EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Dashboard para el seguimiento continuo de las actividades del ciclo de vida de DevOps en la gestión de proyectos de software

presentada por

Ing. Luis Flores Orozco

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Directora de tesis

Dra. Blanca Dina Valenzuela Robles

Cuernavaca, Morelos, México. Junio 2024.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 16/mayo/2024

OFICIO No. DCC/049/2024
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial de **LUIS FLORES OROZCO** con número de control M21CE011, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"DASHBOARD PARA EL SEGUIMIENTO CONTINUO DE LAS ACTIVIDADES DEL CICLO DE VIDA DE DEVOPS EN LA GESTIÓN DE PROYECTOS DE SOFTWARE"** y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

BLANCA DINA VALENZUELA ROBLES
Directora de tesis

RENE SANTAOLAYA SALGADO
Revisor 1

HUMBERTO HERNÁNDEZ GARCÍA
Revisor 2

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante

Cuernavaca, Mor., 22/mayo/2024
No. De Oficio: SAC/179/2024
Asunto: Autorización de impresión de tesis

LUIS FLORES OROZCO
CANDIDATO AL GRADO DE MAESTRO
EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"DASHBOARD PARA EL SEGUIMIENTO CONTINUO DE LAS ACTIVIDADES DEL CICLO DE VIDA DE DEVOPS EN LA GESTIÓN DE PROYECTOS DE SOFTWARE"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz

Agradecimientos

Agradezco a el Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado durante la realización de esta tesis.

Agradezco al Tecnológico Nacional de México campus Centro Nacional de Investigación y Desarrollo Tecnológico (TecNM /CENIDET), que me brindó la oportunidad de realizar mis estudios de maestría.

Quiero agradecerle a mi asesora de tesis, la Dra. Blanca Dina Valenzuela Robles, por su inestimable orientación, paciencia y apoyo a lo largo de todo este proceso. Su experiencia y consejos han sido fundamentales para la culminación de este trabajo.

Agradezco también a los miembros de mi comité revisor, el Dr. René Santaolaya Salgado y al M.C. Humberto Hernández García, por las valiosas contribuciones que hicieron al trabajo final y por el tiempo que dedicaron para revisarlo.

Abstract

Large technology companies such as Amazon, Facebook and Netflix share a fundamental characteristic: their adoption of agile methods, which are the basis of continuous implementation and integration. However, collaboration between development and operations teams is essential to maximize the benefits of these approaches. It is crucial to break down the existing barriers between these areas and foster an environment where both parties can learn from each other, thus improving the integration and continuous deployment of projects.

All teams involved in software management and development should prioritize identifying current production issues or defects. This approach allows you to obtain accurate information and facilitates the implementation of continuous improvements. Continuous improvement, as a fundamental principle of DevOps, drives us to constantly improve our processes, teams and products.

The main objective of this thesis is to provide software development companies with a control tool that allows them to track process activities through the results of the proposed metrics and indicators and graphically visualize the progress status of their projects. DevOps.

To achieve the stated objective, a software tool was developed based on open-source technologies, using the 8 most common processes of the DevOps approach and relying on the IEEE 2675:2021 standard for DevOps.

Resumen

Las grandes empresas tecnológicas como Amazon, Facebook y Netflix comparten una característica fundamental: su adopción de métodos ágiles, que son la base de la implementación e integración continua. Sin embargo, la colaboración entre los equipos de desarrollo y operaciones es esencial para maximizar los beneficios de estos enfoques. Es crucial derribar las barreras existentes entre estas áreas y fomentar un ambiente donde ambas partes puedan aprender mutuamente, mejorando así la integración y el despliegue continuo de los proyectos.

Todos los equipos involucrados en la gestión y desarrollo de software deben priorizar la identificación de los problemas o defectos de producción actuales. Este enfoque permite obtener información precisa y facilita la implementación de mejoras continuas. La mejora continua, como principio fundamental de DevOps, nos impulsa a perfeccionar constantemente nuestros procesos, equipos y productos.

El objetivo principal de esta tesis es proporcionar a las empresas desarrolladoras de software una herramienta de control que les permita dar seguimiento a las actividades de los procesos mediante los resultados de las métricas e indicadores propuestos y visualizar de manera gráfica el estado de avance de sus proyectos DevOps.

Para lograr el objetivo planteado, se desarrolló una herramienta de software con base en tecnologías de código abierto, utilizando los 8 procesos más comunes del enfoque DevOps y apoyándose del estándar IEEE 2675:2021 para DevOps

Tabla de contenido

Índice de figuras

Índice de tablas

CAPÍTULO I

1. Introducción

Las grandes empresas tecnológicas como Amazon, Facebook y Netflix comparten una característica fundamental: su adopción de métodos ágiles, que son la base de la implementación e integración continua. Sin embargo, la colaboración entre los equipos de desarrollo y operaciones es esencial para maximizar los beneficios de estos enfoques [1]. Es crucial derribar las barreras existentes entre estas áreas y fomentar un ambiente donde ambas partes puedan aprender mutuamente, mejorando así la integración y el despliegue continuo de los proyectos.

Es fundamental iniciar la comunicación entre los equipos de desarrollo (Dev) y operaciones (Ops) para eliminar las barreras existentes entre ellos y fomentar un ambiente colaborativo donde ambas partes puedan aprender mutuamente [2]. Trabajar en conjunto posibilita que los desarrolladores, probadores y equipos de operaciones estén al tanto de los problemas de producción y sus soluciones. Esta información puede ser utilizada para establecer un plan de riesgo o de mitigación para futuras versiones del software.

Las empresas de desarrollo de software que adoptan métodos o procesos DevOps deben adaptarlos a sus necesidades individuales, ya que los objetivos de cada empresa pueden variar [3]. Al adoptar este enfoque, es fundamental comprender cómo se modifican los procesos en el ciclo de vida del software, desde el inicio del desarrollo hasta la implementación del producto en un entorno de producción. Esta comprensión proporcionará una base sólida para implementar mejoras continuas y obtener información precisa sobre el rendimiento del proceso.

Al implementar estrategias de DevOps adecuadas, es crucial medir su impacto en los proyectos de desarrollo de software. La mejora continua, como principio fundamental de DevOps, nos impulsa a superarnos cada día, optimizando los procesos, equipos y productos. Aunque identificar áreas de mejora puede ser sencillo, encontrar la mejor solución posible a menudo requiere innovación y creatividad.[3]

Este trabajo de tesis se centró en dos análisis. El primero consistió en identificar los procesos más utilizados dentro del enfoque DevOps, mientras que el segundo tuvo como objetivo identificar las métricas e indicadores más comunes entre la comunidad que adopta este enfoque en sus proyectos de software. Esto se realizó con el propósito de establecer las bases para el desarrollo de un tablero de control (Dashboard) que, a diferencia de otras herramientas con este propósito, se fundamenta en el estándar IEEE 2675:2021 para DevOps.

1.1. Planteamiento del problema

Algunas empresas desarrolladoras de software que han adoptado el enfoque DevOps como parte de su cultura organizacional, pueden descuidar el seguimiento y cumplimiento completo de los principios y actividades de este enfoque. Esto puede ocurrir debido al desconocimiento de indicadores y métricas relevantes o a una comprensión limitada de la cultura DevOps. Esta situación puede resultar en una gestión inadecuada de los proyectos de desarrollo de software y en la incapacidad de cumplir con la mejora continua, es en este sentido donde las oportunidades de mejora pueden verse disminuidas.

1.2. Objetivos

Objetivo general

Medir el desempeño de prácticas DevOps mediante el desarrollo de un tablero de control que dé seguimiento a los procesos del ciclo de vida del desarrollo de software para proponer oportunidades de mejora en la gestión de proyectos de software.

Objetivos específicos

- Establecer los indicadores de desempeño para las prácticas DevOps.
- Establecer las métricas para el proceso de cada práctica
- Dar seguimiento a las actividades de los procesos mediante los resultados de las métricas e indicadores propuestos.

1.3. Justificación

La adopción de prácticas DevOps no se limita únicamente a la automatización de tareas, actividades o subprocesos; también implica mejorar todo el ciclo de vida del desarrollo de software. La falta de indicadores o métricas de calidad desde el inicio del desarrollo hasta la implementación en producción dificulta la evaluación del cumplimiento de la mejora en el ciclo de vida del software.

Esta investigación tiene por objetivo proporcionar una herramienta de control que permita a las empresas desarrolladoras de software dar seguimiento y visualizar de manera gráfica el estado de avance de sus proyectos DevOps.

1.4. Alcances y limitaciones

Alcances

- Al ser instalada en un servidor de aplicaciones el tablero de control podrá estar disponible 24/7.
- Los indicadores y métricas mostradas en el tablero de control en cada etapa son exclusivas de los procesos DevOps.
- Se implementa una métrica y un indicador por cada fase del proceso de desarrollo de proyectos de software en DevOps

Limitaciones

- No se consideran cambios organizacionales o culturales para la adopción de la cultura DevOps en las empresas.
- Se parte del supuesto que las organizaciones ya tienen adoptado DevOps
- No se propone adopción de herramientas para el enfoque DevOps
- No se mide la calidad del software entregado conforme

CAPÍTULO II

2. Marco conceptual

Este capítulo se centra en los conceptos teóricos necesarios para comprender el presente trabajo

2.1. Medición [4]

Es la capacidad de medir el proceso de desarrollo de software, mediante la incorporación de diferentes métricas. El proceso de medir ayudará a aumentar la eficiencia en el desarrollo del producto. Los datos deben ser transparentes, accesibles para todos, significativos y capaces de ser vistos.

2.2. Métrica [4]

Para la literatura de ingeniería de software, el término abarca una variedad de conceptos. Se utiliza para referirse tanto a la acción de cuantificar un fenómeno, como a varios aspectos relacionados con este proceso. Por ejemplo, puede hacer referencia a la cantidad que se está midiendo, al método utilizado para llevar a cabo la medición, los resultados obtenidos, a los modelos que describen las relaciones entre diferentes medidas o incluso a la medición de los propios objetos.

Las métricas de software pueden aplicarse en diversas etapas del proceso de desarrollo, desde la planificación y diseño hasta la implementación, pruebas y mantenimiento. Además, pueden adaptarse a las necesidades específicas de cada proyecto y organización, lo que las convierte en una herramienta versátil y fundamental para la gestión de proyectos de software.

Al proporcionar valores cuantitativos, las métricas de software permiten una evaluación objetiva y sistemática de diferentes aspectos del desarrollo de software. Esto facilita la identificación de áreas de mejora, la toma de decisiones informadas y la evaluación del progreso a lo largo del ciclo de vida del proyecto.

2.2.1. Las métricas de procedimiento [5]

Se enfocan principalmente en aspectos como la duración de un procedimiento, los costos asociados, la eficacia de los métodos empleados, así como a la mejora y la predicción para futuros procedimientos. Estas métricas son utilizadas por los gestores de alto nivel para evaluar el estado del desarrollo. Su aplicación resulta beneficiosa para el control y la gestión integral de todo el proceso de desarrollo.

2.2.2. Métricas de proyecto [5]

Estas métricas se realizan normalmente para un proyecto específico e incluye la escala, el costo, la carga de trabajo, el estado, la capacidad de producción, el riesgo, el grado de satisfacción de los clientes, etc. Sirven principalmente para ajustar el proyecto, evitar los problemas o riesgos y ayudar a optimizar los planes de desarrollo. Y a partir de ahí, mejora la calidad del producto a través de los avances en métodos técnicos y estrategias de gestión.

2.2.3. Métricas de producto [5]

Estas métricas comprenden y gestionar la calidad del producto, con el propósito de prever y mejorar su nivel de calidad. Se focalizan, principalmente, en aspectos como la confiabilidad, la facilidad de mantenimiento, la escala del producto, la complejidad del software, la portabilidad y la documentación, entre otros. Esta métrica se centra en evaluar el producto medio o final de la fase del sistema.

2.2.4. Métricas de complejidad del software [5]

Se refieren a la cantidad de recursos consumidos durante el desarrollo, mantenimiento y uso del software. La complejidad es un factor fundamental que puede desencadenar defectos, siendo la fiabilidad un problema inherente a la complejidad del software. Cuando la complejidad supera ciertos umbrales, los defectos o fallas del software tienden a aumentar rápidamente. La mantenibilidad del software también está estrechamente ligada a su nivel de complejidad.

2.2.5. Indicador [4]

Un indicador es una medida que ofrece una estimación o evaluación de atributos específicos, derivada de un modelo con respecto a las necesidades de información definidas. Los indicadores constituyen la base para el análisis y la toma de decisiones, y son aquellos que se presentan a los usuarios para su consideración.

2.2.6. Indicadores de software [6]

Se define que, un indicador es una métrica, o una combinación de métricas, de software que proporciona una medida para el proceso de producción de software y el producto final. Consiste en asignar valores cuantitativos a los atributos que influyen en el producto o en el proceso.

