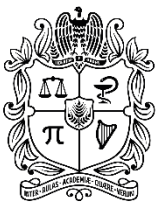


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Desarrollo de una metodología para definir el proceso DevOps y su implementación en la industria de tecnologías de la información

Yennifer Yaneth Montes Caraballo

Universidad Nacional de Colombia
Facultad de Minas, Área Curricular de Sistemas e Informática
Medellín, Colombia
2024

Desarrollo de una metodología para definir el proceso DevOps y su implementación en la industria de tecnologías de la información

Yennifer Yaneth Montes Caraballo

Tesis de investigación presentada como requisito parcial para optar al título de:

Maestría en Ingeniería de Sistemas

Director (a):

Fernán Alonso Villa Garzón, PhD

Línea de Investigación:

Sistemas

Universidad Nacional de Colombia

Facultad de Minas, Área Curricular de Sistemas e Informática

Medellín, Colombia

2024

Dedicatoria

Con profundo agradecimiento, dedico esta tesis a mis padres, Yerlis y Wilfrido, este logro es un testimonio de su inmenso amor y dedicación.

A mi querida hermana Luisa, por su constante apoyo y motivación. A mi tía Narlis, agradecida por sus sabios consejos y profundo cariño.

A Donayd y Bobby por ser mis compañeros de vida. Gracias por hacer este camino más significativo y por compartir cada etapa de este logro conmigo.

Agradecimientos

Quisiera expresar mi profunda gratitud a todas las personas que contribuyeron de manera significativa a la realización de esta tesis. En primer lugar, a la Universidad Nacional de Colombia, por brindarme la oportunidad de llevar a cabo este proyecto de investigación. Su compromiso con la excelencia académica y su invaluable apoyo han sido fundamentales para mi desarrollo profesional.

Mi más sincero agradecimiento se dirige hacia mi director de tesis, el Dr. Fernán Alonso Villa Garzón. Su orientación experta, dedicación y paciencia han sido esenciales en cada etapa de este proceso. Su sabiduría académica y su disposición para compartir sus conocimientos han enriquecido enormemente mi experiencia y han sido cruciales para el éxito de esta investigación.

Además, quiero agradecer a todos los profesores, compañeros de clase y personal administrativo que han contribuido de diversas maneras a mi formación académica. Su influencia positiva ha dejado una marca indeleble en mi trayectoria académica.

A mi familia y amigos, quienes han sido mi red de apoyo incondicional, gracias por su paciencia, aliento y comprensión a lo largo de este camino. Cada uno de ustedes ha desempeñado un papel vital en mi éxito.

Resumen

Desarrollo de una metodología para definir el proceso DevOps y su implementación en la industria de tecnologías de la información

Descripción: En un entorno empresarial donde la tecnología es esencial, la industria de Tecnologías de la Información (TI) busca constantemente maneras de agilizar la entrega de software de calidad. En este contexto, DevOps ha surgido como un enfoque clave, fomentando la colaboración entre equipos de desarrollo y operaciones, la automatización de procesos y la entrega continua de software.

Esta tesis de maestría tiene como objetivo desarrollar una metodología integral para definir y poner en práctica DevOps en organizaciones de TI. A través de una revisión exhaustiva de la literatura y el análisis de las mejores prácticas, se presenta una guía que abarca desde los fundamentos de DevOps hasta su aplicación práctica en entornos empresariales.

Palabras clave: Devops, Arquitectura, Infraestructura, Agilismo.

Abstract

Development of a methodology to define the Devops process and its implementation in the IT industry.

Description: In a business environment where technology is essential, the Information Technology (IT) industry is constantly looking for ways to streamline the delivery of quality software. In this context, DevOps has emerged as a key approach, fostering collaboration between development and operations teams, process automation and continuous software delivery.

This master thesis aims to develop a comprehensive methodology for defining and implementing DevOps in IT organizations. Through a comprehensive literature review and analysis of best practices, a guide is presented that spans from the fundamentals of DevOps to its practical application in enterprise environments.

Keywords: Devops, Architecture, Infrastructure, Agile.

Contenido

Lista de figuras

Pág.

Lista de Símbolos y abreviaturas

Abreviaturas

Abreviatura	Término
DevOps	Development Operations
CI	Continuous Integration
CD	Continuous Deployment
ML	Machine Learning
CM	Continuous Monitoring
IAC	Infrastructure as Code
SAAC	Software as a Code
PAAS	Platform as a Service
SAAS	Software as a Service
TFS	Team Foundation Service
VSC	Visual Studio Code
CLI	Command Line Interface
API	Application Programming Services
AWS	Amazon Web Service
GCP	Google Cloud Platform
SDN	Software Defined Networking
BPMN	Business Process Model and Notation
HCL	Hashicorp Configuration Language
TI	Tecnologías de la Información

1. Introducción

La industria de Tecnologías de la Información (TI) ha experimentado una transformación radical en los últimos años, impulsada por la creciente demanda de soluciones digitales ágiles y eficientes. En este contexto, el enfoque DevOps se ha consolidado como un paradigma fundamental para la entrega de software de alta calidad de manera continua. Este enfoque promueve la colaboración estrecha entre los equipos de desarrollo y operaciones, automatizando procesos y reduciendo los tiempos de entrega, lo que permite a las organizaciones responder de manera más efectiva a las demandas cambiantes del mercado. (Christof Ebert, Gorka Gallardo, Josune Hernantes & Nicolas Serrano, 2016).

Las empresas han optado por el uso de tecnologías vanguardistas, que proporcionen mejoras en las labores diarias, tanto a nivel empresarial como al desarrollo profesional de cada uno de sus empleados. La implementación de nuevas tecnologías genera vacantes que conllevan a nuevas contrataciones dentro del sector tecnológico. La evolución constante de los empleados y las empresas requiere un estudio en el que se visualice las nuevas tendencias en metodologías que brinden mejoras a los procesos existentes.

La capacidad de las organizaciones para prestar servicios y aplicaciones a gran velocidad exige competir eficazmente en el mercado. Las prácticas y herramientas para estos procesos de gestión exigen un modelo rápido y fiable. Los cambios deben comenzar en el nivel de ingeniería de software cuando se crean aplicaciones en la nube, por lo que es necesario automatizar nuestros procesos de DevOps utilizando herramientas de automatización de DevOps en la nube y fuera de la nube. (Dhaya Sindhu Battina, 2020)

En los últimos años se ha visto la necesidad en Colombia de implementar un método de trabajo que ya ha sido implementada a nivel de software en varios países con muchos casos de éxitos, denominada Devops. Para que esta implementación sea exitosa se

requiere realizar varios cambios en la metodología, los procesos y la cultura empresarial. Aplicar un proceso Devops implica que varios equipos de una misma empresa, que antes trabajaban en un molde separado, se unan para formar una cadena organizada capaz de superar los obstáculos diarios.

El desarrollo colaborativo y la operación y mantenimiento de DevOps, basados en la automatización, realizan la entrega continua de software, y mejoran la calidad del producto y la eficiencia de la entrega. Mediante el establecimiento de un canal de entrega, el valor empresarial se formará desde la demanda hasta la entrega final a los usuarios, y se construirá un proceso integrado con la entrega como núcleo, lo que hará que la eficiencia de la entrega del producto sea más eficaz y la calidad más fiable. (Wen, Qian, Liu, 2022)

DevOps es un paradigma de desarrollo de software ampliamente adoptado en la industria (Mann, Brown, Stahnke, Kersten, 2018). Este interés en la adopción se debe a las ganancias en valor comercial reportadas por profesionales de la industria e investigadores académicos (Sciences, 2020). El beneficio más comúnmente informado es la capacidad de implementar versiones más rápido y con mayor frecuencia.

Estudios han demostrado que Devops obtiene como resultado un aumento en la productividad, trabajo en equipo, optimización del software y mejora en la experiencia de usuario. Sin embargo, al ser una metodología reciente, no se tiene una guía estandarizada que explique qué fases posee, cuáles son los desafíos que enfrentan las empresas al implementarlas, cuáles son las herramientas necesarias, como se pueden medir las métricas antes y después de su implementación, entre otros interrogantes (Lewerentz, Bluhm, Daher, Dumke, Grahl, Grün & Werner, 2019).

DevOps reduce la brecha entre los desarrolladores, las operaciones y el usuario final, lo que permite la detección temprana de problemas. En el pasado, después de cada sprint de Scrum, el software funcionaba según las especificaciones, pero éstas no eran validadas por el usuario final real. Con DevOps, podemos implementar el desarrollo continuo y software al usuario final con frecuencia. DevOps también permite a los desarrolladores y operaciones trabajar juntos de forma más eficiente y eficaz (Bass, Weber, Zhu, 2015).

La estrategia de integración continua es muy importante ya que ayuda a evitar incompatibilidades, y a agilizar las pruebas unitarias en el sistema. Estas pruebas pueden descubrir errores en el sistema implementado, por lo que el equipo puede corregir rápidamente el error en el mismo día, como dice Wells: "Reparar problemas pequeños cada pocas horas lleva un poco menos de tiempo que solucionar problemas grandes antes de la fecha límite". Y en relación con las pruebas de aceptación según cronogramas extremos, son especificados por el cliente y el foco siempre está en las características y funcionalidad de la prueba en base a las historias desarrolladas y probadas por el equipo. Los artefactos creados pueden ser: implementarse y usarse en el proceso de prueba estandarizado, descubrir problemas y errores antes del lanzamiento a producción, incluidas también pruebas para UI (interfaz de usuario), carga, integración, API (interfaz de programación de aplicaciones), entre otros (Herrero, 2017).

Después de revisar la investigación de campo se evidencia que las Empresas deben analizar el proceso Devops antes de decidir implementarlo, todos deben ser conscientes de que es un proceso continuo y debe implementarse gradualmente en las empresas y todos los integrantes deben ser parte de eso de esta mejora.

Los profesionales de la tecnología de la información deben estar motivados para proponer nuevas herramientas DevOps, y las empresas deben estar abiertas para que estos profesionales tengan un tiempo fijo para estudios y pruebas. Además, es necesario conocer en profundidad las herramientas que utiliza la empresa. Durante el proceso de integración de un nuevo empleado de tecnologías de la información en la empresa, por ejemplo, se sugiere que se realice una presentación sobre las tecnologías utilizadas, centrándose en el mismo contenido evaluado en el cuestionario de este trabajo.

1.1 Planteamiento del problema

Desarrollar una cultura en torno a DevOps es un reto para todos, y con la revisión de la literatura se pudo observar que las inversiones de las empresas en metodologías ágiles han sido un paso importante hacia la consecución de este objetivo. Sin embargo, es fundamental comprender que la adopción exitosa de DevOps va más allá de la implementación de herramientas y prácticas ágiles, ya que implica una transformación cultural profunda que promueve la colaboración, la comunicación y la responsabilidad

compartida entre los equipos de desarrollo y operaciones. La cultura basada en DevOps se caracteriza por facilitar el agilismo en los equipos de desarrollo, aporta beneficios a los desarrolladores y sus clientes.

