Automatización del Despliegue y Configuración de Infraestructura para aplicaciones Web utilizando Entornos de Computación en la Nube

Monografía para optar al título de Ingeniera de Sistemas
Modalidad Práctica Profesional

María Fernanda Jaramillo Trullo

Director: Esp. Pablo Augusto Magé Imbachí
Asesor de la empresa: Ing. Rafael Esteban Cerón Espinosa

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Ingeniería de Sistemas
Popayán, agosto de 2023

Contenido

Lista de figuras

Lista de tablas

Listado de acrónimos

AC	Adquisición y Configuración.
API	*Application Programming Interface* (Interfaz de programación de aplicaciones)
AWS	Amazon Web Services
BPMN	*Bussiness Process Model and Notation* (Modelo y notación de procesos de negocio)
CD	*Continuous Develivery* (Entrega continua).
CDN	*Content Delivery Network* (Red de distribución de contenidos).
CI	*Continuous Integrations* (Integración continua)
CLI	*Command Line Interface* (Interfaz de línea de comandos)
DEV	Development (Desarrollo)
EC2	*Elastic Cloud Computing*
FaaS	*Function as a Service* (Funcion como servicio)
GCP	Google Cloud Platform
HTTPS	*Hypertext Transfer Protocol Secure* (Protocolo seguro de transferencia de hipertexto)
IaC	*Infrastructure as Code* (Infraestructura como código).
IAM	*Identity and Access Management (Gestión de identidad y acceso)*
PROD	*Production* (Producción)
SSL	*Secure Sockets Layer* (Capa de sockets seguro)
SSH	*Secure Shell*
SQS	*Simple Queue Service* (Servicio de cola simple)
UX	*User Experience* (Experiencia de usuario)
QA	*Quality Assurance (Aseguramiento de la calidad)*.
WAF	*Web Application Firewall* (Firewall de aplicaciones web)

Capítulo 1 - Introducción

Actualmente, la industria de desarrollo de software y tecnología se ha convertido en una de las industrias más importantes dentro del entorno empresarial dada la necesidad de automatizar los procesos y aumentar la competitividad en un mundo que demanda la entrega de productos y servicios de calidad de manera más rápida y eficiente. En consecuencia, los procesos de desarrollo de software han debido ajustarse a esta necesidad de evolución continua para poder cumplir con los requerimientos tanto funcionales como no funcionales de los diversos clientes [1].

Diversas metodologías de desarrollo de software se han propuesto a lo largo de los años siendo cada vez más populares los marcos de trabajo ágiles, los cuales permiten involucrar de manera más activa a los clientes en el proceso de desarrollo y entregar valor más frecuentemente [2] esto como reemplazo a las metodologías tradicionales donde los clientes debían esperar por meses o incluso años para recibir actualizaciones y nuevas entregas [3]. Dentro de esta transición hacia el desarrollo ágil y con la introducción de nuevas herramientas y tecnologías, la industria está dando un paso hacia un paradigma DevOps. Grandes organizaciones como Google, Netflix, Amazon, LinkedIn, Spotify, entre otras, han adoptado prácticas DevOps para liberar software más rápido y de mejor calidad [4].

DevOps integra los mundos de desarrollo y operaciones para facilitar la colaboración efectiva entre equipos y mediante el uso de herramientas de automatización de despliegue y monitorización de infraestructura permite la entrega rápida y continua de nuevas versiones de software [5]. Entre los beneficios que ofrece una adopción efectiva de DevOps, se involucran factores como el incremento de la productividad de las organizaciones llegando a alcanzar hasta 30 veces una mayor tasa de despliegues en ambientes productivos [6], la minimización de los costos de producción, el manejo de la infraestructura como código que permite construir sistemas replicables y desechables, la mejora del desempeño del software y la experiencia de usuario y el aumento de la calidad, eficiencia y estabilidad [7].

Sin embargo, a pesar de la popularidad creciente y los beneficios que tiene DevOps, las organizaciones en la industria de TI tienen retos importantes en la adopción de esta práctica debido a que no existe una visión general clara de los procesos y los métodos de implementación efectiva de DevOps [4]. Algunos de estos retos abarcan: problemas de comunicación y colaboración entre equipos y sus integrantes, en especial entre los desarrolladores y los encargados de operaciones dado que no tienen objetivos comunes [4], [8], [9]; falta de motivación y una cultura de resistencia al cambio debido al esfuerzo que esto representa en un ambiente donde existen diversas tareas por realizar y el tiempo es limitado [4], [9]; falta de entrenamiento y habilidades, en vista de que la práctica de DevOps requiere habilidades de desarrollo y operaciones lo que hace difícil que las organizaciones cuenten con el personal apropiado para el intercambio de conocimiento [3], [4], [8]–[10]; el uso de herramientas difíciles de mantener e integrar y la elección de las mismas de tal forma que se adecúen a los procesos de la organización [3], [9]; la utilización de múltiples entornos que introduce una complejidad importante para ejecutar pipelines de

integración y entrega continua afectando su adopción [9]; la omisión de problemas de seguridad derivados de un control de acceso y manejo de secretos deficientes [10]; por último, cada proceso de adopción es único para cada empresa lo que hace necesaria una planeación adecuada y efectiva [2] – [4], [8], [9].

La empresa colombiana Wizit Mind-Blowing Solutions S.A.S [11] dedicada a la consultoría y desarrollo de software, no ha sido ajena a los retos anteriormente expuestos debido a que en la organización la práctica DevOps no se encontraba debidamente estandarizada y no existía un proceso que guiaran la realización de los despliegues de aplicaciones y configuración de infraestructura de manera eficiente en cada ambiente que se esté utilizando, haciendo que los desarrolladores tuviesen más dificultades en los procesos de prueba y despliegue lo que se traducía en un mayor esfuerzo, tiempo y costo.

En este sentido y con base en lo expuesto anteriormente, se planteó el siguiente interrogante para el desarrollo de la práctica profesional:

¿Cómo agilizar la adopción DevOps para automatizar el despliegue y configuración de infraestructura de las aplicaciones desarrolladas por la empresa WIZIT MIND BLOWING SOLUTIONS S.A.S?

1.2. Objetivos

1.2.1. Objetivo general

Definir un proceso y lineamientos que permitan automatizar el despliegue y configuración de infraestructura de aplicaciones web desarrolladas en la empresa WIZIT MIND BLOWING SOLUTIONS S.A.S

1.2.2. Objetivos específicos

- Caracterizar los problemas y las necesidades de la empresa para adoptar prácticas y herramientas DevOps que permitan agilizar los procesos de despliegue y mantenimiento de aplicaciones web.

- Definir un proceso y un grupo de lineamientos para el despliegue de arquitecturas *serverless* utilizando tecnologías de computación en la nube.

- Diseñar e implementar un piloto en el cual se logre desplegar automáticamente una plataforma web de la organización.

1.3. Metodología

Para el desarrollo de esta práctica profesional se utilizó el marco de trabajo SCRUM que corresponde a enfoque de gestión de proyectos ágil que asiste a los equipos en la estructuración y administración del trabajo a través de un conjunto de valores, principios y prácticas [12]. SCRUM propone que el desarrollo de las actividades se

realice en periodos de tiempo llamados *sprints* que generalmente tienen una duración de entre 1 y 4 semanas.

Adicionalmente, para la adaptación y diseño del proceso se tuvo en cuenta el “Método para definir procesos en organizaciones desarrolladoras de software” propuesto [13].

Por otro lado, la adaptación del proceso de DevOps propuesto para la compañía se basó en el trabajo realizado en “Proceso para fomentar y apoyar la adopción de DevOps en PyMEs de software” propuesto en [14].

Capítulo 2 – Marco teórico y tecnologías

A continuación, se definen los conceptos fundamentales que fueron aplicados a lo largo de la práctica profesional y permitieron definir un proceso para la adopción DevOps en la organización. De igual forma, se incluyen las tecnologías que se utilizaron para llevar a cabo el componente práctico de este trabajo de grado.

2.1. Conceptos fundamentales

En esta sección se presentan los principales conceptos asociados a una cultura DevOps necesarios para el entendimiento del presente trabajo.

2.1.1. DevOps

DevOps corresponde a un conjunto de prácticas, herramientas y filosofías culturales que tienen como objetivo incrementar la capacidad de entregar aplicaciones y servicios a un mayor ritmo y de forma constante [5]. El término DevOps proviene de la combinación de los términos en inglés *development* (desarrollo) y *operations* (operaciones) e implica la integración de estas dos disciplinas en un proceso único que permita la colaboración y comunicación, la automatización del ciclo de vida del software, la integración procesos y la mejora continua, de tal manera que se produzca un mayor rendimiento y un menor tiempo de entrega de productos software de calidad incrementando la satisfacción de los clientes [15].

2.1.2. Integración, entrega y despliegue continuo

La Integración, entrega y despliegue continuo son prácticas que automatizan los pasos que deben realizarse para lanzar nuevas versiones de software, permitiendo una entrega más rápida a los usuarios. Esto mejora la frecuencia de los despliegues, puede aumentar la calidad del código y permite a los equipos centrar su atención en tareas más críticas [15].

2.1.2.1. Integración continua

La integración continua hace referencia a la práctica en la cual diferentes desarrolladores integran sus cambios en una base de código fuente sobre la cual posteriormente se realizan compilaciones y pruebas para comprobar que los nuevos cambios no impacten negativamente el software. En esta práctica de DevOps, comúnmente se usan herramientas de automatización que permiten verificar la correctitud de los nuevos fragmentos de código antes de su integración y luego realizar la compilación del código fuente de forma automática [16]. Sin embargo, más allá del componente técnico, la integración continua supone un enfoque cultural donde los desarrolladores han aprendido buenas prácticas para integrar sus cambios frecuentemente.

2.1.2.2. Entrega continua

La entrega continua comúnmente se conoce como el paso siguiente o extensión de la integración continua y es una práctica en la cual se automatiza el proceso de entrega del software permitiendo que pueda ser liberado a un entorno de prueba o un entorno de producción de forma confiable y ágil. En DevOps, la entrega continua automatiza el proceso desde que se integra el código en un único repositorio, se realizan las pruebas y se comprueban los cambios, hasta la entrega al usuario [17].

2.1.2.3. Despliegue continuo

El despliegue continuo corresponde a la práctica donde los pasos que se llevan a cabo durante la integración y entrega continua, se realizan de forma totalmente automatizada. En este sentido, los cambios que suben los desarrolladores, se testean, integran, construyen y despliegan sin intervención humana [18]. Grandes compañías que han adoptado este enfoque, han implementado estrategias para validar las liberaciones en producción y evitar la indisponibilidad de los servicios como los despliegues *canary* donde las nuevas liberaciones solo están disponibles para una pequeña parte de los usuarios y se requiere de su aprobación para realizar una masificación de los cambios al producto software.

2.1.2.4. Pipelines CI/CD

El pipeline de CI/CD corresponde al flujo de trabajo con pasos automatizados que conecta las prácticas de CI/CD, permitiendo a los desarrolladores construir, probar y lanzar cambios en el código de manera más eficiente y con menos errores. Los pipelines de CI/CD pueden incluir una variedad de herramientas y procesos como sistemas de control de versiones de código fuente, servidores de compilación, frameworks de automatización de pruebas y herramientas de despliegue [18]

2.1.3. Infraestructura como código

La infraestructura como código (Infrastructure as Code - IaC) es una práctica clave de DevOps y se usa en conjunto con la entrega continua (CD). Hace referencia a la administración de la infraestructura (hardware, máquinas virtuales, servicios, contenedores, balanceadores de carga, etc.) mediante un modelo descriptivo en código que permite automatizar el proceso manual relacionado a la gestión de esta. Con este enfoque se construyen archivos de configuración que contienen las especificaciones de infraestructura que se necesiten en un determinado entorno facilitando la gestión, replicación, edición y distribución de estos evitando la discrepancia entre ellos [19].

2.1.4. Arquitecturas Sin Servidor (*Serverless)*

Las arquitecturas sin servidor o *serverless* se refieren a una forma de construir y ejecutar aplicaciones y servicios sin tener que administrar y mantener la infraestructura subyacente de las aplicaciones, bases de datos o sistemas de almacenamiento. En serverless se delega la responsabilidad del aprovisionamiento, escalabilidad, administración y disponibilidad sobre el proveedor en la nube. Entre los beneficios de las arquitecturas serverless se encuentra la minimización de costos debido a que se paga solo por la ejecución real de las funciones o servicios, sin incurrir en gastos por recursos no utilizados como servidores en funcionamiento constante. Por otro lado, los recursos pueden escalar automáticamente en respuesta a fluctuaciones de tráfico o carga de trabajo y adicionalmente, incrementa la productividad en vista de que los desarrolladores no tienen que lidiar con la gestión de servidores [20]

2.1.4.1. Function as a Service (FaaS)

Function as a Service es un servicio de computación en la nube *serverless* que permite desarrollar, ejecutar y administrar funcionalidades de código que a eventos sin tener que administrar la infraestructura subyacente asociada con la construcción y ejecución de microservicios [21].

2.1.5 Computación en la nube

La computación en la nube ofrece recursos de computación bajo demanda que son distribuidos a través de internet, siendo una alternativa que evita que las organizaciones deban encargarse del aprovisionamiento, configuración y gestión de los recursos de manera local [22]

2.2. Tecnologías usadas

En esta sección se listan algunas de las tecnologías usadas a lo largo del desarrollo de esta práctica profesional. Si bien no fueron las únicas herramientas tecnológicas usadas, se consideran las más relevantes.

2.2.1. Herramientas de control de código fuente

- **Git:** Sistema de control de versiones distribuido que permite rastrear cambios sobre un conjunto de archivos de código fuente o cualquier otro tipo de archivos de tal manera que es posible el trabajo colaborativo y simultáneo entre integrantes de un equipo [23].

- **GitHub:** Plataforma que permite alojar y realizar seguimientos a repositorios Git. Adicionalmente, GitHub provee diversas herramientas para la gestión de proyectos y equipos, code review, automatización y colaboración haciendo más sencillo para los desarrolladores trabajar en conjunto en sus proyectos [24].

- **GitLab:** Herramienta para la gestión de repositorios Git e integración continua. Es muy similar a GitHub, la diferencia radica en las características de los servicios que ofrecen ambas plataformas. Adicionalmente GitLab es una plataforma de código abierto y *self-hosted* lo que quiere decir que se puede instalar en un servidor propio [25].

2.2.2. Proveedor en la nube

- **Amazon Web Services:** AWS es una plataforma de computación en la nube desarrollada por Amazon que ofrece soluciones en la nube escalables, confiables y rentables. Algunos productos y servicios de AWS incluyen almacenamiento, administración de bases de datos, servicios de seguridad, networking, herramientas de desarrollo de aplicaciones web y móviles, etc. [26]

2.2.3. Tecnologías de desarrollo

- **JavaScript:** JavaScript es un lenguaje de programación multiparadigma interpretado de alto nivel que permite implementar funcionalidades complejas en páginas web. Inicialmente fue concebido para su uso en navegadores web, sin embargo, su uso se ha extendido al lado del servidor con herramientas como Node.js. [27]

- **Node.js:** Node.js es un entorno de ejecución (runtime) de JavaScript, es decir, permite ejecutar código JavaScript por fuera del navegador [28].

- **React.js:** Es una biblioteca de JavaScript que se utiliza para construir interfaces de usuario [29]

2.2.4. Herramientas de IaC y Frameworks Serverless

- **AWS CloudFormation**: AWS CloudFormation es un servicio de infraestructura como código que permite modelar, aprovisionar y administrar recursos de AWS y terceros a través del uso de plantillas donde se describen los recursos que se deben aprovisionar y la relación entre ellos [30].

- **AWS Serverless Application Model (SAM):** El modelo de aplicación sin servidor es un framework open-source para construir aplicaciones serverless el cual proporciona sintaxis para expresar funciones lambda, APIs, bases de datos y eventos de manera más sencilla [31].

- **Serverless Framework:** Es un framework web open-source que permite a los desarrolladores construir y desplegar aplicaciones serverless de manera más sencilla y eficiente usando diferentes proveedores en la nube como AWS, Azure, GCP [32].

2.2.5. Tecnologías de CI/CD

- **Jenkins:** Jenkins es un servidor de automatización open-source que provee gran variedad de plugins que permiten automatizar las etapas de build, test y deploy de cualquier proyecto software posibilitando a los desarrolladores realizar la integración continua de sus cambios en código fuente y haciendo más sencilla y confiable la entrega de nuevas funcionalidades a los usuarios. Actualmente, Jenkins es la herramienta más usada para CI/CD; sin embargo, también puede ser usada para crear flujos de trabajo personalizados que automaticen tareas complejas como por ejemplo el monitoreo continuo [33].

- **Github Actions:** Github Actions es una plataforma de integración y entrega continua (CI/CD) que hace parte de GitHub. Permite a los desarrolladores automatizar diferentes tareas como construir, testear y desplegar actualizaciones de software. Al igual que Jenkins, con GitHub Actions se pueden crear flujos de trabajo personalizados que automaticen tareas que hagan parte del proceso de desarrollo de software [34]

Capítulo 3 – Diagnóstico del estado inicial

En este capítulo, se presentan los resultados obtenidos durante la fase de recolección de información interna en la organización. El objetivo principal de esta etapa fue analizar cómo se estaban llevando a cabo los procesos de desarrollo, testing y despliegue en la compañía. Además, se evaluó la percepción que tanto los colaboradores como los líderes de TI tenían sobre DevOps, con el fin de identificar el nivel de conocimiento existente acerca de las prácticas DevOps y las tecnologías asociadas.

3.1. Proceso de desarrollo, testing y despliegues

En esta sección se presenta la metodología de desarrollo, testing y despliegue que se estaba llevando a cabo en la compañía durante la fase de recolección de información.

3.1.1. Marco de trabajo

Dentro del proceso de desarrollo de software que se lleva a cabo en la organización se aplica el marco de trabajo SCRUM el cual se basa en aplicar un conjunto de buenas prácticas para trabajar en equipo y que tiene como objetivo maximizar el valor entregado a los clientes.

En términos generales el proceso llevado a cabo consiste en desarrollar de manera iterativa las funcionalidades teniendo en cuenta la priorización del cliente y al terminar cada iteración se evalúan los resultados obtenidos y se aplican acciones correctivas o de mejora para los siguientes ciclos en caso de ser necesario.

A continuación, se precisan los elementos relevantes del marco de trabajo SCRUM [12] aplicada en la organización.

- **Sprints**: Un sprint corresponde al ciclo de trabajo o iteración donde se fijan unos objetivos específicos que se espera que se alcancen en un tiempo determinado. Al final de cada sprint lo que se consigue es un entregable o incremento del producto que aporte valor al cliente. En la mayoría de los proyectos dentro de la organización, los sprints tienen una duración de 2 semanas; sin embargo, esto dependerá del cliente y la naturaleza del proyecto.

- **Roles**: Dentro de los proyectos de la organización se pueden identificar tres roles principales

    - **Scrum Maste**r: Se encarga de gestionar el proyecto, velar que las actividades Scrum se estén llevando a cabo correctamente y ayudar a eliminar impedimentos que puedan afectar el avance del proyecto.

    - **Product Owner**: Vela por los intereses del cliente, es decir, se encarga de optimizar y maximizar el valor del producto. En ese sentido, es el product

owner quien realiza la definición de los requerimientos de acuerdo con las necesidades del negocio y establece su prioridad.

- **Developer Team**: Corresponde a los demás colaboradores que hacen posible el desarrollo técnico del sprint. En algunos proyectos de la organización este equipo está dividido en dos partes: los integrantes del equipo por parte del cliente final y el equipo por parte de la empresa. Esto responde al modelo de contratación *staff augmentation*.

  - **Developer Team (Cliente)**: En este equipo están incluidos los arquitectos de datos, arquitectos de software, DevOps, QAs etc. que se encargan de gestionar la calidad del producto, la infraestructura y datos internos del cliente.

  - **Developer Team (Empresa)**: Corresponde al equipo de desarrolladores de software y QA que se encargan de desarrollar el incremento del producto de software y verificar la calidad que se le entrega al cliente.

- **Eventos o ceremonias del Sprint**: Durante el sprint se llevan a cabo una serie de eventos o ceremonias que corresponden a reuniones con propósitos específicos que contribuyen a la gestión del proyecto.

  - **Sprint Planning:** El sprint planning se realiza al inicio del sprint y tiene como propósito establecer qué requerimientos u objetivos se pretenden alcanzar durante el sprint. Para ello se realiza una sesión de refinamiento de los requisitos con todo el equipo donde se pueden llegar a modificar las historias de usuario de acuerdo con las necesidades de negocio, limitaciones técnicas, recomendaciones o aclaraciones.

  - **Daily Meeting**: Son reuniones cortas de máximo 15 minutos donde cada miembro del equipo da reporte de sus avances, las actividades planificadas para el día y los impedimentos que tiene, esto con el objetivo de dar un seguimiento al desarrollo que permita identificar el progreso del sprint y situaciones que puedan impedir que se alcance el objetivo del sprint. Cabe resaltar que dentro en la mayoría de los proyectos en la organización se realizan dos *daily meeting* distintas:

    - **Daily Interna (Empresa)**: Estas reuniones se realizan de manera diaria con el equipo de desarrollo y líder técnico de la empresa y son necesarias para identificar impedimentos o situaciones que se pueden gestionar de manera interna sin necesidad de la intervención del cliente.

    - **Daily con el cliente**: Estas reuniones se realizan 2 o 3 veces a la semana y se da reporte al cliente (SM, PO y demás stakeholders) sobre los avances del equipo de desarrollo y si es necesario se escalan peticiones para solventar situaciones que estén dificultando continuar con las labores del sprint.

- **Sprint Review:** Las ceremonias de Sprint Review se realizan al final del sprint y su objetivo es mostrar a los diferentes stakeholders el trabajo completado durante el sprint. Normalmente se realiza una demo evidenciando las HU desarrolladas y las funcionalidades integradas.

- **Sprint Retrospective:** Las reuniones de retrospectiva se realizan normalmente después del Sprint Review y su objetivo es que los miembros del equipo discutan acerca de lo que consideran que se hizo bien durante el sprint, los impedimentos o bloqueantes que se presentaron y qué acciones de mejora se podrían implementar en los siguientes sprints.

- **Artefactos:** Dentro del marco de trabajo aplicado en la organización se obtienen algunos elementos llamados artefactos que surgen de la aplicación de SCRUM. A continuación, se listan los más relevantes.

  - **Product Backlog:** Se refiere a un listado de todas las tareas, actividades o requerimientos que se esperan implementar durante el desarrollo del proyecto. Generalmente estos requerimientos están priorizados con base a su complejidad y las necesidades del cliente.

    En los proyectos el product backlog puede dividirse en los requerimientos que se pretenden realizar para tener un producto mínimo viable que salga a producción y los demás requerimientos que serán añadidos en las siguientes versiones el producto software.

  - **Sprint Backlog:** Corresponde al listado de requerimientos que se pretenden realizar durante el sprint.

  - **Incremento del producto:** Hacen referencia a los entregables de cada final de sprint que abarcan los requerimientos implementados por el equipo de desarrollo.

3.1.2. Testing

Para facilitar la descripción de la fase de pruebas que se aplica en la organización se deben tener en cuenta los siguientes términos [35]:

- **Error**: Acción humana que produce un resultado incorrecto. En el desarrollo de software un ejemplo de error es una equivocación en la lógica de programación por parte del desarrollador o un requerimiento mal especificado por parte del analista de requerimientos/product owner.

- **Defecto**: Corresponde a la imperfección de un componente de software causada por un error.

- **Fallo**: Es la manifestación visible de un defecto en la ejecución de la aplicación o producto software.

- **Bug**: El término bug se refiere a los defectos en el software que producen que una aplicación no funcione de la manera esperada

- **Caso de prueba**: Se refiere al conjunto de acciones ejecutadas para verificar que una característica o funcionalidad de un software cumple con lo esperado. En los casos de prueba se documentan las condiciones previas a la prueba, los datos y pasos que se deben llevar a cabo para ejecutar la prueba y los resultados que se esperan del software una vez termine la ejecución.

Durante el proceso de desarrollo de software llevado a cabo en la compañía resulta fundamental asegurar la calidad del producto que es entregado a los clientes; por ello, se realiza una etapa de testing interno que busca identificar defectos en el producto que puedan afectar la experiencia de los usuarios en el entorno productivo.

Los objetivos del proceso de testing interno abarcan:

- Evaluar que todos los productos de trabajo funcionen correctamente.
- Verificar que todos los requerimientos especificados se han implementado correctamente, incluidos los de ámbito legal.
- Generar confianza en el software desarrollado
- Encontrar fallas y defectos.
- Incrementar la calidad del software.

Con el fin de que estos objetivos sean alcanzados en cada proyecto desarrollado por la empresa, cada equipo de trabajo tiene una persona encargada de realizar el proceso de pruebas de los requerimientos que se vayan desarrollando durante un sprint.

El desarrollo del ciclo de testing inicia con la etapa de análisis de pruebas donde se tienen como insumos:

- Los requerimientos que se van a desarrollar durante el sprint; es decir las historias de usuario del sprint backlog.
- El diseño UX de la aplicación (si es necesario)
- Documentación de APIs externas que deban ser integradas (si es necesario).

Esta etapa tiene como objetivo identificar riesgos, omisiones o inexactitudes que puedan ser resueltas o escaladas hacia el cliente con el fin de anticiparse a posibles bloqueantes en el proceso.

Posteriormente, se inicia la etapa del diseño de pruebas donde se crea el listado de los casos de prueba que evalúan el comportamiento que se espera que tenga el producto software bajo determinadas condiciones una vez se termine el desarrollo de un requerimiento.

Tan pronto como los desarrolladores finalicen una historia de usuario (HU) se notifica al tester que la HU o tarea está lista para ser probada por medio del software de gestión del proyecto, iniciando la etapa de ejecución de casos de prueba. Es aquí donde el tester realiza todo el flujo necesario para probar las funcionalidades añadidas al producto software, toma evidencias de los resultados esperados y obtenidos y de

acuerdo a ello establece si el caso de prueba es fallido o exitoso. En este sentido, se pueden dar entonces los siguientes escenarios:

- Si el caso de prueba es exitoso, la HU o tarea puede marcarse como hecha en el software de gestión del proyecto.

- Si el caso de prueba es fallido, inicia la etapa de reporte de *bugs* donde el tester documenta las evidencias del caso de prueba fallido y reporta el bug en el software de gestión del proyecto describiendo el comportamiento y apoyándose de las respectivas evidencias; adicionalmente, la HU o tarea debe establecerse “en progreso” nuevamente.

Cuando se reportan bugs inicia la etapa de gestión y seguimiento donde el desarrollador debe hacer la corrección del defecto o la falla y una vez esté corregido, el tester debe volver a ejecutar los casos de prueba fallidos con el fin de verificar que efectivamente el bug fue resuelto. Si se encuentra que no ha sido corregido en su totalidad, el desarrollador debe realizar las correcciones respectivas y nuevamente solicitar al tester que vuelva a realizar las pruebas. Este proceso es iterativo hasta que el bug sea resuelto en su totalidad.

Todas las etapas anteriormente descritas se repiten hasta finalizar el sprint. A continuación, se muestra una imagen ilustrativa sobre el proceso.

