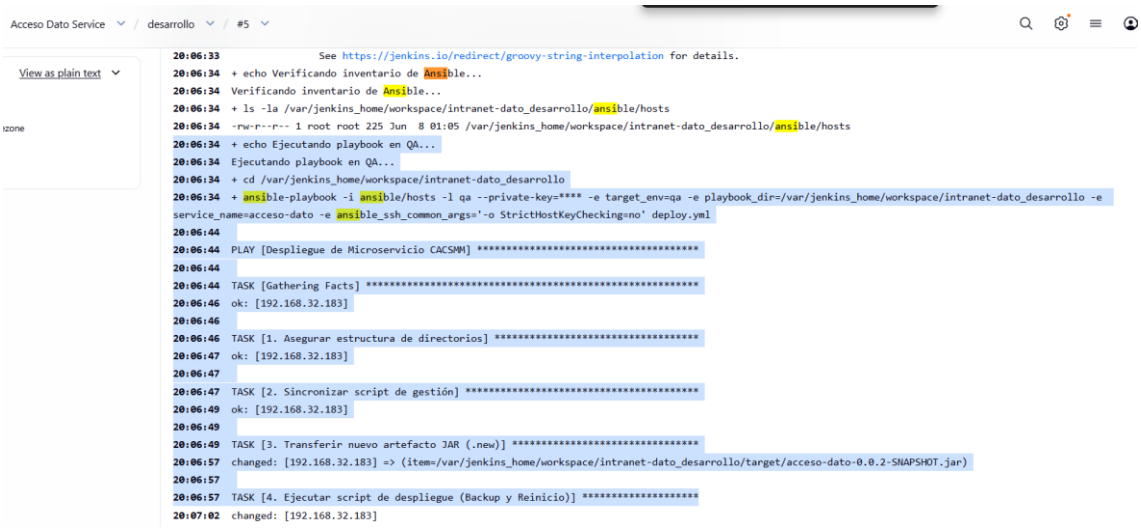


Fuente: SonarQube (2025).

H.2 Evidencia de Despliegue Automatizado en QA y Producción

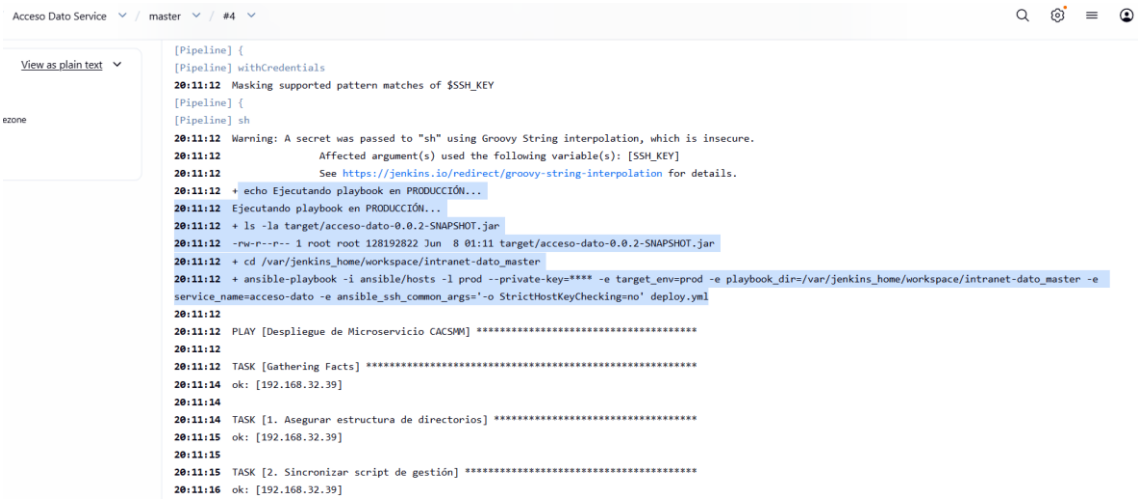
Figura 19

Despliegue automatizado realizado mediante Jenkins y Ansible en el entorno de pruebas.