Los equipos de tecnologías de la información necesitan especializarse cada vez más, preferiblemente gestionándose de tal manera que sea posible separar las actividades diarias de acuerdo con las necesidades, lo que es muy importante para el desarrollo y madurez del equipo en la empresa.

La revisión sistemática de la literatura ha revelado una serie de deficiencias en el campo de la implementación de DevOps en empresas de TI. Estas deficiencias se resumen en los siguientes puntos:

- A pesar de la existencia de diversos modelos en la literatura, no se ha logrado establecer claramente las fases necesarias para llevar a cabo una implementación efectiva de DevOps en el entorno de una empresa de tecnología de la información.

- No se han publicado modelos integrales que aborden de manera exhaustiva la implementación de DevOps, incluyendo los procesos de Integración Continua (CI) y Entrega Continua (CD).

- No se han realizado análisis detallados de las herramientas disponibles para la implementación de DevOps, organizados de acuerdo con las fases del ciclo de desarrollo.

Estos hallazgos ponen de manifiesto la necesidad imperante de desarrollar una metodología que aborde de manera sistemática la implementación de DevOps, definiendo claramente sus ciclos, comparando las herramientas disponibles en función de las etapas del ciclo, y analizando los desafíos que surgen una vez que se ha llevado a cabo la implementación. Esta investigación busca llenar estos vacíos en el conocimiento actual y proporcionar un marco sólido para la implementación exitosa de DevOps en empresas de TI.

El presente trabajo de tesis se enfoca en el desarrollo de una metodología para definir y llevar a cabo la implementación de DevOps en la industria de Tecnologías de la Información. Este proyecto se erige como una respuesta a la necesidad imperante de las organizaciones de TI de adoptar prácticas y herramientas que les permitan ser más ágiles, eficientes y competitivas en un entorno altamente dinámico.

A lo largo de esta investigación, se explorarán los fundamentos teóricos de DevOps, se analizarán las mejores prácticas y las tecnologías que lo respaldan, y se propondrá una metodología específica para guiar a las organizaciones a través del proceso de implementación de DevOps. Se examinarán, además, los beneficios y desafíos asociados con esta transformación, así como los casos de éxito que ilustran su impacto positivo en la industria de TI.

El objetivo principal de esta tesis es brindar a las organizaciones una guía práctica y sólida para adoptar DevOps de manera efectiva, promoviendo así la mejora continua en sus operaciones y su capacidad para entregar soluciones de software innovadoras y de alta calidad en un mundo digital en constante evolución. El trabajo aquí presentado busca, en última instancia, contribuir al avance y la competitividad de la industria de Tecnologías de la Información, al proporcionar una hoja de ruta clara para el desarrollo y la implementación exitosa de DevOps.

1.2 Hipótesis

Es factible desarrollar una metodología que defina y lleve a cabo la implementación de DevOps en la industria de Tecnologías de la Información, permitiendo la integración de las ventajas de los distintos equipos involucrados en el proceso (desarrolladores y operaciones). Esta metodología tiene como objetivo superar las limitaciones individuales de cada uno de los procesos en el ciclo de desarrollo de software y abordar todas las dificultades presentes en la implementación de nuevas tecnologías.

1.3 Objetivos

1.3.1 Objetivo general

Proponer una metodología integral de desarrollo de software basado en operaciones.

1.3.2 Objetivos específicos

- Caracterizar los componentes para una metodología de desarrollo de software basado en operaciones.
- Determinar los agentes para una metodología de desarrollo de software basado en operaciones.
- Especificar los procesos para una metodología de desarrollo de software basado en operaciones.
- Definir una estrategia para implementar la metodología en una empresa de desarrollo de software.

1.4 Alcances

El alcance de esta tesis de maestría es investigar y analizar los desafíos, oportunidades y estrategias para la adopción de DevOps en el contexto de las medianas empresas en Colombia. El estudio tiene como objetivo proporcionar información valiosa sobre cómo estas empresas pueden aprovechar este proceso para mejorar el rendimiento general, la calidad y los tiempos de entrega en el ciclo de desarrollo. La investigación abarcará varios aspectos para la implementación de DevOps, incluidos, entre otros, el proceso, ciclos, herramientas, agentes y componentes.

Para objeto de estudio, se definen empresas medianas como entidades con una plantilla que oscila entre 50 y 500 empleados. Este rango específico de recuento de empleados se elige para apuntar a organizaciones que son lo suficientemente grandes como para tener estructuras operativas establecidas e infraestructuras de TI potenciales, pero que aún pueden enfrentar desafíos distintos en comparación con las empresas más grandes. El estudio tendrá como objetivo diseñar una metodología, que muestre las partes del proceso

y como pueden ser ejecutados por los diferentes agentes en las empresas medianas de TI en Colombia.

1.5 Mapa del documento

El documento se estructura de la siguiente manera: en el Capítulo 2 se explican los agentes presentes en la metodología Devops, se explican sus funciones y principales aportes al proceso. En el capítulo 3 se detallan los componentes que se utilizan en el ciclo de desarrollo, en este capítulo se explican las herramientas que utiliza cada uno de los agentes presentes en el proceso.

El capítulo 4 aborda la definición del proceso DevOps, explicando los procesos y ciclos que los conforman. Para la definición de los procesos se utilizan gráficos BPMN, diseñados utilizando Bizagi Model, que facilitan la comprensión del proceso y su puesta en práctica.

Finalmente, el capítulo 5 recoge las conclusiones derivadas de este trabajo y define los trabajos a futuro en esta línea de investigación.

2. Agentes

La metodología DevOps es un enfoque integral para el desarrollo de software que busca integrar de manera efectiva el desarrollo (Dev) y las operaciones (Ops) en un ciclo de vida de desarrollo continuo (Mendes Calo, Estévez, 2010). Dentro de este marco de trabajo, diferentes agentes desempeñan roles cruciales para lograr una implementación exitosa y eficiente de los principios DevOps. En este contexto, los agentes claves incluyen desarrolladores, probadores, diseñadores, gerentes de proyecto, managers, expertos en seguridad y administradores de nube. A continuación, se explorarán detalladamente las funciones y relevancia de cada uno de estos agentes en el proceso DevOps.

2.1 Gerentes de proyecto

Los gerentes de proyecto desempeñan un rol coordinador y estratégico en el desarrollo de software. Supervisan y coordinan las actividades del equipo, establecen metas y plazos para la entrega de software, y facilitan la comunicación entre los diferentes equipos involucrados. Su relevancia se manifiesta en asegurar que el proyecto se adhiera a los plazos y presupuestos establecidos, facilitar la colaboración entre equipos para lograr objetivos comunes, y contribuir a la gestión eficiente de recursos y riesgos.

2.2 Administradores (Manager)

Los managers tienen un papel estratégico en la implementación exitosa de prácticas DevOps. Definen estrategias y objetivos comerciales, facilitan la alineación entre los equipos de desarrollo y operaciones, y toman decisiones estratégicas para mejorar la eficiencia y la calidad. Su relevancia se evidencia al garantizar que las prácticas DevOps estén alineadas con los objetivos del negocio, impulsar la cultura de colaboración y mejora continua, y facilitar la adopción exitosa de prácticas DevOps a nivel organizativo. (Lwakatare, Kilamo, Karvonen, 2019).

2.3 Diseñadores

Los diseñadores juegan un papel esencial en la creación de una experiencia de usuario efectiva. Participan en la planificación y diseño de interfaces de usuario, colaboran con desarrolladores para garantizar la implementación adecuada de diseños, y aseguran la consistencia y usabilidad del producto final (Waseem M, Liang P, Shahin M, 2021). Su relevancia radica en mejorar la experiencia del usuario a través de un diseño efectivo, facilitar la comunicación visual entre equipos de desarrollo y operaciones, y garantizar que el diseño sea coherente con los objetivos del negocio.

2.4 Desarrolladores

Los desarrolladores desempeñan un papel fundamental en el proceso de desarrollo de software. Escriben código y desarrollan nuevas funcionalidades, colaboran en la automatización de procesos para la entrega continua y participan activamente en la integración continua, pruebas y despliegue (Theunissen T, van Heesch U & Avgeriou P, 2022). Su relevancia radica en asegurar la calidad del código, alineándolo con los requisitos del cliente. Además, contribuyen significativamente a la automatización, acelerando el tiempo de entrega, y colaboran estrechamente con otros equipos para integrar cambios de manera eficiente.

2.5 Analista de calidad (QA)

El equipo de testers desempeña un papel crucial en la garantía de la calidad del software. Realizan pruebas de unidad, integración y aceptación, implementan estrategias de prueba automatizadas, y colaboran en la identificación y corrección de errores (Rubert M, & Farias K, 2022). Su relevancia se manifiesta al asegurar la calidad del software mediante la identificación proactiva de defectos, facilitando la detección temprana de problemas a lo largo del ciclo de desarrollo y contribuyendo a la mejora continua mediante retroalimentación sobre la calidad del software.

2.6 Ingenieros de seguridad

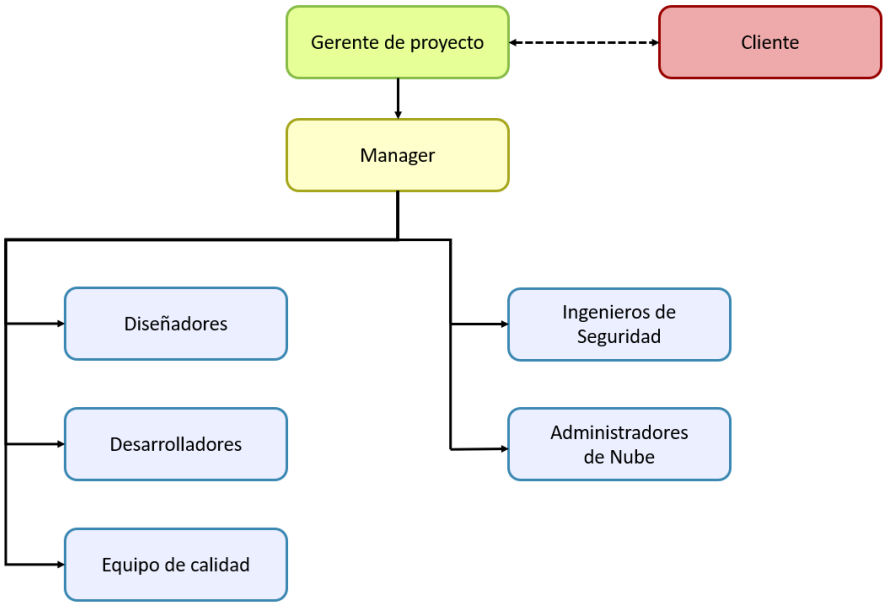
El equipo de seguridad realiza la protección de software y datos. Identifican y evalúan riesgos de seguridad, implementan medidas de seguridad en todas las etapas del desarrollo, y colaboran en la detección y respuesta a posibles amenazas. Su relevancia radica en garantizar la integridad y seguridad del software y los datos, facilitar la integración de la seguridad en el ciclo de vida del desarrollo, y contribuir a la conformidad con estándares y regulaciones.