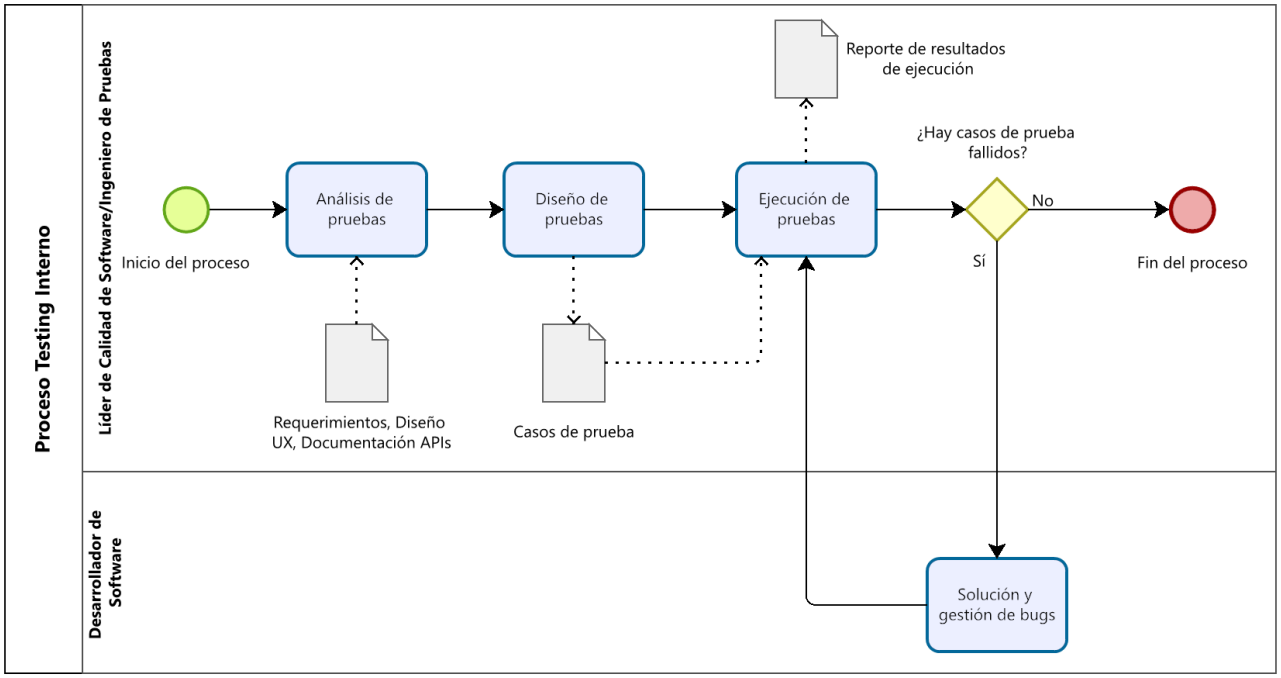

Figura 1. Modelado del proceso de testing interno

3.1.3. Despliegue del software y aspectos relevantes

Una parte fundamental del desarrollo de software comprende el proceso de despliegue, que engloba todas aquellas actividades que garantizan la disponibilidad y uso de los productos de software. En el caso de las aplicaciones web, como los

productos desarrollados por la compañía, el despliegue permite el acceso a las aplicaciones a través del navegador web. Además, en el caso de las APIs, el despliegue posibilita que sean consumidas por diversos clientes, como aplicaciones móviles, aplicaciones web, clientes de escritorio, otras APIs, entre otros.

De acuerdo a la recolección de información acerca de cómo se realizaban los despliegues de las aplicaciones dentro de la organización, se identificó que múltples pasos se estaba llevando de forma manual. Por otro lado, se identificó la existencia de un problema durante el proceso de desarrollo de software en la compañía relacionado con la diversidad de entornos/ambientes que utilizan los clientes. Estos entornos pueden contar con herramientas específicas, configuraciones personalizadas y estar sujetos a políticas de seguridad. Además, se presentan procesos burocráticos donde las tareas de configuración de ambiente, infraestructura y despliegues pueden tomar horas o incluso días para completarse, lo que dificulta y bloquea el flujo de desarrollo.

Por esta razón, la organización consideró de gran importancia establecer sus propios entornos de desarrollo, estandarizar la infraestructura y promover buenas prácticas que faciliten los despliegues en los diversos ambientes. El objetivo de esto es asegurar un entorno funcional que permita a los desarrolladores aumentar su productividad de manera significativa.

A continuación, se mencionan algunos puntos relevantes de cómo se realizaban los despliegues de las aplicaciones:

- Los despliegues deben realizarse hacia los ambientes de la empresa y del cliente. Regularmente, se decide qué tan frecuentes deben ser para los diferentes entornos y pueden ser asignados a uno o varios miembros del equipo.

Despliegues en ambiente de la empresa:

- Para el despliegue de infraestructura se utilizaba la interfaz gráfica del proveedor en la nube, donde se subía la plantilla de infraestructura como código y se ingresaban de forma manual los parámetros necesarios para el despliegue. A continuación, se presenta un diagrama de proceso que representa esta situación.

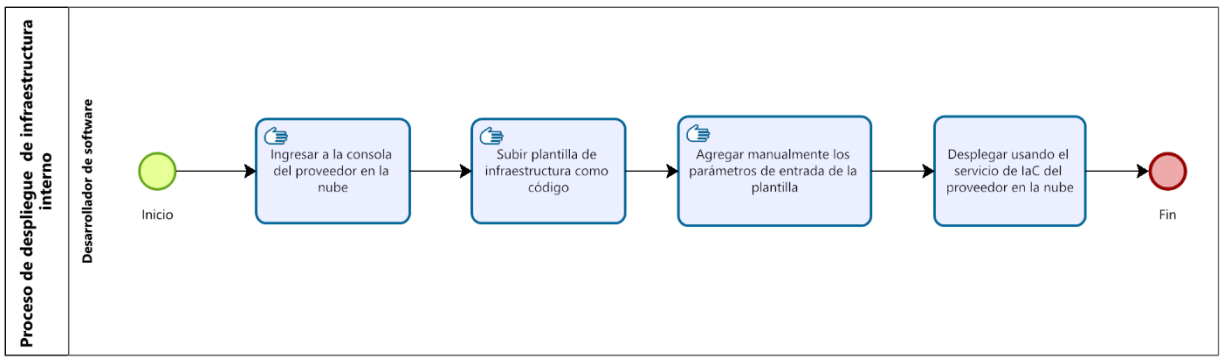

Figura 2. Diagrama de proceso de despliegue de infraestructura

El manejo de infraestructura de esta forma presenta desafíos para mantener la estabilidad del entorno dada su propensión a errores debido a las actividades manuales requeridas. Además, este enfoque también permite que los desarrolladores realicen despliegues de plantillas que no se encuentran en el repositorio remoto, lo que puede dar lugar a diferencias en la versión común que se mantiene entre el resto de los desarrolladores.

Además, es importante destacar que no todos los desarrolladores tienen la capacitación necesaria para gestionar la infraestructura. A pesar de ello, actualmente tienen la libertad de realizar cambios en las plantillas sin seguir un esquema base o estándar, lo que da lugar a numerosas disparidades en la configuración de infraestructura entre los diferentes proyectos. Esta falta de uniformidad dificulta la reutilización efectiva del código de Infraestructura como Código (IaC) y puede llevar a problemas de compatibilidad y mantenimiento en el futuro.

Las circunstancias descritas anteriormente pueden generar inconsistencias y dificultades en el proceso de desarrollo y despliegue de infraestructura. En este punto se hace importante considerar alternativas más sólidas y automatizadas que mejoren la estabilidad y coherencia del entorno de infraestructura, así como es fundamental establecer pautas y procedimientos que aseguren la consistencia en la creación y modificación de las plantillas de IaC.

- El despliegue de backend y frontend los desarrolladores podían hacerlo desde sus máquinas utilizando comandos por consola que requerían realizar configuraciones previas de credenciales del proveedor en la nube. Enseguida, se presenta un diagrama de proceso que describe esta situación.

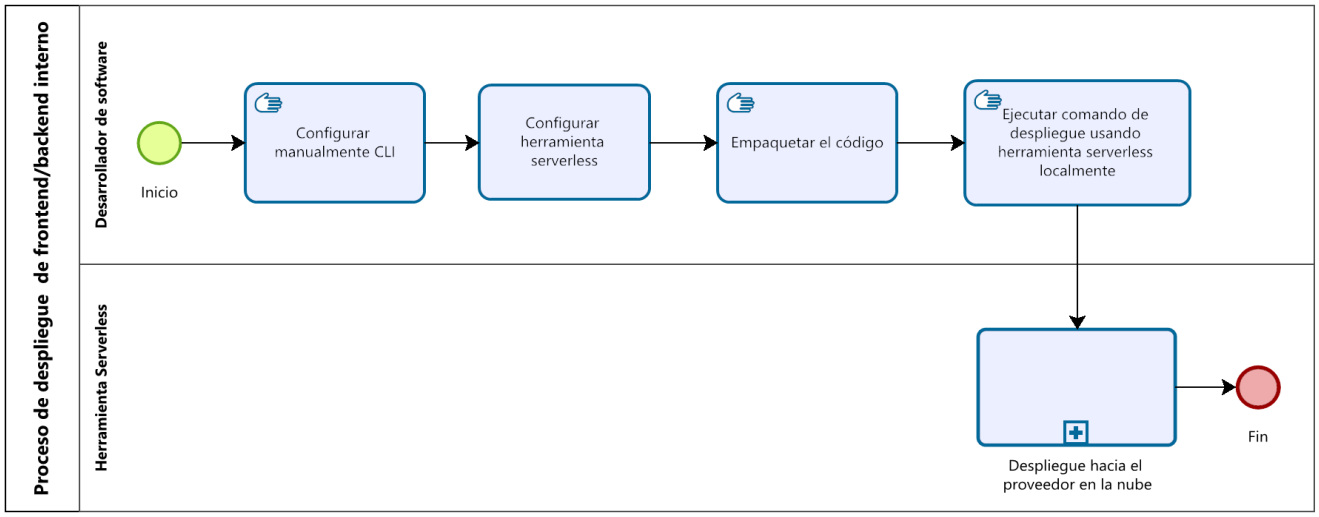

Figura 3. Diagrama de proceso despliegue interno de la organización

En este punto es importante resaltar que una de las principales dificultades al utilizar arquitecturas *serverless*, especialmente en el lado del backend, es la limitación para realizar pruebas desde un entorno local. Como resultado, los desarrolladores vieron la necesidad de llevar a cabo despliegues frecuentes hacia el entorno de computación

en la nube para poder probar los nuevos cambios. En este contexto, la falta de un entorno local adecuado para probar las funciones y servicios *serverless* puede restringir la agilidad y eficiencia del proceso de desarrollo. Los desarrolladores se ven obligados a realizar despliegues completos en la nube para evaluar y depurar su código, lo que puede ser más lento y engorroso en comparación con las pruebas locales tradicionales.

Debido a lo anterior, los desarrolladores tienen la posibilidad de realizar despliegues manuales de sus cambios locales tanto en el backend como en el frontend, sin necesidad de integrarlos al repositorio remoto. Sin embargo, esta situación genera discrepancias entre la versión desplegada y la versión que los desarrolladores tienen en sus entornos locales generando confusiones y dificultando la manutención y estabilidad del ambiente.

Además, la práctica manual del despliegue limita la capacidad de realizar pruebas automáticas y evaluaciones de calidad de manera efectiva. La ausencia de pruebas automatizadas aumenta el riesgo de introducir errores en el código y disminuye la confianza en la calidad del software.

Estas actividades manuales necesarias para los despliegues pueden consumir tiempo y esfuerzo considerable, lo que afecta la productividad del equipo y ralentizar la entrega de nuevas funcionalidades o correcciones.

Por último, este enfoque dificulta la inclusión de nuevos colaboradores al proyecto, ya que deben realizar configuraciones y ajustes específicos para llevar a cabo los despliegues de forma adecuada (que terminan siendo necesarios debido a la dificultad de pruebas en entornos locales).

En conclusión, resulta esencial buscar soluciones que posibiliten la simulación o emulación local de las funciones serverless, lo cual permitiría realizar pruebas y depuraciones de manera más rápida y eficiente. Además, se deben considerar alternativas que promuevan la integración continua, la automatización de pruebas y un enfoque más estructurado en el manejo de los despliegues.

Despliegues en ambientes del cliente:

- En el entorno del cliente se manejan tres ambientes: *desarrollo*, *pruebas* y *producción.* El ambiente de desarrollo (*development* o *dev*) es donde los desarrolladores realizan las pruebas de las funcionalidades que se van integrando al producto software. Por su parte el entorno de pruebas (*nonprod* o *test*) es donde los QA y personas de negocio ejecutan los casos de prueba sobre las nuevas funcionalidades y verifican los flujos completos de la aplicación. Por último, el ambiente productivo (*prod*) corresponde al aplicativo que se disponibiliza hacia los usuarios finales.

- Los despliegues en el ambiente *dev* del cliente se realizan a través de la herramienta de CI/CD y pueden ser ejecutados por cualquier persona del proyecto. Sin embargo, para ambientes *nonprod* y *prod*, los despliegues deben ser ejecutados por el ingeniero DevOps asignado al proyecto. En este caso, el

equipo de desarrollo debe generar un tag de git y el despliegue de componentes debe ser aprobado dentro de la organización del cliente.

3.2. Evaluación de la percepción sobre DevOps

Como parte de la fase de recolección de información se realizaron encuestas para conocer la percepción de los líderes de TI y algunos desarrolladores de la compañía acerca de DevOps. Las encuestas fueron divididas en dos: una para los líderes de TI quienes tienen más experiencia y conocimiento tanto técnico como en gestión de proyectos y otra para los desarrolladores, los cuales cuentan con conocimiento técnico y se encuentran involucrados en diferentes proyectos.

3.2.1. Encuesta para líderes de TI

La encuesta para líderes de TI se compuso principalmente de preguntas abiertas y de selección múltiple cuyo objetivo fue determinar con base en su experiencia el conocimiento acerca de las prácticas DevOps y algunas de las herramientas comúnmente usadas para estas prácticas. El detalle de las preguntas y resultados de la encuesta se encuentra en el **Anexo A. Encuesta percepción de DevOps para Líderes de TI**.

3.2.1.1. Análisis de las respuestas

Se entrevistaron a tres líderes de TI con más de siete años de experiencia y de acuerdo con los resultados obtenidos se deduce lo siguiente:

- Dos de los tres encuestados relacionan estrechamente DevOps con el despliegue y configuración de infraestructura de las aplicaciones web. Sin embargo, no identifican DevOps como un movimiento cultural, más bien se le asocia a un rol específico o una tarea específica de automatización y no se le ve como un conjunto de prácticas que tiene como objetivo final la entrega más rápida y eficiente de incrementos de software de calidad.

- Se identifica correctamente algunas de las actividades principales que se realizan en DevOps incluyendo configuración y definición de arquitecturas escalables y replicables, configuración de ambientes y pipelines que conducen a la automatización de los despliegues y cumplimiento de los estándares de seguridad de las aplicaciones que dependen de la infraestructura. Sin embargo, se dejan de lado diversas actividades que también hacen parte de una cultura DevOps que está presente durante todo el ciclo de vida del software como lo es el monitoreo continuo, el análisis de código o el testing continuo, entre otras.

- De acuerdo a los resultados observados en la Figura 4, 100% de los encuestados reconocen a DevOps como una práctica importante para las organizaciones.

Figura 4. Resultados importancia de DevOps en las organizaciones

- Como se ilustra en la Figura 5 las prácticas más conocidas por los encuestados son la integración continua, entrega continua e infraestructura como código. Las cuales también son las que, en su mayoría, identifican que se aplican en las organizaciones. Sin embargo, en prácticas como continuous feedback, improvement y testing existe una brecha en el conocimiento.

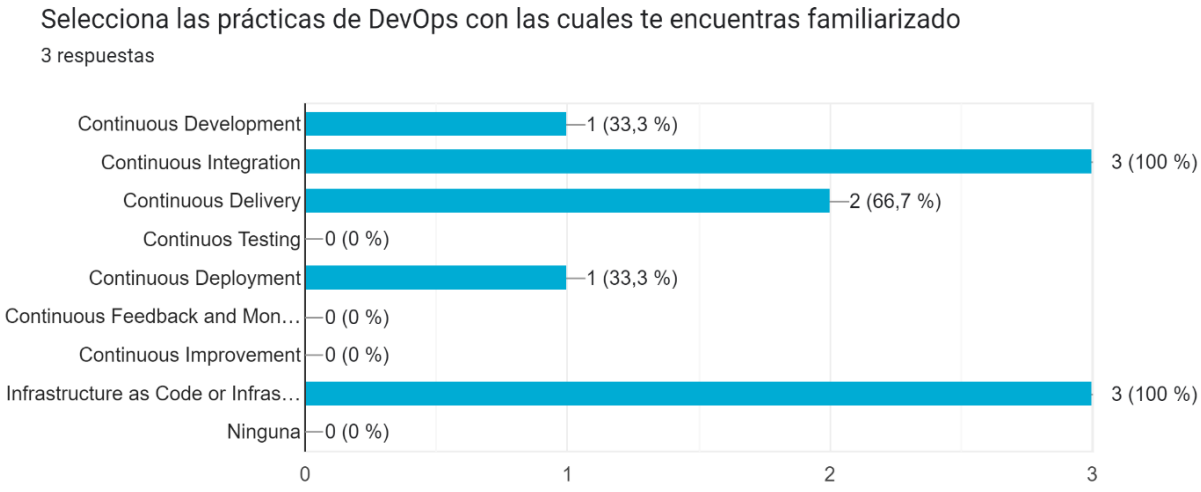

Figura 5. Resultados prácticas DevOps conocidas por líderes de TI

Figura 6. Resultados prácticas DevOps implementadas en la organización

- Como se observa en la Figura 7, Jenkins es la herramienta para CI/CD más conocida por los líderes de TI. Esto se considera lo natural ya que es la herramienta más popular en el mercado para estos propósitos. Adicionalmente, de acuerdo a los resultados presentados en la Figura 8, la herramienta de infraestructura más conocida por los encuestados es AWS CloudFormation.

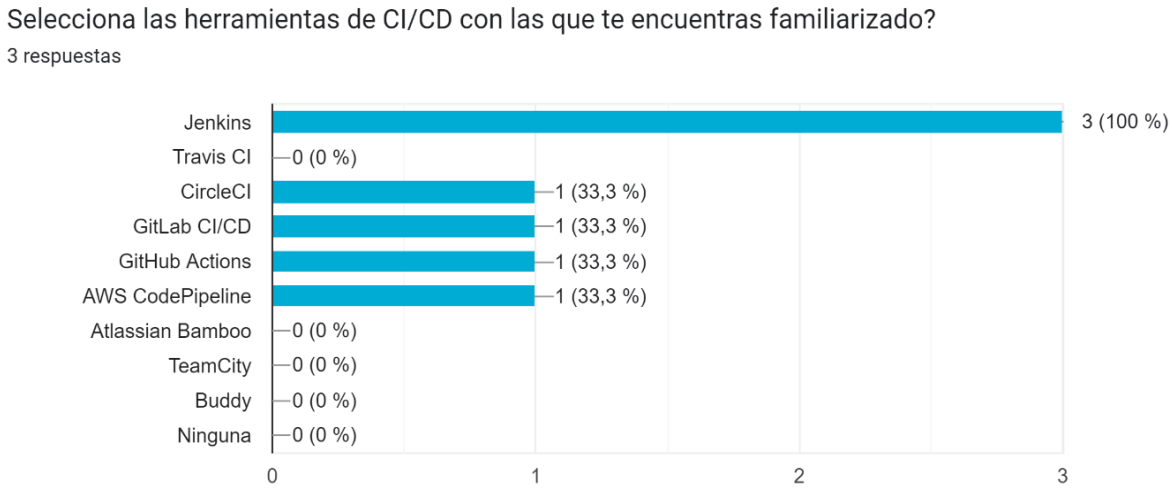

Figura 7. Resultados herramientas de CI/CD conocidas

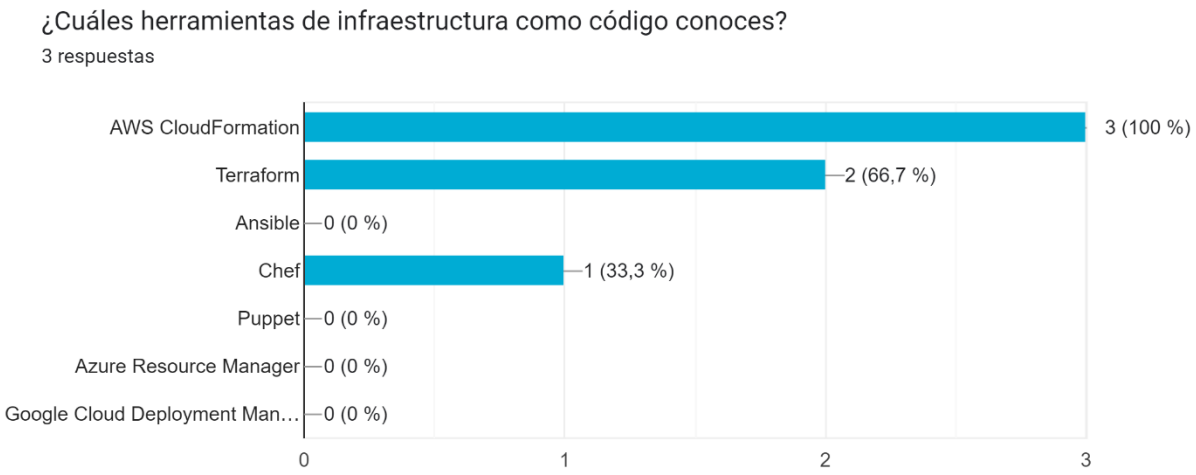

Figura 8. Resultados herramientas de infraestructura como código conocidas

- Según los resultados que se presentan en el gráfico de la Figura 9, las herramientas de pruebas reconocidas por los líderes de TI son principalmente Jest y Selenium

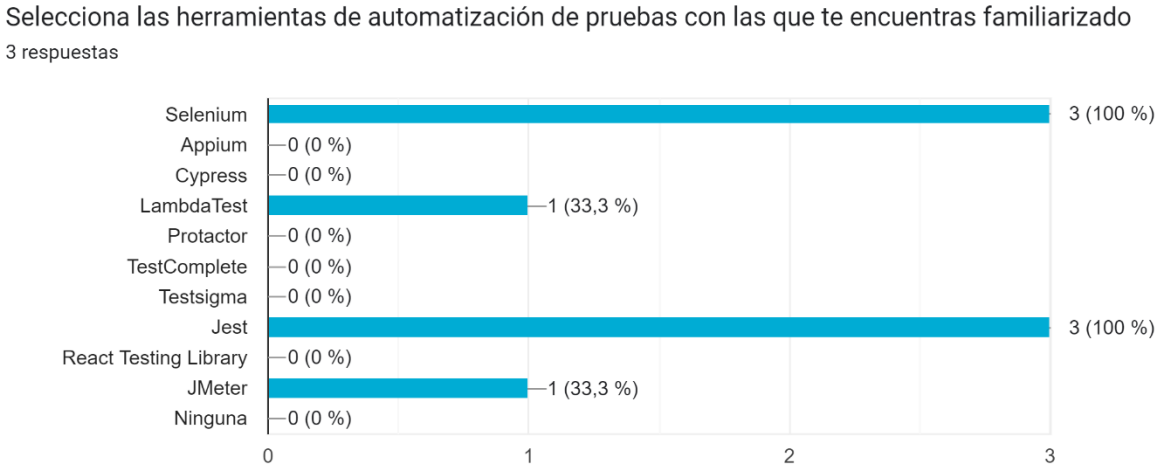

Figura 9. Resultados herramientas de testing conocidas por los líderes de TI

- Como conclusión, se considera que el conocimiento sobre DevOps de los líderes de TI está enfocado al aspecto técnico y está acoplado a las herramientas que presentan mayor conveniencia con las tecnologías usadas en las organizaciones. Sin embargo, el conocimiento sobre las diversas prácticas DevOps es limitado.

3.2.2. Encuesta para desarrolladores

Esta encuesta tenía como objetivo establecer el nivel de conocimiento y experiencia de los desarrolladores con las prácticas y herramientas usadas para aplicar DevOps. La encuesta se compuso de 10 preguntas de selección múltiple con el propósito de obtener información cuantitativa sobre la familiarización de términos y herramientas. El detalle de las preguntas y resultados de la encuesta se encuentra en el **Anexo B. Encuesta sobre percepción de DevOps para desarrolladores**

3.2.2.1. Análisis de las respuestas

La encuesta fue aplicada a 14 desarrolladores de diferentes rangos y se obtuvieron las siguientes conclusiones:

Como se ilustra en la Figura 10, un 57% de los encuestados se inclinan por la definición de DevOps como un conjunto de herramientas y principios que tiene como objetivo la mejora de la colaboración entre los equipos de desarrollo y operaciones. Sin embargo, este no es el objetivo final de DevOps, ya que DevOps supone la adopción no solo de prácticas y herramientas, sino también de un paradigma cultural donde el interés sea entregar valor más rápido y de mejor calidad.

Figura 10. Resultados definición de DevOps

De acuerdo a los resultados presentados en la Figura 11 más del 70% de los encuestados concuerdan con que las actividades principales son la implementación de la integración y entrega continua, diseño y configuración de infraestructura, implementación estrategias de *release* y monitorización. Sin embargo, aproximadamente el 20% tienen un concepto erróneo de las actividades debido a que consideran que dentro de las actividades de DevOps se realiza desarrollo de código incremental, lo cual se sale del objetivo de DevOps.

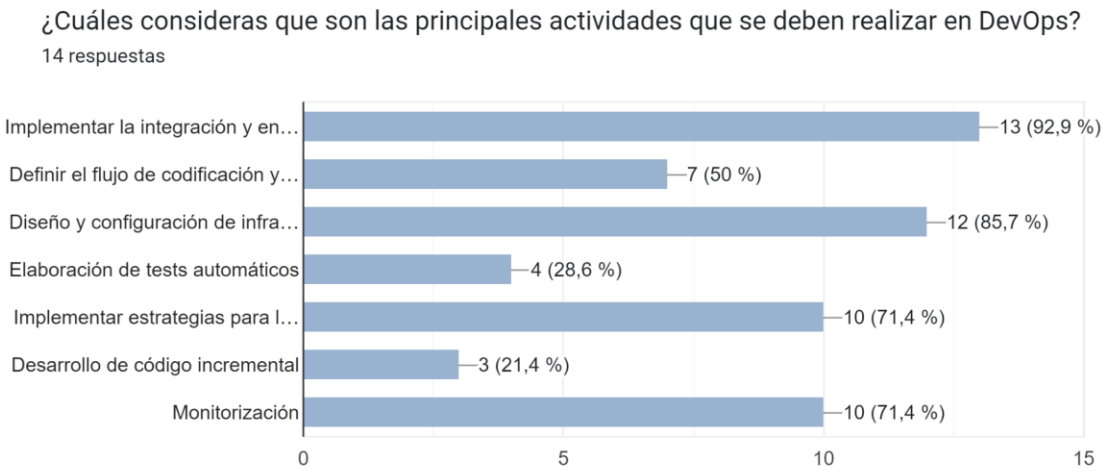

Figura 11.Resultados actividades de DevOps

Acorde a los resultados que se observan en la Figura 12, se evidencia que existe poco conocimiento de las prácticas de DevOps por parte de los desarrolladores ya que menos del 50% de los desarrolladores están familiarizados con CI/CD. Además, tanto la monitorización, así como la infraestructura como código solo es conocida por el 28% de los encuestados. Esto puede deberse a desconocimiento de los términos y conceptos.

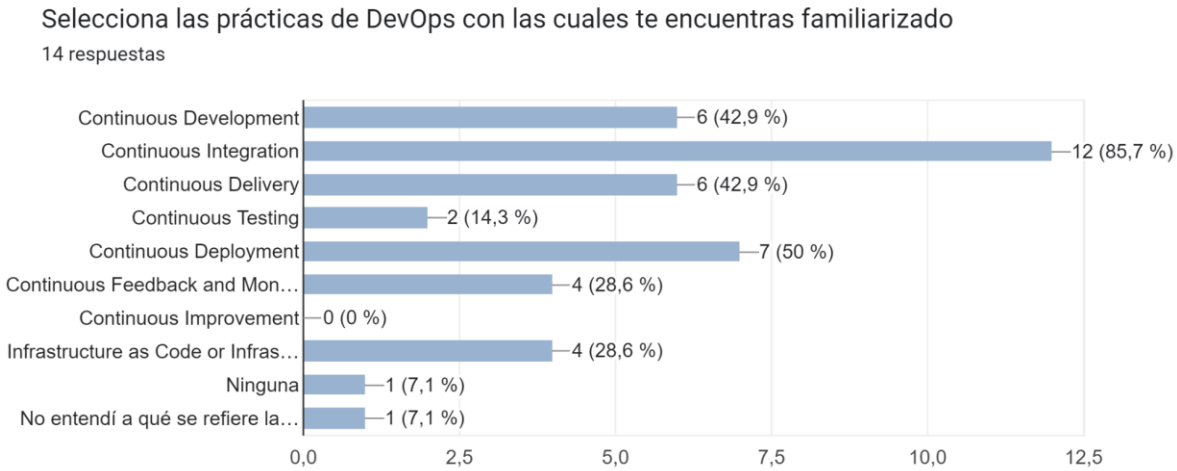

Figura 12. Resultados Prácticas DevOps

Como se observa en la siguiente figura, de acuerdo a la percepción de la mayoría de desarrolladores, en la organización se aplica Continuous Integration y Continuous Development. Las demás solo son reconocidas por menos del 35%. Además, 14% de los encuestados no considera que se realicen prácticas DevOps en las organizaciones.