2.2.7. Ingeniería de software [7]

En la normativa IEEE 12207-2017, se refiere que la ingeniería de software requiere de la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software. Se trata esencialmente de aplicar principios de ingeniería al ámbito del software. La ingeniería de software se centra en entender las necesidades de las partes interesadas y la funcionalidad requerida desde las primeras etapas del ciclo de desarrollo. Esto implica documentar exhaustivamente los requisitos del sistema y llevar a cabo el diseño y la validación del sistema de manera integral, teniendo en cuenta la totalidad del problema a resolver.

2.2.8. El ciclo de vida del desarrollo de software [7]

En la normativa IEEE 12207-2017, se refiere que al período durante el cual se lleva a cabo el proceso de desarrollo de software. Este ciclo abarca todos los detalles del procedimiento de formulación del software, proporcionando un marco estándar para la creación de productos de software y tecnologías de la información. Las organizaciones suelen seguir este procedimiento para crear productos que cumplan con los requisitos y expectativas establecidos.

El ciclo de vida del desarrollo de software abarca una variedad de modelos, como el modelo en cascada, el modelo incremental, el modelo en espiral y el modelo V, entre otros. Estos modelos representan diferentes enfoques para la planificación, el diseño, la implementación y el mantenimiento del software a lo largo de su ciclo de vida. Cada modelo tiene sus propias características y ventajas, y la elección del modelo adecuado depende de los requisitos específicos del proyecto y las preferencias de la organización.

2.2.9. Ingeniería de Software orientada a procesos [6]

La ingeniería de software abarca procesos, métodos y herramientas diseñados para desarrollar sistemas complejos basados en computadoras de manera oportuna y con alta calidad. El proceso de desarrollo de software se compone de cinco actividades fundamentales: comunicación, planificación, modelado, construcción y despliegue, las cuales son aplicables a todos los proyectos de software. La práctica de la ingeniería de software es una actividad orientada a resolver problemas, que se rige por un conjunto de principios fundamentales.

2.3. Definición de DevOps [2]

DevOps es un conjunto de procedimientos en los cuales los desarrolladores y los equipos de operaciones colaboran y se comunican para proporcionar software y servicios de forma ágil, confiable y con niveles superiores de calidad. Implica compartir roles y responsabilidades dentro de un equipo capacitado, que tiene la plena responsabilidad de su servicio y la tecnología asociada. Este enfoque se extiende desde las etapas de desarrollo hasta la implementación y el mantenimiento continuo del producto.

En la normativa IEEE 2675:2021, el término 'DevOps' proviene de la combinación de las abreviaturas 'Dev' (Desarrolladores) y 'Ops' (Operación). Según esta normativa, DevOps se define como el conjunto de principios y métodos destinados a facilitar la comunicación y colaboración entre las diversas partes interesadas. Su objetivo es especificar, desarrollar y operar software, sistemas y servicios, con el fin de lograr una mejora continua en todas las etapas del ciclo de vida.

En la normativa se establece que, el proceso de medición para DevOps está integrado con otras actividades de implementación e integración continua, llevándose a cabo como parte del flujo de valor de CI/CD sin necesidad de recopilación, agregación o informes manuales [8].

El proceso de medición afecta a diversos roles dentro de la organización, incluyendo desarrolladores, probadores, personal de soporte, gerentes de seguridad y administradores. Cada uno de estos roles debe tener la capacidad de implementar y ajustar fácilmente las métricas para respaldar el progreso del trabajo actual, prevenir problemas y asegurar mejoras en el proceso.

2.4. Prácticas DevOps [9]

Las prácticas más utilizadas en el enfoque DevOps son:

Registro y monitoreo: Se lleva a cabo un monitoreo frecuente de las aplicaciones organizacionales para evaluar la satisfacción de los usuarios finales, lo que ayuda a los desarrolladores a determinar las actualizaciones necesarias.

Integración continua: Este proceso se refiere a cómo los desarrolladores ejecutan pruebas automatizadas y fusionan sus códigos. Su objetivo es mejorar la calidad del software detectando y corrigiendo errores de manera más rápida.

Comunicación y colaboración: Los equipos utilizan plataformas de chat de la aplicación para compartir el flujo de trabajo organizacional y realizar un seguimiento de sus proyectos.

Microservicios: Las empresas emplean una interfaz de programación de aplicaciones basada en HTTP para integrar servicios ligeros en una sola aplicación. Cada unidad funcional de la aplicación tiene un propósito específico.

Entrega continua: Este proceso implica la construcción, prueba y publicación ininterrumpida de cambios de código. Suele ir de la mano con la integración continua.

Infraestructura como código: Es una práctica que se utiliza en la gestión de la infraestructura mediante técnicas de desarrollo de software y código, como la entrega e integración continua. Los administradores y desarrolladores interactúan a través de la interfaz de programación de aplicaciones (API) de la nube utilizando herramientas codificadas.

Por lo tanto, según [9], los beneficios de utilizar DevOps como enfoque de desarrollo de software son:

Velocidad: La entrega continua y el uso de microservicios permiten a los equipos lanzar actualizaciones rápidamente, obtener retroalimentación en tiempo real de los clientes y adaptarse ágilmente a cambios en el mercado.

Fiabilidad: La fiabilidad en DevOps se refiere a garantizar que los sistemas, aplicaciones y servicios permanezcan operativos y que funcionen correctamente según lo previsto. Esto asegura que los sistemas puedan recuperarse rápidamente de fallos y operar bajo diferentes niveles de carga.

Ruptura de silos: La colaboración mejora la eficiencia al eliminar tiempos muertos en la transferencia de responsabilidades. Las funciones de codificación e implementación suelen tratarse como una sola unidad integrada.

Entrega rápida: Los productos se lanzan al mercado con celeridad, lo que mejora el retorno de la inversión y la satisfacción del cliente.

Gestión proactiva de riesgos: El equipo identifica rápidamente cualquier error que pueda afectar el éxito de la aplicación y lo soluciona antes de que llegue a los consumidores, minimizando así los impactos negativos.

2.5. Metodología ágil [10]

La metodología ágil es un enfoque iterativo e incremental que permite a los equipos de desarrollo de software adaptarse rápidamente a los cambios en los requisitos del proyecto y proporcionar productos de alta calidad de manera más eficiente y efectiva. Ofrece un enfoque flexible y colaborativo, que se centra en la entrega continua de valor al cliente, la adaptación a los cambios y la mejora constante del producto.

2.6. Scrum [10]

Scrum se centra en mejorar el proceso de desarrollo de software, la calidad del producto, el valor empresarial, el rendimiento, la usabilidad y la eficiencia; al mismo tiempo que busca reducir costos, riesgos e incertidumbres.
También, promueve la participación activa del usuario final, la aceptación del cambio y la entrega iterativa de productos y que puede ser utilizado en combinación con otros métodos. A diferencia de algunos enfoques tradicionales, Scrum también se esfuerza por fomentar factores blandos como la agilidad, la confianza, la motivación, la responsabilidad y la transparencia [11].

2.7. Sprint [11]

Un sprint en metodologías ágiles es, un período de tiempo fijo y enfocado en el que se desarrolla y entrega un incremento de producto funcional. Permite un enfoque iterativo e incremental para el desarrollo de software, lo que ayuda a minimizar los riesgos y a obtener retroalimentación temprana del cliente.

2.8. Product Backlog [11]

El Product Backlog se define como un listado ordenado y priorizado de los requisitos necesarios para la implementación de un proyecto, enlistan las tareas identificadas, se hace visible el trabajo necesario para alcanzar un objetivo establecido, proporcionando una guía para la planificación y ejecución del proyecto.

2.9. Sprint Backlog [11]

El Sprint Backlog es una lista detallada de tareas y actividades que el equipo de desarrollo planea completar durante un Sprint en un marco de trabajo ágil como Scrum. Es una parte fundamental del proceso Scrum, ya que define el trabajo que se va a realizar en el sprint actual.
Este formato permite al equipo de desarrollo mantener una visión organizada y enfocada en las tareas necesarias para completar el sprint de manera exitosa.

CAPÍTULO III

3. Estado del arte

En esta sección se presenta un panorama de lo que se ha investigado y descubierto hasta el momento en relación con el tema.

3.1. Implementación de DevOps

Para llevar a cabo esta investigación, se realiza una revisión del uso del enfoque DevOps en la industria. Se examinan prácticas como la entrega y despliegue continuo, el control de versiones y automatización de datos de prueba y la gestión de proyectos como en [12], [13] y [14]. Además, se analiza la integración de DevOps con contenedores y microservicios para coordinar la entrega y los controles de calidad de una aplicación [15]. También se aborda el impulso del cambio cultural para mejorar la comunicación entre equipos, permitiendo una mayor integración entre las unidades operativas y de desarrollo [16].

DevOps ha sido implementado en otras áreas, como la automatización del aprendizaje aplicando principios DevOps, y la aceleración y aumento de la eficiencia de los procesos al establecer conexiones entre las actividades de la ingeniería [17]. Asimismo, los principios DevOps también están presentes en el ámbito académico en [18] y [19].

Las investigaciones demuestran que las prácticas de DevOps han transformado radicalmente la manera en que las organizaciones desarrollan, entregan y operan software [20]. Este enfoque facilita la transformación digital en las empresas que lo adoptan, permitiéndoles acelerar la entrega de software, reducir los tiempos de inactividad, mejorar la calidad del producto y aumentar la satisfacción del cliente [21] y [22].

3.2. Desarrollo de software con DevOps y metodologías ágiles

3.2.1. Measurement Based Performance Evaluation of DevOps

El artículo propone utilizar la productividad y la eficiencia como métricas y conductores de calidad en DevOps, las métricas resultantes incluyen [23]:

- Reuniones de Sprint: Evalúa la duración y el número de reuniones del equipo durante un sprint específico.
- Rendimiento: Mide las unidades de trabajo completadas por el equipo dentro de un período de tiempo establecido.
- Cobertura de Historias de Usuario: Cuantifica el número de requisitos o cambios cubiertos por el equipo durante el desarrollo de un proyecto.

3.2.2. The Convergence of Scrum and DevOps for an Agile IT Organization

En esta investigación se propone que las metodologías Agile y DevOps son los principales impulsores de cambio en las organizaciones de TI y comparten muchos aspectos comunes. Ambas se caracterizan por ciclos de entrega más rápidos, lanzamientos incrementales más pequeños, retroalimentación para la mejora continua, eliminación de desperdicios e impedimentos. Si bien Agile se centra en las interacciones del equipo, la cultura y los valores también aborda la automatización, DevOps pone énfasis en el flujo de entrega y considera la comunicación y la cultura.

En conclusión, los equipos de desarrollo tienen la autonomía para determinar cómo se organizan, cómo operan y qué herramientas y procesos utilizan, siendo responsables del valor y la calidad de los resultados entregados. Sin embargo, ningún enfoque es universalmente aplicable, y las personas que realizan el trabajo son las más adecuadas para tomar decisiones sobre cómo hacerlo. Se sugiere un enfoque empírico que comienza desde la concepción de la idea hasta la entrega final al cliente, con el objetivo de proporcionar un flujo continuo de valor [24].

3.2.3. The Software Architect and DevOps

Esta investigación señala cuatro preocupaciones principales asociadas con las prácticas de DevOps [25]:

- Realizar cambios rápidos en producción.
- Detectar errores mediante la automatización de pruebas.
- Minimizar o eliminar errores durante el despliegue.
- Identificar y corregir fallas en el sistema.

En conclusión, el artículo argumenta que los miembros de un equipo de trabajo deben adaptarse a diversos roles, lo cual representa un cambio cultural, mientras que la implementación de prácticas DevOps implica también cambios técnicos, como la selección del conjunto adecuado de casos de prueba. Además, subraya que el éxito de la implementación de DevOps depende en gran medida del respaldo proporcionado por las herramientas y la automatización elegidas.

3.2.4. Quality-Aware DevOps Research: ¿Where Do We Stand?

En este artículo se presenta un estudio de los esfuerzos existentes en el ámbito de DevOps, categorizados según la etapa y ciclo de vida en los que principalmente contribuyen.

Esta investigación abarca todas las etapas de DevOps, identificando iniciativas dirigidas a mejorar el diseño arquitectónico, el modelado y la infraestructura como código, así como la integración continua / entrega continua (CI / CD), pruebas y verificación, y la gestión del tiempo de ejecución [13].

En conclusión, la investigación sugiere que el futuro de DevOps se dirige hacia la IA y el ML, impulsado por un uso intensivo de datos. Esto ofrece potenciales beneficios al mejorar la funcionalidad y transformar la forma en que los desarrolladores de sistemas y los administradores pueden diseñar, probar, implementar y mantener los sistemas.

3.2.5. Communication of Changes in Continuous Software Development

En esta investigación se evaluaron los desafíos e impactos de las prácticas de Desarrollo Continuo de Software (CSD, por sus siglas en inglés). Los resultados generales indican que los desafíos y prácticas relacionadas con la comunicación de cambios de software en entornos CSD varían según varios factores, como la frecuencia de los cambios de software y su implementación, así como los equipos, la estructura y el diseño del producto [12].