2.7 Administradores de Nube

Los administradores de nube son fundamentales para la gestión eficiente de la infraestructura en la nube. Implementan y gestionan la infraestructura en la nube, optimizan el rendimiento y la escalabilidad de las aplicaciones, y automatizan tareas relacionadas con la infraestructura (Kumar & Goyal, 2020). Su relevancia se manifiesta al facilitar la implementación y escalabilidad de aplicaciones de manera eficiente, colaborar en la automatización de procesos para la entrega continua, y garantizar la disponibilidad y rendimiento de la infraestructura en la nube.

En la Figura 2-1 se muestran los agentes que pertenecen al ciclo de Devops y como se conectan entre sí.

Figura 2-1: Agentes del ciclo DevOps



2.8 Conclusiones

En conclusión, la metodología DevOps se presenta como un enfoque integral del desarrollo de software, que busca la integración efectiva del desarrollo y las operaciones en un ciclo de vida continuo. En este marco, diversos agentes desempeñan papeles cruciales para lograr el éxito de la implantación.

Los gerentes de proyectos desempeñan un papel coordinador y estratégico, garantizando el cumplimiento de plazos y presupuestos, y facilitando la colaboración entre equipos. Los administradores (Manager), por su parte, definen las estrategias y objetivos de negocio, asegurando la alineación con las prácticas DevOps y promoviendo la mejora continua a nivel organizativo.

Los diseñadores contribuyen a la experiencia del usuario mediante un diseño eficaz, facilitando la comunicación visual entre los equipos y garantizando la coherencia con los objetivos empresariales. Los desarrolladores, esenciales en el proceso, garantizan la calidad del código, automatizan procesos y colaboran estrechamente con otros equipos.

El equipo de probadores, o analistas de calidad, desempeña un papel crucial para garantizar la calidad del software mediante pruebas e identificación proactiva de defectos.

Los ingenieros de seguridad son esenciales para garantizar la integridad y seguridad del software y los datos, integrando la seguridad en el ciclo de desarrollo.

Por último, los administradores de la nube son clave para la gestión eficaz de la infraestructura de la nube, facilitando el despliegue y la escalabilidad de las aplicaciones de forma eficiente, colaborando en la automatización y garantizando la disponibilidad y el rendimiento de la infraestructura.

En conjunto, la colaboración y coordinación eficaz de estos agentes en el marco DevOps es esencial para el éxito en el desarrollo de software, garantizando la entrega eficiente, segura y de calidad de productos y servicios.

3. Componentes

En el dinámico y colaborativo mundo del desarrollo de software, la integración eficiente de diversos componentes y agentes es esencial para alcanzar el éxito en la entrega de productos de alta calidad (Chen, 2015). Cada componente desempeña un papel crucial en el ciclo de vida del desarrollo, contribuyendo de manera única a la consecución de los objetivos organizativos y la satisfacción del cliente. A lo largo de este proceso, diversos agentes, como Gerentes de Proyecto, Administradores, Diseñadores, Desarrolladores, Analistas de Calidad (QA), Ingenieros de Seguridad y Administradores de Nube, desempeñan funciones especializadas que abarcan desde la gestión estratégica hasta la implementación técnica.

Este entramado de agentes y componentes se ve respaldado por una variedad de herramientas tecnológicas diseñadas para optimizar y agilizar cada etapa del ciclo de vida del desarrollo de software. En este capítulo, se explora la diversidad de herramientas que estos profesionales utilizan para colaborar, diseñar, codificar, probar, asegurar y administrar la infraestructura en la nube. A medida que examinamos estas herramientas, se revela la importancia de la sinergia entre los agentes y la tecnología para lograr la entrega continua, la eficiencia operativa y la calidad del producto final en el mundo del desarrollo de software moderno.

3.1 Gerentes de Proyecto

- Trello: Para la gestión de proyectos y tareas.
- Jira: Herramienta ampliamente utilizada para la gestión ágil de proyectos.
- Asana: Para la planificación y seguimiento de proyectos.

3.2 Administradores (Manager)

- Slack: Plataforma de comunicación en equipo.
- Microsoft Teams: Herramienta de colaboración y comunicación.
- Zoom o Microsoft Teams: Para reuniones y videoconferencias.

3.3 Diseñadores

- Adobe Illustrator: Herramienta de diseño y prototipado de experiencias de usuario.
- Sketch: Aplicación de diseño vectorial para interfaces de usuario.
- Figma: Plataforma de diseño colaborativo en la nube.

3.4 Desarrolladores

- Git: Sistema de control de versiones.
- GitHub o GitLab: Plataformas para alojar y revisar el código.
- Visual Studio Code o IntelliJ IDEA: Entornos de desarrollo integrados (IDE) populares.

En el ámbito del desarrollo de software, existen diversos tipos de desarrolladores, cada uno con habilidades especializadas en áreas específicas. A continuación, se describen algunos de los tipos más comunes.

3.4.1 Desarrollador Backend

El rol del desarrollador Backend se concentra en la esencia y el funcionamiento interno de las aplicaciones, abarcando la gestión de servidores, bases de datos y aplicaciones del lado del servidor. Este profesional destaca por su habilidad en lenguajes de programación diversos, entre ellos, Java, Python, Ruby y PHP. Además, posee experiencia significativa en el desarrollo de APIs y la gestión eficiente de bases de datos. Su enfoque especializado asegura la robustez y el rendimiento óptimo de la infraestructura que respalda las aplicaciones.

3.4.2 Desarrollador Frontend

El desarrollador Frontend desempeña un papel fundamental en la creación de la interfaz de usuario y la optimización de la experiencia del usuario. Su labor incluye la traducción del diseño visual de la aplicación a código, asegurándose de que esta sea atractiva y fácil de utilizar. Este profesional destaca por su dominio de lenguajes esenciales como HTML, CSS y JavaScript. Asimismo, posee una familiaridad sólida con frameworks y bibliotecas frontend, tales como React, Angular o Vue. Gracias a estas habilidades, el desarrollador Frontend contribuye de manera crucial a la creación de interfaces intuitivas y visualmente atractivas para mejorar la interacción del usuario con la aplicación.

3.4.3 Desarrollador Full Stack

Con la capacidad de abordar tanto el desarrollo backend como frontend de una aplicación, este profesional demuestra versatilidad para asumir diversos aspectos del proceso de desarrollo, desde el inicio hasta el final. Su conocimiento abarcador en una variedad de lenguajes y tecnologías, tanto en el ámbito frontend como backend, le permite contribuir de manera integral a la construcción de aplicaciones completas. Con experiencia sólida en el desarrollo de extremo a extremo, este desarrollador es un recurso valioso para equipos que buscan una perspectiva holística y una ejecución coherente en todos los aspectos del ciclo de vida de desarrollo de software.

3.4.4 Desarrollador Mobile

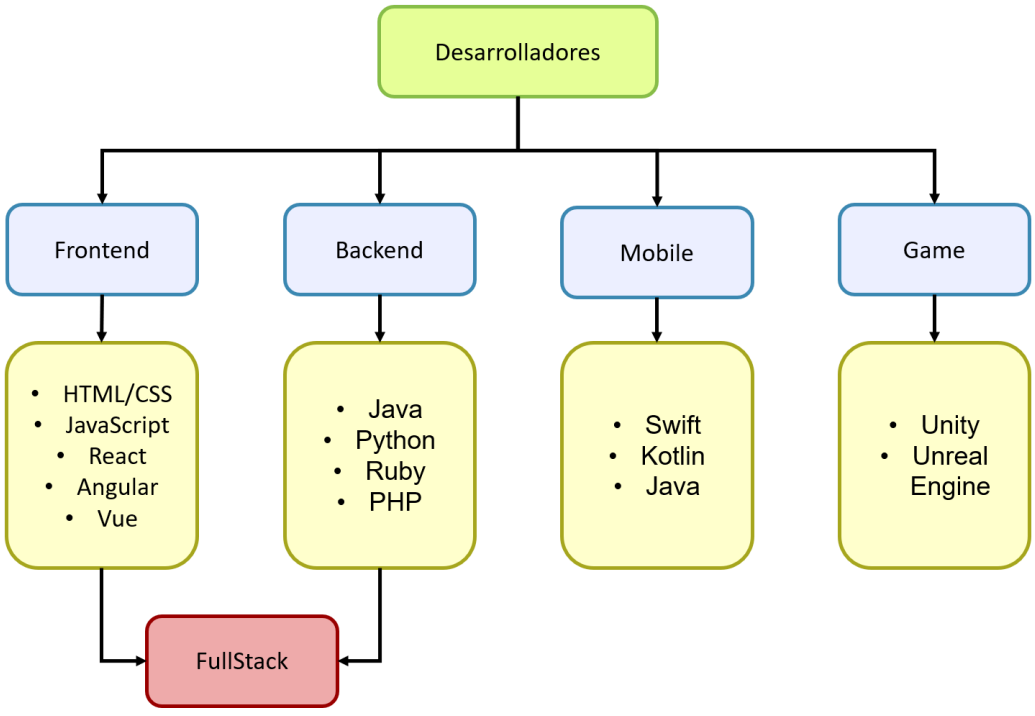
Este desarrollador se especializa en el desarrollo de aplicaciones móviles, concentrándose en plataformas líderes como iOS y Android. Sus responsabilidades incluyen la creación de soluciones adaptadas a las particularidades de estos sistemas operativos. Entre sus habilidades destacan el dominio de lenguajes específicos como Swift para iOS, y Kotlin o Java para Android. Además, demuestra familiaridad con diversos frameworks y entornos de desarrollo móvil, asegurando así la capacidad de diseñar y ejecutar aplicaciones móviles eficientes y centradas en la experiencia del usuario en ambos ecosistemas.