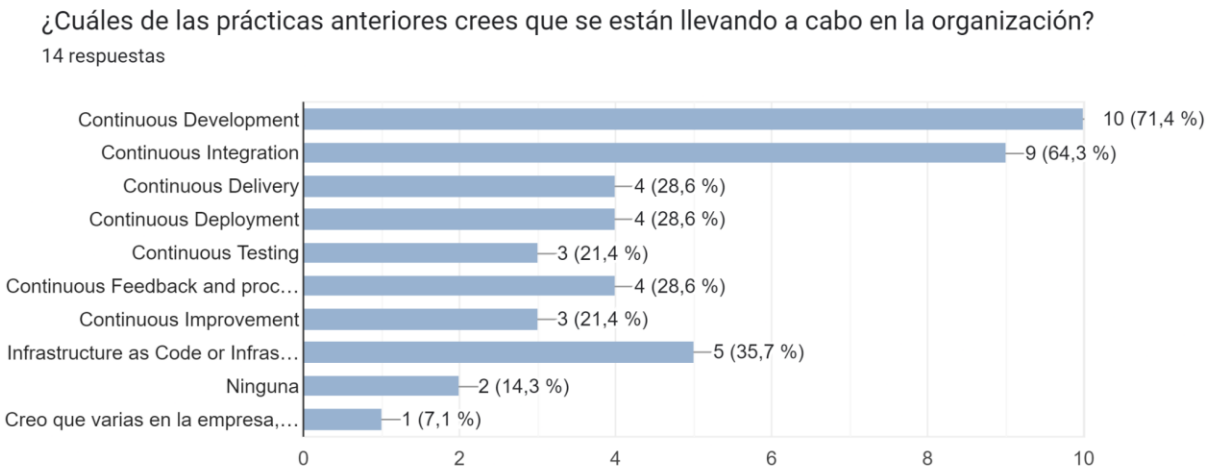

Figura 13.Resultados prácticas DevOps aplicadas en la organización

Por parte de los desarrolladores encuestados se considera que la práctica de DevOps es muy importante para las organizaciones como se puede visualizar en el siguiente gráfico.

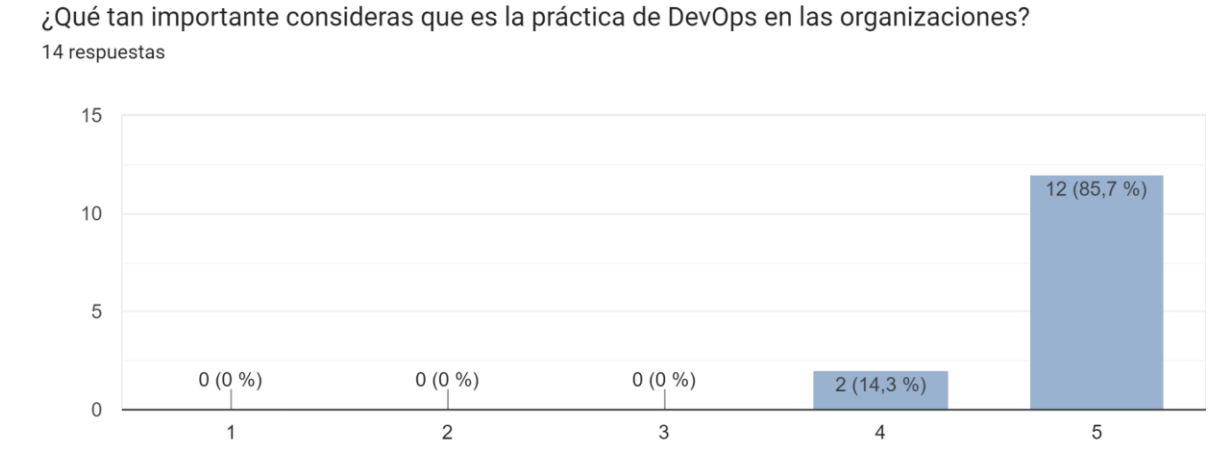

Figura 14. Resultados importancia DevOps en las organizaciones

De acuerdo con los resultados que se visualizan en la Figura 15, se concluye que existe un conocimiento general sobre algunas de las herramientas de CI/CD de la industria como son: GitHub Actions, Gitlab CI/CD, Jenkins y AWS CodePipeline. Sin embargo, el porcentaje de desarrolladores que están familiarizados con ellas sigue siendo bajo. Adicionalmente, como se puede observar en la Figura 16 existe poco conocimiento acerca de herramientas de automatización de pruebas.

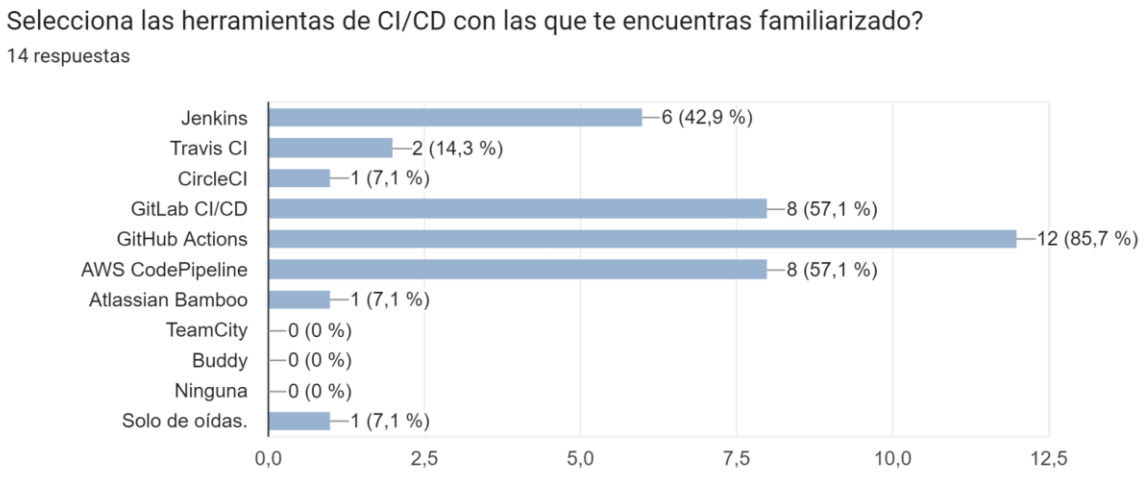

Figura 15. Resultados herramientas de CI/CD

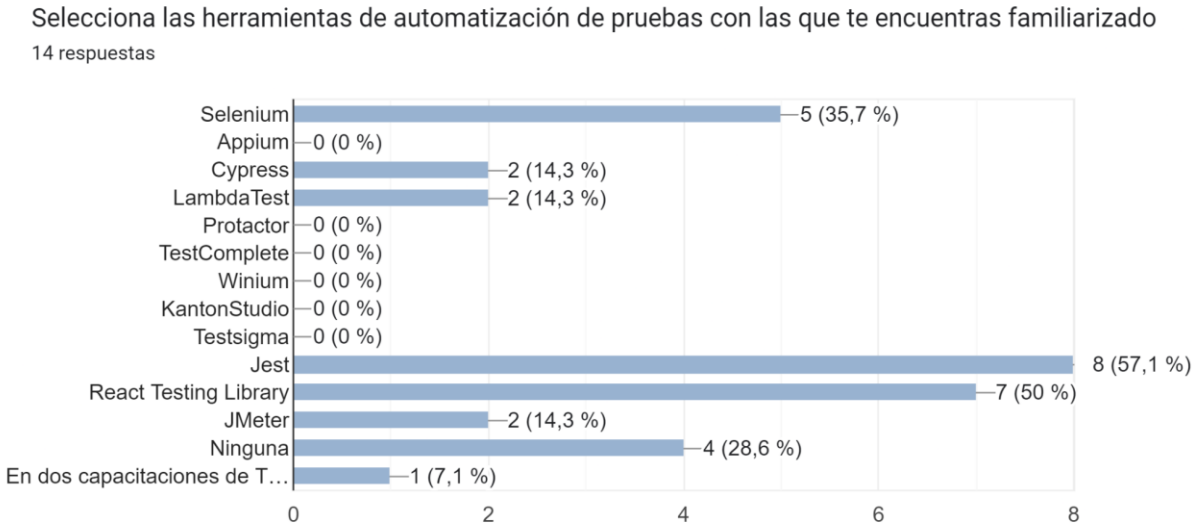

Figura 16. Resultados herramientas de testing

Como se ilustra en la figura 17, los resultados demuestran que la herramienta de infraestructura como código con la que más se encuentran familiarizados los desarrolladores es AWS CloudFormation. Adicionalmente, se observa que 28% no reconocen ninguna herramienta de IaC. Cabe resaltar que se evidencia un problema de conceptos ya que en la pregunta acerca de las prácticas de DevOps, solo 4 de los encuestados respondieron que la conocían; sin embargo, en esta pregunta 8 de los encuestados están familiarizados con CloudFormation la cual es una herramienta de infraestructura como código.

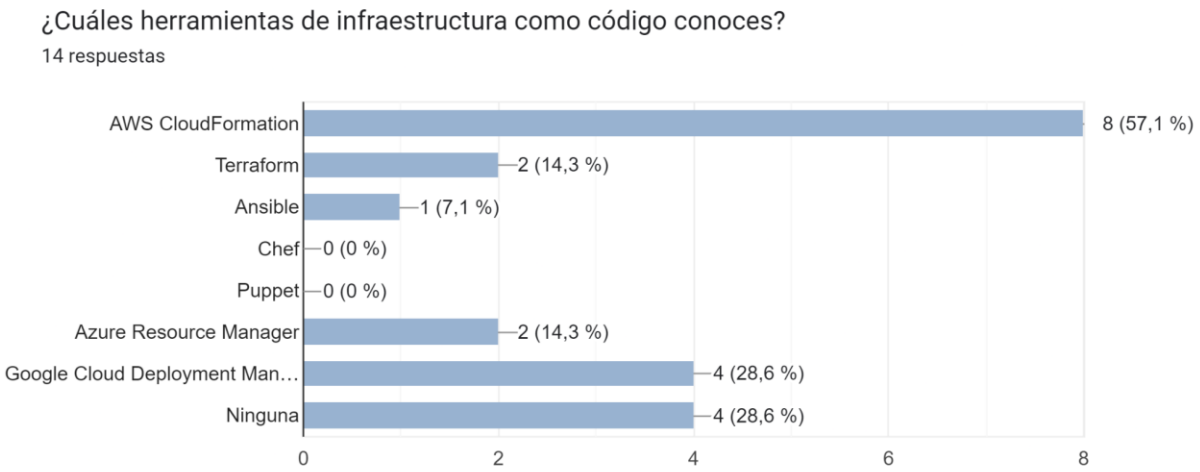

Figura 17. Resultados herramientas de infraestructrura como código

3.3. Resumen de problemáticas identificadas

Basándonos en la información recopilada en este capítulo, se presenta la tabla 1 que resume las problemáticas identificadas en la organización.

Categoría	Problemática identificada
Procesos internos del cliente	Alta burocracia en los procesos de despliegue y gestión de infraestructura por parte de los clientes de la organización, obligando a necesitar ambientes externos a los del cliente para evitar bloqueos en el flujo de desarrollo.
Gestión de Infraestructura	Falta de estandarización y guías para la gestión de infraestructura
	Posibilidad de omitir la centralización de cambios de código de infraestructura en el repositorio remoto
	Inestabilidad y propensión a errores en el manejo de infraestructura
	Despliegue de infraestructura realizando diversos pasos manuales
Despliegue de componentes backend y frontend	Dificultad para configurar ambientes locales con funciones *serverless*, generando así una necesidad de despliegue frecuente.
	Posibilidad de omitir la centralización de cambios de código en el repositorio remoto
	Dificultad para practicar la integración continua
	Realización de procesos manuales repetitivos para despliegue

	que dificultan la manutención de los ambientes de desarrollo.
Conocimiento acerca de DevOps y sus prácticas	Brecha del conocimiento en conceptos y prácticas fundamentales de DevOps por parte de los líderes de TI y desarrolladores

Tabla 1. Resumen problemáticas identificadas en la fase de recolección de información

Capítulo 4 – Diseño del proceso de adopción de DevOps

4.1. Antecedentes

El proceso de adopción de DevOps tomó como referencia el artículo [14] donde se describen 82 actividades, 16 artefactos, 9 roles y se recomiendan 13 herramientas tecnológicas para apoyar y fomentar la adopción de DevOps en las organizaciones.

Para el diseño del proceso propuesto en [14], se realizó un mapeo sistemático de la literatura donde se identificó un alto grado de heterogeneidad en las aproximaciones propuestas para adoptar DevOps, por lo cual fue necesario realizar un proceso de armonización que permitiera identificar de manera ordenada, estructurada y clara el conjunto de elementos del proceso que deben considerar las medianas y pequeñas empresas para implementar DevOps. Adicionalmente, este proceso fue diseñado utilizando el método para definir procesos en empresas desarrolladoras de software propuesto en [13].

El proceso está compuesto por los tres subprocesos de gestión de configuración (GC), integración, entrega y despliegue continuo (CI/CD) y monitoreo continuo (MC). En la Tabla 2, se presenta una breve descripción de los objetivos de cada subproceso.

Subproceso	Descripción
Gestión de la configuración	Establecen las configuraciones necesarias que permitan crear, configurar y desplegar un *pipeline* de integración, entrega, despliegue y monitoreo continuo.
Integración, entrega y despliegue continuo	Sugiere cómo se debe llevar a cabo la ejecución de los *pipelines* de integración, entrega y despliegue continuo.
Monitoreo continuo	Recomienda las actividades para realizar seguimiento continuo al rendimiento de la aplicación que se encuentre desplegada.

Tabla 2.Subprocesos propuestos en el artículo base

Cada subproceso se compone de sus respectivas actividades, las cuales se dividen en actividades fundamentales y complementarias.

- **Actividades fundamentales**: Aseguran que la empresa aplique de forma correcta los subprocesos de gestión de la configuración e integración, entrega y despliegue continuo.

- **Actividades complementarias**: Ayudan a realizar el subproceso de monitoreo continuo y suponen el uso de contenedores, el cual depende de las características y necesidades de la empresa, por lo que estas actividades se consideran opcionales.

En total se proponen 46 actividades fundamentales (Af) y 22 actividades complementarias (Ac). El listado completo y detalle de las actividades para cada subproceso propuesto se pueden consultar en el **Anexo C. Actividades del Proceso para Fomentar la Adopción de DevOps para Pymes**

Adicionalmente, en el proceso se sugiere un total de 16 artefactos que corresponden a cualquier información de entrada o salida que surja de la implementación de los subprocesos y sus respectivas actividades. En el **Anexo D. Artefactos del Proceso para Fomentar la Adopción de DevOps para Pymes**, se presenta el nombre de los artefactos sugeridos y una breve descripción sobre ellos.

Por otro lado, también se definen 9 roles que participan en la ejecución de las diferentes actividades y que forman parte fundamental para que el proceso se pueda llevar a cabo de manera efectiva. Los nombres y descripción de cada rol se encuentran en el **Anexo E. Roles sugeridos del Proceso para Fomentar la Adopción de DevOps para Pymes.**
.

En la Figura 18 se presenta un resumen de la relación e interacción de los diferentes subprocesos donde cada uno está conformado por actividades fundamentales y complementarias, roles, artefactos y herramientas tecnológicas. El proceso inicia con la gestión de la configuración donde se realizan actividades de adquisición y configuración de herramientas e infraestructura que van a permitir soportar e implementar los *pipelines* de CI/CD. Este subproceso propone 14 actividades fundamentales y 15 complementarias, 2 roles y recomienda herramientas tecnológicas como Terraform [36], Chef [37], Ansible [38] entre otras. El siguiente subproceso corresponde a la integración, entrega y despliegue continuo (CI/CD) donde se plantean actividades de desarrollo de los requerimientos, pruebas unitarias, de integración y funcionales, análisis de código, compilación y despliegue a ambientes como QA, preproducción y producción. Este subproceso de CI/CD sugiere 32 actividades fundamentales, 17 actividades complementarias, 5 roles, 6 artefactos y algunas herramientas tecnológicas como Jenkins [33], GitHub [24], SonarQube [39], entre otras. Por último, el subproceso de monitoreo continuo se conecta tanto a la gestión de la configuración como al subproceso de CI/CD en un sentido bilateral ya que se monitorea tanto la infraestructura como las aplicaciones desarrolladas y desplegadas; de esta forma, si durante el subproceso de monitoreo se identifican alertas o fallas, deberían corregirse en los subprocesos ligados a este.

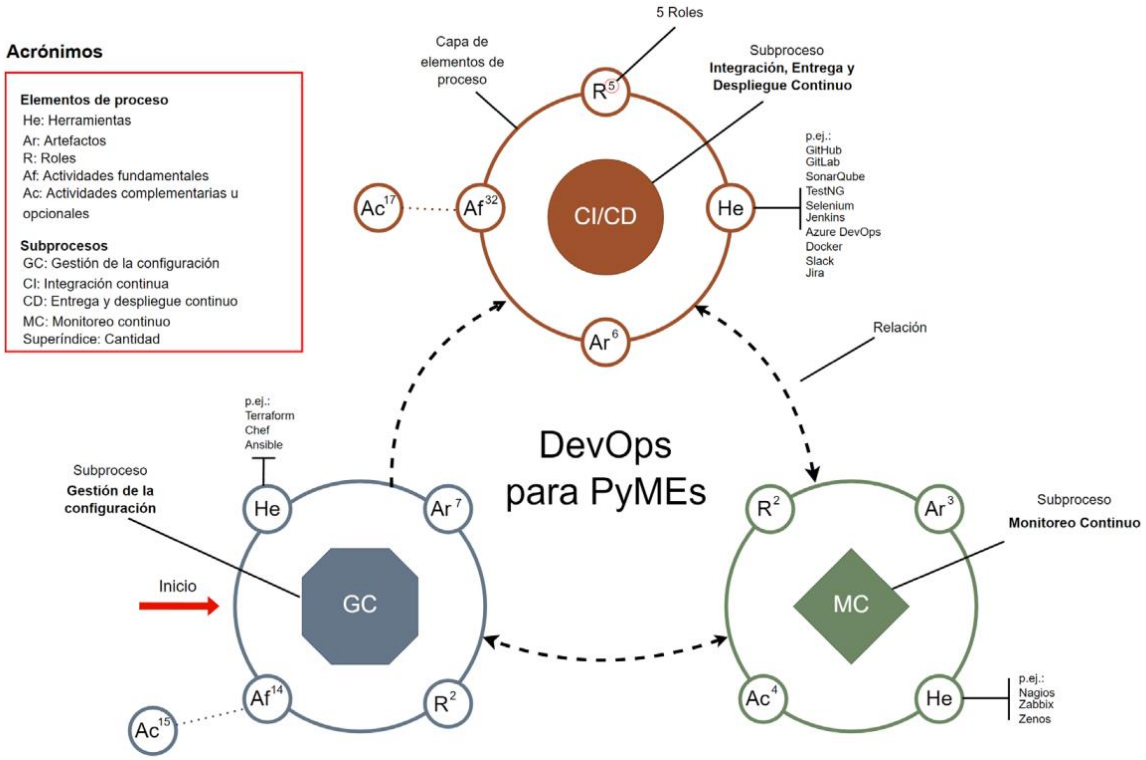

Figura 18. Proceso para apoyar la implementación de DevOps propuesto en [14]

Imagen tomada de [14]

El modelado del proceso e información más detallada se puede consultar en el siguiente link: https://bit.ly/47knDdF

4.2. Adaptación del proceso al contexto de Wizit Mind Blowing Solutions S.A.S

El proceso descrito en la sección anterior fue analizado en conjunto con un experto de la empresa donde se identificó que los roles, actividades y artefactos propuestos no se ajustaban en su totalidad al contexto de la organización; por lo que fue necesario realizar una adaptación del mismo teniendo en cuenta las necesidades de la empresa y de sus clientes.

Para la adaptación del proceso, se llevaron a cabo una serie de sesiones de trabajo con el equipo técnico, donde como primera instancia se realizó un filtrado de los roles, subprocesos, actividades y artefactos; posteriormente, se propusieron modificaciones y se añadieron nuevos componentes de tal forma que el proceso fuera adecuado para el desarrollo de software que se lleva a cabo en la organización.

4.2.1. Filtrado y definición de los roles

De los 9 roles propuestos en el proceso mencionado anteriormente (Anexo E), se descartaron los siguientes roles:

- Coordinador de Liberaciones
- Administrador de Sistemas

La exclusión de estos dos roles se basa en el siguiente criterio:

- **Inexistencia y poca relevancia de acuerdo a los procesos de la organización**: Estas dos figuras, actualmente, no forman parte de los procesos que se llevan a cabo en la organización, y no se considera relevantes incluirlas ya que, se ha determinado que en la práctica y siendo fieles a los principios de agilismo, su presencia podría ocasionar retrasos en la entrega de productos software y en el soporte al cliente.

Por otro lado, el rol de Líder de desarrollo y Arquitecto de Software fueron unidos en un solo rol, esto debido al siguiente criterio:

- **Capacitación del Líder de Desarrollo**: En la actualidad, de acuerdo a la experiencia, conocimiento y preparación en Arquitectura de Software que tiene el líder de desarrollo dentro de la compañía, está capacitado para tomar las decisiones de arquitectura que sean las más convenientes conforme al contexto del producto software en construcción. Adicionalmente, el líder de desarrollo desempeña un papel de mentor para el resto del equipo de desarrollo en materia de arquitectura y buenas prácticas.

Adicionalmente, se ha incorporado el rol del Líder de TI o Líder de Ingeniería, quien actúa como el representante del área técnica y de ingeniería dentro de la organización debido a su amplia experiencia y habilidad en el campo técnico. La inclusión de esta figura se considera fundamental, ya que representa a los expertos de la empresa, y su opinión y gestión se consideran extremadamente relevantes para el proceso de DevOps. Su participación garantiza una visión más completa y estratégica para la implementación de las prácticas de DevOps, asegurando un enfoque sólido y coherente en la integración de desarrollo y operaciones.

En la Tabla 3 se presenta el resumen de los seis roles propuestos y sus responsabilidades teniendo en cuenta el conocimiento contextual que se tiene de la organización.

Roles	Descripción
Arquitecto de software/Líder de desarrollo	Dentro de la organización corresponde al desarrollador con más experiencia y conocimiento dentro del equipo que se encarga de tomar las decisiones de arquitectura debido a su preparación en el campo y establece estándares y herramientas a utilizar. De la misma forma, sirve de guía a los demás desarrolladores del equipo para que todos puedan cumplir con las tareas propuestas.
Desarrollador de software	Encargado de desarrollar los requisitos funcionales y no funcionales que cumplan con las necesidades de negocio de los diferentes clientes de la organización.
Facilitador	Hace referencia a la persona encargada de la gestión del

	equipo y de velar que se alcancen los objetivos del proyecto eliminando los impedimentos que se puedan presentar en el camino. En la mayoría de proyectos de la organización corresponde a la figura del Scrum Master o Project Management.
Ingeniero de pruebas	Encargado de diseñar y ejecutar casos de prueba sobre las funcionalidades desarrolladas a lo largo del proyecto para asegurar que el producto software cumpla con los requerimientos de negocio y los estándares de calidad.
Ingeniero DevOps	Rol que se encarga de fomentar la colaboración y optimización de los procesos que se llevan a cabo durante el ciclo de vida de desarrollo de software ejecutando diversas prácticas y herramientas tecnológicas que guíen en la adopción de DevOps.
Líder de TI	Figura que lidera el área técnica y de ingeniería dentro de la organización, en virtud de su amplia experiencia técnica. Su responsabilidad principal es liderar y orientar al equipo técnico, brindando mentoría y guía para el desarrollo profesional de los integrantes del área.
Usuario final	Usuario o sistema que hará uso del producto software desarrollado.

Tabla 3. Roles propuestos para el proceso de DevOps de la compañía

4.2.2. Adaptación y definición de subprocesos

En el caso de los subprocesos no fue necesario realizar un filtrado o exclusión, sin embargo, se agregó un nuevo subproceso y se realizaron ajustes en el alcance y compromisos de cada subproceso propuesto. En total, se plantearon 4 subprocesos, cuyo detalle se encuentra en la Tabla 4.

Subproceso	Descripción
Adquisición y configuración	Constituye las actividades de elección y adquisición de herramientas y configuración de las mismas que permitan ejecutar pipelines de CI/CD y gestionar la infraestructura de los diferentes proyectos de la organización.
Gestión de infraestructura	Establece como llevar a cabo el aprovisionamiento de los recursos de infraestructura como código y la gestión óptima de estos recursos en el proveedor de servicios de cómputo en la nube.
Integración, entrega y despliegue continuo	Sugiere cómo llevar a cabo la ejecución del pipeline de integración, entrega y despliegue continuo.
Monitoreo continuo	Establece las actividades de monitoreo a recursos de infraestructura y productos software en curso o ya desarrollados dentro de la organización.

Tabla 4. Subprocesos propuestos para el proceso de DevOps de la compañía

4.2.3. Filtrado y definición de actividades

Con base en las actividades propuestas para cada subproceso en [14], se identificó la necesidad de realizar ciertos ajustes para adaptarlas a las particularidades y requerimientos específicos de la organización. Para lograr esto, se trabajó en conjunto con el experto de la organización con el objetivo de construir los criterios de exclusión de las actividades.

- **Criterio de exclusión 1 – Alta granularidad:** Se opta por excluir aquellas actividades que tienen un alto grado de granularidad y que pueden ser simplificadas al fusionarlas en una única actividad más general. El objetivo principal de esta decisión es reducir la complejidad del proceso, buscando una mayor eficiencia y claridad en las etapas involucradas. Al agrupar tareas afines en una sola actividad, se facilita su gestión y se evita una excesiva fragmentación que podría dificultar la comprensión y ejecución del proceso de manera efectiva.

- **Criterio de exclusión 2 – Uso de contenedores:** Dentro del alcance actual para el proceso de DevOps, se han excluido las actividades relacionadas con el uso de contenedores, ya que los esfuerzos están focalizados en arquitecturas *serverless* dado que son más comunes y pertinentes para los proyectos de la empresa. En este contexto, se pretende que las actividades se centren en optimizar y gestionar eficientemente las arquitecturas *serverless* dándole un enfoque específico del proceso de DevOps que se quiere implementar en la organización.

Tipos de Actividades

En este caso se mantuvo el modelo de dos tipos de actividades como en el proceso base, las cuales son:

- **Actividades fundamentales (Af)**: Comprenden parte fundamental de los subprocesos y son necesarias para la ejecución adecuada de los mismos.

- **Actividades opcionales (Ao)**: Aquellas actividades que dependen del contexto de los proyectos, herramientas o prácticas que se estén llevando a cabo.

Es importante destacar que, al definir las actividades para la organización en comparación con el proceso original, se adoptó un enfoque más flexible en cuanto al nombrado y descripción de las mismas. Es decir, las actividades diseñadas específicamente para la organización comienzan a divergir de las actividades planteadas en el proceso original. Esta divergencia se debe a que, durante las sesiones de trabajo conjunto con el experto de la organización para definir las actividades, se priorizó la relevancia, simplicidad y pertinencia de acuerdo a lo requerido por la organización sobre las actividades propuestas inicialmente en el proceso original. El objetivo primordial fue adaptar el proceso de DevOps de manera

más adecuada a las características particulares de la organización, lo que resultó en una versión personalizada y más ajustada a su contexto.

4.3. Proceso de Adopción de DevOps para Wizit Mind Blowing Solutions S.A.S

En esta sección se presenta la descripción del Proceso de Adopción de DevOps para la organización cuyo objetivo es fomentar la inclusión de prácticas DevOps que entre otros beneficios permitan automatizar el despliegue y configuración de infraestructura de las aplicaciones web desarrolladas por la compañía en un entorno de computación en la nube.