En conclusión, se sugiere que a medida que aumenta la frecuencia de los cambios lanzados, aumenta la necesidad de prácticas de comunicación automáticas y asincrónicas para transmitir los cambios de software. Por otro lado, en el caso de versiones menos frecuentes, se pueden utilizar prácticas de comunicación sincrónicas o manuales, como reuniones estructuradas, notas de lanzamiento y correos electrónicos.

3.3. Las métricas en DevOps

3.3.1. The Goal Question Metric Approach

Se presenta un enfoque sistemático utilizado en la ingeniería de software y otros campos para establecer objetivos, formular preguntas relevantes y definir métricas para evaluar el progreso hacia esos objetivos. Esto permite a los equipos de desarrollo de software establecer una base sólida para la toma de decisiones informadas y la mejora continua del proceso de desarrollo.

Se basa en tres elementos principales [26]:

Objetivos: Son las metas que se desean alcanzar en un proyecto o proceso específico. Estos objetivos deben ser claros, medibles y relevantes para el contexto del proyecto.

Preguntas: Son interrogantes específicas que ayudan a comprender y evaluar el progreso hacia los objetivos establecidos. Estas preguntas se formulan de manera que puedan ser respondidas mediante la recopilación y análisis de datos.

Métricas: Son medidas cuantitativas o cualitativas que se utilizan para evaluar el rendimiento y el cumplimiento de los objetivos. Las métricas se seleccionan para proporcionar información relevante sobre el estado y el progreso del proyecto.

En resumen, el enfoque GQM sigue un proceso iterativo que involucra la identificación de objetivos, la formulación de preguntas relacionadas con esos objetivos, la selección de métricas apropiadas y la recopilación y análisis de datos para responder a esas preguntas.

3.3.2. Defining software quality metrics for Agile and DevOps - SD Times

En el artículo se examina un informe titulado "Las métricas de calidad de software definitivas para Agile y DevOps", se muestran los resultados de una encuesta realizada a líderes empresariales en Agile y DevOps, que sobresalen en la medición y la búsqueda de la automatización en todo el ciclo de vida del software, los resultados incluyen las cuatro mejores prácticas empleadas por los líderes de DevOps, según los hallazgos del informe [27]:

- Asignación adecuada de presupuestos de prueba y enfoque en el desarrollo de habilidades.
- Implementación de pruebas continuas para satisfacer las demandas de frecuencia de lanzamiento y respaldar la entrega continua.
- Inclusión de probadores como parte de equipos de entrega integrados.
- Implementación de pruebas tempranas en el desarrollo.

3.3.3. The Research on Software Metrics and Software Complexity Metrics

En este análisis, señala la importancia de integrar métricas de software a lo largo de todo el ciclo de vida del desarrollo de software. El documento argumenta que el proceso de desarrollo de software, que abarca desde los requisitos y diseños hasta los programas y pruebas, puede ser medido y analizado utilizando métodos métricos.
Según el artículo, las métricas de software pueden ser herramientas efectivas para monitorear y mejorar la calidad del producto de software, lo que a su vez facilita la planificación y el mantenimiento futuros [5].

En conclusión, se propone la utilización de métodos métricos adecuados para la evaluación y mejora de la calidad. Se presentan dos métodos de medición de la complejidad del software: el método de McCabe y las métricas orientadas a objetos.

3.3.4. DevOps Metrics-Case Eficode

En este artículo, se enfoca en las métricas de DevOps y cómo pueden ser utilizadas para mejorar y respaldar el trabajo del equipo de desarrollo, teniendo en cuenta los indicadores y datos necesarios para alcanzar los objetivos estratégicos y operativos de una empresa.

Se distinguen dos tipos principales de métricas para DevOps [14]:

Métricas operativas, que se centran en la rentabilidad, la productividad, la programación y la mejora continua.

Métricas estratégicas, alineadas con los objetivos estratégicos de la empresa y enfocadas en la satisfacción del cliente.

En conclusión, las métricas de DevOps están estrechamente ligadas tanto a los objetivos estratégicos como, a la gestión continua del desarrollo por parte del equipo de desarrollo de software.

3.3.5. Metrics-driven devops

En esta investigación, analizan el informe anual "State of DevOps", el cual es una encuesta dirigida a toda la industria TI , proporciona evidencia de que la entrega de software desempeña un papel crucial en las organizaciones impulsadas por la tecnología de alto rendimiento [28].

En la investigación n se identifican dos tipos de métricas aplicables a DevOps:

Las métricas basadas en el sistema: Se refieren a datos provenientes de diversos sistemas de registro que componen un flujo de valor de entrega de software de extremo a extremo del ciclo de vida del software y se evalúan tres criterios:

- **Completitud:** ¿Los datos capturados son lo suficientemente completos como para proporcionar visibilidad, métricas e informes que son el objetivo de la iniciativa?
- **Exhaustividad:** ¿Se capturan suficientes datos?
- **Exactitud:** ¿Están los datos suficientemente correlacionados para ser correctos?

las métricas basadas en encuestas: Se refieren a datos sobre sistemas y personas, estas encuestas se envían a las personas que trabajan en los propios sistemas y están íntimamente familiarizadas con el desarrollo de software y el sistema, se valoran según dos criterios:

- **Cohesión:** Los datos basados en encuestas son particularmente buenos para proporcionar una visión completa y holística de los sistemas.

- **Exactitud:** El diseño y la medición de encuestas se pueden aprovechar para proporcionar buenos datos y conocimientos sobre los sistemas y la cultura.

3.4. Tabla de trabajos relacionados

La Tabla 1 muestra un resumen de las características encontradas respecto al uso del enfoque DevOps entre los trabajos relacionados y el trabajo de esta tesis.

El enfoque DevOps y sus procesos sirven como guía, apoyo y complemento en la gestión, desarrollo y administración del software, como se muestra en la Tabla 1 Por ejemplo, en los trabajos [29] y [21] , se han utilizado los procesos de entrega e integración continua de DevOps para mejorar sus procesos de información y aprendizaje. En los trabajos [24], [28], [15], [12], [6] y [23], se combina el enfoque DevOps con procesos de desarrollo de software, de análisis y evaluaciones, así como con modelos como QUAMOCO y TAPI, para mejorar la productividad y la eficiencia del desarrollo de software.

Por otro lado, en los trabajos [13], [30] y [19], se realizan tablas de análisis del enfoque DevOps con el fin de establecer y dar a conocer los beneficios de adoptar dicho enfoque. Además, los trabajos [27], [5], [14] y [4] presentan guías y recomendaciones para organizaciones TI que deseen implementar el enfoque DevOps en sus desarrollos de software.

Asimismo, los trabajos [26], [31] y [28] proponen métricas que pueden implementarse en los desarrollos de software que se guíen bajo el enfoque DevOps. Es importante destacar que el tablero de control desarrollado en esta tesis está basado en el IEEE Standard para DevOps, y hasta el momento no existe una herramienta que complemente los 8 procesos y el estándar de DevOps. Ninguno de los trabajos relacionados tiene esta característica.

Tabla 1 Comparativa de trabajos relacionados

Referencia	Prácticas *DevOps*	Mejora continua	Métricas e indicadores de calidad	Producto resultante
Karamitsos I[16]	integración y entrega continua	SEMMA	NO	Proceso de aprendizaje automático
Batra P[23]	Entrega y despliegue continuo	No	Productividad y la eficiencia	Método de evaluación
Haindl P[31]	Registro, monitoreo e integración continua	No	QUAMOCO	Modelo de calidad de software
Yang D[18]	Integración e implementación continua	No	No	Sistema de información educativa basado en DevOps
Sánchez M[17]	Medición Automatización Intercambio Cultura	No	No	Tabla de análisis DevOps
Leite L[20]	Definición DevOps	No	No	Tabla de análisis DevOps
Al-Zahrani S[29]	Entrega continua	No	No	Tabla de beneficios DevOps
Groll Jayne[24]	Definición DevOps	No	No	Recomendaciones para equipos ágiles/DevOps
Bass[25]	Cambios rápidos en producción. Automatización de pruebas.	No	No	Recomendaciones para las organizaciones y arquitectos de software en el uso de DevOps
Muñoz M[21]	Medición Automatización Intercambio Cultura	No	No	Guía para la implementación DevOps en organizaciones TI bajo el estándar ISO / IEC 29110
Alnafessah A[32]	Entrega e integración continua	No	No	Análisis DevOps
Marijan D[30]	Entrega e integración continua	No	No	Metodología de optimización integrada en las fases de desarrollo, pruebas y operaciones
Wang Y[19]	Integración continua	No	TAPI	Análisis TAPI aplicado a DevOps
Schafer[27]	Construcción Validaciones funcionales Pruebas de integración Pruebas de regresión	Si	1.Presupuestos de prueba 2.La implementación de pruebas continuas y apoyar la entrega continua 3.La inclusión de probadores 4. Implementar las pruebas antes en el desarrollo	Lista de métricas aplicadas a DevOps
Tu Honglei[5]	No	Si	McCabe y el método métrico C&K	Lista de tipos de métricas De procedimiento De proyecto De complejidad de software
Behm R[14]	Entrega y desarrollo continuo	Si	Métricas operativas Métricas estratégicas	Lista de métricas aplicadas a DevOps

Forsgren N[33]	Practicas comunes de DevOps	No	Métricas basadas en el sistema Métricas basadas en encuestas	Lista de métricas aplicadas a DevOps
Poniszewska-Marańda A[15]	Entrega e integración continua	No	No	Gia de métodos de implementación de clúster en Kubernetes que se pueden aplicar en el enfoque DevOps
Cardoso T[12]	prácticas de CSD (desarrollo continuo de software)	Si	No	Evaluación de los desafíos e impactos de las prácticas de CSD (desarrollo continuo de software)
Basili V[26]	No	No	Nivel conceptual Nivel operativo Nivel cuantitativo	Guía para obtención de métricas basadas en objetivos
Esta tesis	Prácticas más comunes DevOps Planificación Codificación Compilación Pruebas Lanzamiento Implementación Operaciones Monitoreo	Si	Tasa de errores Frecuencia de implementación Tiempo medio de recuperación (MTTR)	Software de control y seguimiento para proyectos de desarrollo de software bajo el esquema DevOps

CAPÍTULO IV

4. Metodología de solución

La metodología de solución propuesta implica el desarrollo de un tablero de control con base en la metodología Scrum, con el objetivo de dar seguimiento a las actividades y los procesos, para los interesados en la gestión de proyectos de software en entornos DevOps. Este tablero de control actúa como una herramienta centralizada que proporciona una visión integral del estado y el progreso de los proyectos, así como de las métricas clave asociadas con la implementación de prácticas DevOps.

4.1. Tabla de análisis de procesos DevOps

En los trabajos previamente analizados se identifica que la mayoría de autores reconocen de manera general cuáles son las prácticas o procesos propios del enfoque DevOps, aunque en muchos casos no establecen una definición del proceso, en la Tabla 2 se presenta un resumen de los procesos identificados en la literatura consultada, lo que brinda una visión general de las prácticas predominantes en el ámbito de DevOps. Esta recopilación permite comprender la amplitud y la diversidad de los procesos empleados en diferentes contextos y proporciona un marco de referencia para el análisis comparativo y la identificación de tendencias en la implementación de DevOps.

Tabla 2 Concentrado de procesos DevOps

Proceso	IEEE [8]	B.Alnamlah [9]	I.Karamitsos [16]	R.Chatley [34]	M.Senapathi [35]	D.Teixeira [36]	V.Ivanov [37]	T.Düllmann [38]
Automatización de construcción	x				x			
Automatización de infraestructura	x				x			
Automatización de la implementación	x				x			
Automatización de métricas	x				x			
Automatización de pruebas	x				x	x		
Automatización de recuperación	x				x			
Automatización de seguimiento	x				x			
Codificación	x					x		
Comunicación y colaboración	x	x						
Construcción	x					x	x	
Desarrollo colaborativo y continuo	x				x			
Despliegue Continuo	x			x	x		x	
Desarrollo	x		x		x			
Entrega continua	x	x	x	x		x	x	x
Gestión de configuración para código e infraestructura	x				x			
Implementación continua	x		x		x	x		x
Infraestructura como código	x	x						
Integración continua	x	x	x	x	x		x	
Inyección de fallos.								x
Lanzamiento	x		x		x	x		
Liberación primaria (Canary Releasing).								