3.4.5 Desarrollador de Juegos (Game Developer)

Este profesional se dedica al diseño y desarrollo de juegos digitales, desempeñando un papel esencial en la creación de experiencias interactivas envolventes. Sus responsabilidades abarcan la concepción y ejecución de gráficos, la implementación de física y la definición de mecánicas de juego que garanticen la diversión y la interactividad del usuario. Además, cuenta con habilidades especializadas, incluyendo conocimientos profundos en motores de juego como Unity o Unreal Engine. Su competencia en la programación, utilizando lenguajes como C++ o C#, respalda la creación de juegos innovadores y de alta calidad que cautivan a los jugadores con su atractivo visual y experiencias de juego únicas.

La Figura 3-1 muestra los tipos de desarrolladores necesarios en las medianas empresas, este esquema muestra las principales herramientas que utilizan cada uno de los roles descritos.

Figura 3-1: Tipos de desarrolladores



3.5 Analista de calidad (QA)

- Selenium: Para automatización de pruebas de software.
- JUnit o TestNG: Frameworks de prueba para Java.
- Postman: Herramienta para probar APIs.

3.6 Ingenieros de seguridad

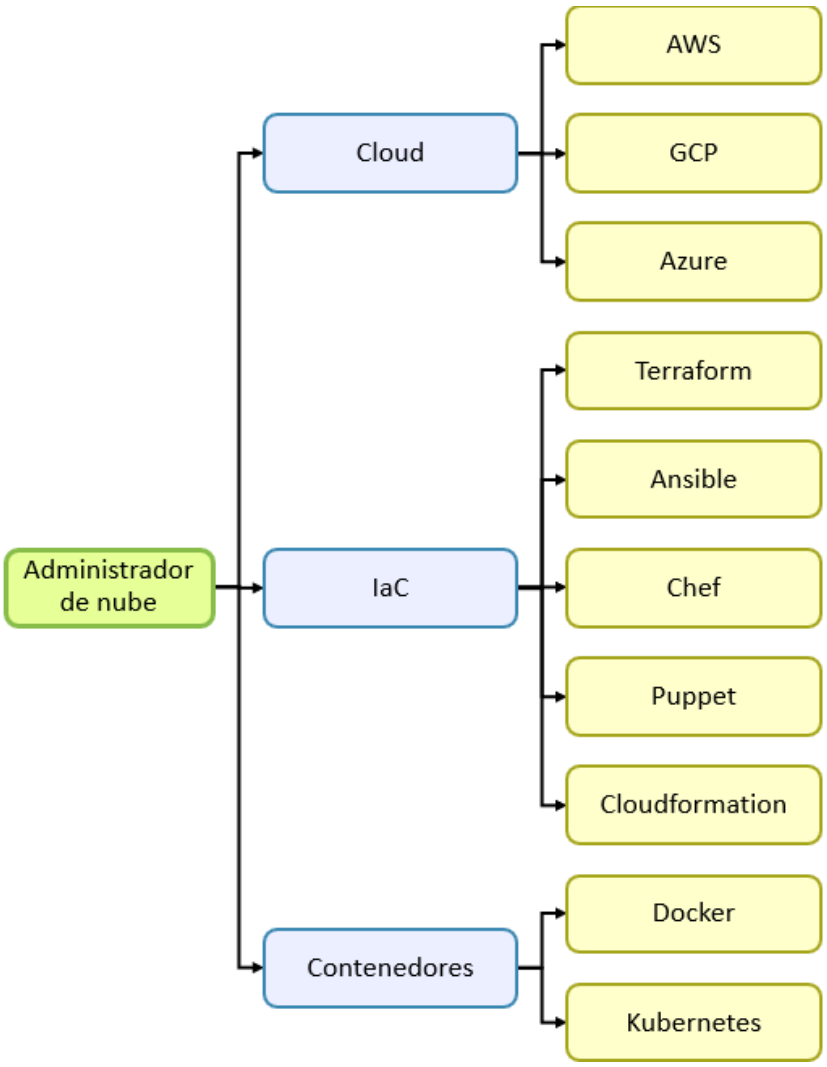
- Burp Suite: Herramienta para pruebas de seguridad en aplicaciones web.
- Wireshark: Analizador de protocolos de red.
- Nessus: Herramienta de escaneo de vulnerabilidades.

3.7 Administradores de Nube

- Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform (GCP): Plataformas de servicios en la nube.
- Terraform o Ansible: Herramientas de infraestructura como código (IaC).
- Docker y Kubernetes: Para contenerización y orquestación de aplicaciones.

La Figura 3-2 describe las funciones que realiza un administrador de nube, se dividen en tres funciones principales y se especifican las herramientas utilizadas en cada uno de estos procesos

Figura 3-2: Administradores de nube



3.8 Conclusiones

En conclusión, en el dinámico mundo del desarrollo de software, la colaboración eficiente entre varios agentes y la integración de herramientas tecnológicas especializadas son esenciales para lograr el éxito en la entrega de productos de alta calidad. Cada componente, desde los gestores de proyectos hasta los administradores de la nube, desempeña un papel crucial en el ciclo de vida del desarrollo, contribuyendo de forma única a los objetivos de la organización y a la satisfacción del cliente.

La diversidad de herramientas tecnológicas utilizadas por estos profesionales, como Trello, Jira, Slack, Git, Selenium y muchas más, refleja la complejidad y variedad de tareas a lo largo del proceso de desarrollo. La sinergia entre agentes y tecnología se hace evidente al examinar cómo se utilizan estas herramientas para colaborar, diseñar, codificar, probar, asegurar y gestionar la infraestructura de la nube.

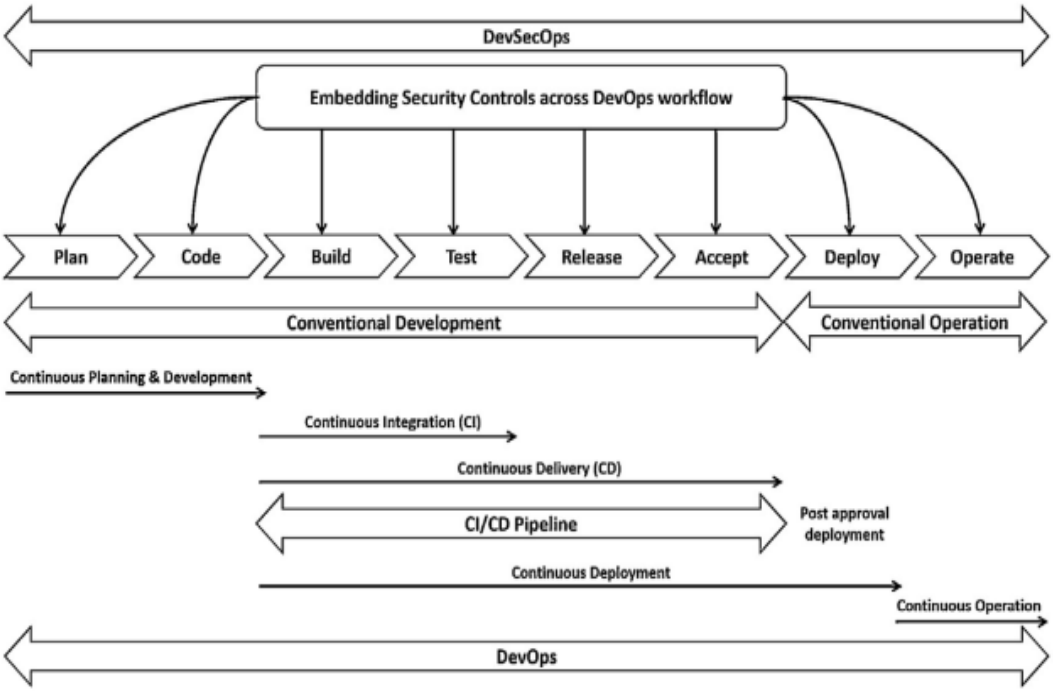
La importancia de la coordinación entre agentes y tecnologías se manifiesta en la capacidad para lograr la entrega continua, la eficiencia operativa y la calidad del producto final en el desarrollo de software moderno. La combinación de funciones especializadas y herramientas específicas demuestra cómo la cooperación entre agentes y la aplicación estratégica de la tecnología son esenciales para afrontar los retos y cumplir las normas del sector en constante evolución. En última instancia, esta interacción armoniosa impulsa la innovación, la eficiencia y la satisfacción del cliente en el competitivo entorno del desarrollo de software.

4. CI / CD

La investigación determinó que las fases de la metodología Devops son cíclicas, es decir repetitivas constantemente, iniciando en la planificación, pasando por procesos de pruebas y compilación automatizados. Por último, se realiza el despliegue de los nuevos componentes y el monitoreo de las respectivas actualizaciones, de tal manera que se puedan solventar de manera eficiente y eficaz realizando una instalación escalonada.

Kumar y Goyal, realizan un modelo estandarizado en el que se evidencian las fases del ciclo Devops incluyendo procesos de seguridad. Este proceso se observa en la figura 4-1.

Figura 4-1: Modelo del ciclo Devops con proceso de seguridad



Fuente: Kumar, Goyal, 2020

Los componentes clave en el ciclo de DevOps incluyen los siguientes procesos:

4.1 Desarrollo (Development)

El proceso de desarrollo de software es una parte crítica del ciclo de vida del desarrollo de software, y abarca desde la concepción de nuevas características o mejoras hasta su implementación. Aquí se detallan las principales actividades relacionadas con el desarrollo de software.

4.1.1 Requisitos y Diseño

Antes de comenzar el desarrollo, es esencial tener una comprensión clara de los requisitos del proyecto. Esto implica la definición de las funcionalidades que se deben desarrollar y la creación de diseños o especificaciones que guíen la implementación. Esto puede incluir diagramas, prototipos, documentación técnica y otros artefactos.

4.1.2 Codificación

La codificación es el proceso de escribir el código fuente del software. Los programadores o desarrolladores utilizan lenguajes de programación y herramientas de desarrollo para traducir los requisitos y el diseño en código ejecutable. Durante este proceso, los desarrolladores deben seguir buenas prácticas de codificación, como mantener el código limpio, utilizar nombres de variables descriptivos y aplicar patrones de diseño cuando corresponda.