4.3.1. Modelado de proceso

A continuación, se presenta el modelado de proceso mediante notación BPMN del Proceso de DevOps para Wizit Mind-Blowing Solutions S.A.S. Por motivos de espacio, el diagrama debe dividirse en varias imágenes. El diagrama completo puede visualizarse en https://bit.ly/43V2oMy

Como se puede apreciar en la Figura 19, el proceso comienza con la verificación de si el subproceso de Adquisición y Configuración ha sido previamente ejecutado, dado que este subproceso se realiza una sola vez con el propósito de obtener las herramientas configuradas para aplicar DevOps. En caso de que este subproceso no se haya llevado a cabo, el Líder de TI e Ingeniero DevOps proceden a su ejecución, lo cual resulta en la obtención de las herramientas configuradas necesarias para implementar DevOps.

Una vez completado este paso, el Analista de requisitos avanza a la fase de análisis, definición y refinamiento de los requisitos del sprint, teniendo en cuenta el marco de trabajo SCRUM. Durante esta etapa, se obtiene una lista detallada y una descripción exhaustiva de los requisitos funcionales, no funcionales y los requisitos de infraestructura que se deben cumplir en el desarrollo del proyecto.

Posteriormente, como se muestra en la Figura 20, el Ingeniero de Pruebas se encarga de planificar y crear casos de prueba basados en los requerimientos obtenidos en la fase anterior. De manera paralela, el Arquitecto de Software/Líder de Desarrollo realiza una validación para determinar si existen nuevos requerimientos de arquitectura. En caso afirmativo, se inicia el subproceso de Definición de Arquitectura, donde se actualiza o define el Diagrama de Arquitectura y se establecen los patrones de arquitectura a aplicar. Una vez completado lo anterior, el Ingeniero DevOps procede con la ejecución del subproceso de Gestión de Infraestructura, que tiene como resultado la creación de la pila de recursos en la nube necesarios para el proyecto. Ambos flujos de trabajo convergen en el Subproceso de Integración, Entrega y Despliegue Continuo.

Conforme se muestra en la Figura 21, el proceso avanza con la ejecución del subproceso de Entrega y Despliegue Continuo, llevado a cabo por el Ingeniero DevOps y el Equipo de Desarrollo. En esta etapa, se desarrollan y liberan nuevas

versiones del producto software de manera eficiente, utilizando automatización y siguiendo buenas prácticas de desarrollo. Finalmente, se lleva a cabo de manera cíclica el subproceso de Monitoreo Continuo, el cual supervisa toda la arquitectura desplegada en el entorno de computación en la nube.

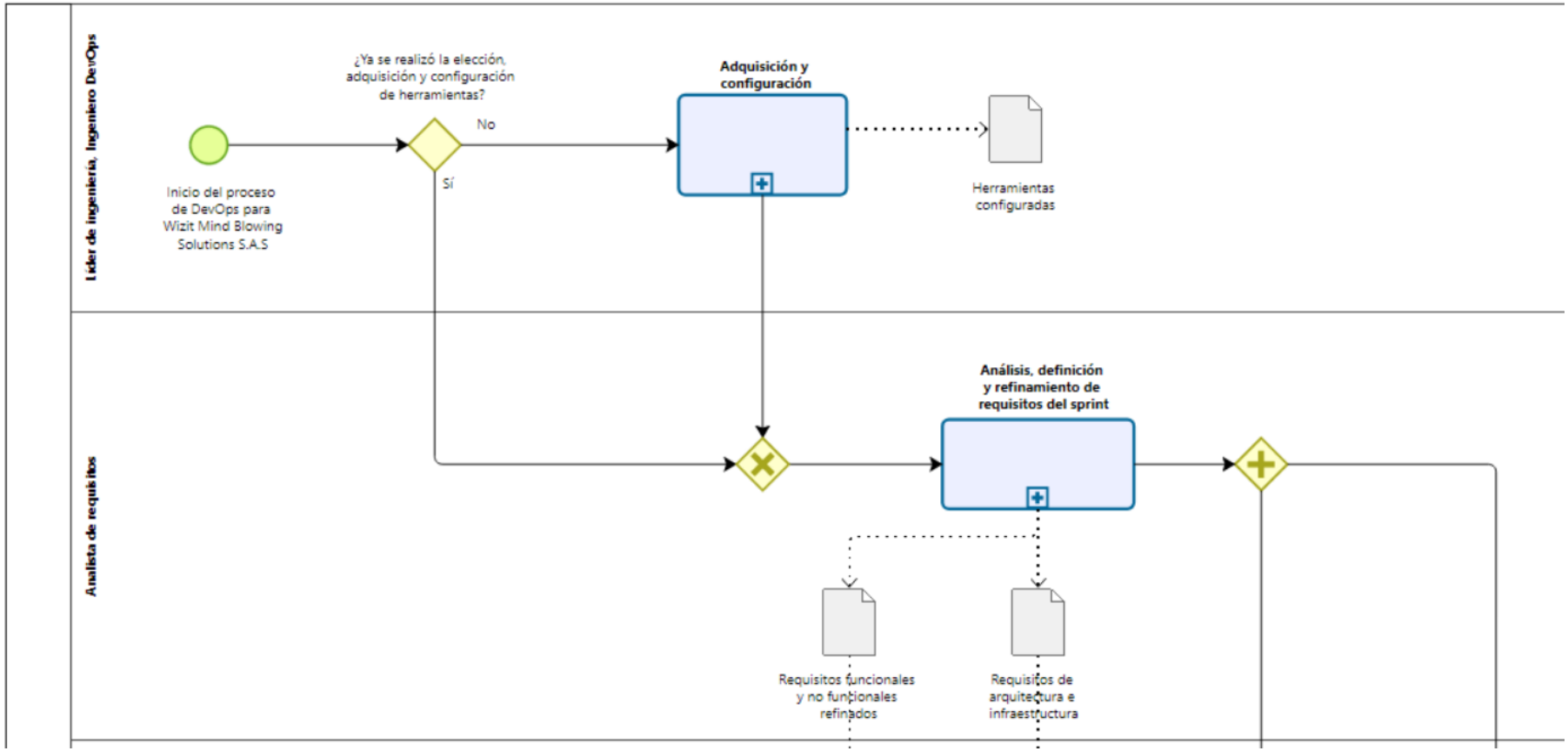

Figura 19. Primera parte del modelado de proceso de DevOps propuesto para la organización

Nota: Es importante destacar que por motivos de simplicidad del modelado general se omitieron o resumieron los artefactos procedentes de la ejecución de los subprocesos.

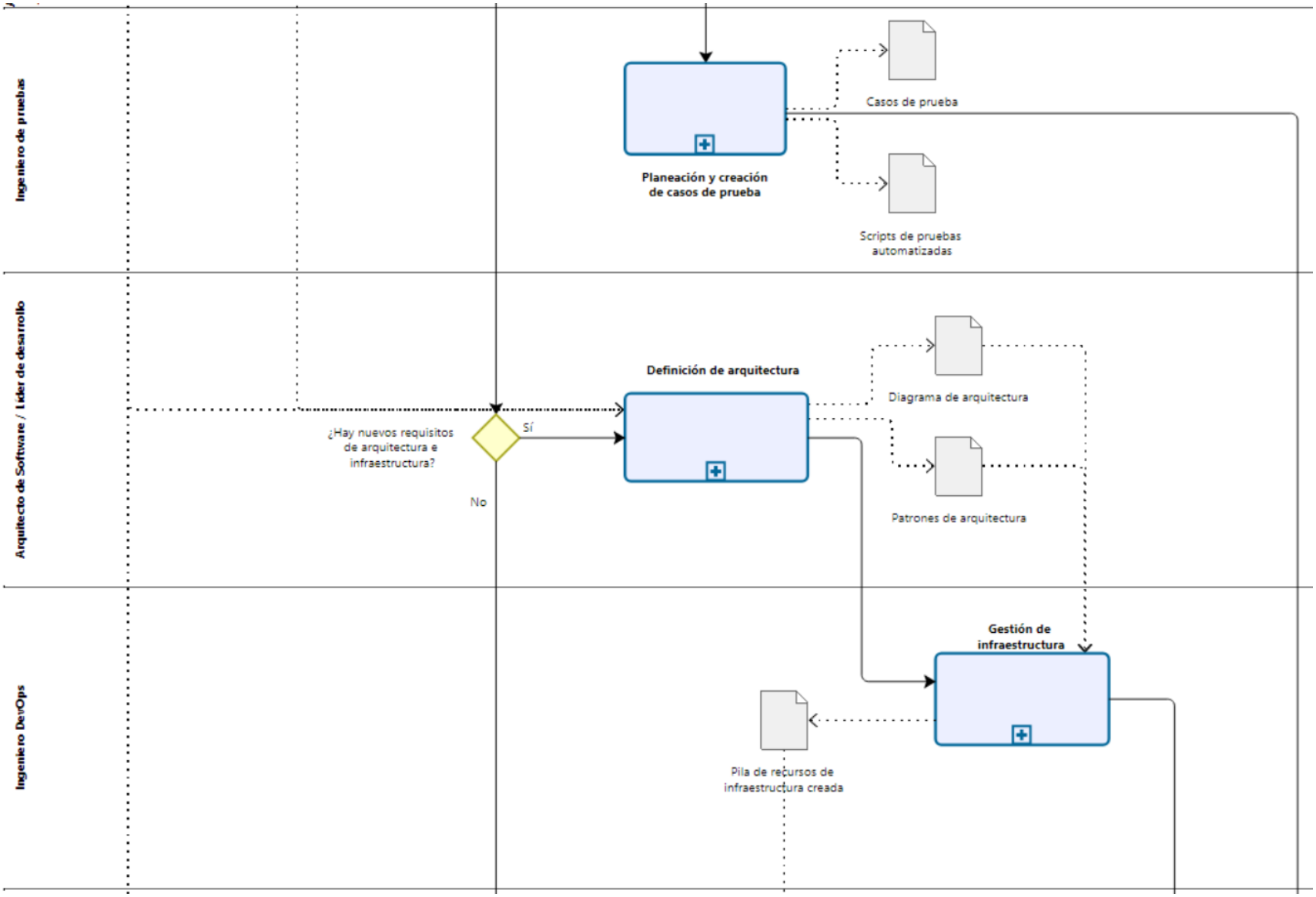

Figura 20. Parte 2 del modelado de proceso de DevOps propuesto para la organización

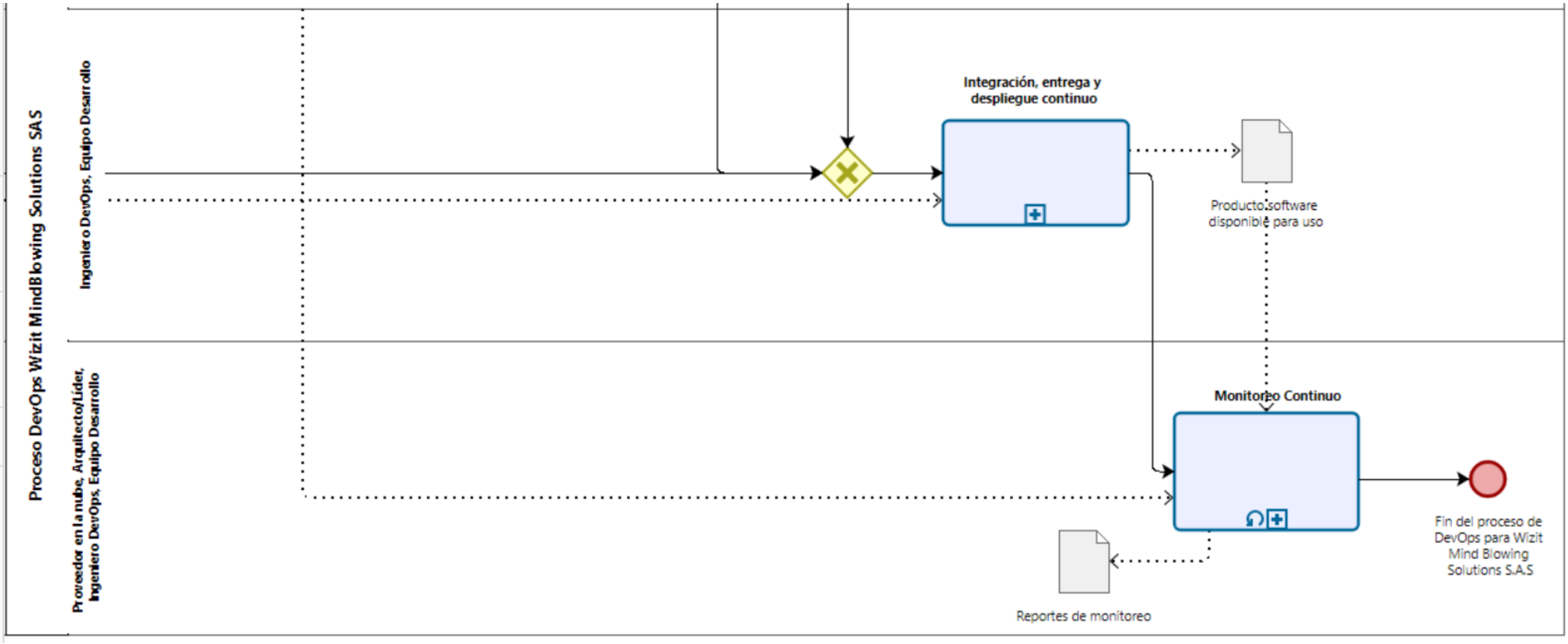

Figura 21. Parte 3 del modelado de proceso de DevOps propuesto para la organización

En las secciones subsecuentes se describirán de manera más detallada los subprocesos propuestos para implementar DevOps en la organización.

4.3.2. Subproceso de Adquisición y Configuración.

El subproceso Adquisición y Configuración constituye una agrupación de actividades que se llevan a cabo una sola vez durante la implementación de DevOps. Su objetivo radica en establecer las herramientas necesarias y configurar adecuadamente los recursos para llevar a cabo las operaciones de DevOps en la organización. Es importante señalar que, a medida que la industria evoluciona con el tiempo, es posible explorar nuevas herramientas y tecnologías para aplicar DevOps, especialmente para pequeñas y medianas empresas (PYMEs) como es el caso, donde se tiene la flexibilidad de adaptarse a las demandas cambiantes del mercado.

4.3.2.1. Modelado de proceso para el subproceso de Adquisición y Configuración

En la Figura 22, se presenta el modelado de proceso mediante notación BPMN que describe las actividades e interacción de este subproceso.

El subproceso comienza con una interacción conjunta entre el Líder de TI y el Ingeniero DevOps, quienes definen las herramientas y proveedores en la nube que serán considerados para la elección posterior. De manera secuencial, ambos roles establecen los criterios que se deberán tener en cuenta en dicha selección.

Una vez que se ha establecido el listado de herramientas y proveedores elegibles, y se han definido los criterios para la elección, el líder de TI, el Ingeniero DevOps y los Desarrolladores de Software trabajan en conjunto y de manera paralela para realizar la selección del proveedor en la nube y las herramientas que serán aplicadas en el proceso de DevOps.

Paralelamente, el líder de TI se encarga de la gestión para adquirir las herramientas y el proveedor seleccionado. En esta actividad el líder de TI puede incluir a demás roles de la organización con el propósito de evaluar y establecer acuerdos, licencias y contratos necesarios para asegurar una adquisición conveniente y exitosa.

Una vez adquiridas las herramientas y el proveedor, se procede a realizar las configuraciones necesarias para su correcto funcionamiento, incluyendo las herramientas de Integración Continua y Entrega Continua (CI/CD). Asimismo, se configuran aquellas herramientas que requieran ajustes específicos para adaptarse al entorno de trabajo.

Con todas estas acciones completadas, el subproceso de Adquisición y Configuración llega a su fin, y se da paso a la siguiente etapa en la implementación de DevOps en el proyecto.

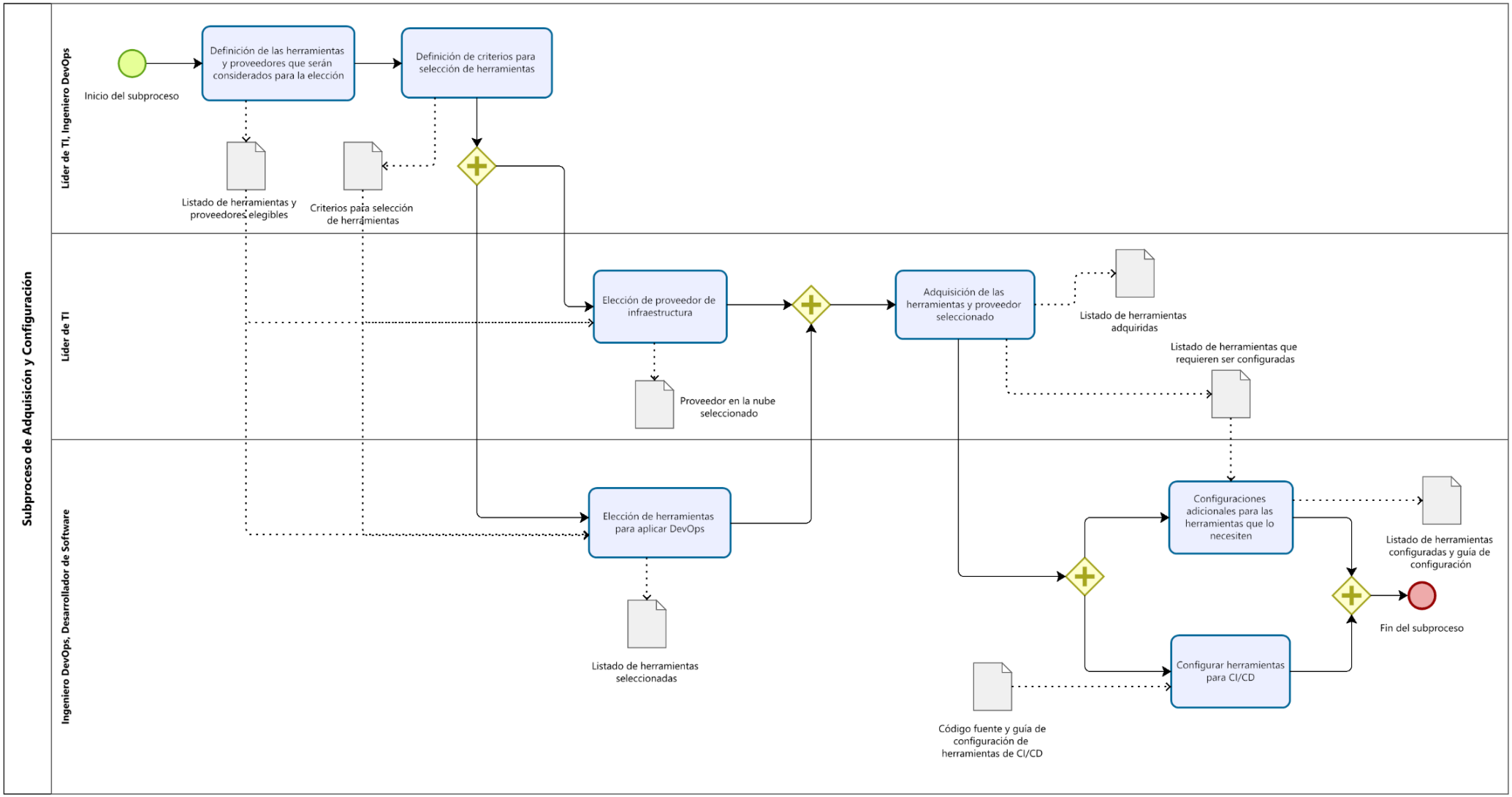

Figura 22. Modelado de proceso para el subproceso de Adquisición y Configuración

4.3.2.2. Actividades y artefactos del subproceso de Adquisición y Configuración

En la Tabla 5 se proporciona un listado de las actividades sugeridas en este subproceso, así como su tipo. Es importante destacar que la descripción detallada de cada actividad y su ejecución se presentarán en el capítulo 5, donde se aborda la implementación del proceso de DevOps específico para la organización.

Subproceso	Actividad	Tipo	
		Af	Ao
Adquisición y configuración	AC-1. Definición de las herramientas y proveedores que serán considerados para la elección	X	
	AC-2. Definición de los criterios para la selección de herramientas	X	
	AC-3. Elección de proveedor de infraestructura	X	
	AC-4. Elección de herramientas para aplicar DevOps	X	
	AC-5. Adquisición de las herramientas y proveedor seleccionado	X	
	AC-6. Configurar herramientas para CI/CD	X	
	AC-7. Configuraciones adicionales para las herramientas que lo necesiten		X

Tabla 5. Actividades del subproceso de adquisición y configuración

En la siguiente tabla se presenta el listado de los artefactos con su respectiva descripción del subproceso de Adquisición y Configuración.

Subproceso	Artefacto	Descripción
Adquisición y configuración	Listado de herramientas y proveedores elegibles	Lista de herramientas y proveedores que serán consideradas para aplicar DevOps de acuerdo a las tecnologías emergentes que se consideran pertinentes para la organización.
	Criterios para la selección de herramientas	Criterios de selección de las herramientas para aplicar DevOps que deberán tenerse en cuenta para hacer una elección acertada y acorde a las necesidades de la organización
	Listado de herramientas y proveedores seleccionados	Lista de herramientas y proveedores que fueron elegidos para aplicar prácticas DevOps en la organización y que serán adquiridas

	Listado de herramientas adquiridas	Lista de las herramientas efectivamente adquiridas a partir de la selección de las herramientas
	Listado de herramientas que requieren ser configuradas	Lista de herramientas que necesitan ser pre configuradas
	Código fuente y guía de configuración de herramientas de CI/CD	Repositorio donde se almacena el código fuente y guía de configuración para las herramientas de CI/CD que así lo hayan requerido.
	Herramientas adquiridas y configuradas	Herramientas que han sido adquiridas y han pasado por un proceso de configuración.

Tabla 6. Artefactos subproceso de Adquisición y configuración

4.3.3. Subproceso de Gestión de Infraestructura

El subproceso de gestión de infraestructura surge de la agrupación y definición de actividades que suponen la administración de los recursos de infraestructura en la nube durante el desarrollo de los diversos proyectos de la organización. En este punto es relevante mencionar que las actividades de gestión de infraestructura que se sugieren en este apartado tienen el enfoque de Infraestructura como Código, donde se busca que el manejo de la infraestructura se realice de una manera eficiente permitiendo crearla, replicarla y desecharla fácilmente. Adicionalmente, se proponen actividades que automatizan el despliegue de la infraestructura de tal forma que se reduzca la intervención humana con el fin de aumentar la productividad y disminuir la propensión a fallos que se tenía en el manejo de infraestructura inicialmente en la organización.

4.3.3.1. Modelado de proceso para el subproceso de Gestión de Infraestructura

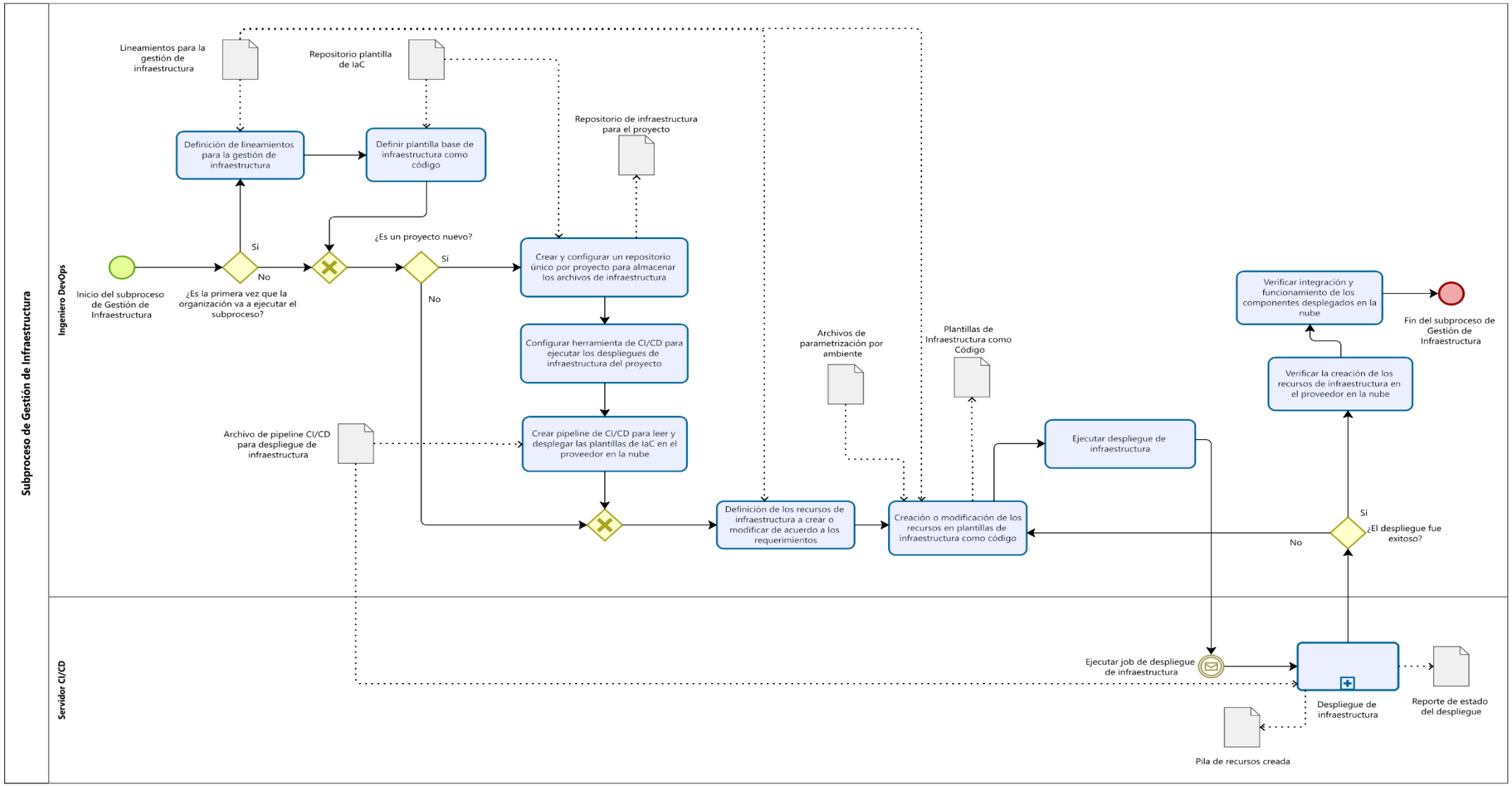

Figura 23. Modelado de proceso Gestión de Infraestructura

En el diagrama de proceso presentado en la Figura 23, la interacción inicia con una compuerta exclusiva donde se evalúa si es la primera vez que se ejecuta el proceso. Esto es crucial para verificar la existencia de lineamientos y plantillas base destinadas a la gestión de infraestructura. En caso de que sea la primera ejecución del subproceso, el Ingeniero DevOps debe establecer los lineamientos que regirán dicha Gestión de Infraestructura. El objetivo de esto es lograr una estandarización que permita configurar la infraestructura de manera uniforme y eficiente bajo un marco común. De esta manera, se busca alcanzar una mayor productividad, consistencia y eficiencia en todo el proceso.

Posteriormente, se evalúa si es el proyecto es nuevo, esto es relevante para determinar si se requieren configuraciones específicas de la herramienta de despliegue (CI/CD) para el nuevo proyecto. En caso afirmativo, se procede a crear el repositorio de código fuente, configurar la herramienta de CI/CD y elaborar el archivo del pipeline que guiará los pasos a seguir para el despliegue de la infraestructura.

Una vez se hayan realizado las configuraciones previas, si fuesen necesarias, el Ingeniero DevOps debe determinar los recursos de la nube que serán creados o modificados, teniendo en cuenta los requerimientos de arquitectura. Posteriormente, procederá a la implementación de dichos recursos utilizando plantillas de infraestructura como código.

Cuando se han completado todos los cambios necesarios se procederá al despliegue de los recursos en el proveedor de la nube. El evento de inicio del despliegue dependerá del tipo de disparador que haya sido definido en la configuración de la herramienta de CI/CD. Entre los eventos más comunes que pueden actuar como disparadores se encuentran los siguientes:

- **Disparador manual**: Este método implica que el Ingeniero DevOps inicie el despliegue desde la interfaz gráfica del servidor de CI/CD, generalmente mediante un botón o una acción manual.