Medición continua	x				x			x
Microservicios:		x						
Monitoreo continuo del comportamiento del usuario y realimentación	x				x			
Monitoreo continuo y mejoramiento	x				x			
Monitoreo.	x		x			x		x
Operación	x					x		
Planeación	x		x			x		
Proceso de adquisición	x							
Proceso de análisis del sistema	x							
Proceso de análisis empresarial o de misión	x							
Proceso de aseguramiento de la calidad	x							
Proceso de definición de arquitectura	x							
Proceso de definición de necesidades y requisitos de las partes interesadas	x							
Proceso de definición de requisitos del sistema / software	x							
Proceso de definición del diseño	x							
Proceso de eliminación	x							
Proceso de evaluación y control del proyecto	x							
Proceso de gestión de decisiones	x							
Proceso de gestión de la calidad	x							
Proceso de gestión de la cartera	x							
Proceso de gestión de la configuración	x							
Proceso de gestión de la información	x							
Proceso de gestión de la infraestructura	x							
Proceso de gestión de recursos humanos	x							
Proceso de gestión de riesgos	x							
Proceso de gestión del conocimiento	x							
Proceso de gestión del modelo de ciclo de vida	x							
Proceso de implementación	x							
Proceso de integración	x							
Proceso de mantenimiento	x							
Proceso de medición	x							
Proceso de operación	x							
Proceso de planificación del proyecto	x							
Proceso de suministro	x							
Proceso de transición	x							
Proceso de validación	x							
Proceso de verificación	x							
Pruebas	x					x	x	
Pruebas A/B.	x							x
Pruebas continuas	x				x			
Recuperación de fallas de servicio sin demora	x				x			
Registro y monitoreo	x	x						
Supervisión	x						x	

4.2. Tabla de análisis de métricas DevOps

En la Tabla 3 se realiza un análisis de métricas e indicadores agrupados por fase, conforme al enfoque DevOps, con el propósito de dar seguimiento al plan de trabajo se presenta el resultado de la información consultada.

Tabla 3 Concentrado de métricas e indicadores DevOps

Fase DevOps	Métrica	indicador	Descripción	Trabajos
Planeación	Cambio de estimación	puntos de historia y cambios entre iteraciones	Mide los cambios realizados en las estimaciones de esfuerzo	[13]
	Cambio de prioridad	*	Mide los cambios de prioridad de PBI's.	[39]
	Completitud de datos	Número de elementos presentes	Datos necesarios y relevantes para un proceso	[40]
	Índice de Mejora	estimaciones de esfuerzo	Muestra cuántos PBI han tenido un cambio de estado positivo,	[39]
	Pronóstico de alcance	*	Se ejecuta el alcance inicial del sprint	[39]
	Tamaño del proyecto Restante	*	Estimación del esfuerzo total que aún debe realizarse	[39]
	Tamaño del proyecto	Número de PBI abiertos	Mide Estimación del esfuerzo total o número de PBI.	[40]
	Esfuerzo por función	Número de funciones	Supervisa que el esfuerzo se centre en funciones de desarrollo y se prioricen las funciones correctas	[40]
	Cohesión	Alto/bajo	Fuerza de relación entre módulos	[41]
Codificación	Errores por día	Número de errores	Presenta el progreso temporal de los errores en el producto	[40]
	Conteo de asignaciones	Número de asignaciones	Son los cambios enviados a la fuente de código principal	[42]
Construcción	Número de pruebas prioritarias	Pruebas automatizadas	*	[8]
	Compilaciones de código exitosas	Número de compilaciones exitosas	*	[8]
	Unidad de aprobación / falla	*	*	[8]
	Número total de defectos identificados	Número total de defectos	*	[8]
	Cobertura de código	*	*	[8]
Pruebas	Registro de fallas en las pruebas	Número total de fallas	El uso de esta métrica cambia mucho según la cultura del equipo de desarrollo	[39]
	Número de pruebas realizadas	Número total de pruebas	La cantidad de pruebas en una única asignación también es vital.	[42]
	Tasa de defectos	Número total de defectos encontrados	*	[8]
Pruebas de integración	Requerimientos cubiertos por pruebas API	Número total de requisitos cubiertos en las pruebas	*	[8]
	Nuevos defectos API encontrados	Número total de defectos encontrados	*	[8]
	Densidad de errores API	*	*	[8]
	Tasa de prueba / error en pruebas API	*	*	[8]
	Cobertura de código funcional /cobertura de riesgos API (tie)	*	*	[8]
Pruebas de regresión	Porcentaje de casos de pruebas automatizadas	Porcentaje del 0 al 100	*	[8]
	Cobertura de defectos en las pruebas	*	*	[8]
	Número total de defectos identificados en las pruebas	Número de defectos encontrados	*	[8]

	Número de casos de prueba ejecutados	Número total de casos de prueba	*	[8]
	Cobertura de casos de prueba	Número de casos cubiertos	*	[8]
Lanzamiento	Retrocesos en la implementación y frecuencia de las fallas	Tiempo en horas	Incluyen la falta del alcance de los requisitos de una aplicación, por orden administrativa y por los errores que podrían haber escapado a las pruebas	[42]
	Pronóstico de alcance	Tasa de mejora y el Tamaño restante del proyecto	Mide qué tan bien se siguen las iteraciones esperadas	[40]
	Tasa de despliegue	Tiempo en horas	Cuando se crea una nueva versión de su aplicación, y la velocidad a la que se producen es una indicación general de la productividad de su equipo	[43]
Despliegue	Velocidad de las implementaciones	Tiempo en horas	Mide el tiempo necesario para completar una sola implementación.	[43]
	Agregado de PBI	Número de nuevos PBI	Número porcentual de nuevos PBI en un sprint	[40]
	Frecuencia de despliegue	Tiempo en horas	*	[8]
Operación	PBI rechazados	Número de eliminados	Eliminación de PBI acumulados debido al cambio de alcance	[44]
	Disponibilidad de la aplicación	Tiempo en horas disponible	Los usuarios deben tener acceso ininterrumpido a su aplicación las 24 horas del día	[43]
	Plazo de ejecución de la versión	Tiempo en horas	Mide el tiempo que transcurre desde que se recibe hasta que se resuelve un ticket	[43]
	Tasa de respuesta a los tickets	Tiempo en horas	Monitorear la tasa de respuesta a los tickets (incidentes descubiertos en software puesto en producción)	[43]
	Volumen de problemas (número de problemas) y tiempo de resolución	Número total	*	[8]
	Disponibilidad de los registros (log del sistema)	Tiempo en horas disponible	*	[8]
Monitoreo	Tráfico y uso de aplicaciones	*	El siguiente enfoque debe ser monitorear el tráfico y el uso de la aplicación	[42]
	El número de entradas	*	Los usuarios proveen comentarios al enviar las entradas a su equipo de soporte	[42]
	Tiempo medio de recuperación (MTTR)	Tiempo en horas	*	[8]
	Tiempo medio de detección (MTTD)	Tiempo en horas	*	[8]
Validación funcional	Requisitos cubiertos por las pruebas	Número total de requisitos	*	[45]
	Conteo crítico de defectos funcionales	Número de defectos	*	[45]
	Tasa de aprobación/ falla	*	*	[45]
	Densidad de errores	*	*	[8]
	Cobertura de riesgos	*	*	[8]
Otros	Exhaustividad	% Porcentaje	Registro de eventos vitales	[41]
	Exactitud	Nivel de correlación de datos	Precisión de los datos	[41]
	Cambio de PBI	Tiempo de cambio en horas	Descripciones cambiadas y cantidad de esfuerzo	[44]
	Horas utilizadas	Tiempo en horas	Indica la división entre diferentes áreas de desarrollo	[46]
	Costos de infraestructura	*	Se utiliza el escalado automatizado y la creación de nuevos entornos	[46]
	Nuevos elementos en Wiki	*	Mide el intercambio de conocimientos dentro de la empresa	[44]
	Tiempo y volumen de cambio	Tiempo en horas	*	[8]
	Tasa de fallos en los cambios	*	*	[8]
	Tiempo de aprobación	Tiempo en horas	*	[8]

	Tiempo para parchear(reparar) las vulnerabilidades	Tiempo en horas	*	[8]
	Cumplimiento del control de retención	*	*	[8]
	Recuento de hallazgos de los requisitos de aseguramiento del software	*	*	[8]
	Tráfico de aplicaciónes	*	*	[8]
	Detalles del vector de ataque (IP, seguimiento de la pila, tiempo, tasa de ataque)	*	*	[8]
	Utilización de recursos	*	*	[8]
*No definido, se menciona de manera general en el enfoque de DevOps				

En resumen, cada fase del ciclo de vida del desarrollo de software, desde la planificación hasta la implementación y el monitoreo, ha sido examinada en busca de métricas específicas que permitan evaluar y mejorar continuamente el proceso. Estas métricas abarcan aspectos como el tiempo de entrega, la frecuencia de despliegue e implementación, la estabilidad del sistema, la calidad del código, la eficiencia operativa y la satisfacción del cliente, entre otros.

4.3. Propuesta de procesos y métricas aplicables

Tras el análisis de la información recopilada, se ha elaborado una lista de métricas e indicadores correspondientes al enfoque DevOps. En la Tabla 4 se presenta detalladamente un listado de estas métricas e indicadores, organizados por fase, que servirán como referencia durante el desarrollo de la herramienta de tablero de control.

Tabla 4 Propuesta de métricas e indicadores

Fase DevOps	Métrica aplicable	Tipo de escala	Fórmula	Indicador aplicable	Descripción
Planificación	Esfuerzo por puntos de historia	De proporción, cociente o razón	Se establecen mediante un consenso	El número de puntos de historias que están siendo completadas, ya sea por día, por semana o en toda una iteración	El número de puntos de historias completadas por proyecto será la suma de los puntos de historia por cada iteración [10].
Codificación	Tasa de defectos	De proporción, cociente o razón	TD = Númdefectos / líneas-código X100	Es la suma total de defectos encontrados a lo largo del ciclo de vida en DevOps	Presenta el progreso temporal de los defectos en el producto [8].
Construcción	Tasa de compilaciones de código exitosas	De proporción, cociente o razón	TCE = NúmTotalCompiExito / NumTotalCompilaciones X100	Es el número de compilaciones de código exitosas	Esta métrica ayuda a llevar un control del código [8].
Pruebas	Tasa de éxito de pruebas	De proporción, cociente o razón	TasaExt=NúmTotalPrueExito / NúmTotalPruebas X100	Mide el porcentaje de pruebas que se ejecutan sin errores	Una alta tasa de éxito indica una buena calidad de las pruebas y del software [22].
Lanzamiento	Tiempo promedio de despliegue	De proporción, cociente o razón	TD = TiempoTotal / NúmTotalLanzamientos	Mide el tiempo promedio de despliegue por lanzamiento.	Un tiempo de despliegue corto indica una entrega rápida y eficiente del software [16].
Despliegue	Frecuencia de despliegue	De proporción, cociente o razón	FD=NúmeroTotalDespliegues / TiempoTotal	Mide el tiempo en horas necesario para completar una sola implementación en producción	La frecuencia de despliegue mide qué tanto se acerca o se aleja el objetivo [8].
Operación	Tiempo medio de detección (MTTD)	De proporción, cociente o razón	Tiempo total para detectar todos los errores durante un período de tiempo=Td Número total de errores=Ne MTTD= Td / Ne	Es el tiempo en horas que tarda el equipo en descubrir una incidencia en un ambiente productivo.	Cuando ocurren problemas, es importante identificarlos rápidamente [8].
Monitoreo	Tiempo medio de recuperación (MTTR)	De proporción, cociente o razón	Tiempo total desde el descubrimiento de la falla hasta la resolución en horas=Td Número total de reparaciones=Nr MTTR= Td / Nr	Es el tiempo en horas que se tarda en reparar un sistema defectuoso	Esta métrica ayuda a rastrear cuánto tiempo lleva recuperarse de los fallos [8].

4.4. Metodología de desarrollo para el tablero de control DevOps

4.4.1. Sprint 0

En Scrum, las épicas surgen al inicio del proyecto cuando las historias de usuario representan funcionalidades de alto nivel que requieren una descripción inicial sin detallar [11]. Estas épicas deben desglosarse en historias más pequeñas y manejables que puedan gestionarse de acuerdo con los principios y técnicas ágiles. En la Tabla 5 se presenta la épica del tablero de control, la cual ha sido elaborada tras el análisis de todas las funcionalidades requeridas.