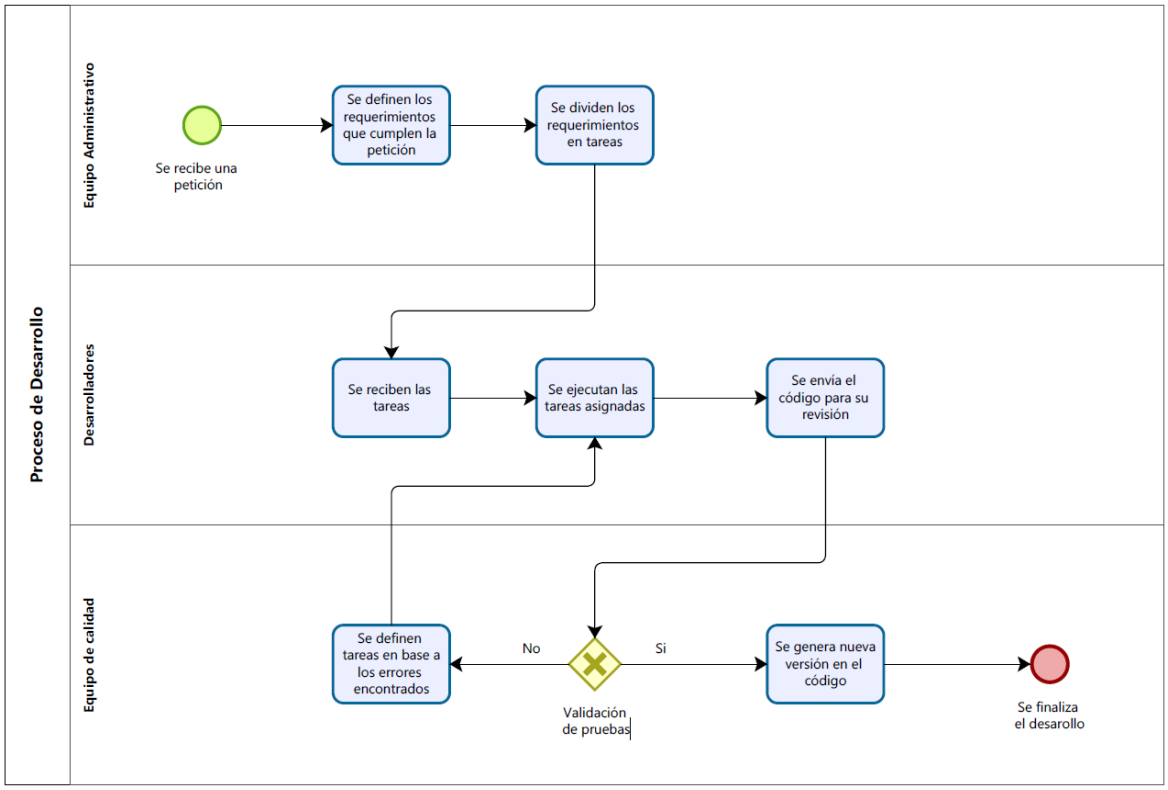
4.1.3 Integración de Código

En entornos de desarrollo colaborativo, es común que múltiples desarrolladores trabajen en diferentes partes del código. La integración de código se refiere al proceso de combinar el código de diferentes desarrolladores en una única base de código (Rajapakse, R. N., Zahedi, M., Babar, M. A, 2022). Esto puede realizarse de manera continua (integración continua) o en etapas planificadas. La integración de código puede llevarse a cabo

utilizando sistemas de control de versiones como Git y sistemas de integración continua como Jenkins.

La figura 4-2 muestra el proceso de desarrollo que se utiliza el ciclo Devops. En este esquema intervienen los desarrolladores, administradores y el equipo de calidad. Este diagrama permite analizar las diferentes funciones y como se interconectan entre sí.

Figura 4-2: Modelo del proceso de Desarrollo



4.2 Pruebas (Testing)

El proceso de pruebas en el desarrollo de software es una parte crítica para garantizar que el software sea de alta calidad, seguro y cumpla con los requisitos del usuario (Najihi, S., Elhadi, S., Abdelouahid, R. A., 2022). A continuación, se explica cada tipo de prueba en profundidad.

4.2.1 Pruebas Funcionales

Las pruebas funcionales son un tipo de prueba que se enfoca en verificar que el software cumple con las especificaciones funcionales definidas. Estas pruebas se realizan para garantizar que el software realice las funciones esperadas y se comporta de acuerdo con los requisitos del usuario. Aquí se muestran los pasos clave en el proceso de pruebas funcionales

4.2.1.1 Planificación de Pruebas Funcionales

En esta etapa, se define un plan de pruebas que identifica los casos de prueba a ejecutar, los datos de entrada, los resultados esperados y los criterios de aceptación.

4.2.1.2 Diseño de Casos de Prueba

Se crean casos de prueba específicos que evalúan diferentes aspectos del software, como la entrada de datos, la navegación, la lógica de negocio y las funcionalidades clave.

4.2.1.3 Ejecución de Pruebas

Los casos de prueba se ejecutan de acuerdo con el plan de pruebas. Cada prueba se registra y se documentan los resultados, incluyendo cualquier defecto encontrado.

4.2.1.4 Seguimiento y Corrección de Defectos

Los defectos encontrados durante las pruebas funcionales se registran, se priorizan y se asignan para su corrección. Luego se realizan pruebas de regresión para asegurarse de que las correcciones no introduzcan nuevos problemas.

4.2.1.5 Validación y Verificación

Una vez que se han corregido los defectos, se verifica y valida que el software funcione de acuerdo con los requisitos y especificaciones.

4.2.2 Pruebas de Rendimiento

Las pruebas de rendimiento se centran en evaluar cómo el software se comporta en términos de velocidad, capacidad y estabilidad bajo diferentes cargas y condiciones. Estas

pruebas son esenciales para garantizar que el software pueda manejar la demanda del mundo real sin problemas. Aquí se explican los pasos clave en el proceso de pruebas de rendimiento

4.2.2.2 Definición de Objetivos de Rendimiento

Establece objetivos claros de rendimiento, como tiempos de respuesta específicos o capacidades de procesamiento.

4.2.2.3 Creación de Escenarios de Prueba

Diseña escenarios que simulan diferentes situaciones, como cargas de usuarios simultáneos, volúmenes de datos y patrones de tráfico.

4.2.2.4 Ejecución de Pruebas de Rendimiento

Los escenarios de prueba se ejecutan utilizando herramientas de prueba de rendimiento. Esto puede incluir pruebas de carga, estrés, escalabilidad y resistencia.

4.2.2.5 Análisis de Resultados

Se analizan los resultados para identificar cuellos de botella, tiempos de respuesta lentos o problemas de rendimiento.

4.2.2.6 Optimización y Ajustes

Si se encuentran problemas de rendimiento, se realizan ajustes en el código o la infraestructura para mejorar el rendimiento.

4.2.2.7 Validación de Cumplimiento

Se verifica que el software cumpla con los objetivos de rendimiento establecidos en la etapa de definición de objetivos.

4.2.3 Pruebas de Seguridad

Las pruebas de seguridad se centran en identificar vulnerabilidades y garantizar que el software sea resistente a amenazas y ataques cibernéticos. Estas pruebas son

fundamentales para proteger la integridad, confidencialidad y disponibilidad de los datos. Aquí se detallan los pasos en el proceso de pruebas de seguridad:

4.2.3.1 Identificación de Amenazas Potenciales

Comprende las amenazas que podrían afectar al software y sus datos, como ataques de inyección SQL, cross-site scripting (XSS), ataques de denegación de servicio, entre otros.

4.2.3.2 Diseño de Casos de Prueba de Seguridad

Crea casos de prueba que simulan diferentes tipos de ataques y vulnerabilidades conocidas.

4.2.3.3 Ejecución de Pruebas de Seguridad

Los casos de prueba de seguridad se ejecutan para identificar vulnerabilidades y problemas de seguridad.

4.2.3.4 Análisis de Resultados

Se analizan los resultados para identificar vulnerabilidades y debilidades en el software.

4.2.3.5 Corrección de Vulnerabilidades

Las vulnerabilidades identificadas se corrigen y se realizan pruebas de validación de seguridad para asegurarse de que se hayan abordado correctamente.

4.2.3.6 Pruebas de Penetración

En algunos casos, se pueden realizar pruebas de penetración más avanzadas para simular ataques reales y evaluar la resistencia del software a intrusiones.

4.2.3.7 Informe de Seguridad

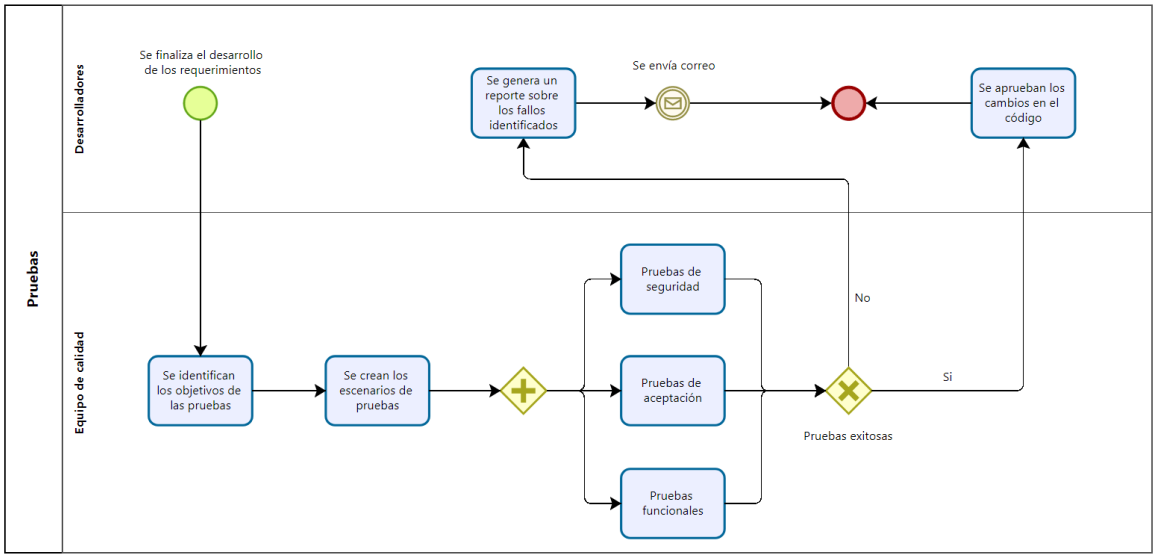
Se genera un informe de seguridad que documenta las vulnerabilidades encontradas y las medidas tomadas para corregirlas.

4.2.3.8 Cumplimiento de Normativas

Si el software debe cumplir con regulaciones específicas (por ejemplo, GDPR, HIPAA), se verifica el cumplimiento durante las pruebas de seguridad.

La figura 4-3 muestra el proceso de pruebas que se utiliza el ciclo Devops. En este esquema intervienen los desarrolladores y el equipo de calidad. Este diagrama permite analizar las diferentes funciones y como se interconectan entre sí.

Figura 4-3: Modelo del proceso de Pruebas



4.3 Integración Continua (Continuous Integration, CI)

La integración continua (CI) es una práctica clave en DevOps que se centra en automatizar la integración de cambios de código en un repositorio compartido de forma continua y frecuente, lo que permite detectar y solucionar problemas de manera temprana. A continuación, se detalla el proceso:

4.3.1 Gestión de Repositorio

En la fase inicial, el código fuente del proyecto se almacena en un sistema de control de versiones, como Git. Cada desarrollador contribuye al repositorio utilizando ramas de código separadas para nuevas funcionalidades o correcciones de errores.