- **Disparador automático utilizando git:** Se pueden configurar disparadores para acciones específicas de git, como push y merge a una rama. La ventaja de este enfoque es que cualquier cambio realizado en el repositorio de código fuente desencadena automáticamente el despliegue por parte del servidor de CI/CD. Esto agiliza el proceso y asegura que las actualizaciones sean implementadas de manera oportuna

- **Disparador temporizado:** En este caso, se establece una hora y frecuencia específicas para que se ejecute el despliegue automáticamente. Sin embargo, para el caso de este subproceso, no es recomendable este disparador, ya que está sujeto a posibles esperas que podrían retrasar la entrega de las implementaciones.

El servidor de CI/CD recibe la señal de que debe realizar el despliegue y ejecuta el subproceso de despliegue de infraestructura, cuya representación bajo notación BPMN se puede observar en la siguiente figura:

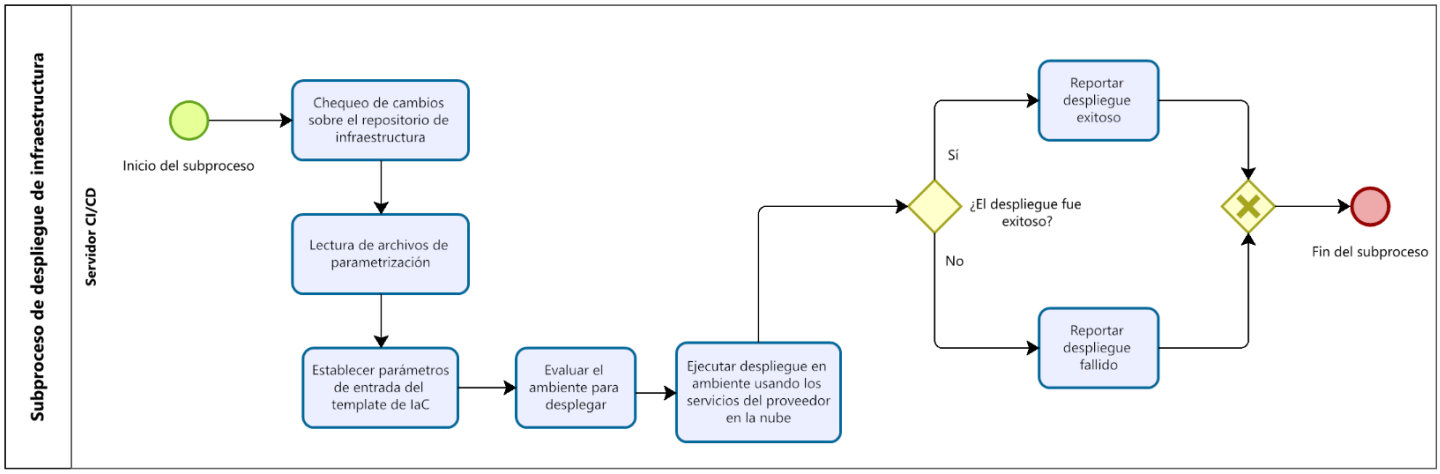

Figura 24. Diagrama del subproceso de despliegue de infraestructura

Tras la ejecución del subproceso de despliegue, el Ingeniero DevOps evalúa si este ha sido exitoso o ha fallado, basándose en el reporte generado por el propio subproceso de Despliegue de Infraestructura. En caso de que el despliegue haya fallado, es responsabilidad del Ingeniero DevOps realizar las correcciones o modificaciones necesarias sobre las plantillas de infraestructura como código, tantas veces como sea requerido, hasta lograr un despliegue exitoso.

Cuando se obtiene un reporte de despliegue exitoso, el Ingeniero DevOps procede a verificar la creación o modificación de los recursos de infraestructura en la nube. Además, se lleva a cabo una verificación del correcto funcionamiento de dichos recursos. Este paso de verificación es crucial para asegurar que todo se encuentre en buen estado y en concordancia con los requerimientos establecidos. Solo después de completar exitosamente estas etapas, el subproceso de Gestión de Infraestructura llega a su conclusión.

4.3.3.2. Actividades y artefactos del subproceso de Gestión de Infraestructura

En la Tabla 7 se proporciona un listado de las actividades sugeridas en este subproceso, así como su tipo. Es importante destacar que la descripción detallada de cada actividad y su ejecución se presentarán en el capítulo 5, donde se aborda la implementación del proceso de DevOps específico para la organización.

Subproceso	Actividad	Tipo	
		Af	Ao
	GI-0. Definición de lineamientos para la gestión de infraestructura		X
	GI-1. Definir una plantilla base de infraestructura como código para ser compartida con los demás colaboradores		X
	GI-2. Crear y configurar un repositorio único por proyecto para	X	

Gestión de infraestructura	almacenar los archivos de gestión de infraestructura.		
	GI-3. Configurar herramienta de CI/CD para ejecutar los despliegues de infraestructura incluyendo origen y disparador del despliegue	X	
	GI-4. Crear pipeline de CI/CD para leer y desplegar las plantillas de IaC en el proveedor en la nube	X	
	GI-5. Definir los recursos de infraestructura que se deben crear o modificar de acuerdo a los requerimientos de arquitectura e infraestructura definidos en etapas anteriores	X	
	GI-6. Crear o modificar los recursos de infraestructura en forma de plantillas de Infraestructura como Código.	X	
	GI-7. Ejecutar despliegue de infraestructura mediante eventos (push en rama especificada o disparador manual)	X	
	GI-8. Despliegue de infraestructura	X	
	GI-8A. Chequeo de cambios sobre el repositorio de infraestructura	X	
	GI-8B. Lectura de archivos de parametrización	X	
	GI-8C. Establecer parámetros de entrada del template de IaC	X	
	GI-8D. Evaluar el ambiente a desplegar	X	
	GI-8E. Ejecutar despliegue usando los servicios del proveedor en la nube	X	
	GI-8F. Reportar estado del despliegue (fallido o exitoso)	X	
	GI-9. Verificar la creación de los recursos de infraestructura en el proveedor en la nube	X	
	GI-10. Verificar integración y funcionamiento de los componentes desplegados en la nube		X

Tabla 7. Actividades del subproceso Gestión de Infraestructura

En la siguiente tabla se presenta el listado de los artefactos con su respectiva descripción del subproceso de Gestión de Infraestructura.

Subproceso	**Artefacto**	**Descripción**	**Tipo**	
			Entrada	**Salida**
	Diagrama de arquitectura	Diagrama que define la interacción entre los componentes y recursos de infraestructura del sistema.	x	

Gestión de la infraestructura	Requisitos de infraestructura	Listado de los requisitos que se necesitan aprovisionar dentro de la infraestructura.	x	
	Archivos de parametrización por ambiente	Ficheros que contienen parámetros de entrada por cada ambiente específico como nombres de recursos, referencias a otros recursos, etc.	x	
	Plantillas de Infraestructura como código	Archivo de configuración con la definición en código de los recursos de infraestructura requeridos		x
	Archivo que describe pipeline de despliegue de infraestructura	Archivo de definición de los pasos que se deben ejecutar de forma automática para el despliegue de los recursos definidos en la plantilla de IaC.		x
	Pila de recursos creada	Conjunto de recursos creados en el proveedor en la nube.		x

Tabla 8. Artefactos subproceso Gestión de Infraestructura

4.3.4. Subproceso de Integración, entrega y despliegue continuo

El subproceso de integración, entrega y despliegue continuo comprende un conjunto de actividades fundamentales que dirigen y agilizan el ciclo de vida del desarrollo del software, con el objetivo primordial de proporcionar valor de manera eficiente y rápida. Las actividades de este subproceso abarcan desde las configuraciones iniciales, donde se prepara el entorno de trabajo, hasta la codificación, en la que se construye el software siguiendo los requisitos establecidos para posteriormente realizar pruebas exhaustivas que garanticen la calidad y funcionalidad del producto, y finalmente, procede con el despliegue del software en los distintos entornos hasta producción, poniéndolo a disposición de los usuarios finales. Al integrar y automatizar estos pasos, se logra una entrega de valor continua, facilitando la adaptación a los cambios y manteniendo una alta eficiencia en todo el proceso de desarrollo.

Entre los puntos relevantes a mencionar sobre este subproceso se encuentran que:

- Las actividades propuestas para este subproceso contemplan el uso del marco de trabajo SCRUM como ya es estándar por parte de la organización.

- Se sugiere el uso del flujo de trabajo GitFlow [40] para el manejo de ramas en Git.

- Se propone como práctica de desarrollo el desarrollo basado en pruebas (*Test driven development - TDD)* [41] donde se codifican primero las pruebas unitarias y luego se desarrollan los requerimientos con el propósito de crear código de mejor calidad.

- Se sugiere la obligatoriedad de la revisión de código (*code review*) por parte de los demás colaboradores del proyecto (o externos al proyecto) con el

propósito de impartir conocimiento y promover las buenas prácticas en el proceso de codificación.

- Se plantea que como parte de la práctica de Integración Continua, que los desarrolladores estén constantemente integrando su código a los repositorios.

- Se automatiza el despliegue de los componentes backend y frontend de los proyectos

4.3.4.1. Modelado del subproceso de Integración, entrega y despliegue continuo

La Figura 25 y 27 muestran el modelado del subproceso de Integración, Entrega y Despliegue Continuo.

El subproceso comienza con una pregunta clave: ¿es un nuevo proyecto? En caso afirmativo, el desarrollador de software crea y configura los repositorios para almacenar el código fuente del proyecto. Luego, el Ingeniero DevOps entra en acción y configura los trabajos en la herramienta de CI/CD específicamente para estos nuevos repositorios, además de crear el archivo pipeline de CI/CD con los pasos necesarios para el despliegue.

A continuación, el desarrollador de software analiza los requerimientos a desarrollar y comienza a crear las pruebas unitarias bajo el enfoque de Desarrollo basado en pruebas. Una vez finalizado el análisis, se procede con el desarrollo de los respectivos requerimientos, gestionados en el repositorio según el flujo de trabajo gitflow.

Al culminar el desarrollo, se agregan los cambios al repositorio remoto, lo que desencadena el evento para ejecutar el pipeline de CI/CD, permitiendo el despliegue de los componentes, ya sea frontend o backend, según sea necesario.

El despliegue de componentes corresponde a un subproceso destacado en la Figura 26. En este proceso, el servidor de CI/CD lleva a cabo una serie de pasos cruciales. En primer lugar, se verifica si hay cambios en el repositorio remoto y, en caso de encontrarlos, se realiza un análisis estático del código para generar un informe detallado de las incidencias encontradas, que pueden incluir code-smells, vulnerabilidades de seguridad, complejidad ciclomática, cobertura de pruebas unitarias, entre otros.

A continuación, se ejecutan las pruebas unitarias para asegurar la calidad y funcionalidad del código. Si estas pruebas arrojan algún fallo, se reporta el despliegue como fallido. En cambio, si las pruebas unitarias se completan exitosamente, se procede al despliegue de los componentes en el entorno de computación en la nube, y se reporta el estado del despliegue.

En caso de que el despliegue falle en el entorno de desarrollo (dev), el desarrollador debe identificar las causas del fallo y realizar los ajustes necesarios en el código para intentar el despliegue nuevamente. Una vez resuelto el problema, se genera una versión inmutable del código para llevar los cambios al entorno de no producción (nonprod).

Luego, el ingeniero DevOps procede a realizar el despliegue en el entorno de nonprod utilizando la herramienta de CI/CD, especificando la versión del tag correspondiente. Una vez que el despliegue se ha realizado, los ingenieros de prueba pueden comenzar a ejecutar los casos de prueba relacionados con las funcionalidades entregadas en el entorno de nonprod. Se genera un reporte con los resultados de las pruebas, y en caso de encontrar errores o bugs, los desarrolladores deben realizar las correcciones necesarias y enviar nuevas versiones para volver a ser probadas.

Cuando la versión ha sido aprobada por el equipo de calidad (QA), se solicita al ingeniero DevOps que proceda con el despliegue de los artefactos de código en el entorno productivo. Este proceso garantiza que el código ha sido probado y validado adecuadamente antes de ser implementado en el entorno de producción, lo que ayuda a reducir riesgos y asegura que solo las versiones estables y probadas sean desplegadas para el uso final de los usuarios.

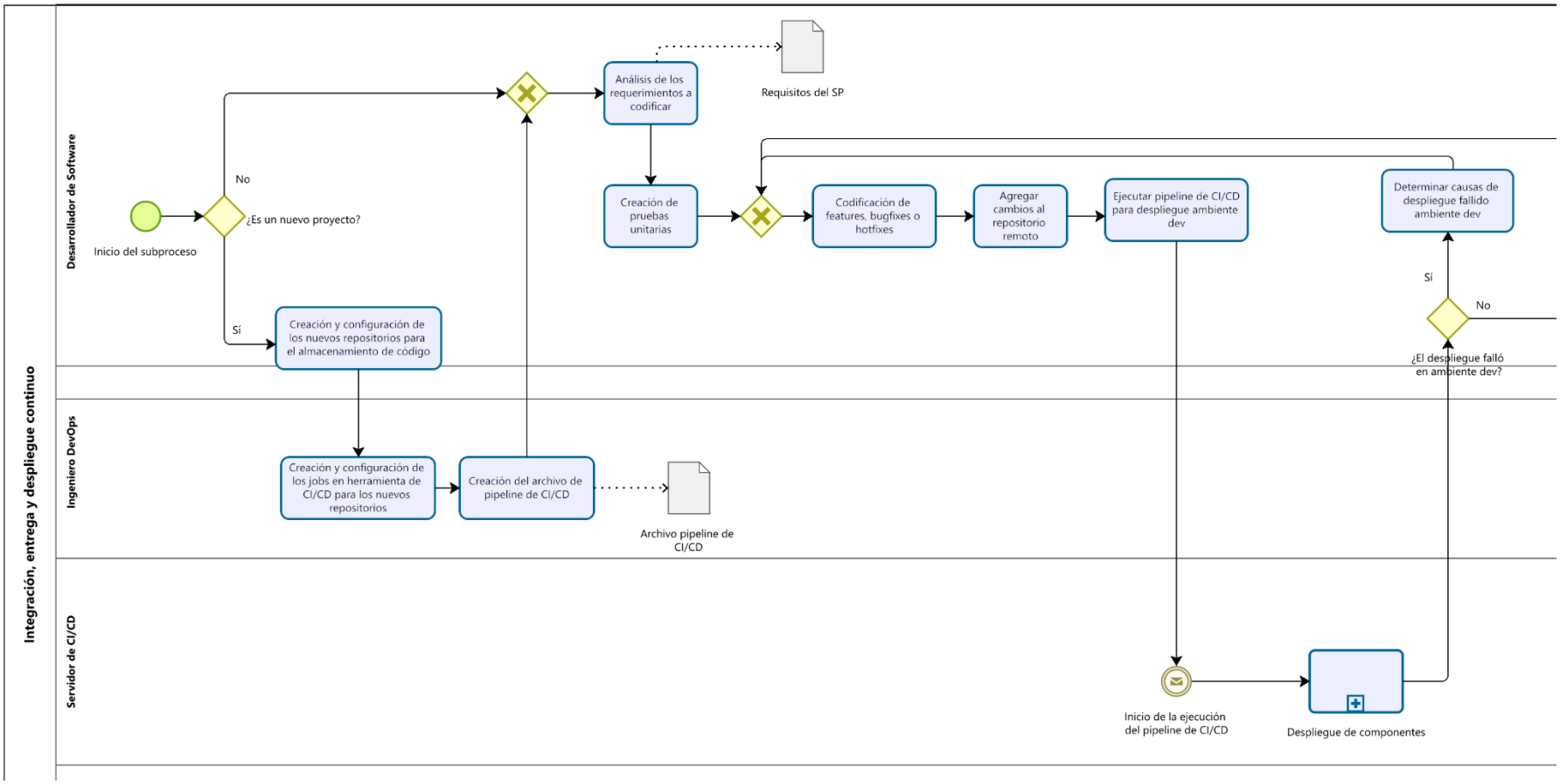

Figura 25. Parte 1 Diagrama de proceso Integración, entegra y despliegue continuo

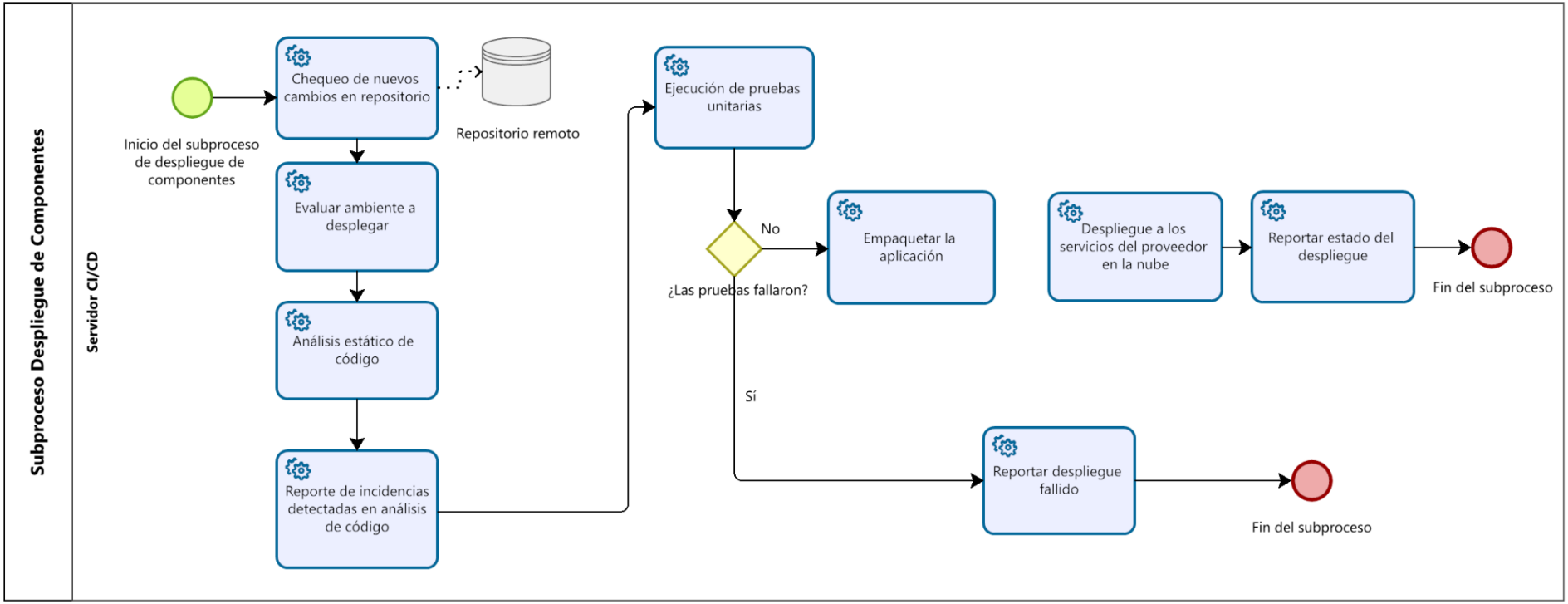

Figura 26. Subproceso de despliegue de componentes

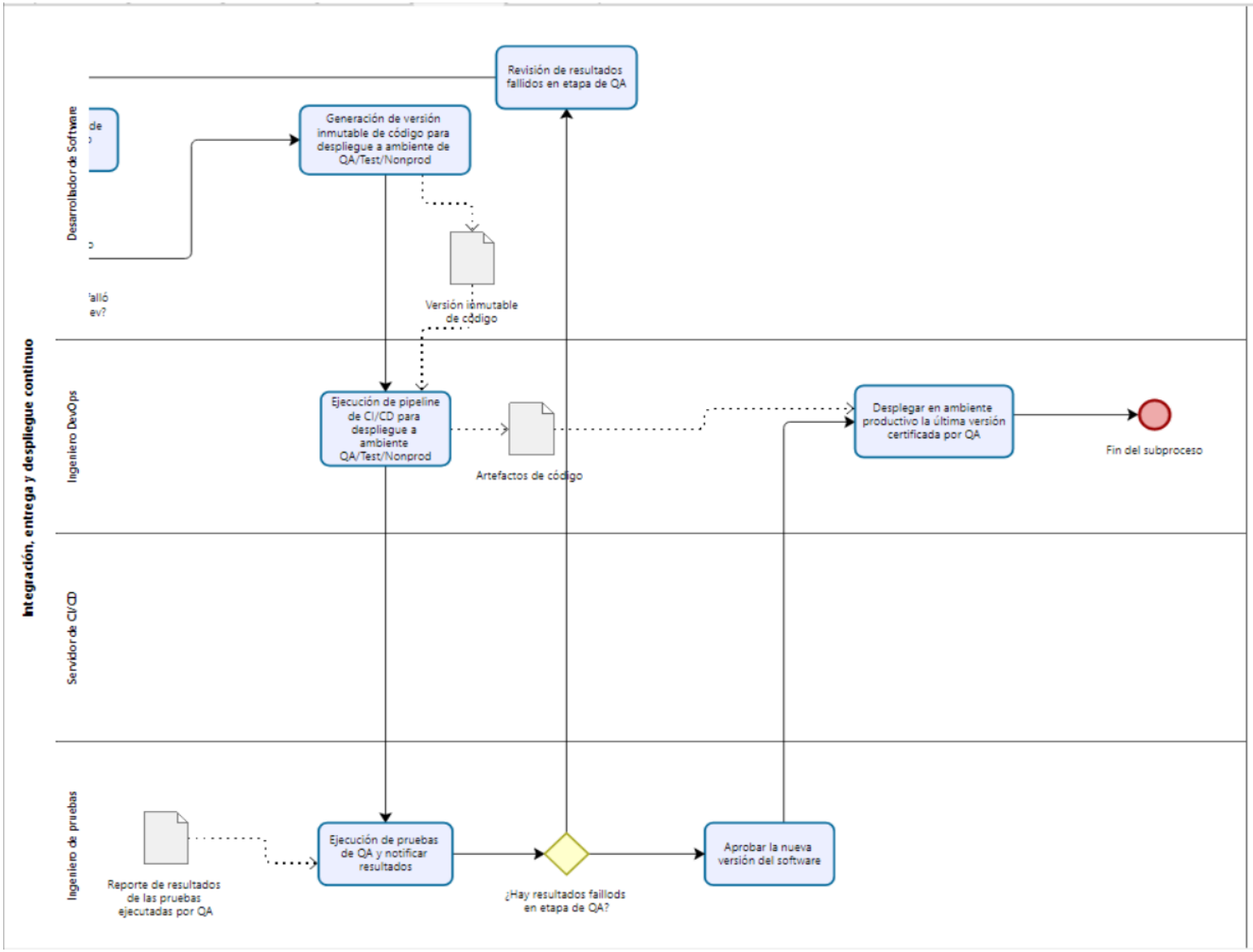

Figura 27. Parte 2 Diagrama de proceso Integración, entrega y despliegue continuo

4.3.4.2. Actividades y artefactos del subproceso de Integración, entrega y despliegue continuo

En la Tabla 9 se proporciona un listado de las actividades sugeridas en este subproceso, así como su tipo. Es importante destacar que la descripción detallada de cada actividad y su ejecución se presentarán en el capítulo 5, donde se aborda la implementación del proceso de DevOps específico para la organización.

Subproceso	Actividad	Tipo	
		Af	Ao
Integración, entrega y despliegue continuo	CICD-1. Creación y configuración de los nuevos repositorios de un proyecto.	X	
	CICD-2. Creación y configuración de los jobs en herramienta de CI/CD para los nuevos repositorios	X	
	CICD-3. Creación del archivo de pipeline de CI/CD que guían la automatización del despliegue para cada repositorio del proyecto teniendo en cuenta las tecnologías y tipo de componente a desplegar (backend y frontend)	X	
	CICD-4. Análisis de los requerimientos a codificar	X	
	CICD-5. Creación de pruebas unitarias	X	
	CICD-6. Codificación de features, bugfixes o hotfixes según corresponda	X	
	CICD-7. Agregar cambios al repositorio remoto	X	
	CICD-8. Ejecutar pipeline de CI/CD para despliegue ambiente dev	X	
	CICD-8a. Chequeo de nuevos cambios en repositorio	X	
	CICD-8b. Análisis estático de código		x
	CICD-8c. Reporte de incidencias detectadas en análisis de código		x
	CICD-8d. Empaquetar la aplicación	X	
	CICD-8e. Despliegue a los servicios del proveedor en la nube	X	
	CICD-8f. Reporte de estado del despliegue (fallido o exitoso)	X	
	CICD-9. Generación de versión inmutable de código para despliegue a ambiente de QA/Test/Nonprod		X

	CICD-10. Ejecución de pipeline de CI/CD para despliegue a ambiente QA/Test/Nonprod	X	
	CICD-11. Ejecución de pruebas de QA y notificar resultados	X	
	CICD-12. Revisión y solución de resultados fallidos en etapa de QA	X	
	CICD-13. Aprobar la nueva versión del software	X	
	CICD-14. Desplegar en ambiente productivo la última versión certificada por QA	X	

Tabla 9. Actividades subproceso de Integración, entrega y despliegue continuo

A continuación, en la Tabla 10 se presenta el listado de los artefactos que utiliza el subproceso de Integración, entrega y despliegue continuo, con su respectiva descripción y tipo.

Subproceso	Artefacto	Descripción	Tipo	
			Entrada	Salida
Integración, entrega y despliegue continuo	Especificación de requisitos a codificar	Especificación de los requisitos en forma de Historias de Usuario que se van a codificar durante el ciclo de desarrollo ágil.	X	
	Reporte de los resultados de las pruebas unitarias	Informe de las pruebas unitarias exitosas y fallidas.		X
	Versión inmutable de código	Punto específico dentro de un repositorio que marca una nueva versión o release del proyecto. Ejemplo: tags de Git.		X
	Reporte de resultados de las pruebas ejecutadas por QA	Reporte de la ejecución de casos de prueba por parte del equipo de QA.		X
	Artefactos de código	Comprimido de los archivos de código fuente		X

Tabla 10. Artefactos subproceso Integración, entrega y despliegue continuo

4.3.5 Subproceso de Monitoreo Continuo

Como su nombre lo indica el subproceso de monitoreo continuo se compone de actividades de monitoreo sobre los productos software desarrollados o en proceso de desarrollo y la infraestructura que soporta esos productos con el objetivo de identificar riesgos y fallos que puedan afectar la estabilidad y costos de las aplicaciones.

4.3.5.1. Modelado del proceso y actividades

La Figura 28 muestra el modelado del proceso para el subproceso de monitoreo continuo. En este caso, el proceso se inicia con la ejecución paralela del monitoreo continuo de los componentes software desplegados en la nube en los diversos entornos. Si se produce un error en algún componente software, como el fallo de un servicio interno o un error que afecte el funcionamiento de los servicios, el error se registra y reporta.

A continuación, el equipo de desarrollo debe identificar e informar la causa del error en el componente software y regresar al subproceso de Integración, Entrega y Despliegue Continuo para resolverlo.

De manera simultánea, el proveedor en la nube realiza un monitoreo continuo de la infraestructura. Si se presenta un error en la infraestructura, se genera una alerta que reporta el error correspondiente. En este punto, el ingeniero DevOps debe identificar e informar las causas del error de infraestructura y regresar al proceso de Gestión de Infraestructura para ejecutar la solución del fallo, en caso de ser necesario.

Este proceso de monitoreo continuo garantiza la detección temprana de problemas tanto en el software como en la infraestructura, lo que permite al equipo de desarrollo y al equipo de DevOps actuar rápidamente para resolver los errores y mantener la estabilidad y disponibilidad del sistema en todo momento.