Tabla 5 Épica para este proyecto tablero de control DevOps

Épica: Tablero de control para la gestión de proyectos DevOps					
ID: HU01	Prioridad: Alta	Tamaño: XL	Tipo: Funcional	Técnico: Software	Iteración: 1
Narrativa					
Desarrollar una herramienta de software con base en tecnologías de código abierto, que permita dar seguimiento y evaluar el cumplimiento de los objetivos de calidad en el desarrollo de software. COMO Administrador de proyectos de software bajo el enfoque DevOps QUIERO Un tablero de control que permita administrar y visualizar gráficamente proyectos de software bajo el enfoque DevOps PARA Dar seguimiento a los proyectos de software y evaluar el cumplimiento de los objetivos de calidad en el desarrollo de dichos proyectos con la adopción de prácticas de DevOps					
Notas					
· Se considera una métrica por proceso					
Criterios de Aceptación					
· La herramienta debe mostrar gráficos de procesos propuestos asociada a cada proyecto. · De acuerdo con la evolución del proyecto se visualizan las gráficas correspondientes a cada fase evaluada. · Cada gráfica debe mostrar el comportamiento de los proyectos de cada una de las fases propuestas de acuerdo con las métricas e indicadores asignados. · La combinación de colores en la interfaz debe ser amigable con el usuario. · El sistema debe ser responsivo. · El sistema recibe de manera manual los datos de entrada en cada proceso del enfoque DevOps.					
Tareas					
· Para el proceso de Planeación realizar el análisis de los requerimientos · Para el proceso de Codificación diseñar la arquitectura del tablero de control e interfaz de usuario · Para el proceso de Construcción codificar en un lenguaje de programación el tablero de control · Para el proceso de Pruebas realizar pruebas automatizadas unitarias, funcionales y de integración del tablero de control · Para el proceso de Lanzamiento integrar las versiones de código · Para el proceso de Despliegue implementar una versión del tablero de control en el servidor · Para el proceso de Operación monitorear el comportamiento del tablero de control productivo · Para el proceso de Monitoreo recopilar información de los resultados obtenidos a lo largo del proyecto · Definir indicadores y métricas para cada una de las prácticas o etapas DevOps. · Describir las prácticas DevOps que se van a medir (8 prácticas actuales) · Establecer los indicadores de desempeño para las prácticas DevOps. · Establecer las métricas para el proceso de cada práctica.					

4.5. Product backlog general para el tablero de control DevOps

La Figura 1 muestra el diagrama del Product Backlog general para el tablero de control DevOps.

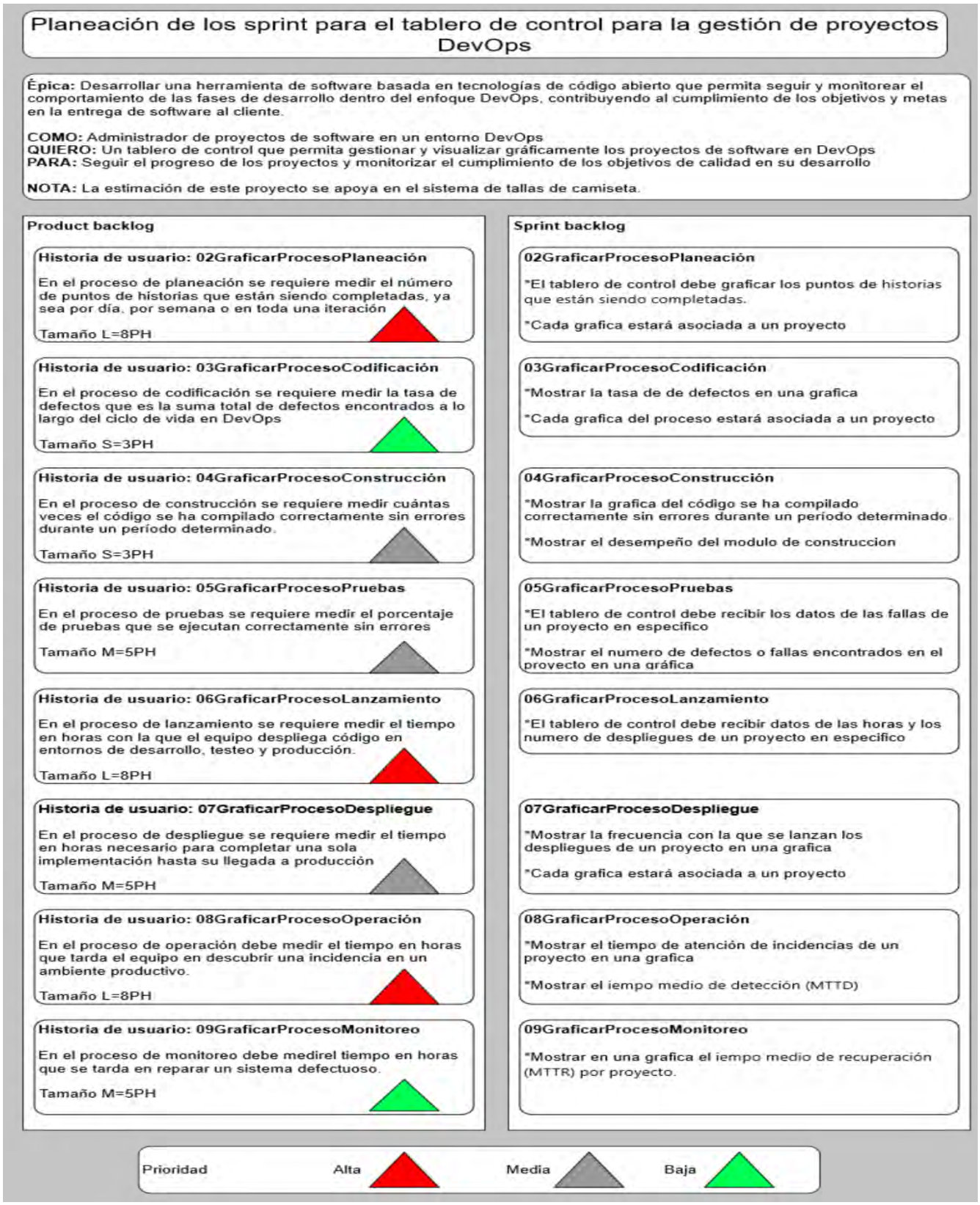

Figura 1 Product Backlog general para el tablero de control DevOps

4.6. Diagrama de despliegue

La Figura 2 tiene por objetivo visualizar la arquitectura, los componentes y enlaces de comunicación del tablero de control DevOps.

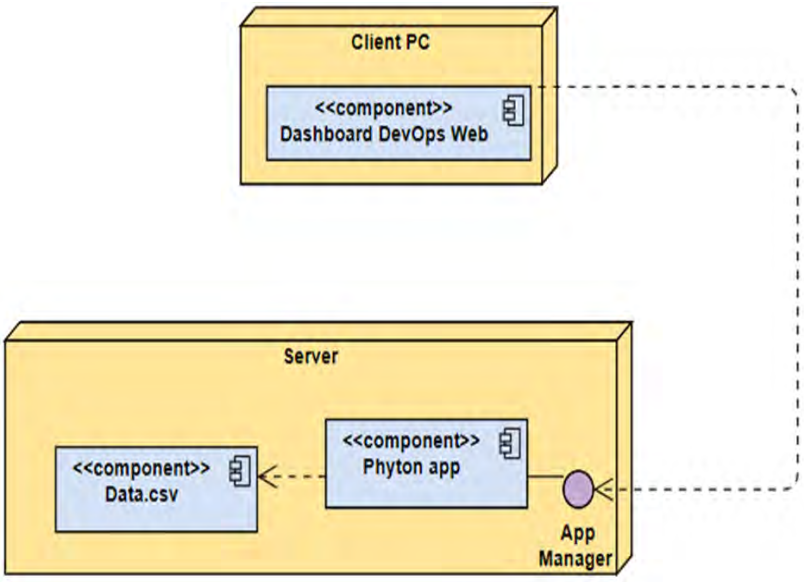

Figura 2 Diagrama de despliegue del tablero de control DevOps

Se invita al lector a explorar el apartado de anexos para obtener una perspectiva más profunda sobre los temas discutidos en este capítulo, así como para acceder a recursos adicionales que pueden ser útiles para la comprensión y aplicación de los conceptos presentados.

CAPÍTULO V

5. Resultados

En este capítulo se presentan los resultados obtenidos a partir del análisis de datos recopilados y de la aplicación de metodologías mencionadas con anterioridad.

5.1. Dataset para la implementación de indicadores y métricas del sistema

Para mostrar y probar el sistema se han cargado datos publicados por ISBSG que es una organización, con sede en Australia. tiene como objetivo proporcionar a las organizaciones información valiosa y confiable sobre el desempeño y la productividad en el desarrollo y mantenimiento de software. Empresas de TI de todo el mundo comparten los datos de sus proyectos con ISBSG, mantiene dos repositorios de datos: uno para proyectos de desarrollo y otro para proyectos de mantenimiento y soporte [47].

La Tabla 6 muestra los datos de treinta y un proyectos utilizados para las pruebas del tablero de control.

Tabla 6 Datos ISBSG para las pruebas de la herramienta de control y seguimiento del proceso DevOps

Núm.	ID del proyecto SBSG	Esfuerzo total de M&S	Unidad de esfuerzo	KSLOC total, ingresado	Defectos totales	Velocidad de entrega	Plan de Esfuerzo	Effort Build (construcción)	Effort Test (pruebas)
1	10198	464	horas	8	23	15.2	8	0	6
2	10200	544	horas	9	20	170.4	2	7	4
3	10221	311.5	horas	115	8	57	6	45	5
4	10228	1788	horas	4	75	88.6	9	55	4
5	10314	294	horas	55	0	8.1	1188	24497	17620
6	10322	306	horas	20	14	160.2	4	9	28
7	10327	22423	horas	4	1311	59.4	327	4700	2038
8	10331	423	horas	5	39	18.1	7	2	51
9	10337	1012	horas	393.696	463.91	10	25	224	34
10	10341	430	horas	33	54	57.1	51	2	15
11	10346	230	horas	0	21	57.3	6	19	2
12	10377	3120	horas	1000	1793.121	17.4	4	9	12
13	10409	1987	horas	0	30.866	35.5	56	308	784
14	10452	172	horas	7	11	19.3	4	3	55
15	10464	1369	horas	63	223	8	40	7	77
16	10467	221	horas	5	9	5	86	7	6
17	10480	1608	horas	89	59	6	91		44
18	10495	1314	horas	86	188	9	70	701	5
19	10512	174	horas	1067	20	3	0		33
20	10517	4	horas	38.923	45	34.1		1198.9	413.4
21	10566	40780	Horas	4.5	6	12.2			1
22	10589	1440	horas	86	232	3			0
23	10610	1618	horas	0	97	89.8			8

24	10952	174	horas	57	687.379	30.8			3
25	11248	6306	horas	5	288	31		725	973
26	11091	1218	horas	7	205	62		456	8
27	11100	1079	horas	1	51	1		842	5
28	11334	358	horas	0	13	3.1	97	1547	2100
29	11603	20	horas	0.959	0	74	98	3165.2	2101
30	11621	379	horas	0	2	195.9	99	3413.5	2102
31	11898	279	horas	0	0	16.7	100	3413.5	2103

5.2. Características generales del Tablero de control para DevOps

El tablero de control tiene como objetivo proporcionar una herramienta para visualizar los datos de manera efectiva, lo que facilita la interpretación y comunicación de información importante. En este tablero de control se pueden representar diversos tipos de información de los proyectos desarrollados bajo el esquema DevOps.

En la Figura 3 se puede visualizar de manera general el tablero de control desarrollado para DevOps

Figura 3 Pantalla de inicio

El tablero de control genera gráficas en formato PNG, incluyendo gráficos de barras, gráficos de líneas, gráficos de dispersión, gráficos circulares, entre otros.
Esto permite al usuario seleccionar el tipo de gráfica más adecuado para visualizar sus datos y utilizar las gráficas en presentaciones, informes u otros documentos.
Beneficios del uso del tablero de control DevOps:

- Facilita la visualización y comprensión de datos.
- Permite tomar decisiones informadas basadas en la información presentada de manera visual.
- Mejora la comunicación de resultados y hallazgos a través de gráficas claras y concisas.

Aplicaciones potenciales:

- Análisis de datos.
- Presentaciones de informes y resultados de investigaciones.
- Visualización de tendencias y patrones.
- Monitoreo de métricas.