4.3.2 Ejecución de Pruebas Automatizadas

4.3.2.1 Pruebas Unitarias

Cuando un desarrollador envía un cambio al repositorio, las pruebas unitarias se ejecutan automáticamente en un entorno aislado para verificar si el nuevo código funciona correctamente. Las pruebas unitarias se centran en comprobar funciones, métodos o clases individuales.

4.3.2.1 Pruebas de Integración

Además de las pruebas unitarias, se pueden ejecutar pruebas de integración automatizadas para evaluar cómo interactúan las diferentes partes del sistema cuando se combinan. Esto ayuda a identificar problemas causados por la integración de código.

4.3.2.1 Pruebas de Aceptación Automatizadas

Estas pruebas se centran en la funcionalidad global del software y en cómo se comporta en situaciones del mundo real. Comprueban si el sistema cumple con los criterios de aceptación definidos.

4.3.3 Generación de Informes

Después de que las pruebas se ejecutan, se generan informes que indican si las pruebas han pasado o fallado. Estos informes proporcionan detalles sobre los errores encontrados.

4.3.4 Notificación de Resultados

Se configuran notificaciones automatizadas para informar a los miembros del equipo de desarrollo sobre los resultados de las pruebas. Esto puede incluir notificaciones por correo electrónico, mensajes en sistemas de chat, o integración con herramientas de gestión de proyectos.

4.3.5 Despliegue Continuo

Si todas las pruebas pasan con éxito y se alcanzan los criterios de calidad definidos, el código se considera apto para el despliegue. En algunos casos, se implementa automáticamente en un entorno de producción o preproducción. Esto se conoce como despliegue continuo (CD).

4.3.6 Retroalimentación y Corrección

Si las pruebas fallan o se identifican problemas, se notifica al equipo de desarrollo, que debe abordarlos de inmediato. Este proceso de retroalimentación y corrección es esencial para mantener la calidad del software y garantizar que los problemas se solucionen antes de llegar a producción.

4.3.7 Registro y Auditoría

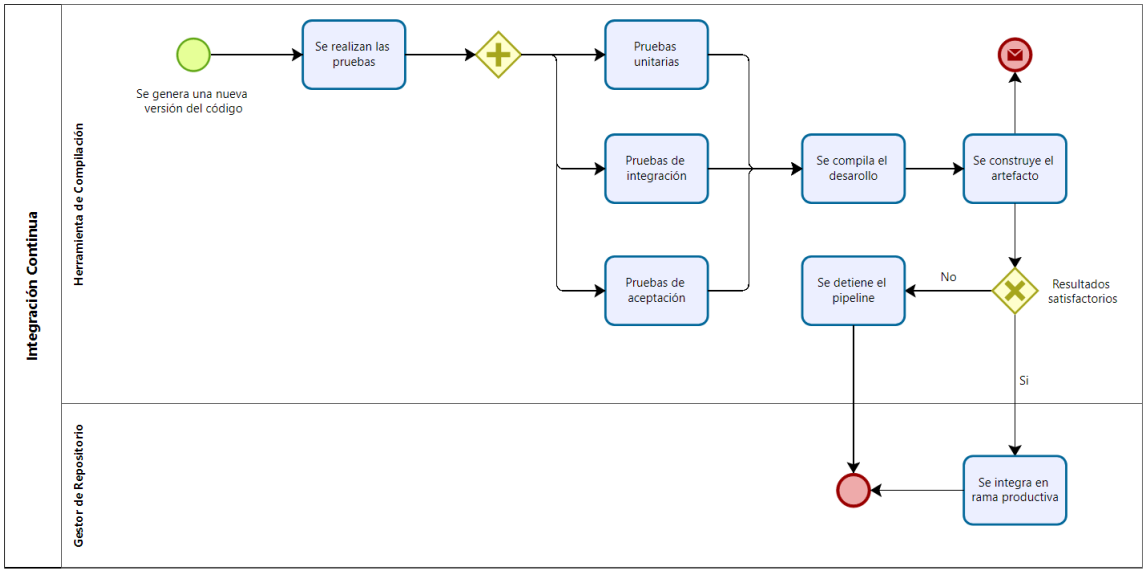
Se realiza un seguimiento de todos los cambios realizados en el sistema, incluyendo quién realizó el cambio, qué cambios se realizaron y cuándo se realizaron. Esto es importante para fines de auditoría y para entender el historial del código.

4.3.8 Ciclo Continuo

El proceso de CI es continuo y se repite cada vez que se envía un cambio al repositorio. Esto garantiza que el código se integre y pruebe regularmente, lo que ayuda a detectar y resolver problemas tempranamente y mantener la calidad del software a lo largo del tiempo.

La figura 4-4 muestra el proceso de Integración continua que se utiliza el ciclo Devops. En este esquema intervienen los gestores de repositorio y las herramientas de compilación escogida para cada proyecto. Este diagrama permite analizar las diferentes funciones y como se interconectan entre sí.

Figura 4-4: Modelo del proceso de Integración Continua



4.4 Entrega Continua (Continuous Delivery, CD)

La entrega continua (CD, por sus siglas en inglés, Continuous Delivery) es una práctica fundamental en DevOps que se enfoca en la automatización de la entrega de software a entornos de prueba y producción de manera frecuente y confiable.

4.4.1 Desarrollo y Pruebas Iniciales

El proceso comienza con el desarrollo de nuevas características o correcciones de errores. Estos cambios se desarrollan en un entorno de desarrollo local o en ramas separadas del repositorio de código fuente.

Se ejecutan pruebas unitarias y de integración automáticamente para validar que las nuevas funcionalidades funcionen correctamente y no afecten negativamente a las funcionalidades existentes.

4.4.2 Control de Versiones y Repositorio

Una vez que los cambios han pasado las pruebas iniciales, se fusionan en la rama principal del repositorio de código (por ejemplo, la rama "main" o "master" en Git).

El control de versiones asegura que cada cambio esté registrado y rastreado, lo que facilita la auditoría y la reversión de cambios si es necesario.

4.4.3 Automatización de la Construcción

La construcción del software se automatiza utilizando herramientas de compilación (por ejemplo, Jenkins, Travis CI, CircleCI). Esto incluye la compilación del código, la generación de artefactos (como archivos binarios o contenedores) y la creación de paquetes de distribución.

4.4.4 Pruebas Automatizadas Adicionales

Luego de la construcción, se ejecutan pruebas adicionales, como pruebas de aceptación automatizadas, pruebas de rendimiento y pruebas de seguridad. Estas pruebas validan que el software funcione correctamente y cumpla con los criterios de calidad y seguridad.

4.4.5 Despliegue en Entorno de Prueba (Staging)

Una vez que el software ha superado con éxito las pruebas automatizadas, se despliega automáticamente en un entorno de prueba (también conocido como entorno de preproducción o staging). Este entorno es una réplica del entorno de producción y se utiliza para realizar pruebas finales antes de la implementación en producción.

4.4.6 Pruebas Manuales y Validación

En el entorno de prueba, se pueden realizar pruebas manuales adicionales por parte del equipo de control de calidad (QA) y otros stakeholders. Esto incluye pruebas de usuario final y validación de que el software cumple con los requisitos y expectativas.

4.4.7 Aprobación de la Versión

Después de las pruebas y validación en el entorno de prueba, la versión se somete a una revisión y aprobación. Las partes interesadas, como el equipo de desarrollo, el equipo de operaciones y el equipo de gestión, pueden dar su aprobación para la implementación en producción.

4.4.8 Despliegue en Producción

Si la versión recibe la aprobación necesaria, se implementa automáticamente en el entorno de producción. Este proceso suele ser automatizado y garantiza que la versión desplegada en producción sea idéntica a la probada en el entorno de prueba.

4.4.9 Monitorización y Retroalimentación

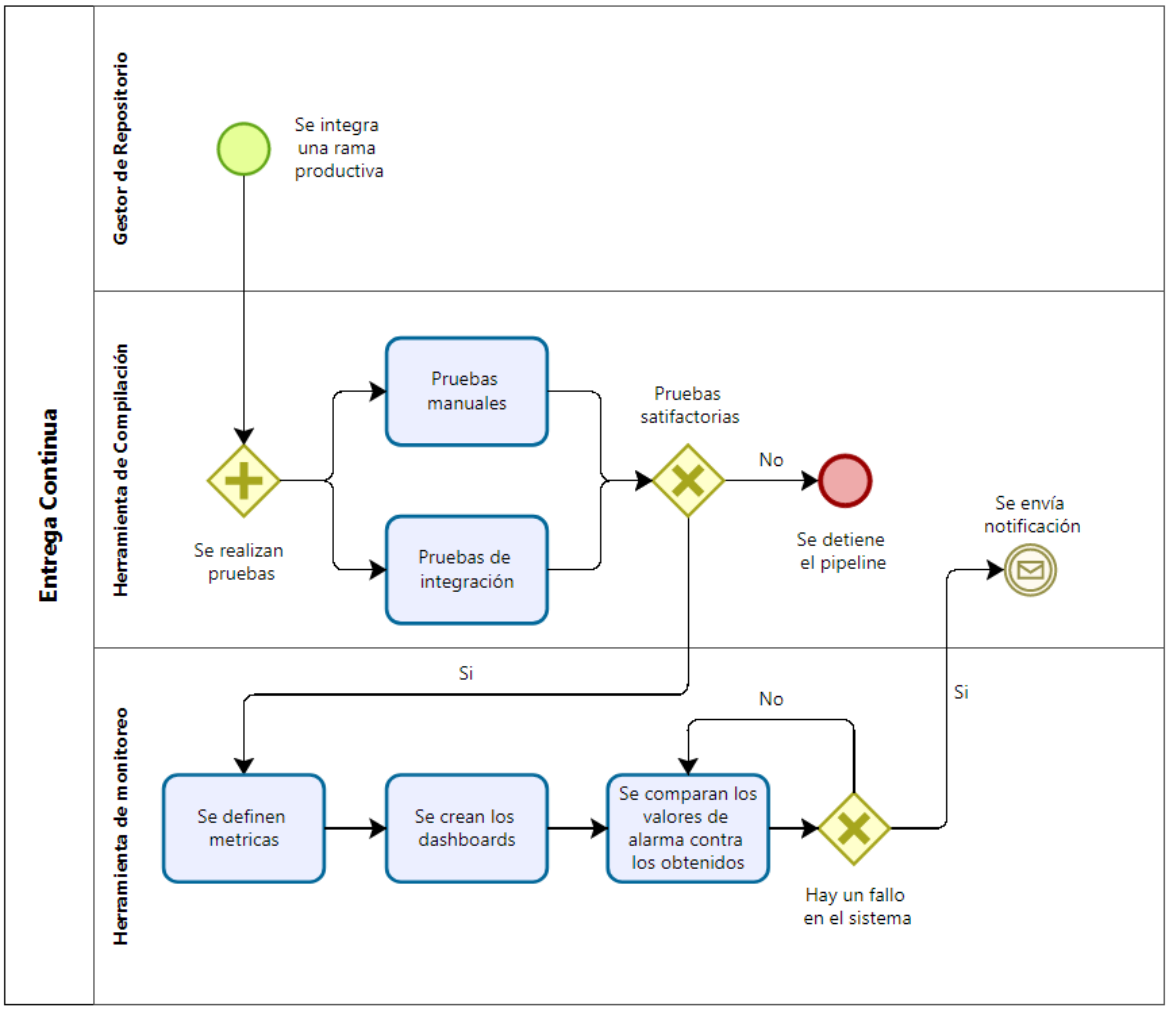
Después de la implementación en producción, se realiza un monitoreo constante para detectar problemas o degradación del rendimiento. Cualquier problema identificado se aborda inmediatamente, lo que puede incluir la reversión a la versión anterior si es necesario.