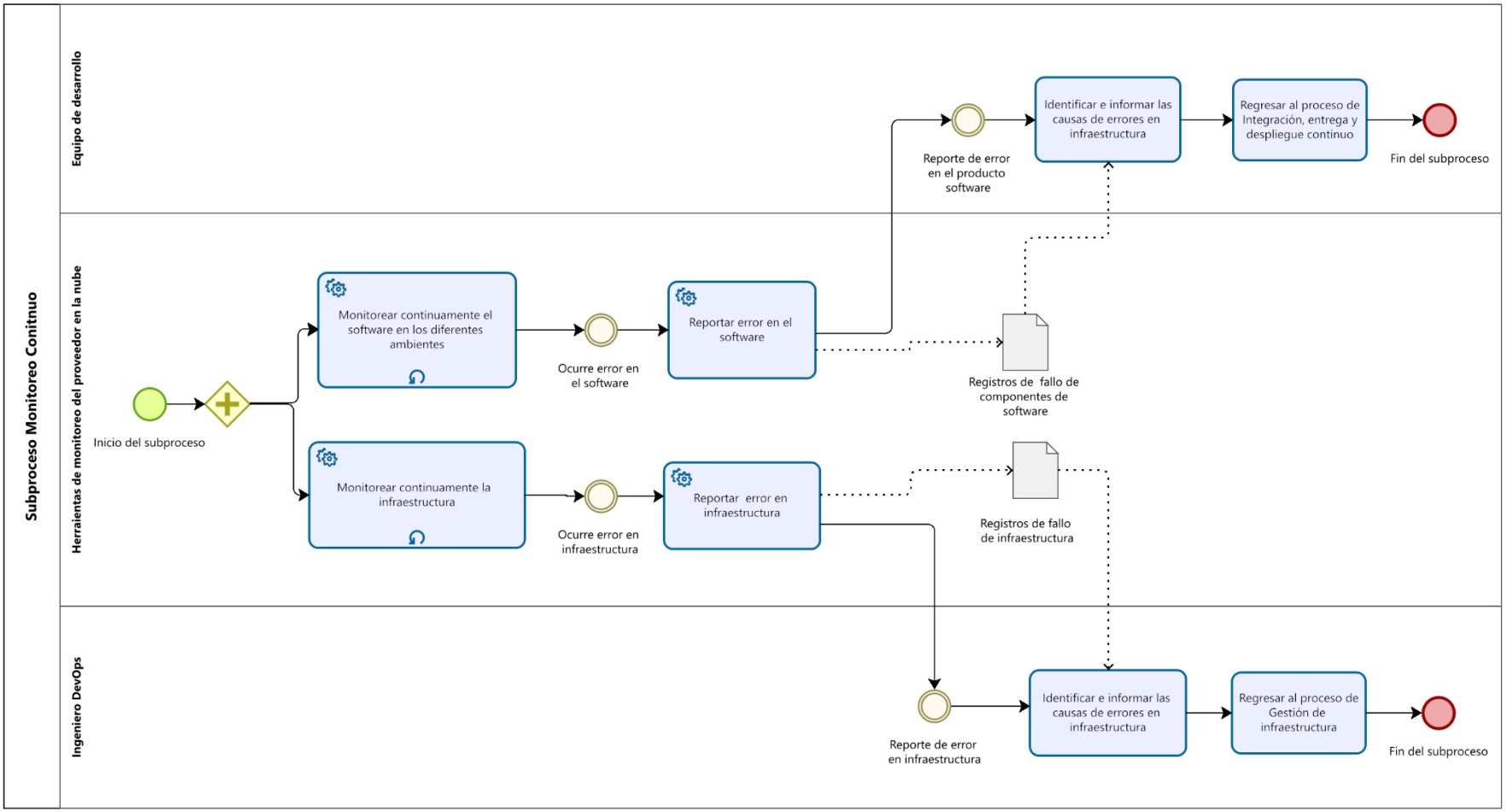

Figura 28. Subproceso de monitoreo continuo

En la Tabla 11 se presenta el listado de las actividades del subproceso de monitoreo continuo.

Subproceso	Actividad	Tipo	
		Af	Ao
Monitoreo continuo	MC-1. Monitorear continuamente el software en los diferentes ambientes (dev, nonprod/QA/test, prod)	x	
	MC-2. Monitorear continuamente la infraestructura para encontrar errores o picos de botella	x	
	MC-3. Reportar errores ocurridos en los componentes software	x	
	MC-4. Reportar errores ocurridos en la infraestructura	x	
	MC-5. Identificar e informar las causas de errores en componentes software	x	
	MC-6. Identificar e informar las causas de errores en infraestructura	x	

Tabla 11. Actividades del subproceso Monitoreo Continuo

Capítulo 5 – Ejecución del proceso de Adopción de DevOps

En este capítulo se presenta la ejecución del proceso de Adopción de DevOps para Wizit MindBlowing Solutions S.A.S. Para ello, se detalla el desarrollo de las actividades de cada subproceso incluyendo las actividades preliminares como la elección de las herramientas a utilizar y la configuración de las mismas. Así mismo, se describe la implementación de un proyecto de referencia desde la etapa de diseño de una arquitectura serverless en AWS, el desarrollo y despliegue de sus funcionalidades bajo el marco de trabajo SCRUM.

5.1. Glosario de términos y tecnologías del capítulo

A continuación, se presenta una breve definición de términos y tecnologías clave que se mencionan a lo largo del capítulo y que son necesarios para la comprensión del mismo.

1. **API**: Es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones y sistemas se comuniquen entre sí. Sirve como una interfaz de programación que define los métodos, formatos de datos y estructuras que pueden ser utilizados para el intercambio de información y la interacción entre distintos componentes de software [42]

2. **Content Delivery Network**: Es una infraestructura distribuida compuesta por servidores ubicados estratégicamente en diferentes ubicaciones geográficas. Su objetivo principal es acelerar la entrega de contenido web, como imágenes, videos, archivos y otros recursos estáticos, a los usuarios finales. Funciona almacenando en caché copias de contenido estático en servidores distribuidos en diferentes partes del mundo. Cuando un usuario solicita un recurso, el CDN redirige la solicitud al servidor más cercano teniendo en cuenta su ubicación geográfica [43].

3. **Docker**: Plataforma de código abierto que permite la creación, distribución y ejecución de aplicaciones en contenedores. Docker permite empaquetar una aplicación y todas sus dependencias en una unidad llamada contenedor, lo que garantiza que la aplicación se ejecute de manera consistente en cualquier entorno [44].

4. **Nginx:** Es un servidor web ligero y de alto rendimiento que también se puede utilizar como proxy inverso y balanceador de carga. Es conocido por su capacidad para manejar eficientemente una gran cantidad de solicitudes simultáneas y por su eficiencia en el uso de recursos [45].

5. **AWS CloudFormation:** Es un servicio de AWS que permite aprovisionar y gestionar recursos de manera automatizada en la nube. CloudFormation utiliza plantillas en formato JSON o YAML para describir la infraestructura y las

configuraciones necesarias, lo que facilita la creación y el mantenimiento de entornos complejos y escalables [30].

6. **AWS IAM (Identity and Access Management)**: Es un servicio de AWS que proporciona control de acceso y administración de identidades para los recursos de AWS. IAM permite gestionar usuarios, grupos y roles, y define políticas para controlar qué acciones pueden realizar los usuarios y qué recursos pueden acceder [46].

7. **Rol IAM:** Es una entidad de seguridad que se puede asignar a usuarios, grupos o servicios dentro de una cuenta de AWS. Su objetivo principal es permitir a las entidades asumir temporalmente ese rol para obtener acceso a recursos específicos. Los Roles IAM se definen con políticas IAM, que especifican los permisos y acciones que se les otorgan [47].

8. **Política IAM:** Una Política IAM es un conjunto de declaraciones que definen los permisos y acciones permitidas o denegadas para usuarios, grupos o roles IAM en AWS. Proporcionan un control granular sobre los permisos, lo que permite definir qué recursos y acciones son accesibles para cada entidad [48].

9. **AWS SQS (Simple Queue Service):** Es un servicio de colas de mensaje distribuidos altamente escalable de AWS. SQS permite el envío y recepción de mensajes entre componentes de aplicaciones distribuidas y desacopla los sistemas para que puedan funcionar de manera independiente y a su propio ritmo [49].

10. **AWS DynamoDB**: Es un servicio de base de datos NoSQL de AWS. DynamoDB es conocido por su escalabilidad, rendimiento y baja latencia. Permite almacenar y consultar grandes cantidades de datos de manera rápida y confiable [50].

11. **AWS CloudWatch**: Es un servicio de monitoreo y observabilidad de AWS. CloudWatch recopila y rastrea métricas, registros y eventos de los recursos y aplicaciones de AWS, lo que permite visualizar y analizar el rendimiento operativo y la salud de los sistemas [51].

12. **AWS Lambda**: Es un servicio de computación sin servidor de AWS. Lambda permite ejecutar código sin aprovisionar ni administrar servidores, y se escala automáticamente en función de la demanda. Es comúnmente utilizado para ejecutar funciones y tareas de backend en respuesta a eventos o solicitudes [52].

13. **AWS WAF (Web Application Firewall)**: Es un servicio de firewall de aplicaciones web administrado por AWS. WAF protege las aplicaciones web contra ataques comunes, como inyecciones de SQL y cross-site scripting (XSS), mediante la inspección y filtrado del tráfico HTTP/HTTPS [53].

14. **AWS Cognito:** Es un servicio de AWS que proporciona autenticación, autorización y gestión de usuarios para aplicaciones web y móviles. Cognito

permite agregar fácilmente funciones de registro, inicio de sesión por medio de redes sociales y control de acceso a las aplicaciones [54].

15. **AWS S3 (Simple Storage Service)**: Es un servicio de almacenamiento de objetos altamente escalable de AWS. S3 permite almacenar y recuperar grandes cantidades de datos de manera segura y duradera, y proporciona una interfaz simple para acceder y administrar los objetos almacenados [55].

16. **AWS Route 53:** Es un servicio de DNS (Sistema de Nombres de Dominio) altamente escalable y confiable de AWS. Route 53 permite registrar y gestionar nombres de dominio, así como también dirigir el tráfico de Internet a los recursos de AWS, como instancias EC2, balanceadores de carga y buckets de S3 [56].

17. **AWS CloudFront**: Es un servicio de entrega de contenido (CDN) global de AWS. CloudFront permite distribuir contenido web de manera eficiente a través de una red global de ubicaciones de borde (edge locations), lo que reduce la latencia y mejora el rendimiento al servir contenido estático y dinámico, como imágenes, videos y archivos [57].

18. **AWS Certificate Manager**: Es un servicio de AWS que permite gestionar y desplegar certificados SSL/TLS para utilizar en servicios y recursos de AWS. Certificate Manager simplifica la adquisición, renovación y gestión de certificados, lo que garantiza la comunicación segura entre los usuarios y los recursos de AWS, como los sitios web [58].

19. **AWS API Gateway:** Es un servicio completamente administrado de AWS que permite la creación, el despliegue y la administración de APIs de forma segura y escalable. API Gateway actúa como un punto de entrada para las solicitudes de API, gestionando el enrutamiento y la seguridad [59].

5.2. Ejecución del subproceso de Adquisición y Configuración

Como se definió en el capítulo anterior, el subproceso de adquisición y configuración es la primera etapa que debe ejecutarse en el proceso de adopción de DevOps. En esta etapa participan los roles de Líder de Desarrollo/Arquitecto de Software e Ingeniero DevOps con el propósito de definir un listado de herramientas y proveedores en la nube que serán esenciales para aplicar DevOps dentro de la organización.

5.1.1. Definición y elección de las herramientas que serán consideradas para la práctica de DevOps (AC-1, AC-2, AC-3)

Como parte de la ejecución de las actividades AC-1, AC-2 y AC-3 (Ver capítulo 4) se realizó un proceso de vigilancia tecnológica (VT) que de acuerdo a la definición formal corresponde a un “proceso organizado, selectivo y permanente, de captar información del exterior y de la propia organización sobre ciencia y tecnología, seleccionarla, analizarla, difundirla y comunicarla, para convertirla en conocimiento para tomar

decisiones” [60]. El proceso de vigilancia tecnológica estuvo enmarcado en la norma UNE 166006:2018 [60] que se compone de cuatro principales etapas:

1. Identificación de necesidades, fuentes de información y medios de acceso
2. Búsqueda y tratamiento de la información
3. Puesta en valor de la información
4. Distribución y almacenamiento

- **Etapa de Identificación de necesidades, fuentes y medios:** En esta fase se definen las necesidades de la vigilancia tecnológica; es decir qué se quiere vigilar y por qué, se consideran las fuentes de la información a utilizar y los medios con los que se cuenta para ejecutar la vigilancia.

  A continuación, se describen cada uno de los elementos que se identificaron en esta fase para el caso específico de la organización:

  - **Necesidad de información:** Se estableció la necesidad de vigilar las herramientas tecnológicas y proveedores en la nube que puedan usarse para ejecutar prácticas DevOps dentro de la organización; esto con el objetivo de identificar tecnologías emergentes o de vanguardia, que sean propicias para los objetivos de crecimiento, competitividad e innovación de la organización.

  - **Fuentes y medios de información:** Se consideraron fuentes tanto académicas como empresariales que permitieran tener diversidad de información para evitar sesgos. Los medios principales de consulta fueron los siguientes:

    1. Información interna de la organización teniendo en cuenta la experiencia de los expertos y colaboradores y las peticiones de los clientes.
    2. Bases de datos científicas (búsqueda de artículos científicos)
    3. Recursos publicados en internet por la comunidad DevOps (blogs y newsletters)

    Es importante destacar que, para la organización, las fuentes de información más relevantes son aquellas que estén enmarcadas dentro de un ámbito empresarial, así como los conocimientos adquiridos por la compañía a lo largo de su experiencia.

- **Etapa de búsqueda y tratamiento de información.** En esta fase se realiza la búsqueda y recopilación de la información utilizando las fuentes definidas en la etapa anterior.

  - **Búsqueda inicial:** La búsqueda inicial comprendió el proceso de consulta y recopilación de la información considerando como filtro básico únicamente la eliminación de las herramientas repetidas.

    La consulta en las bases de datos científicas se realizó bajo la cadena de búsqueda: “Devops” and “Tools” and “Adoption” y como criterio de

inclusión se tuvo en cuenta únicamente los artículos que fuesen publicados después del 2019 y se excluyeron aquellos que posterior a realizar una rápida revisión del artículo no proveían un listado de herramientas. De esta fuente de información se escogieron cinco artículos principalmente que se consideraron los más relevantes y que más información podrían aportar a la recolección.

Por otro lado, se consultaron recursos creados por la comunidad DevOps en internet, de los cuales se destacó *DevOps Period Table* [61] creada por aproximadamente 18000 practicantes de DevOps. En este recurso se describen por categorías cuáles son las herramientas que más usan los ingenieros DevOps alrededor del mundo.

Por último, se consultó con el experto de la compañía, las herramientas tecnológicas que, de acuerdo a la experiencia y objetivos de la organización, deberían ser consideradas en el listado.

De esta búsqueda inicial se identificó un total de 59 herramientas y 3 proveedores en la nube. Los resultados de esta primera etapa pueden consultarse en el **Anexo F. Listado de la búsqueda inicial de herramientas para aplicar DevOps**.

- **Tratamiento y filtro de la información:** Como parte del tratamiento de la información, se realizó la clasificación de las herramientas en categorías de acuerdo a sus características y funcionalidades. Las herramientas se distribuyeron en 10 categorías.

  Posteriormente, en conjunto con el experto de la organización se refinó el listado de herramientas categorizadas y los proveedores en la nube obtenidos en la fase anterior, considerando los siguientes criterios:

1. Soporte, mantenimiento y actividad de la comunidad: ¿Existe documentación oficial y soporte brindado por los proveedores de la herramienta? ¿Se cuenta con una vasta comunidad en la cual apoyarse?
2. Curva de aprendizaje: ¿Qué tan fácil y rápido sería aprender o adaptarse al uso de la herramienta?
3. Relación Costo/Funcionalidades: ¿Qué funcionalidades ofrece la herramienta por el precio establecido?
4. Tiempo en el mercado/Madurez: ¿Es un software estable? ¿Cumple con las cualidades necesarias para utilizarlo a nivel empresarial?
5. Librerías, Plugins o Complementarios: ¿Se pueden extender las funcionalidades de la herramienta a través de librerías, plugins o complementos?
6. Aceptación o Interés del cliente por la herramienta:¿El cliente ha solicitado que se utilice la herramienta o tiene interés en ella? ¿Ya está siendo utilizada?

   El filtro con el experto de la organización dejó un total de 32 herramientas y 3 proveedores en la nube. Este listado puede ser

consultado en el **Anexo G. Listado de herramientas después de filtro con experto**

Posteriormente, se ejecutaron las actividades AC-2 y AC-3, por medio de una encuesta a 13 colaboradores entre los cuales se encontraban desarrolladores más experimentados a nivel técnico y colaboradores con experiencia en gestión de proyectos, a los cuales se les pidió dar una puntuación de 1 a 5 sobre la disposición y pertinencia para usar las herramientas listadas en el Anexo G. La encuesta puede ser consultada en el **Anexo H. Encuesta sobre selección de herramientas para aplicar DevOps.**

Los resultados y análisis de la encuesta produjeron el listado final de herramientas que se muestra a continuación en la Tabla 12.

Categoría	Descripción	Herramientas escogidas
Source Code Management	Las herramientas de control de código fuente nos permiten realizar la gestión y seguimiento histórico de los cambios o modificaciones realizados sobre una base de código compartida.	Github [24] Gitlab [62]
Infrastructure as Code & Configuration Management Tools	Las herramientas de infraestructura como código nos permiten aprovisionar y gestionar nuestros recursos de infraestructura a través de archivos de configuración fácilmente replicables.	AWS Cloudformation [30]
Continuous Integration & Continuous Delivery (CI/CD) Tools	El objetivo de estas herramientas es automatizar en gran parte o en su totalidad la intervención humana manual que tradicionalmente se podría necesitar para agregar nuevas funcionalidades en código a distintos ambientes. Con un pipeline de CI/CD los cambios en el código fuente pueden ser automáticamente testeados, integrados y desplegados.	Jenkins [33] Github Actions [34] Gitlab CI [62]
Monitoring and	El objetivo de las herramientas	AWS CloudWatch

Logging Tools	de monitoreo y logging es permitirnos llevar una trazabilidad de lo que está sucediendo dentro de nuestra arquitectura ya sea para prevenir inconvenientes o detectar fallos.	[51]
Automated Testing Tools	Las herramientas de testing automatizado ejecutan scripts que nos sirven para validar si un software funciona correctamente y cumple con los requerimientos.	Selenium [63]
Development Testing Tools	Las herramientas de testing para desarrollo nos ayudan a verificar que el código que estamos escribiendo es correcto y hace exactamente lo que se supone que debería.	Jest [64]
Correctitud	Las herramientas de correctitud realizan análisis estático del código y reportan aquellas incidencias que son susceptibles a mejoras con el objetivo de tener un código con mejor nivel de optimización	Sonarqube [39]
Collaboration & Communication	La comunicación y colaboración es un pilar fundamental en los proyectos de TI ya que permite que los colaboradores trabajen hacia el logro de una misma meta intercambiando ideas y ofreciendo diferentes perspectivas para brindar soluciones.	Slack [65] Google chat [66]
Planning & Knowledge Sharing	Las herramientas de planeación e intercambio de conocimiento nos van a permitir estructurar, organizar, compartir el trabajo y hacer seguimientos efectivos de tal forma que la gestión de nuestros proyectos sea la más adecuada.	Trello [67] Monday [68]

Cloud Provider	Los proveedores de servicios en la nube corresponden a las compañías que ofrecen infraestructura, plataformas, aplicaciones o servicios de almacenamiento basados en la nube.	Amazon Web Services [26] Google Cloud Platform [66]

Tabla 12. Herramientas escogidas para aplicar DevOps en la organización

- **Puesta en valor de la información.** Esta etapa se considera en aquellos casos donde una vez obtenidos los resultados de la vigilancia tecnológica, se concluye que la información recopilada no es suficiente para satisfacer las necesidades planteadas inicialmente. En esta situación, es necesario realizar un análisis más exhaustivo que incluya la integración de datos provenientes de diversas fuentes, así como la interpretación de lo que es relevante para las necesidades de información. El objetivo es obtener una combinación de información que sea más beneficiosa para los propósitos de la vigilancia.

  Esta fase no fue necesaria en el proceso desarrollado para la organización, ya que la información recopilada y tratada fue suficiente para satisfacer los objetivos planteados para la vigilancia tecnológica.

- **Distribución y almacenamiento**. Los productos de vigilancia e inteligencia se deben distribuir entre los diversos miembros de la organización. Asimismo, se debe velar porque los datos se almacenen de forma que se recupere en el futuro para nuevas actualizaciones.

  El listado definitivo de las herramientas escogidas posterior al proceso de búsqueda, filtrado y refinamiento, se publicó en forma de infografía en la plataforma de *Planning* y *Knowledge Sharing* de la organización visible para todos sus miembros. Adicionalmente, los documentos del proceso de vigilancia tecnológica fueron compartidos con los líderes de la compañía en un espacio de Google Drive.

5.1.2. Adquisición y configuración de las herramientas seleccionadas

A continuación, se describe el proceso de adquisición y configuración de las herramientas y proveedores en la nube que se seleccionaron en la ejecución de las actividades anteriores.

5.1.2.1. Adquisición de las herramientas y proveedor seleccionados (AC-4, AC-5)

- **Contratación de los proveedores en la nube**: Los dos proveedores en la nube seleccionados ya se encontraban contratados con anterioridad por la organización. Sin embargo, posterior a la elección del proveedor y por peticiones de los clientes, se empezó a utilizar Amazon Web Services como principal proveedor en la nube

- **Adquisición de las herramientas**: En este caso, no fue necesario realizar una contratación específica, ya que las herramientas elegidas estaban incluidas en los servicios de AWS o eran herramientas de uso libre.

5.1.2.2. Configuración de las herramientas de CI/CD (AC-5):

Como se mencionó en la Tabla 12, las herramientas escogidas para aplicar CI/CD correspondieron Jenkins, Github Actions y Gitlab CI. A continuación, se describirán los procesos de configuración para cada herramienta:

- **Configuración de Gitlab CI & Github Actions**: Tanto GitLab CI como GitHub Actions son herramientas administradas por sus respectivas compañías proveedoras (GitLab, Inc. y GitHub, Inc.), por lo que no se necesitó realizar configuraciones iniciales. En este caso, solo se revisaron las cuentas de la organización para asegurarse de que tuvieran los permisos necesarios para ejecutar estos servicios.

- **Configuración Jenkins:** Anteriormente, la organización no disponía del uso de Jenkins para llevar a cabo los despliegues de los proyectos. Por esta razón, se configuró una instancia de Jenkins utilizando Docker. Además, se utilizó el servicio de Amazon EC2 para alojar esta instancia.

  Para este caso, se procedió primero a configurar una instancia de Amazon EC2 con las siguientes especificaciones, adaptadas a las necesidades de la organización:

  - 2 vCPUs, 4 GiB, 3.3 GHz Intel Xeon Scalable processor
  - Sistema Operativo Amazon Linux.
  - La instancia está configurada para permitir el tráfico de HTTPS, HTTP y SSH desde cualquier lugar, esto para permitir la conexión a través de SSH y acceder a ella mediante una dirección IP pública.

  Para la instancia de Jenkins se propuso utilizar una arquitectura Master-Slave, la cual consiste en distribuir la carga de trabajo en varios nodos con el objetivo de mejorar tanto la escalabilidad como la eficiencia del sistema.

  En esta arquitectura existe un nodo maestro (master) encargado de gestionar la interfaz de usuario de Jenkins, la planificación de trabajos, el seguimiento del progreso y la gestión de los resultados.

  Adicionalmente, existe uno o más nodos esclavos (slave) que se encargan de realizar trabajos que consumen más recursos, como por ejemplo los *builds*, las pruebas y el despliegue. Los nodos esclavos son máquinas independientes que se conectan al nodo maestro mediante una conexión segura SSH. El nodo maestro es quien se encarga de “repartir el trabajo” a los nodos esclavos.

  Esta arquitectura permite la escalabilidad horizontal, lo que significa que es posible agregar más nodos esclavos para manejar una mayor carga de trabajo.

Además, permite la flexibilidad para ejecutar diferentes tipos de trabajos en diferentes nodos, lo que mejora la eficiencia y el rendimiento de Jenkins.

Algunas características importantes de los nodos esclavos de Jenkins son:

- Se utilizan para distribuir la carga de trabajo y de esta forma mejorar la eficiencia de Jenkins.
- Los nodos esclavos pueden ser cualquier tipo de máquina que ejecute un sistema operativo compatible con Java. Por ejemplo, se pueden utilizar máquinas virtuales, servidores físicos o contenedores.
- Los nodos esclavos pueden tener diferentes sistemas operativos, configuraciones de software y hardware, lo que permite realizar pruebas en diferentes entornos.
- Los nodos esclavos pueden estar ubicados en cualquier parte de la red, siempre y cuando puedan comunicarse con el nodo maestro a través de una conexión segura SSH.
- Los nodos esclavos pueden ser configurados con diferentes conjuntos de herramientas y entornos de compilación, lo que permite la flexibilidad para ejecutar diferentes tipos de trabajos en diferentes nodos.

En la Figura 29 se presenta una ilustración de la arquitectura

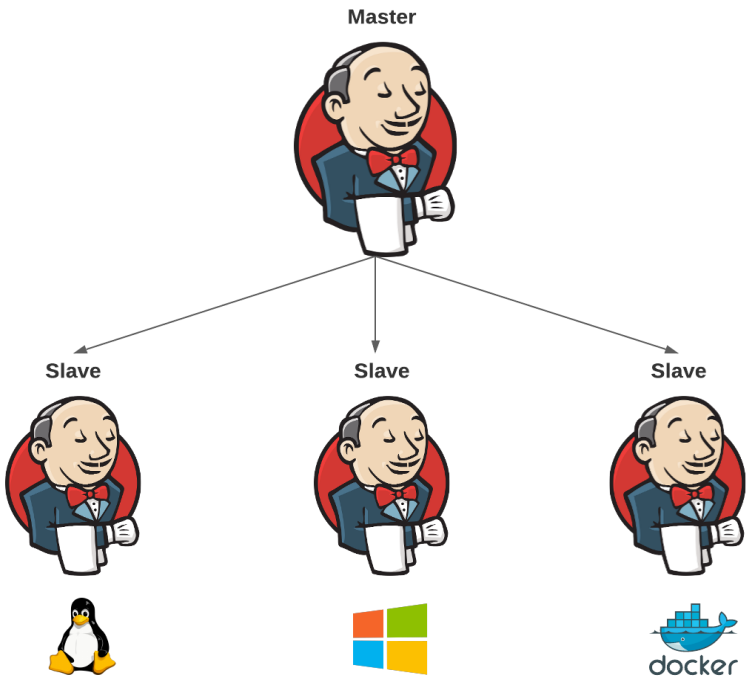

Figura 29. Representación arquitectura Master-Slave Jenkins

Para la implementación de esta arquitectura se decidió utilizar docker y docker-compose debido a las ventajas que estas herramientas ofrecen en términos de personalización, adaptabilidad a las necesidades de la organización y la portabilidad para el despliegue de soluciones en diferentes entornos de ejecución que cuenten con la herramienta instalada.