La Figura 4 corresponde la fase de planeación en DevOps, muestra el número de puntos de historia para los proyectos con id 10589, 12907, 10341, 11100 y 10467 que han sido completados

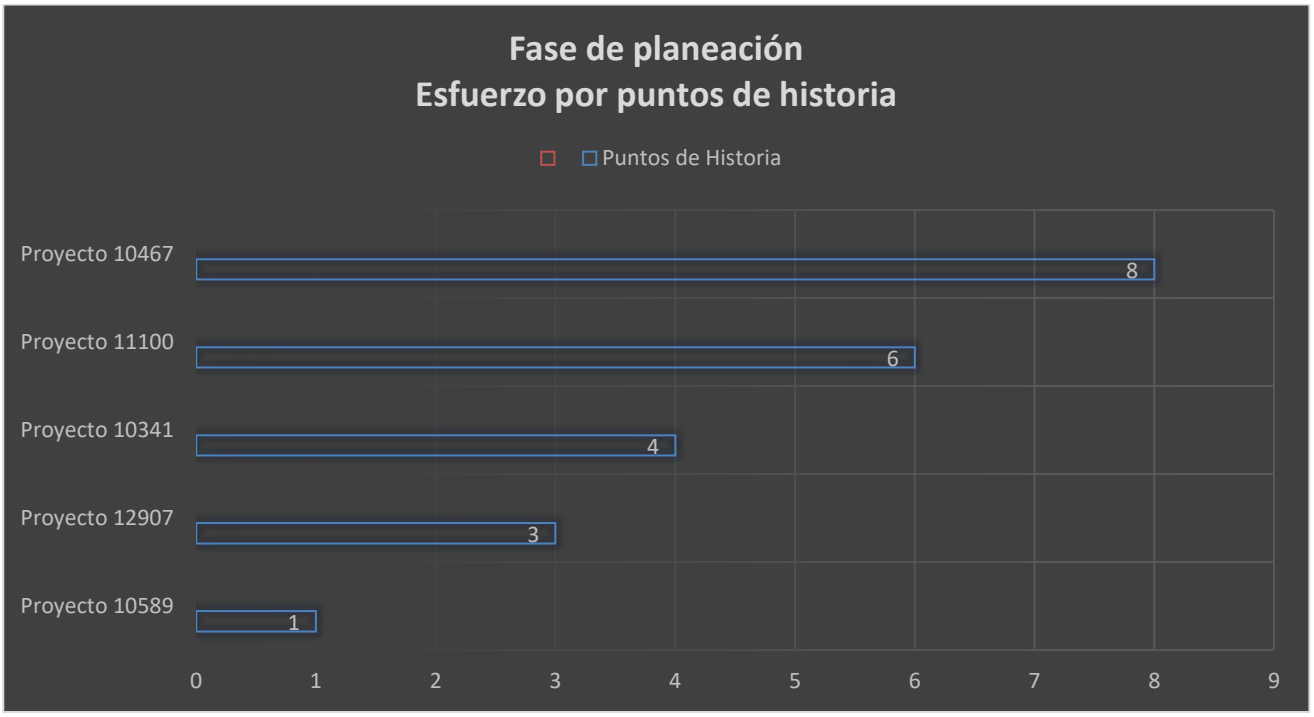

Figura 4 Esfuerzo por puntos de historia en la fase de planeación

La Figura 5 corresponde a la fase de codificación en DevOps, muestra la tasa de defectos para los proyectos con id 10337, 10495 y 10314

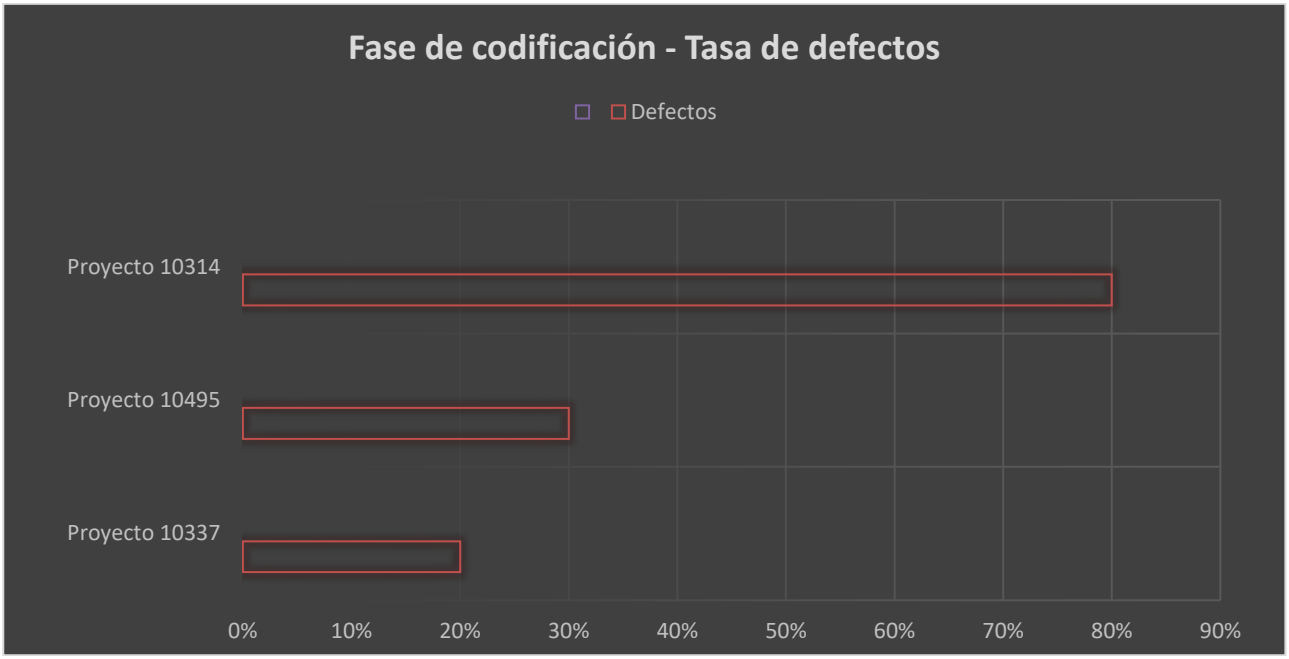

Figura 5 Tasa de defectos en la fase de codificación

La Figura 6 corresponde a la fase de construcción en DevOps, muestra la tasa compilaciones de código exitosas para los proyectos con id 10517, 11603, 11334, 10377 y 10200

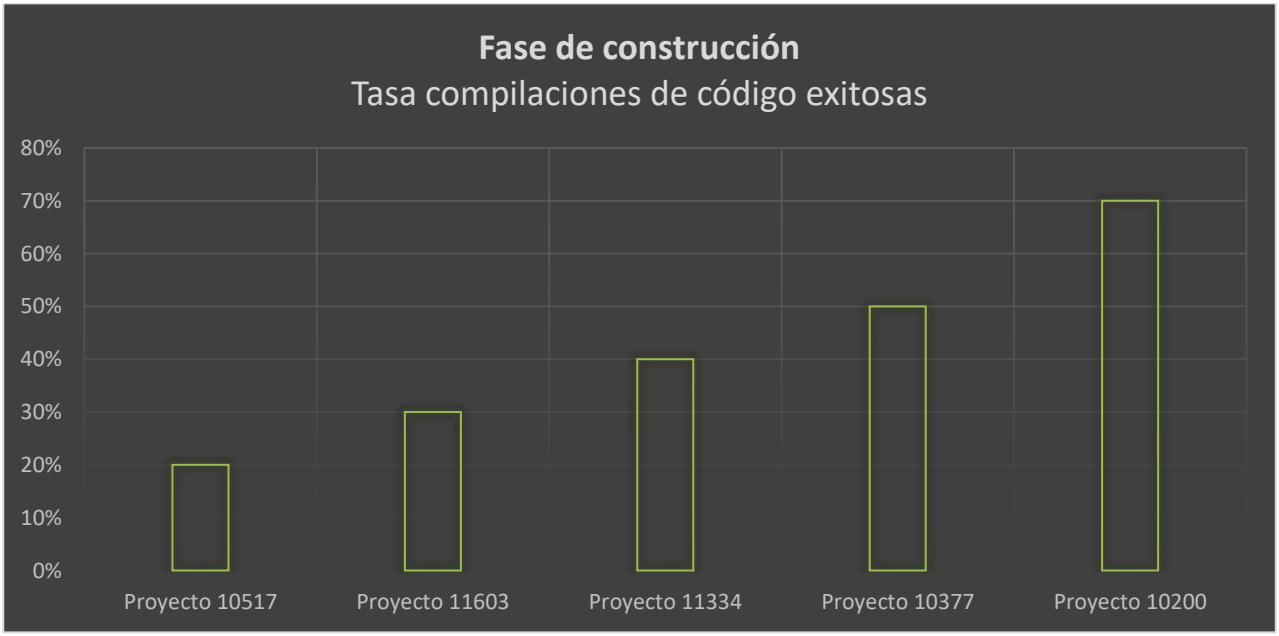

Figura 6 Tasa compilaciones de código exitosas en la fase de construcción

La Figura 7 corresponde a la fase de pruebas en DevOps, muestra la tasa de éxito de pruebas para los proyectos con id 10464 y 10512

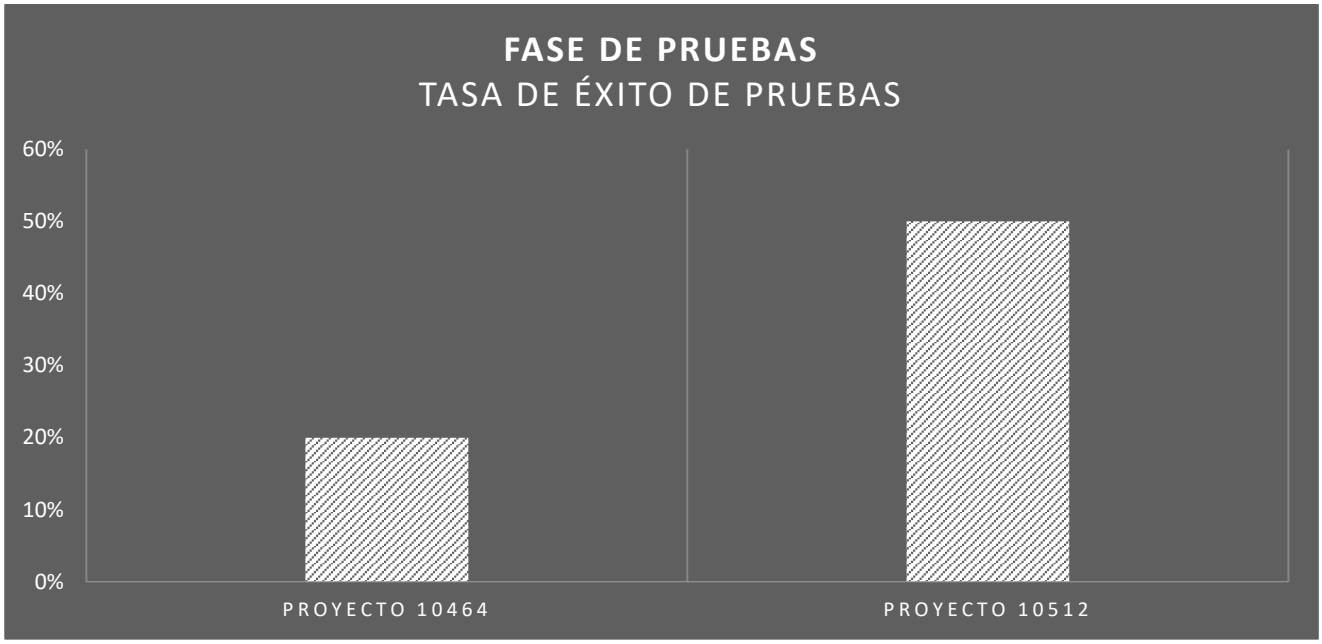

Figura 7 Tasa de éxito de pruebas en la fase de pruebas

La Figura 8 corresponde a la fase de lanzamiento en DevOps, muestra el tiempo promedio de despliegue de los proyectos con id 11898, 10314, 10566, 10377 y 12907

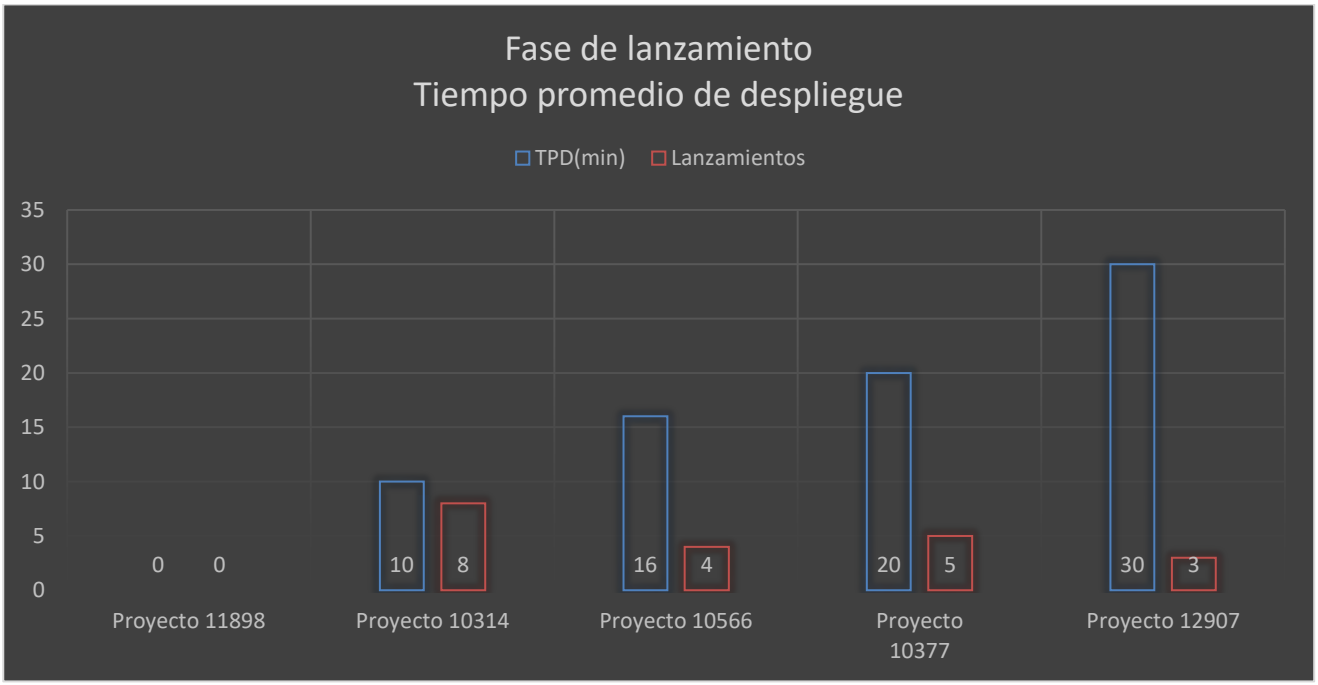

Figura 8 Tiempo de despliegue en la fase de lanzamiento

La Figura 9 corresponde a la fase de despliegue en DevOps, muestra la frecuencia de despliegues para los proyectos con id 11334 y 10314