4.4.10 Ciclo Continuo

La entrega continua es un proceso continuo que se repite cada vez que se realiza un cambio en el código. Esto garantiza que el software se mantenga actualizado, confiable y seguro a lo largo del tiempo.

La figura 4-5 muestra el proceso de entrega continua que se utiliza el ciclo Devops. En este esquema intervienen los gestores de repositorio, las herramientas de compilación y las herramientas de monitoreo escogidas para cada proyecto. Este diagrama permite analizar las diferentes funciones y como se interconectan entre sí.

Figura 4-5: Modelo del proceso de Entrega Continua



4.5 Automatización (Automation)

La automatización es un pilar fundamental de DevOps. Implica la automatización de tareas repetitivas, como la compilación, pruebas, despliegue y monitoreo, lo que acelera el proceso y reduce errores humanos.

4.6 Infraestructura como Código (Infrastructure as Code)

El proceso de Infraestructura como Código (IaC) es una práctica fundamental en DevOps que se centra en la gestión y provisión automatizada de servidores y recursos de infraestructura utilizando código. Esta práctica permite tratar la infraestructura de la misma

manera que se trata el código de aplicación, lo que resulta en una mayor agilidad, consistencia y escalabilidad en el desarrollo y despliegue de aplicaciones (Rahman, Mahdavi-Hezaveh, 2019) A continuación, se detalla el proceso de IaC.

4.6.1 Definición de la Infraestructura como Código

El proceso comienza con la definición de la infraestructura requerida en forma de código. Este código puede estar escrito en lenguajes específicos de IaC, como HashiCorp Configuration Language (HCL) para Terraform o YAML para herramientas como Ansible y Kubernetes.

4.6.2 Repositorio de Código de Infraestructura

El código de infraestructura se almacena en un repositorio de código fuente, como un repositorio Git. Esto asegura que el código de infraestructura sea versionado y gestionado de la misma manera que el código de aplicación.

4.6.3 Automatización del Provisionamiento

Herramientas de IaC, como Terraform, Ansible, Puppet o Chef, se utilizan para automatizar el provisionamiento de recursos de infraestructura. Estas herramientas permiten describir los recursos necesarios, como servidores, redes, bases de datos y servicios, y desplegarlos de manera automática y consistente.

4.6.4 Pruebas de Infraestructura

Antes de desplegar la infraestructura en entornos de producción o prueba, se pueden ejecutar pruebas de infraestructura automatizadas para validar que la configuración definida en el código de infraestructura sea correcta y cumpla con los requisitos. Esto incluye verificar que los servidores se estén configurando adecuadamente y que las redes estén correctamente definidas.

4.6.5 Despliegue de Infraestructura en Entornos de Prueba

Una vez que las pruebas de infraestructura han pasado con éxito, la infraestructura definida en código se despliega en entornos de prueba o preproducción. Esto se realiza de manera automática y controlada.

4.6.6 Gestión de Configuración

Además de la creación inicial de la infraestructura, las herramientas de IaC pueden utilizarse para gestionar cambios de configuración a lo largo del tiempo. Esto garantiza que la infraestructura siga siendo coherente y cumpla con los requisitos a medida que evoluciona el software.

4.6.7 Escalabilidad y Alta Disponibilidad

La IaC facilita la escalabilidad y alta disponibilidad mediante la definición de políticas y reglas en el código de infraestructura. Los recursos pueden escalar automáticamente en respuesta a aumentos de demanda o fallas, garantizando la continuidad del servicio.

4.6.8 Monitorización y Retroalimentación

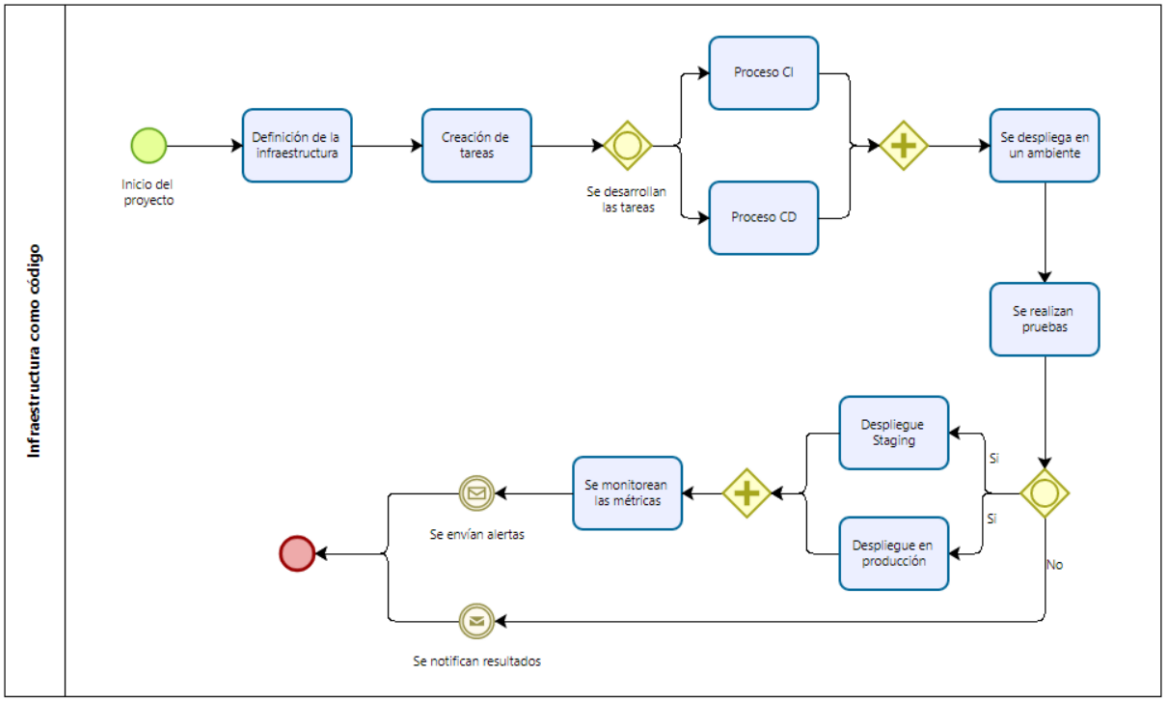
La infraestructura desplegada se monitorea continuamente para detectar problemas o desviaciones de rendimiento. Cualquier problema identificado se aborda automáticamente o se informa para su resolución.

4.6.9 Ciclo Continuo

El proceso de IaC es continuo y se repite cada vez que se realice un cambio en el código de infraestructura o en los requisitos. Esto garantiza que la infraestructura se mantenga actualizada y cumpla con las necesidades en constante cambio del software.

La figura 4-6 muestra el proceso de creación de la infraestructura como código que se utiliza en el ciclo Devops. En este esquema intervienen los administradores de nubes. Este diagrama permite analizar las diferentes funciones y como se interconectan entre sí.

Figura 4-6: Modelo del proceso de Infraestructura como código



4.7 Monitoreo y Retroalimentación (Monitoring and Feedback)

La supervisión constante es crucial para la identificación temprana de problemas en producción. Los datos y la retroalimentación se utilizan para realizar mejoras continuas en el software y los procesos.

4.8 Colaboración (Collaboration)

La comunicación efectiva y la colaboración entre los equipos de desarrollo y operaciones son esenciales en DevOps. La cultura de colaboración fomenta la resolución de problemas de manera eficiente y la toma de decisiones informadas.

Estos componentes trabajan en conjunto para lograr los objetivos de DevOps, que incluyen la entrega continua, la mejora de la calidad del software, la reducción de tiempos de entrega y la capacidad de respuesta a las cambiantes demandas del mercado.

4.9 Conclusiones

En conclusión, el desarrollo de software es un proceso complejo que abarca desde la concepción de nuevas funcionalidades hasta su implementación, siendo esencial en el ciclo de vida del desarrollo de software. La fase de desarrollo implica la definición clara de requisitos y diseños, la codificación del software y la integración eficaz del código, con la colaboración de desarrolladores, administradores y equipos de calidad.

Las pruebas garantizan la calidad del software. Las pruebas funcionales se centran en verificar el cumplimiento de las especificaciones, las pruebas de rendimiento evalúan el comportamiento bajo diferentes cargas y las pruebas de seguridad identifican vulnerabilidades y garantizan la resistencia a las amenazas. La automatización de pruebas, la entrega continua y la infraestructura como código aceleran el proceso, reducen los errores y garantizan la coherencia en el desarrollo y la implantación de aplicaciones.

La integración continua (CI) se centra en la automatización de la integración del código y las pruebas, proporcionando información rápida y detectando los problemas en una fase temprana. La entrega continua (CD) se centra en la automatización de la entrega de software a entornos de prueba y producción, garantizando la coherencia y fiabilidad de la implantación.

La automatización es un pilar fundamental de DevOps, que abarca tareas repetitivas como la compilación, las pruebas, el despliegue y la supervisión. La infraestructura como código (IaC) permite la gestión automatizada y eficiente de la infraestructura, proporcionando agilidad, coherencia y escalabilidad.

En resumen, la combinación de desarrollo, pruebas, automatización y prácticas como CI, CD e IaC forman un marco sólido que impulsa la eficiencia, la calidad y la velocidad en el desarrollo de software en el contexto de la metodología DevOps. Este enfoque integral permite afrontar los retos del desarrollo de software moderno y adaptarse a las demandas cambiantes del sector.