En la siguiente imagen se muestra una representación simplificada de la implementación de la instancia Jenkins para la organización

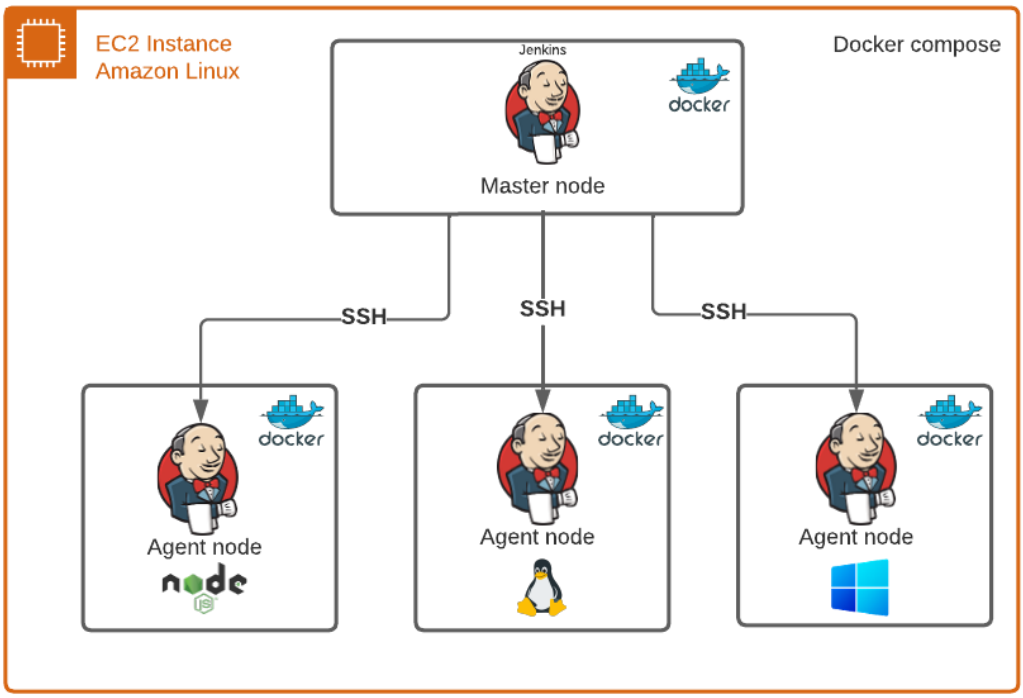

Figura 30. Representación Arquitectura Jenkins para la organización

Para comenzar con la configuración de la arquitectura Master-Slave de Jenkins, se llevó a cabo la instalación de Docker y Docker-Compose en la instancia de Amazon Linux EC2. Luego, se procedió a definir un archivo de docker-compose con la especificación del contenedor del nodo maestro y los nodos esclavos, incluyendo sus configuraciones de volumen y puertos necesarios para una conexión segura mediante SSH. También se realizaron otras configuraciones relevantes para la correcta ejecución de los contenedores en la instancia de EC2.

Una vez se ejecutaron los contenedores se realizó una configuración de proxy reverso para que se realice la redirección desde la IP pública de la instancia de EC2 hacia el contenedor del nodo maestro de Jenkins utilizando Nginx. En la figura 31 se presenta una ilustración de la configuración.

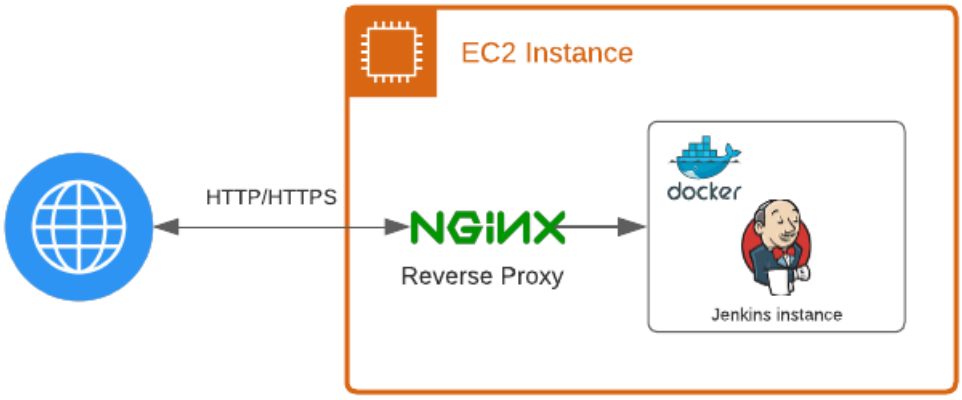

Figura 31. Ilustración proxy inverso configurado para la organización

Posterior a la implementación con docker y la configuración del proxy reverso se ejecutaron las configuraciones correspondientes desde la interfaz gráfica de Jenkins donde se agregaron los nodos esclavos y plugins necesarios para la ejecución de los *jobs*. Para la organización, se configuró un nodo esclavo que posibilite la ejecución de Node.js y la conexión a AWS, permitiendo el despliegue de código fuente Javascript en funciones lambda y en aplicaciones frontend utilizando librerías como React y Vue. Luego, se realizaron pruebas con *jobs* simples que se ejecutaron de forma correcta.

5.1.2.3. Configuraciones adicionales para las herramientas que lo necesiten (AC-6)

Las herramientas seleccionadas no requirieron configuraciones adicionales o específicas, ya que principalmente se trataban de herramientas gratuitas y libres de uso o específicas para cada proyecto. Por lo tanto, su configuración dependerá de los nuevos desarrollos si no habían sido integradas anteriormente.

5.3. Ejecución del subproceso Gestión de infraestructura

En esta sección se presenta la implementación de las actividades del proceso de gestión de infraestructura donde se realizaron actividades de definición de buenas prácticas y plantillas. Posteriormente se realizó un piloto de implementación técnica con un proyecto para un cliente de la organización.

5.3.1. Actividades preliminares del proceso de gestión de infraestructura (GI-0 y GI-1)

Las actividades preliminares del proceso de gestión de infraestructura corresponden a actividades de única aplicación al inicio del subproceso.

- **Definición de lineamientos para la gestión de infraestructura (GI-0)**: Se

definieron las siguientes pautas para la correcta gestión de la infraestructura de los proyectos de la organización:

- **Repositorio exclusivo para infraestructura:** Debe existir un repositorio exclusivo para el almacenamiento de los archivos de gestión de infraestructura con la siguiente estructura:

    - Archivo README: Este fichero debe especificar el nombre e identificador único del proyecto y la estructura de archivos presentes dentro del repositorio y la funcionalidad de cada uno.

    - Archivo de parametrización: Archivo JSON que contenga los valores de los parámetros por ambiente que serán usados como parámetros de entrada en el template de infraestructura como código

    - Archivo de infraestructura como código: Archivo YAML (plantilla CloudFormation) con la definición de los recursos de la nube que serán creados en un stack.

    - Archivo de definición de pipeline de CI/CD: Fichero con sintaxis YAML o Groovy dependiedo de la herramienta que será usado por la herramienta de CI/CD y contiene la descripción de los pasos para llevar a cabo el proceso de despliegue de infraestructura.

- **Nombrado de los recursos y nombrado lógico en archivo de IaC:**

    - Usar nombres descriptivos y cortos que permitan identificar fácilmente qué hace cada recurso. Los nombres de los recursos deben fijarse en una notación **pascal case.** Ejemplo: UserFilesS3Bucket

    - Mantener una misma nomenclatura para el nombre lógico de los recursos. Por ejemplo, agregar prefijos que permitan identificar a qué aplicativo corresponde ese recurso. Los nombres lógicos de los recursos deben fijarse en una notación **kebab case**. Ejemplo: co-users-project-files-s3-bucket

- **Definición de inputs y outputs**

    - **Parámetros de entrada**: Se deben definir parámetros de entrada en el template de infraestructura cuyos valores sean leídos desde un archivo de parametrización. El objetivo de esto es poder personalizar el *stack* para diferentes entornos (desarrollo, pruebas o producción) sin tener que modificar la plantilla del *stack* cada vez que se va a realizar un despliegue.

    - **Parámetros de salida:** Se deben definir los parámetros de salida relevantes que podrían ser usados para monitoreo o por otros

recursos.

- **Seguimiento del AWS Well Architected Framework y sus seis pilares (o su equivalente en otros proveedores de la nube):** AWS Well-Architected Framework describe los conceptos clave, los principios de diseño y las prácticas recomendadas de arquitectura para diseñar y ejecutar cargas de trabajo en la nube. Para la definición de cualquier arquitectura es necesario tener en cuenta todas las recomendaciones y pilares que proporcionan los proveedores en la nube [69]

- **Definir una plantilla de infraestructura base para ser compartida con los demás colaboradores (GI-1):** El objetivo de esta actividad fue disponer una plantilla base para el manejo de la infraestructura con los recursos básicos que son comúnmente utilizados. Como se indicó en los lineamientos propuestos en la actividad anterior, en este repositorio se definió:

    - **Plantilla de CloudFormation:** Describe los recursos de AWS que en este caso son los básicos para la definición de una arquitectura que tenga componente frontend y backend. Estos recursos pueden ser Roles IAM, políticas IAM, Buckets S3, SQS, Api Gateway, tablas de DynamoDB, configuración CloudFront, etc.

    - **Archivo de parametrización**: En este archivo JSON se describen los valores que serán usados como parámetros de entrada en el template de CloudFormation.

    - **Archivo README.md:** Archivo que contiene la descripción de uso del template.

    - **Archivo de definición de Pipeline de CI/CD:** Fichero que contiene los pasos para el despliegue de la infraestructura base.

En la plantilla de CloudFormation se definieron los siguientes recursos básicos:

- **Rol y políticas IAM:** Se define un rol IAM con sus respectivas políticas para otorgar o denegar los permisos necesarios a los recursos de AWS de acuerdo a las necesidades del proyecto y siguiendo el principio de privilegio mínimo.

- **CloudFront Distribution:** Se define una distribución de CloudFront que corresponde a un grupo de servidores donde el contenido que se va a disponer a los usuarios finales es almacenado temporalmente (caché) en edge locations. En este caso, la distribución de CloudFront va a tener como origen del contenido un bucket de S3.

- **S3 Bucket (Cloudfront):** Se define un bucket de S3 para almacenar el contenido de la aplicación web que se va a distribuir a través de CloudFront. Para los proyectos de la organización, el contenido almacenado corresponde al empaquetado (dist) del frontend con librerías como React y Vue.js.

- **Lambda Edge:** Se define una función Lambda Edge que actúa como un interceptor para las peticiones; antes de que la petición llegue a CloudFront, la lambda se ejecuta, revisa la petición que es recibida y verifica que se estén enviando las cabeceras con el contenido permitido. Esto se hace como parte de configuración de seguridad para identificar tempranamente amenazas y garantizar la protección de los usuarios finales.

- **Api Gateway:** El API Gateway actúa como una capa intermedia entre las aplicaciones y los servicios de backend. Permite a las aplicaciones cliente acceder a los servicios de backend a través de una API pública. En este caso la definición del API Gateway incluye la definición del REST API, API Gateway Resource, métodos y API Gateway Stage y deployment.

Estos archivos se pusieron en un repositorio en Github de la organización y fueron compartidos con los demás colaboradores.

En el siguiente link de GitHub se encuentra la plantilla base (simplificada por cláusulas contractuales de confidencialidad) que fue publicada para la compañía: Adjuntar link

Para la ejecución de las siguientes actividades de este subproceso se inició un plan piloto con un proyecto de la organización que brinde una arquitectura de referencia y los insumos iniciales para los proyectos contratados por los clientes; este proyecto requería la construcción de una aplicación web con React y el consumo de APIs REST con Javascript.

5.3.2. Configuraciones preliminares para nuevo proyecto

GI-2. Crear y configurar un repositorio único por proyecto para almacenar los archivos de gestión de infraestructura.

Este repositorio fue creado con base en la plantilla que se definió en la actividad anterior. Se realizaron las adaptaciones adecuadas para el proyecto como nombres de recursos, identificadores del proyecto, valores de parámetros de entrada, entre otros.

GI-3. Configurar herramienta de CI/CD definiendo tipos, origen y desencadenadores para ejecutar despliegues de infraestructura

Esta tarea previa es necesaria para herramientas como Jenkins. Un job en Jenkins se refiere a una tarea o trabajo que se ejecuta dentro del sistema de automatización de Jenkins y la configuración incluye los siguientes pasos desde la interfaz gráfica de Jenkins:

1. Definir un nombre para el job. Los nombres deben ser descriptivos con los prefijos que permitan identificar al proyecto y usando la notación Kebab case. Ejemplo: co-it-users-management-deploy

2. Definir el tipo de job, que en este caso para la mayoría de los proyectos de la organización corresponde a un job tipo Pipeline el cual permite definir trabajos

más complejos y elaborados, que pueden incluir múltiples etapas y pasos lo que da una mayor personalización y flexibilidad.

3. Especificar el origen del código que para los proyectos de la organización corresponde al repositorio de código fuente.

4. Configurar los desencadenadores del job. El disparador del job puede ser manual a través de la interfaz gráfica de Jenkins. También puede establecerse bajo un horario programado, cuando se produce un cambio en el código fuente, cuando se recibe una solicitud de otro servicio o aplicación, o en respuesta a otros eventos.

Por otro lado, en herramientas como GitLab CI y GitHub Actions, esta configuración se realiza a través del archivo de descripción del pipeline, lo que significa que no es necesario recurrir a una interfaz específica o hacer ajustes manuales. En el caso particular de este proyecto, el evento que dispara el despliegue es el push en la rama principal del repositorio.

GI-4. Crear un pipeline de CI/CD para leer y desplegar las plantillas de IaC en el proveedor en la nube

Para la ejecución de esta actividad se ha tenido en cuenta la creación de los pipelines para dos de las herramientas de CI/CD: Jenkins y Github Actions. Sin embargo, ambos archivos tienen los mismos pasos.

Los pasos que se definen para el despliegue la infraestructura independientemente de la herramienta son:

1. Verificación de cambios en el repositorio de código fuente
2. Configuración de credenciales para acceder AWS
3. Verificación de ambiente a desplegar
4. Lectura de parámetros desde el archivo de parametrización
5. Despliegue de plantilla de infraestructura a la nube con los parámetros asignados (para este paso se usan funciones dispuestas por la comunidad de cada herramienta).

5.3.3. Definición y gestión de recursos de infraestructura del proyecto

GI-5. Definición de los recursos de infraestructura del proyecto

Para llevar a cabo esta actividad, se requiere la participación del arquitecto/líder de desarrollo y del ingeniero DevOps. El arquitecto/líder de desarrollo tiene la responsabilidad de traducir los requisitos funcionales y no funcionales de la aplicación en una arquitectura basada en la nube. Por su parte, el ingeniero DevOps se encargará de traducir dicha arquitectura en infraestructura como código. Esta labor permitirá la automatización del despliegue y gestión de la aplicación en el entorno de la nube.

En la Figura 32, se presenta una abstracción de una arquitectura de referencia para

un proyecto. Es importante destacar que, no es posible compartir la arquitectura del proyecto y los requisitos contratados debido a los acuerdos de confidencialidad firmados entre la empresa y sus clientes.

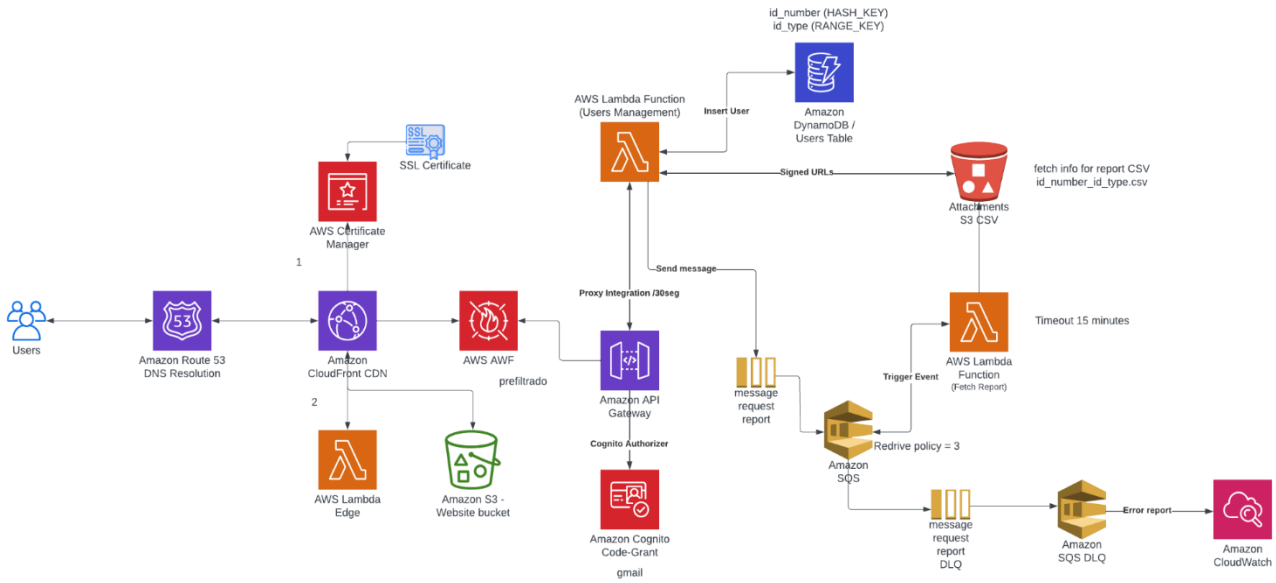

Figura 32. Arquitectura Serverless de referencia para un proyecto de la organización

En términos generales, la arquitectura está diseñada para un aplicativo que permite la gestión de operaciones síncronas como la creación, edición, eliminación de registros, inicio de sesión de usuarios y también puede soportar actividades que requieren la ejecución de procesos asíncronos (procesamiento de pagos, envío de correos, generación de reportes en diferentes formatos, entre otros).

Por ejemplo, el funcionamiento de la arquitectura presentada en la Figura 32 podría ajustarse al contexto de gestión de usuarios y generación de reportes en formato csv siguiendo la siguiente interacción:

1. Desde el navegador los usuarios acceden al dominio de la aplicación web a través de una URL. La resolución de DNS se realiza con Amazon Route 53.

2. Utilizando AWS Certificate Manager se obtiene el certificado SSL que sirve para garantizar que cualquier información que se transmita entre el sitio web y el navegador del usuario esté protegida mediante cifrado simétrico.

3. Las peticiones de los usuarios, las recibe el servicio Cloudfront quien en principio ejecuta una función Lambda Edge que en este caso realiza verificación y reescritura de cabeceras.

4. Las peticiones ya procesadas por lambda edge son devueltas a la distribución de cloudfront. Cloudfront se encarga de entregar el contenido a los usuarios los cuales interactúan con él a través de la interfaz gráfica de la aplicación web. El origen del contenido de la distribución en este caso está configurado por un bucket de S3 el cual contiene el empaquetado del frontend.

5. Como parte de la interacción de los usuarios con la aplicación web se realizan peticiones hacia el backend por medio de un API REST. Estas deben pasar primero por AWS WAF el cual protege las aplicaciones web alojadas en Amazon Web Services (AWS) contra amenazas comunes en línea, como ataques de inyección de SQL, cross-site scripting (XSS), ataques de fuerza bruta y otros tipos de ataques web.

6. Las peticiones verificadas por AWS WAF serán recibidas por el API Gateway quien se integra con Amazon Cognito para verificar la autenticación del usuario.

7. Dentro de las peticiones que los usuarios finales pueden realizar desde la interfaz está la gestión de usuarios como la agregación, edición, eliminación y listado (CRUD). Estas operaciones de registro de datos se realizan a través de una función lambda que se encarga de almacenar los datos en una tabla de DynamoDB

8. Otro de los requerimientos es la generación de reportes asíncronos, estos reportes pueden tener una gran volumetría de datos. Debido a que el API Gateway limita el tiempo para dar respuesta a una solicitud a 30 segundos fue necesario definir un método para encolar solicitudes, en este caso utilizando colas de SQS. Los mensajes de SQS desencadenan eventos en otra función lambda que se encarga de la generación de reportes para posteriormente disponibilizar la descarga de los mismos desde el aplicativo o mediante el envío de un correo electrónico.

GI-6. Crear o modificar los recursos en archivos de infraestructura como código del proyecto

Una vez que la arquitectura ha sido definida y los recursos básicos han sido añadidos en los archivos de gestión de infraestructura, se procede a modificar dichos archivos para agregar los recursos adicionales que puedan ser necesarios y los lineamientos establecidos en la actividad GI-0.

5.3.4. Despliegue y verificaciones post-despliegue

GI-7. Ejecutar despliegue de infraestructura mediante eventos

La ejecución del despliegue está dada por el disparador. En el caso del despliegue de infraestructura por Github Actions está configurado para el evento de push sobre la rama principal (*main*); esto quiere decir que cuando un colaborador suba cambios a esta rama, automáticamente Github Actions se encarga de ejecutar los pasos de despliegue.

Por otro lado, en los ambientes de algunos clientes es necesario ejecutar los despliegues de forma manual siendo desencadenado desde un botón en la interfaz gráfica de Jenkins. Al disparar el evento, Jenkins se encarga de ejecutar los pasos de despliegue.

En esta parte se ejecutan los pasos definidos en el archivo pipeline de CI/CD establecido en la actividad GI-4.

GI-8. Verificar la creación de los recursos de infraestructura en el proveedor en la nube

Una vez la herramienta de CI/CD ha finalizado la ejecución del despliegue y ha sido exitosa, es posible evidenciar la creación o actualización de los elementos en la pila de recursos de infraestructura del proyecto. Para mayor comodidad, generalmente esto se realiza directamente desde la interfaz gráfica del proveedor en la nube.

GI-9. Verificar integración y funcionamiento de los componentes desplegados en la nube

Una vez verificada la creación de los componentes, se procede a hacer pruebas simples de la integración y el correcto funcionamiento y configuración de los componentes. Dependiendo del tipo de recurso, esto se puede realizar directamente desde la consola de AWS o utilizando herramientas externas como un cliente API, navegador, etc.

5.4. Ejecución del subproceso Integración, entrega y despliegue continuo

En esta sección se presenta la ejecución de las actividades del subproceso Integración, entrega y despliegue continuo. En este subproceso se ven involucrados los roles de: arquitecto/líder de desarrollo, desarrollador e ingeniero de pruebas.

5.4.1. Configuraciones preliminares para nuevo proyecto

CICD-1. Creación y configuración de los nuevos repositorios de un proyecto.

Para el caso del proyecto de referencia que se compone de diferentes funciones lambda y una interfaz gráfica, la distribución de repositorios es la siguiente:

- Se crea un solo repositorio el cual agrupa el código de las funciones lambda (backend)

- Se crea un repositorio para almacenar el código fuente del frontend.

CICD-2. Creación y configuración de los jobs en herramienta de CI/CD para los nuevos repositorios

Como se mencionó para el despliegue de infraestructura, la configuración de jobs corresponde a una configuración de herramientas como Jenkins. En este caso para cada repositorio se siguen los mismos pasos mencionados anteriormente.

CICD-3. Creación del archivo de pipeline de CI/CD (backend y frontend)

La ejecución de esta actividad implica la creación de archivos que describen los pipelines que automatizan el despliegue de cada repositorio del proyecto, teniendo en cuenta las tecnologías y el tipo de componente que se va a desplegar. En el caso del proyecto específico del piloto, se consideraron tanto el backend como el frontend.

Los pasos que se llevan a cabo dentro del pipeline se describirán en la actividad CICD-8.

5.4.2. Codificación de requisitos

CICD-4. Análisis de los requerimientos a codificar

Como parte fundamental del proceso de desarrollo de software, se encuentra el análisis previo de los requerimientos a desarrollar; el cual resulta esencial para garantizar que el software se construya de manera correcta y satisfaga las necesidades del cliente.

Estos requerimientos son descritos en forma de historias de usuario y son definidos por el Product Owner dentro del marco de trabajo Scrum. El product owner es responsable de representar las necesidades del cliente y de priorizar los requisitos a desarrollar en cada sprint y de que estos sean claros para el equipo de desarrollo.

La realización de este análisis previo a la codificación permite que el equipo de desarrollo de los proyectos de la organización tenga una visión clara y establezca un marco de trabajo que se alinea con las necesidades de los clientes. Esto facilita que el equipo enfoque sus esfuerzos de manera efectiva y garantice que el resultado final cumpla con las expectativas y requisitos de los clientes.

CICD-5. Creación de pruebas unitarias

Posterior al análisis de los requerimientos a codificar, como estándar se establece que se deben codificar pruebas unitarias antes de iniciar el desarrollo de las nuevas funcionalidades, lo que es conocido como Desarrollo Guiado por Pruebas (en inglés *Test Driven Development - TDD*). Esto obedece a las prácticas de testing continuo y facilita la integración continua dentro del marco de DevOps debido a que cuando se escriben pruebas, se puede garantizar que las pruebas se ejecuten de manera automática cada vez que se realiza un cambio en el código. Esto ayuda a identificar rápidamente cualquier problema de integración que pueda surgir.

Cabe destacar que para la creación de pruebas unitarias se utilizó la herramienta Jest tal como se definió en la elección de herramientas para aplicar DevOps.

CICD-6. Codificación de features, hotfixes o releases

Posterior a la creación de las pruebas unitarias, se realiza la codificación de *features*, *bugfixes* o *hotfixes y releases* según corresponda. Esta terminología corresponde a la terminología usada según el flujo de trabajo Gitflow [40], el cual se centra en el uso de diferentes ramas (branches) de Git para organizar el flujo de codificación en

equipo.

De acuerdo a Gitflow cada término corresponde a:

- **feature**: Se refiere a las nuevas funcionalidades o características del software.

- **bugfix o hotfix:** Hace referencia a la solución de bugs sobre la rama principal normalmente en ambiente productivo.

- **release:** Corresponden a ramas para preparar la próxima versión estable del software.

CICD-7. Code Review y Agregar cambios al repositorio remoto

Una vez terminada la codificación de funcionalidades o solución de bugs, se agregan los cambios al repositorio remoto donde se solicita un proceso de Code Review, que consiste en la práctica de solicitar a otro u otros desarrolladores la revisión del código para detectar errores, problemas de calidad, oportunidades de mejora, entre otros aspectos. Esta es una práctica beneficiosa para la integración continua porque puede permitir detectar errores y problemas de calidad en el código de forma temprana, lo que ayuda a garantizar que el código fuente que se integra en sea de alta calidad y esté libre de errores.

Una vez verificados los cambios en el proceso de CodeReview se integra el nuevo código fuente a la rama de desarrollo en el repositorio remoto. Esto se hace varias veces durante el día de tal forma que no se integren cambios demasiado grandes de código que dificulten realizar el seguimiento y detección de errores de forma temprana.

5.4.3. Despliegue de componentes

CICD-8. Ejecutar pipeline de CI/CD para despliegue ambiente dev

En primera instancia se ejecuta el pipeline de despliegue en el ambiente de desarrollo *(dev)* para propósitos de pruebas por parte de los desarrolladores. Sin embargo, los ambientes de test y producción implementan los mismos pasos; estos pasos son definidos en el archivo de pipeline y son ejecutados por la herramienta de CI/CD.

- **CICD-8a. Chequeo de nuevos cambios en repositorio:** La herramienta de CI/CD realiza la verificación y obtención de los últimos cambios integrados en el repositorio de código fuente.

- **CICD-8b. Análisis de código:** Corresponde a un análisis de código estático para detectar errores, vulnerabilidades, malas prácticas de programación y cumplimiento de estándares. La herramienta para el análisis de código estático en este caso es SonarQube y entre las cosas que analiza se encuentra:

    - **Bugs**: errores en el código que pueden causar un comportamiento inesperado o incorrecto del software.

    - **Vulnerabilidades**: problemas de seguridad en el código que pueden ser explotados por atacantes para acceder a información sensible o dañar el sistema.

    - **Code smells**: prácticas de programación que pueden indicar problemas en el código, como código duplicado, complejidad excesiva, falta de cohesión, alto acoplamiento, etc.

    - **Cobertura de pruebas**: la cantidad de código que es cubierto por pruebas unitarias.

    - **Cumplimiento de estándares de codificación**: SonarQube puede verificar que el código cumpla con estándares de codificación predefinidos, como por ejemplo, cumplir con el estilo de codificación definido por el lenguaje de programación utilizado.

    - **Métricas de complejidad**: por ejemplo la complejidad ciclomática, el número de líneas de código, la cantidad de parámetros de una función, entre otros.

- **CICD-8c. Reporte de incidencias detectadas en análisis de código**: SonarQube genera un reporte de los hallazgos durante el análisis del código. Estos reportes permiten identificar qué cambios deben realizarse para mejorar la calidad del código o subsanar vulnerabilidades.

- **CICD-8d. Empaquetar la aplicación:** La herramienta de CI/CD crea un paquete que contiene todos los archivos y dependencias necesarios para distribuir la aplicación hacia los entornos en la nube.

- **CICD-8e. Despliegue a los servicios del proveedor en la nube:** La herramienta de CI/CD realiza el despliegue del empaquetado hacia los servicios del proveedor en la nube como AWS CloudFront y AWS Lambda según se haya configurado en los pasos especificados en el archivo del pipeline.

- **CICD-8f. Reporte de estado del despliegue (fallido o exitoso):** La herramienta de CI/CD informa si la operación de despliegue fue exitosa o fallida, lo cual permite conocer el estado de los despliegues y tomar medidas rápidas en caso de que algo haya salido mal.