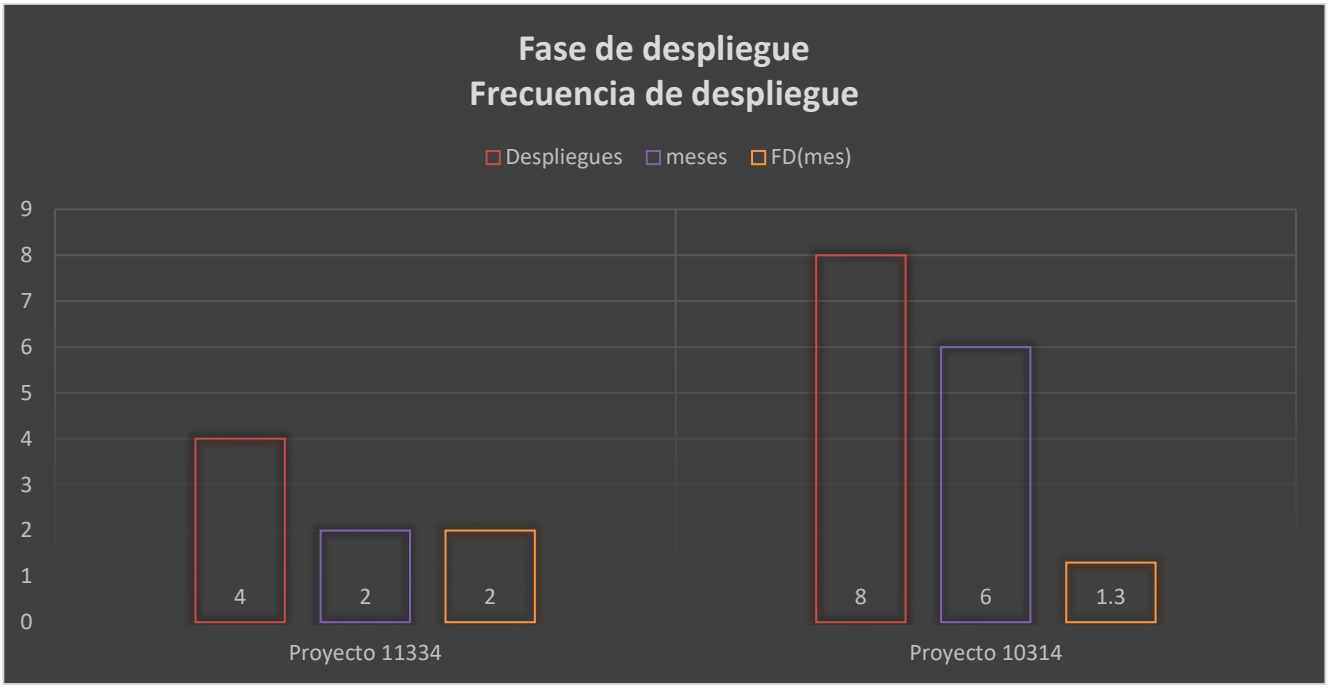

Figura 9 Frecuencia de despliegues en la fase de despliegue

La Figura 10 corresponde a la fase de operación en DevOps, muestra el tiempo medio de detección (MTTD) para el proyecto con id 10327

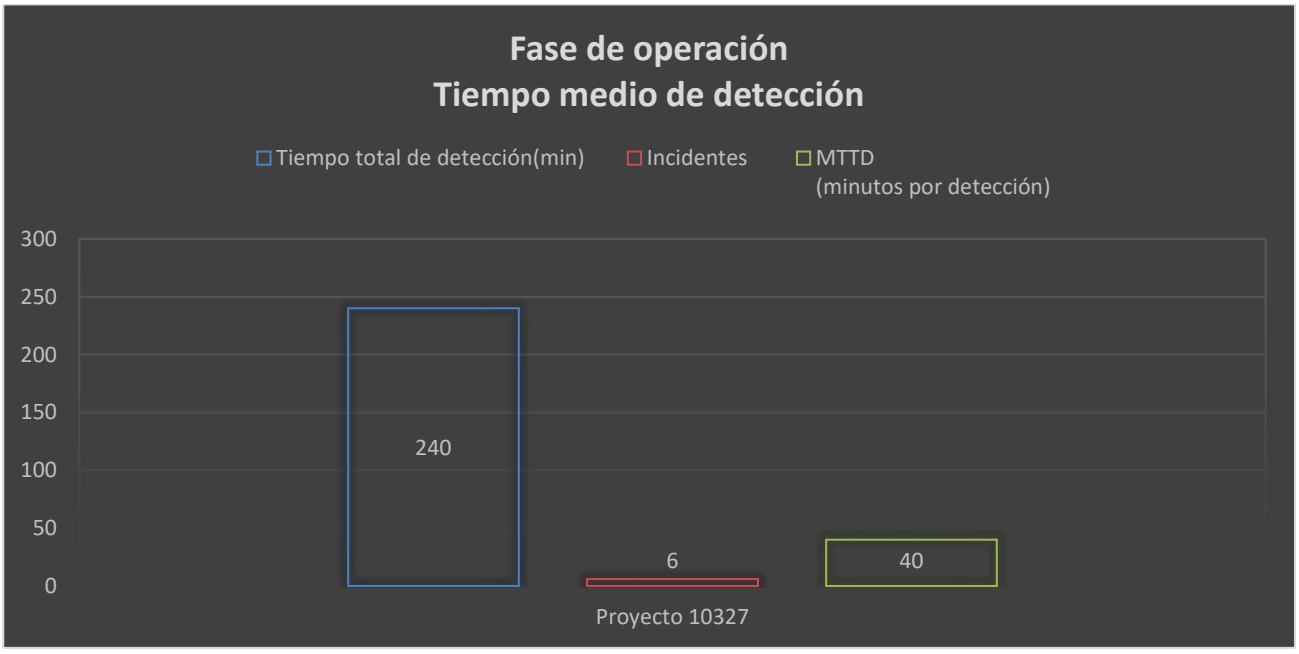

Figura 10 Tiempo medio de detección (MTTD) en la fase de operación

La Figura 11 corresponde a la fase monitoreo en DevOps, muestra el tiempo medio de recuperación (MTTR) para el proyecto con

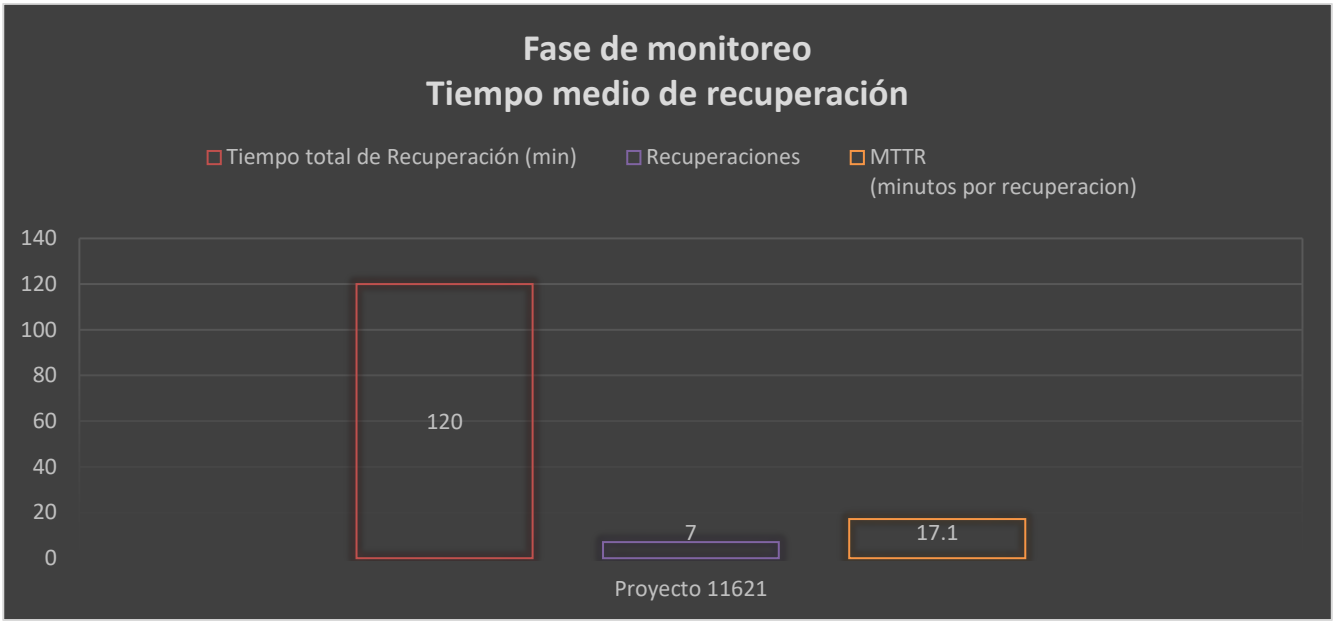

Figura 11 Tiempo medio de recuperación (MTTR) para la fase de monitoreo

La Figura 12 muestra las métricas para DevOps implementadas en el tablero de control para esta investigación

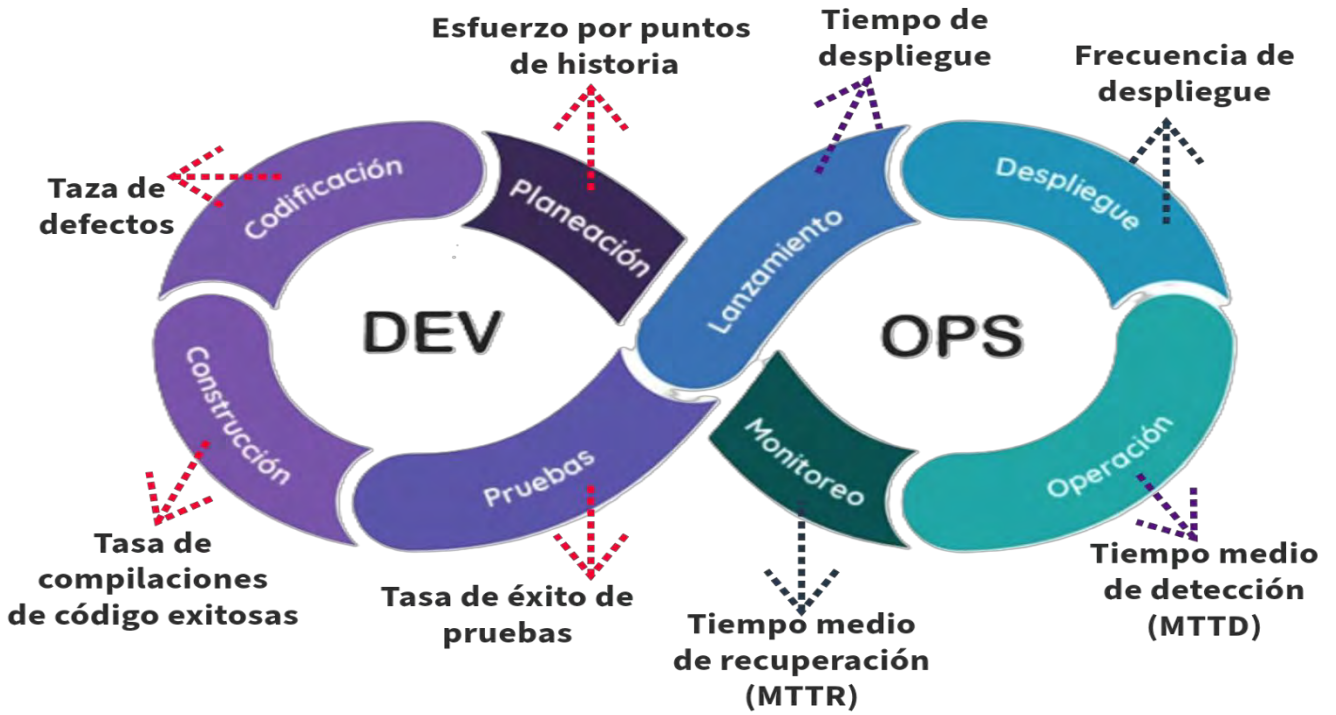

Figura 12 Métricas aplicadas al tablero de control

CAPÍTULO VI

6. Resultados y trabajos futuros

6.1. Conclusiones

El uso del tablero de control con métricas DevOps realizado como resultado de esta investigación proporciona beneficios que incluyen transparencia, visibilidad y optimización del ciclo de desarrollo de software. El análisis y la selección de prácticas ampliamente utilizadas por la industria y las mencionadas en el estándar IEEE 2675:2021 para DevOps, dan como resultado que los indicadores y métricas propuestos y detallados en la metodología de solución, específicamente en la Tabla 4, además de ser relevantes y útiles, cumplen con los objetivos planeados en este proyecto de la siguiente manera:

Para cumplir con el objetivo específico: "Establecer indicadores de desempeño para las prácticas DevOps", los procesos de desarrollo de software e indicadores de desempeño se identificaron mediante el análisis expuesto en la Tabla 2 del capítulo 4 de los cuales se tomaron los siguientes: Planificación, codificación, construcción, pruebas, lanzamiento, despliegue, operación y monitoreo.