5. Conclusiones y recomendaciones

5.1 Respuesta a la pregunta de investigación

Para concluir, la investigación realizada ha demostrado la viabilidad de desarrollar una metodología integral para implantar DevOps en el sector de las tecnologías de la información. Esta metodología está diseñada para facilitar la integración aprovechando las ventajas de los equipos de desarrollo y operaciones, superando las limitaciones inherentes a los procesos individuales dentro del ciclo de desarrollo de software. La metodología propuesta proporciona un enfoque sistemático que no sólo aborde los retos planteados por la convergencia del desarrollo y las operaciones, sino que también gestione las complejidades asociadas a la adopción de nuevas tecnologías.

Al vincular equipos tradicionalmente aislados, la metodología DevOps pretende fomentar la colaboración, la comunicación y la eficiencia a lo largo del ciclo de vida del desarrollo de software. Reconoce la importancia de armonizar los esfuerzos de los desarrolladores, que contribuyen a crear nuevas funcionalidades, y los equipos de operaciones, responsables de garantizar la estabilidad y fiabilidad del software en producción.

Por otro lado, la metodología diseñada pretende ser adaptable y responder a la naturaleza dinámica del panorama de las tecnologías de la información. Reconociendo la necesidad de afrontar y superar los obstáculos que suelen acompañar a la implantación de tecnologías emergentes. A través de un enfoque sistemático, la metodología propuesta se esfuerza por crear un marco que no sólo mejore la colaboración entre el desarrollo y las operaciones, sino que también agilice el proceso general de desarrollo de software.

En esencia, esta investigación contribuye al campo en evolución de DevOps ofreciendo una metodología práctica y estructurada que las organizaciones del sector de las tecnologías de la información pueden emplear para integrar y optimizar sus procesos de desarrollo y operaciones. Los resultados ponen en evidencia el potencial para mejorar la eficiencia, reducir los errores y fomentar la innovación en el desarrollo de software, contribuyendo en última instancia a la implementación exitosa de las prácticas DevOps en la industria.

5.2 Cumplimiento de objetivos

5.2.1 Objetivo general

La investigación realizada ha abordado con éxito el objetivo general de proponer una metodología integral de desarrollo de software basada en operaciones. A través de una exploración sistemática de varias facetas dentro del desarrollo y las operaciones de software, la metodología presentada en esta tesis ofrece un enfoque consolidado para mejorar la eficiencia, la colaboración y la eficacia general del ciclo de vida del desarrollo de software.

5.2.2 Objetivos específicos

Esta investigación ha alcanzado el cumplimiento de sus objetivos específicos al abordar de forma exhaustiva los componentes claves, los agentes, los procesos y la estrategia de implementación de una metodología de desarrollo de software basada en operaciones.

El primer objetivo se centraba en caracterizar los componentes necesarios para dicha metodología. A través de un análisis en profundidad del ciclo de vida del desarrollo de software, las consideraciones operativas y los avances tecnológicos, la investigación identificó y delineó los componentes críticos que forman la base de una metodología eficaz de desarrollo de software basada en operaciones. Esta caracterización proporciona una clara comprensión de los elementos esenciales que contribuyen al éxito de la metodología propuesta.

El segundo objetivo pretendía determinar los agentes que intervienen en una metodología de desarrollo de software basada en operaciones. La investigación profundizó en las funciones y responsabilidades de las distintas partes interesadas, incluidos desarrolladores, personal de operaciones, gestores y otros agentes relevantes. Al reconocer la importancia de la colaboración y la comunicación entre estos agentes, esta investigación evidenció la necesidad de un esfuerzo cohesivo y sinérgico para implantar y mantener con éxito la metodología propuesta.

El tercer objetivo esperaba especificar los procesos integrantes de una metodología de desarrollo de software basada en operaciones. Al examinar cada fase del ciclo de vida del desarrollo de software, desde la recopilación de requisitos hasta la implantación y el mantenimiento, la tesis esbozó procesos específicos que contribuyen a la integración sin fisuras del desarrollo y las operaciones. Esta especificación proporciona una guía práctica para las organizaciones que deseen implantar la metodología, garantizando un enfoque sistemático y eficiente del desarrollo de software.

El cuarto y último objetivo se centraba en definir una estrategia para implantar la metodología en una empresa de desarrollo de software. A partir de los conocimientos adquiridos mediante la caracterización de los componentes, la determinación de los agentes y la especificación de los procesos, la tesis propuso una estrategia integral que tiene en cuenta la cultura organizativa, la gestión del cambio y la mejora continua. Esta estrategia ofrece una hoja de ruta para las empresas que deseen realizar la transición hacia un enfoque de desarrollo de software basado en operaciones.

Bibliografía

[1] Battina, D. S. (2020). Devops, A New Approach To Cloud Development & Testing. International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN, 2349-5162.

[2] Bizagi. (2022). documentación de modelamiento de infrastructura. Obtenido de Bizagi, One Platform; Every Process

[3] BPMN. (2022) Obtenido de BPMN Specification - Business Process Model and Notation

[4] Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia, Valencia, 1-8.

[5] Díaz-de-Arcaya, J., Torre-Bastida, A. I., Miñón, R., & Almeida, A. (2023). Orfeon: An AIOps framework for the goal-driven operationalization of distributed analytical pipelines. Future Generation Computer Systems, 140, 18-35.

[6] Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. Ieee Software, 33(3), 94-100.

[7] Elsevier. (12 de 10 de 2021). Palabras clave como representación del conocimiento. Obtenido de https://www.youtube.com/watch?v=x1ju1m8VrQI&t=5408s

[8] Erich, F., Amrit, C., & Daneva, M. (2014). Report: Devops literature review. University of Twente, Tech. Rep.

[9] Fowler, M., & Highsmith, J. (2001). The agile manifesto. Software development, 9(8), 28-35.

[10] Gil, A. C. (2002). How to design research projects. São Paulo, 5(61), 16-17.

[11] González, A. (2021). Cleverdata - Concertos básicos de machine learning. Obtenido de https://cleverdata.io/conceptos-basicos-machine-learning/

[12] Hernández Bejarano, M., & Baquero Rey, L. E. (2020). Ciclo de vida de desarrollo ágil de software seguro. Editorial Los Libertadores.

[13] IEEE std 610.12-1990. (1990). IEEE Standard Glossary of Software Engineering Terminology.

[14] Kersten, M. (2018). A cambrian explosion of DevOps tools. IEEE Software, 35(02), 14-17.

[15] Kitchenham, B., Brereton, B., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering – A systematic literature review. Information and Software Technology 51, 7-15.

[16] Kitchenham, B., Dyba, T., & Jorgensen, M. (2004). Evidence-based software engineering, in: Proceedings of the 26th International Conference on Software Engineering (ICSE'04). IEEE Computer Society, Washington DC, USA, 273–281.

[17] Kumar, R., & Goyal, R. (2020). Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). Computers & Security, 97, 101967.

[18] Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. ACM Computing Surveys (CSUR), 52(6), 1-35.

[19] Lewerentz, M., Bluhm, T., Daher, R., Dumke, S., Grahl, M., Grün, M., ... & Werner, A. (2019). Implementing DevOps practices at the control and data acquisition system of an experimental fusion device. Fusion Engineering and Design, 146, 40-45.

[20] LIFE.ART.TECH. (2017). lifeartech.wordpress.com. Obtenido de https://lifeartech.wordpress.com/2017/08/16/isoiecieee-42010-descripcion-de-arquitectura-intro/

[21] Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. Journal of Systems and Software, 157, 110384.

[22] Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., ... & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217-230.

[23] Mendes Calo, K., Estévez, E. C., & Fillottrani, P. R. (2010). Evaluación de metodologías ágiles para desarrollo de software. In XII Workshop de Investigadores en Ciencias de la Computación.

[24] MINTIC. (2021). Arquitectura TI Colombia. Obtenido de https://mintic.gov.co/arquitecturati/630/w3-article-8736.html

[25] Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100308.

[26] Najihi, S., Elhadi, S., Abdelouahid, R. A., & Marzak, A. (2022). Software Testing from an Agile and Traditional view. Procedia Computer Science, 203, 775-782.

[27] Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., Rueda, J. R., & Pantano, J. C. (2017, September). Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

[28] Plant, O. H., van Hillegersberg, J., & Aldea, A. (2022). Rethinking IT governance: Designing a framework for mitigating risk and fostering internal control in a DevOps environment. International Journal of Accounting Information Systems, 45, 100560.

[29] Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. Information and Software Technology, 108, 65-77.

[30] Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. Information and software technology, 141, 106700.

[31] Report, Computer Science Department. Keele University (TR/SE-0401) and National ICT Australia Ltd. (0400011T.1).

[32] Rubert, M., & Farias, K. (2022). On the effects of continuous delivery on code quality: A case study in industry. Computer Standards & Interfaces, 81, 103588.

[33] Rzig, D. E., Hassan, F., & Kessentini, M. (2022). An empirical study on ML DevOps adoption trends, efforts, and benefits analysis. Information and Software Technology, 152, 107037.

[34] Smith, S. (2013). Digital signal processing: a practical guide for engineers and scientists. Elsevier.

[35] Sun, Z., Li, Y., & Wen, L. (2022). DevOps and Neural Network Based Full Lifecycle Management Model for Power Information Systems. Procedia Computer Science, 208, 642-649.

[36] Taylor, R., Medvidovic, N., & Dashofy, E. (2009). Software Architecture: Foundations, Theory, and Practice. Wiley Publishing.

[37] Theunissen, T., van Heesch, U., & Avgeriou, P. (2022). A mapping study on documentation in Continuous Software Development. Information and software technology, 142, 106733.

[38] Truong, H. L., & Klein, P. (2020). Devops contract for assuring execution of iot microservices in the edge. Internet of Things, 9, 100150.

[39] Uribe, E. H., & Ayala, L. E. V. (2007). Del manifiesto ágil sus valores y principios. Scientia et technica, 13(34), 381-386.

[40] Waseem, M., Liang, P., Shahin, M., Di Salle, A., & Márquez, G. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. Journal of Systems and Software, 182, 111061.

[41] Wen, L., Qian, H., & Liu, W. (2022). Research on Intelligent Cloud Native Architecture and Key Technologies Based on DevOps Concept. Procedia Computer Science, 208, 590-597.

[42] Yang, D., Wang, D., Yang, D., Dong, Q., Wang, Y., Zhou, H., & Hong, D. (2020). DevOps in practice for education management information system at ECNU. Procedia Computer Science, 176, 1382-1391.

[43] Zhou, X., Li, S., Cao, L., Zhang, H., Jia, Z., Zhong, C., ... & Babar, M. A. (2023). Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry. Journal of Systems and Software, 195, 111521.