Fuente: Jenkins y Ansible (2025).

Figura 20
Despliegue automatizado realizado en el entorno de producción posterior a la aprobación del pipeline.



Fuente: Jenkins y Ansible (2025).

H.3 Resumen de Indicadores Operativos Posteriores a DevOps

Tabla 5
Indicadores observados después de la implementación de DevOps.

Indicador	Valor
Tiempo promedio de despliegue	37 min
Reducción del tiempo de despliegue	77.98 %
Frecuencia de despliegue	Bajo demanda
Uso de CI/CD	Sí
Uso de SonarQube	Sí
Uso de Ansible	Sí
Rollback automatizado	No
Quality Gate	Sí

Fuente: *Elaboración propia (2025).*

El rollback automatizado no formó parte del alcance principal de la implementación. No obstante, el uso de GitLab y Jenkins permite recuperar una versión anterior mediante la reejecución controlada del pipeline sobre un commit estable, por lo que se considera un mecanismo de recuperación semiautomatizado. Por ello, se recomienda incorporar rutinas de rollback automatizado en futuras versiones del pipeline.

UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGIA Y CIVIL

ACTA DE SUSTENTACIÓN DE TESIS Nº 70-2026-FIMGC

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

En la Universidad Nacional de San Cristóbal de Huamanga, en la ciudad de Ayacucho, en cumplimiento a la **Resolución Decanal No 421-2026-FIMGC-D**, a los **veintidós días del mes de septiembre de 2026**, siendo las **10:09 a.m.**, reunidos en el **Auditorio de la Escuela Profesional de Ingeniería Minas**, bajo la presidencia del **Dr. Víctor Félix FLORES MORENO**, y los miembros: **Dr. Manuel Avelino LAGOS BARZOLA**, **Mg. Karel PERALTA SOTOMAYOR** y **Mg. Oscar MENESES YARANGA**, actuando como secretario docente el **Ing. Saul Walter RETA-MOZO FERNANDEZ**, para proceder a la sustentación de tesis para optar el **Título Profesional de Ingeniero de Sistemas**, del Bachiller en Ingeniería de Sistemas:

RONALD ANDERSON MORENO HINOSTROZA

Quien presentó la tesis denominada:

"Principios DevOps para el despliegue automatizado de un módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena - Ayacucho, 2025"

Los señores miembros del jurado luego de expuesta la tesis y absueltas las preguntas, deliberaron y declararon:

Aprobado con 17 (diecisiete)

Siendo las **11:16 a.m.** del día **22 de septiembre del 2026**, culmina el acto de sustentación de tesis, y en conformidad de lo actuado los miembros del jurado firmamos al pie del presente.

Dr. Víctor Félix FLORES MORENO
Presidente

Mg. Karel PERALTA SOTOMAYOR
Miembro

Mg. Oscar MENESES YARANGA
Miembro

Dr. Manuel Avelino LAGOS BARZOLA
Miembro - Asesor

Ing. Saul Walter RETAMOZO FERNANDEZ
Secretario docente de la FIMGC

FACULTAD DE INGENIERÍA
DE MINAS Y CIVIL
Av. Independencia S/N
Ciudad Universitaria
Central Tel. 066 312510
Anexo 151

CONSTANCIA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN

CONSTANCIA Nº 02-2026-CEMC-FIMGC/UNSCH

El que suscribe; responsable verificador de originalidad de trabajos de tesis de pregrado con el software Turnitin, en segunda instancia para las **Escuelas Profesionales** de la **Facultad de Ingeniería de Minas, Geología y Civil**; en cumplimiento a la **Resolución de Consejo Universitario Nº 039-2021-UNSCH-CU**, Reglamento de Originalidad de Trabajos de Investigación de la Universidad Nacional San Cristóbal de Huamanga y **Resolución Decanal Nº 507-2026-FIMGC-D**, deja constancia de originalidad de trabajo de investigación, que el/la Sr./Srta.