5.4.4. Entrega a ambiente de QA y pruebas

CICD-9. Generación de versión inmutable de código para despliegue a ambiente de QA/Test/Nonprod

Una vez se ha integrado y desplegado el código en ambiente de desarrollo, se han realizado las pruebas respectivas del funcionamiento en el entorno en la nube y se ha comprobado inicialmente que los comportamientos de la aplicación son los esperados, se debe realizar el paso a ambiente de Test o Nonprod, el cual

corresponde al ambiente más parecido a producción donde los ingenieros de pruebas ejecutan las validaciones respectivas de las aplicaciones.

Para el despliegue de los proyectos es necesario generar versiones inmutables de código para los despliegues hacia el ambiente de pruebas. Estas versiones inmutables corresponden a Tags de Git que contienen la última versión estable del software candidata para un paso a producción (si se cumplen con las condiciones de QA).

La generación de tags git se automatizó con un script de bash que se encarga de realizar la sincronización del repositorio, lee la versión del producto software desde el archivo *package.json* y ejecuta los comandos de git para la generación del tag. Esto fue muy beneficioso y acorde a la cultura DevOps, debido a que la generación manual de los tags podía llegar a ser tardada dependiendo de cuántos componentes se iban a promover a ambiente de pruebas.

CICD-10. Ejecución de pipeline de CI/CD para despliegue a ambiente QA/Test/Nonprod

Como se mencionó anteriormente los pasos para los despliegues en los diferentes ambientes son los mismos. La diferencia en el ambiente de test y producción es que el despliegue se realiza a partir de la versión inmutable de código (tag de git).

CICD-11. Ejecución de pruebas de QA y notificar resultados

Tan pronto como se dispone de la última versión estable del aplicativo en ambiente de pruebas, el equipo de QA comienza a ejecutar los casos de prueba previamente diseñados, en caso de encontrar errores, se genera un reporte de bugs correspondiente. Este reporte permitirá solucionar los problemas identificados en la ejecución fallida y garantizar un correcto funcionamiento del aplicativo.

CICD-12. Revisión y solución de resultados fallidos en etapa de QA

Por parte del equipo de desarrollo se revisan los resultados de las ejecuciones fallidas, las evidencias de las mismas y se procede a la solución de los bugs reportados para nuevamente iterar sobre el proceso de CI/CD.

CICD-13. Aprobar la nueva versión del software

Cuando se han solucionado todas las incidencias encontradas en etapa de QA y esto haya sido certificado por los ingenieros de pruebas del proyecto, se genera una aprobación y certificación de la nueva versión del software para ser desplegada a ambiente productivo.

5.4.5. Despliegue a ambiente productivo

CICD-13. Desplegar la última versión certificada y aprobada por QA a ambiente productivo

Las versiones estables que hayan sido certificadas por QA, son promovidas a

producción para la validación en principio de usuarios piloto y posteriormente su masificación.

5.5. Ejecución del subproceso de Monitoreo Continuo

A continuación, se describe la ejecución de las actividades del subproceso de monitoreo continuo.

5.5.1. Monitoreo continuo de componentes

MC-1. Monitorear continuamente el software en los diferentes ambientes (dev, nonprod/QA/test, prod)

Como parte del monitoreo continuo del proyecto piloto, se configuró el almacenamiento de logs en puntos críticos del software, tales como conexiones a bases de datos, servicios de AWS y llamadas a servicios externos. Los logs almacenan las trazas de los procesos que realiza el aplicativo de forma que se pueda realizar seguimiento de operaciones exitosas o errores.

Este monitoreo se realiza en diferentes ambientes, incluyendo los de desarrollo y pruebas, con el fin de detectar y evitar la replicación de fallas en el ambiente de producción. De esta forma, se asegura la calidad y estabilidad del sistema en todo momento.

De acuerdo a la elección de herramientas para monitoreo continuo la herramienta utilizada fue AWS CloudWatch.

MC-2. Monitorear continuamente la infraestructura

El monitoreo de la infraestructura del proyecto es una tarea muy importante para garantizar el correcto funcionamiento y disponibilidad de la aplicación web. Esto incluye el monitoreo de la utilización de los recursos de los servicios de AWS que componen la infraestructura del proyecto. Algunos ejemplos de los recursos que se pueden monitorear son el almacenamiento, las unidades de capacidad de lectura y escritura, la CPU, la RAM, el tráfico de red, entre otros. El monitoreo se realiza con AWS CloudWatch donde para recursos específicos se generaron alarmas que se activan cuando hay un error o uso mayor al esperado de los recursos.

La actividad de monitoreo continuo de la infraestructura tiene como objetivo principal la detección temprana de posibles cuellos de botella, riesgos de indisponibilidad, sobrecostos, picos de tráfico o posibles sobrepasos de los límites establecidos por el proveedor de servicios en la nube, entre otros.

5.5.2. Reporte e identificación de errores de infraestructura (MC-3, MC4)

Una vez se identifican riesgos o errores que se den en la infraestructura, AWS CloudWatch genera una alerta que se reporta en la cuenta. En este caso, el ingeniero DevOps (puede ser en conjunto con el Arquitecto de Software) debe identificar las

causas, reportarlas e iniciar nuevamente la ejecución del subproceso de Gestión de Infraestructura en caso de que el error pueda solucionarse con alguna configuración de específica. De lo contrario, se debe escalar el problema con el líder del área de ingeniería para que se le dé gestión.

5.5.2. Reporte e identificación de errores de componentes software (MC-5, MC6)

Cuando se reportan fallas en los componentes de software, AWS CloudWatch guarda los registros del error. En este caso, los desarrolladores de software deben identificar las causas, reportarlas e iniciar nuevamente la ejecución del subproceso de Gestión de Infraestructura en caso de que el error sea a causa de una falla en el código o en el uso de alguno de los servicios de AWS. De lo contrario, se debe escalar el problema con el líder del área de ingeniería para que se le dé gestión.

5.6. Resultados de aplicación del proceso de adopción de DevOps

Durante la aplicación del proceso de adopción de DevOps se logró:

- Configuración y despliegue de instancia de la herramienta de CI/CD Jenkins usando recursos en la nube y tecnologías de contenerización.

- Obtener una arquitectura de una aplicación web de referencia en AWS incluyendo API REST, recursos para procesamiento asíncrono, almacenamiento, gestión de operaciones CRUD y acceso y disponibilización de la aplicación en internet.

- Generación de templates de infraestructura como código y pipelines de CI/CD.

- Generación de lineamientos para la gestión de infraestructura y proceso de CI/CD

- Aplicar del flujo de trabajo y gestión de repositorios GitFlow.

- Incorporación de prácticas de CodeReview y TDD en el flujo de desarrollo de software.

Los entregables fueron compartidos con la empresa para usarse como referencia en los diferentes proyectos de la organización.

Capítulo 6. Conclusiones y trabajo futuro

6.1. Conclusiones

Durante la práctica profesional, se automatizaron los despliegues y la configuración de infraestructura adaptando y aplicando las prácticas DevOps. Este enfoque permitió estandarizar y proporcionar una guía para que los colaboradores de la organización pudieran integrar estas prácticas en el proceso de desarrollo de software.

En primera instancia, el desarrollo de la práctica profesional implicó llevar a cabo una fase inicial de recolección de información para identificar procesos actuales y retos o dificultades existentes en el flujo de desarrollo que practicaba la compañía. Durante esta etapa, se identificaron tres problemáticas fundamentales que requerían abordarse de manera efectiva: el uso de despliegues manuales de código e infraestructura, la falta de lineamientos claros para la gestión de la infraestructura, el flujo de desarrollo, la integración y entrega continua, y la brecha de conocimiento existente en relación con las prácticas DevOps junto con las herramientas necesarias para implementarlas.

Con el objetivo de atender las necesidades y problemáticas de la organización, se propuso un proceso de adopción de DevOps. El proceso resultó necesario para la organización debido a que DevOps se ha convertido en un requisito en el mundo empresarial actual donde la eficiencia y agilidad de entrega de software es fundamental para mantener la competitividad. Este enfoque mejora significativamente la experiencia de los clientes, lo que a su vez aumenta su satisfacción.

El proceso propuesto fue adaptado a partir de [14] para crear un marco compuesto por cuatro subprocesos, 50 actividades, 7 roles, 17 artefactos y un total de 14 herramientas y 2 proveedores en la nube escogidos para aplicar DevOps. El objetivo principal de este proceso fue fomentar la adopción de DevOps dentro de la organización, teniendo en cuenta los propósitos, desafíos y particularidades específicas de la compañía. Las actividades, roles y artefactos del proceso se seleccionaron cuidadosamente para abordar los puntos críticos y maximizar los beneficios de DevOps en la organización, asegurando la participación adecuada de los diferentes equipos y áreas funcionales involucradas en el proceso.

El alcance del proceso de DevOps planteado está netamente enfocado para arquitecturas *serverless*, de esta forma, no todos los proyectos en los cuales se contrate a la organización se podrán ajustar al proceso, ya que esto depende de la naturaleza del cliente y proyecto; sin embargo, la demanda del mercado recibida por la organización está más enfocada a las arquitecturas *serverless*. Este tipo de arquitecturas ofrecen diferentes beneficios que son deseables por las empresas y los clientes entre los cuales se encuentran una mayor escalabilidad, costos optimizados, facilidad de mantenimiento, velocidad de desarrollo y flexibilidad tecnológica. Esto ofrece la posibilidad a la organización de ser más ágil, eficiente y competitiva en el mercado actual.

Se realizó la implementación del proceso la cual implicó llevar a cabo un proyecto piloto donde se desarrolló una aplicación web con una arquitectura *serverless* en

AWS. En este proyecto piloto se logró automatizar el 80% de las actividades que se estaban realizando de forma manual debido a que a pesar de los beneficios que trae la automatización y las prácticas de DevOps, dentro del flujo hay tareas que no es posible automatizar por completo como por ejemplo: el manejo de secretos, la integración con algunos servicios de AWS, pruebas que es necesario realizarlas de forma manual y aún es necesaria una aprobación manual del despliegue en ambiente productivo para evitar despliegues accidentales.

Por último, la integración de prácticas DevOps en el marco de trabajo Scrum ofreció una solución integral y eficiente para enfrentar los desafíos del desarrollo de software en la organización. La combinación de agilidad, estandarización, automatización y colaboración mejoró significativamente la calidad, la velocidad y la satisfacción del cliente. Asimismo, el proceso de adopción de DevOps se adaptó adecuadamente a las particularidades de la empresa, garantizando una implementación exitosa y sostenible de estas prácticas innovadoras.

6.2. Lecciones aprendidas

Las lecciones aprendidas descritas en esta sección se basan en un conjunto de desafíos y experiencias que se presentaron a lo largo del desarrollo de la práctica profesional; que permitieron que se genere un conocimiento tanto a nivel profesional como personal, entre las cuales se encuentra:

- Las prácticas profesionales permiten adquirir y desarrollar diversas capacidades requeridas en un contexto empresarial real tanto a nivel técnico como personal.

- Es fundamental tener un apoyo constante del asesor y demás actores por parte de la organización en todas las etapas incluyendo la documentación de la práctica profesional; asegurando así que lo planteado, desarrollado y documentado cumpla con las expectativas y reglamentos de la organización.

- Es necesario que se dé una convergencia y trabajo conjunto entre el mundo empresarial y académico cuando se llevan a cabo prácticas profesionales que permitan que ambas partes se involucren activamente en todo el proceso; para así maximizar los beneficios de la experiencia de la práctica.

- Se percibió que existe una diferencia significativa del entorno académico con el entorno empresarial en materia de tecnologías aplicadas. Esto generó que la curva de aprendizaje durara más de lo esperado debido a que las tecnologías usadas en la academia tendieron a ser más antiguas.

- El trabajo académico “Proceso para fomentar y apoyar la adopción de DevOps en Pymes de Software” [14] realizado por egresados y profesores del alma máter representó un gran aporte para la organización, evidenciando la influencia que puede tener la academia e investigación dentro de un contexto empresarial real, ofreciendo soluciones a problemáticas que se presentan dentro de este entorno.

- Se comprendió que no todas las necesidades y requerimientos de una organización pueden ser abordados desde un solo enfoque, ya que, en este caso, el proceso de DevOps únicamente tiene el alcance para arquitecturas *serverless* que, si bien se adapta a la mayoría de los proyectos de la organización, no es aplicable para todos los proyectos, teniendo que buscar alternativas y particularizaciones dependiendo de la naturaleza del cliente y proyecto.

- Se comprendió que, si bien las prácticas manuales dentro de un proyecto software pueden causar errores, no todo es automatizable y es necesario que algunos procesos sigan haciéndose con intervención humana.

- El desarrollo de esta práctica profesional permitió interiorizar y ver en acción los conocimientos adquiridos en la academia acerca del ciclo de vida del software (planeación, desarrollo, pruebas, despliegue y mantenimiento) así como el diseño e implementación de arquitecturas de software y gestión de proyectos informáticos.

- Se consideró que es esencial fomentar el desarrollo de las habilidades interpersonales desde la academia; debido a que, en un contexto real se debe tener la capacidad de comunicación efectiva, liderazgo y gestión de las emociones (entre otras) que permitan un mejor relacionamiento con clientes, jefes y colaboradores en una organización.

6.3. Trabajo futuro

De acuerdo al alcance del trabajo realizado la práctica profesional, se realizan las siguientes recomendaciones para trabajo futuro:

- Realizar seguimiento de las prácticas cada seis meses con el objetivo de evaluar si se deben realizar ajustes al proceso de DevOps.

- Continuar con la vigilancia de tecnologías de vanguardia que puedan ser incluidas en las prácticas de DevOps.

- Integrar prácticas de seguridad en el proceso de DevOps bajo un enfoque de DevSecOps.

- Incursionar en la aplicación de prácticas de Inteligencia Artificial y Aprendizaje Automático para las operaciones de TI (AIOps).

Referencias

[1] J. Humble and J. Molesky, “Why enterprises must adopt devops to enable continuous delivery,” *Cutter IT Journal*, vol. 24, no. 8, pp. 6–12, 2011.

[2] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A survey of DevOps concepts and challenges,” *ACM Comput Surv*, vol. 52, no. 6, 2019, doi: 10.1145/3359981.

[3] M. Zulfahmi Toh, S. Sahibuddin, and M. N. Mahrin, “Adoption issues in DevOps from the perspective of continuous delivery pipeline,” *ACM International Conference Proceeding Series*, vol. Part F1479, pp. 173–177, 2019, doi: 10.1145/3316615.3316619.

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

[5] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “DevOps,” 2016.

[6] A. Agarwal, S. Gupta, and T. Choudhury, “Continuous and Integrated Software Development using DevOps,” *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, no. June, pp. 290–293, 2018, doi: 10.1109/icacce.2018.8458052.

[7] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, “DevOps adoption benefits and challenges in practice: A case study,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10027 LNCS, pp. 590–597, 2016, doi: 10.1007/978-3-319-49094-6_44/COVER.

[8] M. Krey, “DevOps Adoption: Challenges & Barriers,” *Proceedings of the 55th Hawaii International Conference on System Sciences*, vol. 7, 2022, doi: 10.24251/hicss.2022.877.

[9] M. Shameem, “A Systematic Literature Review of Challenges Factors for Implementing DevOps Practices in Software Development Organizations: A Development and Operation Teams Perspective,” *Evolving Software Processes*, no. December, pp. 187–199, 2022, doi: 10.1002/9781119821779.ch9.

[10] M. Z. Toh, S. Sahibuddin, and M. Naz’ri Mahrin, “The Challenges and Mitigation Strategies of Using DevOps during Software Development,” *Papers.Ssrn.Com*, vol. Part F1479, no. 1, pp. 173–177, 2019, [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4004335

[11] Wizit MindBlowing Solutions S.A.S, “The BitBang Company.” https://thebitbang.company/

[12] C. Técnicas Aplicadas, M. I. Vinicio Estrada-Velasco, J. I. Alexandra Núñez-Villacis, and W. I. Clemente Cunuhay-Cuchipe, “Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software,” *Dominio de las Ciencias, ISSN-e 2477-8818, Vol. 7, Nº. Extra 4, 2021 (Ejemplar dedicado a: AGOSTO ESPECIAL), pág. 54*, vol. 7, no. 4, p. 54, 2021, doi: 10.23857/dc.v7i4.2429.

[13] D. Quintana, “Método para definir procesos en organizaciones desarrolladoras de Software,” Universidad del Cauca, 2017.

[14] C. E. Orozco, C. Pardo, K. Zuñiga, and S.-C. Certuche, “Proceso para fomentar y apoyar la adopción de DevOps en PyMEs de software,” *Revista Científica, ISSN 0124-2253, ISSN-e 2344-8350, Vol. 45, Nº. 3, 2022 (Ejemplar dedicado a: september-december), págs. 422-437*, vol. 45, no. 3, pp. 422–437, 2022, doi: 10.14483/23448350.19644.

[15] R. Jabbari, N. Bin Ali, K. Petersen, and B. Tanveer, “What is DevOps? A systematic mapping study on definitions and practices,” *ACM International Conference Proceeding Series*, vol. 24-May-201, 2016, doi: 10.1145/2962695.2962707.

[16] A. Debbiche, M. Dienér, and R. B. Svensson, “Challenges when adopting continuous integration: A case study,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8892, pp. 17–32, 2014, doi: 10.1007/978-3-319-13835-0_2/COVER.
[17] S. A. I. B. S. Arachchi and I. Perera, “Continuous integration and continuous delivery pipeline automation for agile software project management,” *MERCon 2018 - 4th International Multidisciplinary Moratuwa Engineering Research Conference*, pp. 156–161, Jul. 2018, doi: 10.1109/MERCON.2018.8421965.
[18] B. Laster, “Continuous Integration vs. Continuous Delivery vs. Continuous Deployment.”
[19] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, “DevOps: Introducing infrastructure-as-code,” *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, pp. 497–498, Jun. 2017, doi: 10.1109/ICSE-C.2017.162.
[20] L. Jiang, Y. Pei, and J. Zhao, “Overview of Serverless Architecture Research,” *J Phys Conf Ser*, vol. 1453, no. 1, Mar. 2020, doi: 10.1088/1742-6596/1453/1/012119.
[21] D. Taibi, N. El Ioini, C. Pahl, and J. R. S. Niederkofler, “Patterns for serverless functions (Function-as-a-Service): A multivocal literature review,” *CLOSER 2020 - Proceedings of the 10th International Conference on Cloud Computing and Services Science*, pp. 181–192, 2020, doi: 10.5220/0009578501810192.
[22] “What is Cloud Computing.” https://aws.amazon.com/what-is-cloud-computing/?nc1=h_ls (accessed Aug. 01, 2023).
[23] “Git.” https://git-scm.com/ (accessed Aug. 01, 2023).
[24] “GitHub: Let’s build from here · GitHub.” https://github.com/ (accessed Aug. 01, 2023).
[25] “The DevSecOps Platform | GitLab.” https://about.gitlab.com/ (accessed Aug. 01, 2023).
[26] “¿Qué es AWS?” https://aws.amazon.com/es/what-is-aws/ (accessed Aug. 01, 2023).
[27] “JavaScript | MDN.” https://developer.mozilla.org/es/docs/Web/JavaScript (accessed Aug. 01, 2023).
[28] “Node.js.” https://nodejs.org/es (accessed Aug. 01, 2023).
[29] “React.” https://react.dev/ (accessed Aug. 01, 2023).
[30] “What is AWS CloudFormation? - AWS CloudFormation.” https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html (accessed Aug. 01, 2023).
[31] “AWS Serverless Application Model - Amazon Web Services.” https://aws.amazon.com/es/serverless/sam/ (accessed Aug. 01, 2023).
[32] “Serverless: Develop & Monitor Apps On AWS Lambda.” https://www.serverless.com/ (accessed Aug. 01, 2023).
[33] “Jenkins.” https://www.jenkins.io/ (accessed Aug. 01, 2023).
[34] “Documentación sobre las Acciones de GitHub - Documentación de GitHub.” https://docs.github.com/es/actions (accessed Aug. 01, 2023).
[35] E. Q. TBBC, “Documento de proceso de testing interno de la organización,” 2023.
[36] “Terraform by HashiCorp.” https://www.terraform.io/ (accessed Aug. 01, 2023).
[37] “Configuration Management System Software - Chef Infra | Chef.” https://www.chef.io/products/chef-infra (accessed Aug. 01, 2023).
[38] “Ansible is Simple IT Automation.” https://www.ansible.com/ (accessed Aug. 01, 2023).
[39] “Code Quality Tool & Secure Analysis with SonarQube | Sonar.” https://www.sonarsource.com/products/sonarqube/ (accessed Aug. 01, 2023).
[40] J. Cwiklinski, “git-flow Documentation Release 1.0,” 2020.
[41] A. Araújo, “Test Driven Development Fortalezas y Debilidades.” [Online]. Available: http://www-pao.ksc.nasa.gov/history/mercury/mercury-overview.htm
[42] M. Santoro, L. Vaccari, and D. Smith, “Web Application Programming Interfaces (APIs): general-purpose standards, terms and European Commission initiatives

APIs4DGov study-digital government APIs: the road to value-added open API-driven services," doi: 10.2760/675.
[43] Folasade Y. Ayankoy, Olubukola Otushile, and Onome Blaise Ohwo, "A Review on Content Delivery Networks and Emerging Paradigms," *International Journal of Scientific and Engineering Research* , 2018, Accessed: Aug. 01, 2023. [Online]. Available: https://www.researchgate.net/publication/345494833_A_Review_on_Content_Delivery_Networks_and_Emerging_Paradigms
[44] "Docker: Accelerated Container Application Development." https://www.docker.com/ (accessed Aug. 01, 2023).
[45] "Advanced Load Balancer, Web Server, & Reverse Proxy - NGINX." https://www.nginx.com/ (accessed Aug. 01, 2023).
[46] "¿Qué es IAM? - AWS Identity and Access Management." https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/introduction.html (accessed Aug. 01, 2023).
[47] "Roles de IAM - AWS Identity and Access Management." https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/id_roles.html (accessed Aug. 01, 2023).
[48] "Políticas y permisos en IAM - AWS Identity and Access Management." https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/access_policies.html (accessed Aug. 01, 2023).
[49] "What is Amazon Simple Queue Service? - Amazon Simple Queue Service." https://docs.aws.amazon.com/es_es/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html (accessed Aug. 01, 2023).
[50] "What is Amazon DynamoDB? - Amazon DynamoDB." https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html (accessed Aug. 01, 2023).
[51] "¿Qué es Amazon CloudWatch? - Amazon CloudWatch." https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html (accessed Aug. 01, 2023).
[52] "¿Qué es AWS Lambda? - AWS Lambda." https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html (accessed Aug. 01, 2023).
[53] "AWS WAF - AWS WAF, AWS Firewall Manager, y AWS Shield Advanced." https://docs.aws.amazon.com/es_es/waf/latest/developerguide/waf-chapter.html (accessed Aug. 01, 2023).
[54] "¿Qué es Amazon Cognito? - Amazon Cognito." https://docs.aws.amazon.com/es_es/cognito/latest/developerguide/what-is-amazon-cognito.html (accessed Aug. 01, 2023).
[55] "¿Qué es Amazon S3? - Amazon Simple Storage Service." https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/Welcome.html (accessed Aug. 01, 2023).
[56] "¿Qué es Amazon Route 53? - Amazon Route 53." https://docs.aws.amazon.com/es_es/Route53/latest/DeveloperGuide/Welcome.html (accessed Aug. 01, 2023).
[57] "¿Qué es Amazon CloudFront? - Amazon CloudFront." https://docs.aws.amazon.com/es_es/AmazonCloudFront/latest/DeveloperGuide/Introduction.html (accessed Aug. 01, 2023).
[58] "¿Qué es AWS Certificate Manager? - AWS Certificate Manager." https://docs.aws.amazon.com/es_es/acm/latest/userguide/acm-overview.html (accessed Aug. 01, 2023).
[59] "¿Qué es Amazon API Gateway? - Amazon API Gateway." https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html (accessed Aug. 01, 2023).

[60] “UNE 166006:2018 Gestión de la I+D+i: Sistema de vigilancia e i...” https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0059973 (accessed Aug. 01, 2023).
[61] “Periodic Table of DevOps Tools | Digital.ai,” *https://digital.ai/*, Accessed: Aug. 01, 2023. [Online]. Available: https://digital.ai/learn/devops-periodic-table/
[62] “GitLab CI/CD | GitLab.” https://docs.gitlab.com/ee/ci/ (accessed Aug. 01, 2023).
[63] “Selenium.” https://www.selenium.dev/ (accessed Aug. 01, 2023).
[64] “Jest · Delightful JavaScript Testing.” https://jestjs.io/ (accessed Aug. 01, 2023).
[65] “¿Qué es Slack? | Slack.” https://slack.com/intl/es-co/help/articles/115004071768-%c2%bfQu%c3%a9-es-Slack- (accessed Aug. 01, 2023).
[66] “Google Chat: Mensajería grupal y colaboración | Google Workspace.” https://workspace.google.com/intl/es-419/products/chat/ (accessed Aug. 01, 2023).
[67] “Qué es Trello: descubre sus funciones, usos y todo lo que ofrece | Trello.” https://trello.com/es/tour (accessed Aug. 01, 2023).
[68] “¿Qué es monday.com? – Support.” https://support.monday.com/hc/es/articles/115005310945--Qu%C3%A9-es-monday-com- (accessed Aug. 01, 2023).
[69] “AWS Well-Architected Framework - AWS Well-Architected Framework.” https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html (accessed Aug. 02, 2023).

Anexo A. Encuesta sobre percepción de DevOps para líderes de TI

El Anexo A presenta el listado de las preguntas realizada a los tres líderes de TI encuestados, así como las respuestas de cada uno de ellos.

Disponible en el siguiente enlace:
https://drive.google.com/drive/folders/1b1QlRE6NYKC7xkZWXb2-DD6-RYT1A0MT?usp=drive_link

Anexo B. Encuesta sobre percepción de DevOps para desarrolladores

El Anexo B presenta el listado de las preguntas realizada a los 14 desarrolladores de software encuestados, así como las respuestas de cada uno de ellos.

Disponible en el siguiente enlace:
https://drive.google.com/drive/folders/1b1QlRE6NYKC7xkZWXb2-DD6-RYT1A0MT?usp=drive_link

Anexo C. Actividades del Proceso para Fomentar la Adopción de DevOps para Pymes

El Anexo C presenta el listado de actividades propuestas por subproceso en el proceso base. Es importante resaltar que este anexo forma parte integral de los anexos presentados en el artículo base.

Disponible en el siguiente link: https://bit.ly/43VCGHw

Anexo D. Artefactos del Proceso para Fomentar la Adopción de DevOps para Pymes

El Anexo D presenta el listado de artefactos de entrada y salida propuestos en el proceso base. Es importante resaltar que este anexo forma parte integral de los anexos presentados en el artículo base.

Disponible aquí: https://bit.ly/3KpCKbG

Anexo E. Roles sugeridos del Proceso para Fomentar la Adopción de DevOps para Pymes.

El Anexo E presenta el listado y descripción de los roles propuestos en el proceso base. Es importante resaltar que este anexo forma parte de las tablas presentadas en el artículo del proceso original.

Disponible aquí:
https://drive.google.com/file/d/1DWdlMUsyrpHHqq4OVTPPI87MR25KsZMt/view?usp=drive_link

Anexo F. Listado de la búsqueda inicial de herramientas para aplicar DevOps.

Este anexo presenta el listado de herramientas de DevOps posterior a la búsqueda inicial sin ningún procesamiento aún realizado.

Disponible aquí:
https://drive.google.com/drive/folders/1d3kojDq0pyxiCInRxwHOSlkzZ1htNhsL?usp=sharing

Anexo G. Listado de herramientas después de filtro con experto

El anexo G presenta el listado de las herramientas que se considerarán para la elección de DevOps después de un filtro inicial con el experto de la organización.

Disponible aquí: https://drive.google.com/drive/folders/1d3kojDq0pyxiCInRxwHOSlkzZ1htNhsL?usp=sharing

Anexo H. Encuesta sobre selección de herramientas para aplicar DevOps

El anexo H presenta la descripción y resultados de la encuesta aplicada a los desarrolladores sobre la elección de herramientas para aplicar DevOps.

Disponible aquí: https://drive.google.com/drive/folders/1b1QlRE6NYKC7xkZWXb2-DD6-RYT1A0MT?usp=sharing