Nombres y Apellidos	:	Ronald Anderson MORENO HINOSTROZA
Escuela Profesional	:	INGENIERÍA DE SISTEMAS
Título de la Tesis	:	"Principios DevOps para el despliegue automatizado de un módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025"
Evaluación de la Originalidad	:	**7%** Índice de Similitud
Identificador de la entrega	:	3087152004

Por tanto, según los Artículos 12, 13 y 17 del Reglamento de Originalidad de Trabajos de Investigación, es **PROCEDENTE** otorgar la **Constancia de Originalidad** para los fines que crea conveniente.

En señal de conformidad y verificación se firma la presente constancia

Ayacucho, 11 de noviembre de 2026

Mg.Celia Edith Martinez Córdova

Con depósito para Sustentación y Tramites
C.c. Archivo

FACULTAD DE INGENIERIA DE MINAS, GEOLOGIA Y CIVIL
Av. Independencia S/N Ciudad Universitaria
Central Tel. 066 312510
Anexo 151

“Principios DevOps para el despliegue automatizado de un módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025”

por Ronald Anderson Moreno Hinostroza

Fecha de entrega: 11-nov-2026 05:09p. m. (UTC-0500)

Identificador de la entrega: 3087152004

Nombre del archivo:
_Principios_DevOps_para_el_despliegue_automatizado_de_un_módulo_de_gestión_de_papeletas_basado_en_microservicios_en_la_Cooperativa_Santa_María_Magdalena_Ayacucho_20
(3.19M)

Total de palabras: 25795

Total de caracteres: 166100

“Principios DevOps para el despliegue automatizado de un módulo de gestión de papeletas basado en microservicios en la Cooperativa Santa María Magdalena – Ayacucho, 2025”

INFORME DE ORIGINALIDAD

7%	6%	2%	3%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Nacional de San Cristóbal de Huamanga Trabajo del estudiante	2%
2	hdl.handle.net Fuente de Internet	1%
3	repositorio.unitec.edu Fuente de Internet	1%
4	dspace.ups.edu.ec Fuente de Internet	1%
5	revistas.unisimon.edu.co Fuente de Internet	<1%
6	biblio.ingenieria.usac.edu.gt Fuente de Internet	<1%
7	Aguilar, Carmela Romero|Alarcón, Andric Alejandro Vargas|Zegarra, Richar Sandro Balboa|Sanchez et al. "Business Consulting: Aplicaciones de Marcos de Referencia para el Desarrollo de Software en MS4M S.A.C.", Pontificia Universidad Catolica del Peru (Peru), 2024 Publicación	<1%
8	risti.xyz Fuente de Internet	<1%
9	api.dspace.itb.edu.ec Fuente de Internet	<1%
10	repositorio.continental.edu.pe Fuente de Internet	<1%
11	apirepositorio.unu.edu.pe Fuente de Internet	<1%
12	atarazana.files.wordpress.com Fuente de Internet	<1%
13	repositorio.une.edu.pe Fuente de Internet	<1%

14	documents.mx Fuente de Internet	<1 %
15	repositorio.uisrael.edu.ec Fuente de Internet	<1 %
16	rua.ua.es Fuente de Internet	<1 %
17	Submitted to Universidad Católica Santo Toribio de Mogrovejo Trabajo del estudiante	<1 %
18	core.ac.uk Fuente de Internet	<1 %
19	repositorio.unh.edu.pe Fuente de Internet	<1 %
20	www.coursehero.com Fuente de Internet	<1 %
21	Belinda Lema-Cachinell, Emma Delgado-Saeteros. "AVANCES DE INVESTIGACIÓN 2026/Nro4", ACVENISPROH Académico, 2026 Publicación	<1 %
22	Submitted to Universidad TecMilenio Trabajo del estudiante	<1 %
23	documentop.com Fuente de Internet	<1 %

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		