Para cumplir con el objetivo específico: "Establecer métricas para el proceso de cada práctica", el análisis que ofrece la Tabla 3 del capítulo 4 muestra las métricas e indicadores agrupados por fase de los cuales se tomaron los siguientes: Esfuerzo por puntos de historia, tasa de defectos, tasa compilaciones de código exitosas, tasa de éxito de pruebas, tiempo de despliegue, frecuencia de despliegue, tiempo medio de detección (MTTD) y tiempo medio de recuperación (MTTR).

Para cumplir con el objetivo específico: "Dar seguimiento a las actividades de los procesos mediante los resultados de las métricas e indicadores propuestos". Con el desarrollo e implementación de los indicadores y métricas propuestos descritos en la Tabla 4 del capítulo 4 fue posible monitorear y observar el estado de proyectos de desarrollo, mostrando en su proceso en cada etapa.

De manera general al cumplir con los objetivos descritos anteriormente, se puede medir el desempeño de prácticas DevOps mediante el uso del tablero de control que da seguimiento a los procesos del ciclo de vida del desarrollo de software para proponer oportunidades de mejora en la gestión de proyectos de software.

6.2. Trabajos futuros

A continuación, se presentan algunas posibles investigaciones y trabajos futuros relacionados.

Continuar mejorando el tablero de control desarrollado en este proyecto, incorporando más indicadores y métricas en áreas como la seguridad, rendimiento del equipo, calidad del código también graficas más avanzadas para optimizar su usabilidad y rendimiento.

Investigar estrategias y herramientas para mejorar la automatización de pruebas y la integración de seguridad en los pipelines de entrega continua, con el objetivo de mejorar la calidad del software y la seguridad de los sistemas.

Hacer uso de la IA generativa para la generación de narrativas de propuestas de mejora continua en cada fase donde se implementan las métricas e indicadores.

Bibliografía

[1] I. Kumara *et al.*, "The do's and don'ts of infrastructure code: A systematic gray literature review", *Inf Softw Technol*, vol. 137, sep. 2021, doi: 10.1016/j.infsof.2021.106593.

[2] Pulasthi Perera, Roshali Silva, Indika Perera, y Sri Lanka pulasthioshan@gmail.com roshalisilva@gmail.com indika@cse.mrt.ac.lk, *Improve Software Quality through Practicing DevOps*. 2017.

[3] "DevOps como impulsor de la innovación - Ibermatica digital". Consultado: el 18 de junio de 2021. [En línea]. Disponible en: https://ibermaticadigital.com/devops-como-impulsor-de-la-innovacion/

[4] A. Abran, *Software metrics and software metrology*. Wiley, 2010.

[5] T. Honglei, S. Wei, y Z. Yanan, "The Research on Software Metrics and Software Complexity Metrics", en *2009 International Forum on Computer Science-Technology and Applications*, 2009, pp. 131–136. doi: 10.1109/IFCSTA.2009.39.

[6] Pressman Roger, *Ingeniería de software un enfoque practico*, Septima edicion.

[7] "ISO - ISO/IEC/IEEE 12207:2017 - Systems and software engineering — Software life cycle processes". Consultado: el 6 de junio de 2021. [En línea]. Disponible en: https://www.iso.org/standard/63712.html

[8] "IEEE Standard for DevOps:Building Reliable and Secure Systems Including Application Build, Package, and Deployment", *IEEE Std 2675-2021*, pp. 1–91, 2021, doi: 10.1109/IEEESTD.2021.9415476.

[9] B. Alnamlah, S. Alshathry, N. Alkassim, y N. S. M. Jamail, "The necessity of a lead person to monitor development stages of the DevOps pipeline", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, núm. 1, pp. 348–353, 2021, doi: 10.11591/IJEECS.V21.I1.PP348-353.

[10] "Software Development with Scrum- A Bibliometric".

[11] *A guide to the Scrum Body of knowledge (SBOK Guide).*

[12] T. E. Cardoso, A. R. Santos, R. Chanin, y A. Sales, "Communication of Changes in Continuous Software Development", *11th International Conference on Software Business, ICSOB 2020*, vol. 407. Springer Science and Business Media Deutschland GmbH, School of Technology, PUCRS, Porto Alegre, RS, 90619-900, Brazil, pp. 86–101, 2021. doi: 10.1007/978-3-030-67292-8_7.

[13] A. Alnafessah, A. U. Gias, R. Wang, L. Zhu, G. Casale, y A. Filieri, "Quality-Aware DevOps research: where do we stand?", *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3064867.

[14] R. Behm, P. Peura, y E. Uotila, "DevOps Metrics-Case Eficode DevOps Metrics-Case Eficode View project Smart Homes and In-Home Care for Elderly and People with Disabilities View project", doi: 10.13140/RG.2.2.19368.19207.

[15] A. Poniszewska-Marańda y E. Czechowska, "Kubernetes cluster for automating software production environment", *Sensors*, vol. 21, núm. 5, pp. 1–24, mar. 2021, doi: 10.3390/s21051910.

[16] I. Karamitsos, S. Albarhami, y C. Apostolopoulos, "Applying devops practices of continuous automation for machine learning", *Information (Switzerland)*, vol. 11, núm. 7, pp. 1–15, 2020, doi: 10.3390/info11070363.

[17] M. Sánchez-Gordón y R. Colomo-Palacios, "Characterizing DevOps culture: A systematic literature review", *18th International Conference on Software Process Improvement and*

Capability Determination, SPICE 2018, vol. 918. Springer Verlag, Østfold University College, Halden, 1757, Norway, pp. 3–15, 2018. doi: 10.1007/978-3-030-00623-5_1.
[18] D. Yang *et al.*, "DevOps in practice for education management information system at ECNU", en *24th KES International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, KES 2020*, M. Cristani, C. Toro, C. Zanni-Merk, R. J. Howlett, L. C. Jain, y L. C. Jain, Eds., Elsevier B.V., 2020, pp. 1382–1391. doi: 10.1016/j.procs.2020.09.148.
[19] Y. Wang, M. Pyhajarvi, y M. V Mantyla, "Test Automation Process Improvement in a DevOps Team: Experience Report", en *13th IEEE International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2020*, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 314–321. doi: 10.1109/ICSTW50294.2020.00057.
[20] L. Leite, C. Rocha, F. Kon, D. Milojicic, y P. Meirelles, "A survey of DevOps concepts and challenges", *ACM Comput Surv*, vol. 52, núm. 6, 2019, doi: 10.1145/3359981.
[21] M. Muñoz, M. Negrete, y M. Arcilla-Cobián, "Using a platform based on the basic profile of iso/iec 29110 to reinforce devops environments", *Journal of Universal Computer Science*, vol. 27, núm. 2, pp. 91–110, 2021, doi: 10.3897/jucs.65080.
[22] P. Haindl y R. Plosch, "Towards Continuous Quality: Measuring and Evaluating Feature-Dependent Non-Functional Requirements in DevOps", en *2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019*, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 91–94. doi: 10.1109/ICSA-C.2019.00024.
[23] P. Batra y A. Jatain, "Measurement Based Performance Evaluation of DevOps", en *2020 International Conference on Computational Performance Evaluation, ComPE 2020*, S. Paul y J. K. Verma, Eds., Institute of Electrical and Electronics Engineers Inc., 2020, pp. 757–760. doi: 10.1109/ComPE49325.2020.9200149.
[24] Groll Jayne, "The Convergence of Scrum and DevOps for an Agile IT Organization", 2017. [En línea]. Disponible en: https://puppet.com/resources/whitepaper/state-of-devops-report.
[25] L. Bass, "The Software Architect and DevOps", *IEEE Softw*, vol. 35, núm. 1, pp. 8–10, 2017, doi: 10.1109/MS.2017.4541051.
[26] V. R. Basili, G. Caldiera, y H. D. Rombach, "THE GOAL QUESTION METRIC APPROACH".
[27] Ian C. Schafer, "Defining software quality metrics for Agile and DevOps - SD Times".
[28] A. Lê-Quôc, "Metrics-driven devops", *Cutter IT Journal*, vol. 24, núm. 12, pp. 24–29, 2011, [En línea]. Disponible en: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84860305087&partnerID=40&md5=4b48104f07181f27139494db2c75dab6
[29] S. Al-Zahrani y B. Fakieh, "How devops practices support digital transformation", *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, núm. 3, pp. 2780–2788, 2020, doi: 10.30534/ijatcse/2020/46932020.
[30] D. Marijan, M. Liaaen, y S. Sen, "DevOps Improvements for Reduced Cycle Times with Integrated Test Optimizations for Continuous Integration", en *42nd IEEE Computer Software and Applications Conference, COMPSAC 2018*, L. C.-H., T. Conte, L. Liu, T. Akiyama, K. Hasan, E. Tovar, H. Takakura, W. Claycomb, S. Cimato, Y. J.-J., Z. Zhang, S. I. Ahamed, S. Reisman, C. Demartini, y M. Nakamura, Eds., IEEE Computer Society, 2018, pp. 22–27. doi: 10.1109/COMPSAC.2018.00012.
[31] P. Haindl y R. Plosch, "Focus Areas, Themes, and Objectives of Non-Functional Requirements in DevOps: A Systematic Mapping Study", en *46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, A. Martini, M. Wimmer, y A. Skavhaug,

Eds., Institute of Electrical and Electronics Engineers Inc., 2020, pp. 394–403. doi: 10.1109/SEAA51224.2020.00071.
[32] A. Alnafessah, A. U. Gias, R. Wang, L. Zhu, G. Casale, y A. Filieri, “Quality-Aware DevOps research: where do we stand?”, *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3064867.
[33] N. Forsgren y M. Kersten, “DevOps metrics”, *Commun ACM*, vol. 61, núm. 4, pp. 44–48, abr. 2018, doi: 10.1145/3159169.
[34] R. Chatley y I. Procaccini, “Threading DevOps Practices through a University Software Engineering Programme”, en *32nd IEEE Conference on Software Engineering Education and Training, CSEE and T 2020*, M. Daun, E. Hochmuller, S. Krusche, B. Brugge, y B. Tenbergen, Eds., Institute of Electrical and Electronics Engineers Inc., 2020, pp. 90–94. doi: 10.1109/CSEET49119.2020.9206211.
[35] M. Senapathi, J. Buchan, H. Osman, y S. S. Innovation, “DevOps capabilities, practices, and challenges: Insights from a case study”, en *22nd International Conference on Evaluation and Assessment in Software Engineering, EASE 2018*, Association for Computing Machinery, 2018. doi: 10.1145/3210459.3210465.
[36] D. Teixeira, R. Pereira, T. A. Henriques, M. Silva, y J. Faustino, “A Systematic Literature Review on DevOps Capabilities and Areas”, *International Journal of Human Capital and Information Technology Professionals*, vol. 11, núm. 2, pp. 1–22, feb. 2020, doi: 10.4018/ijhcitp.2020040101.
[37] V. Ivanov y K. Smolander, “Implementation of a DevOps pipeline for serverless applications”, *19th International Conference on Product-Focused Software Process Improvement, PROFES 2018*, vol. 11271 LNCS. Springer Verlag, Dream Broker Oy, Helsinki, Finland, pp. 48–64, 2018. doi: 10.1007/978-3-030-03673-7_4.
[38] T. F. Düllmann, C. Paule, A. Van Hoorn, y A. C. M. S. I. C. S. I. T. C. on S. E. (TCSE), “Exploiting devops practices for dependable and secure continuous delivery pipelines”, en *4th ACM/IEEE International Workshop on Rapid Continuous Software Engineering, RCoSE 2018*, IEEE Computer Society, 2018, pp. 27–30. doi: 10.1145/3194760.3194763.
[39] A. Mishra y Z. Otaiwi, *DevOps and software quality: A systematic mapping*, vol. 38. Elsevier Ireland Ltd, 2020. doi: 10.1016/j.cosrev.2020.100308.
[40] W. Mallouli, A. R. Cavalli, A. Bagnato, E. M. de Oca, y C. and C. (INSTICC) Institute for Systems and Technologies of Information, “Metrics-driven devsecops”, en *15th International Conference on Software Technologies, ICSOFT 2020*, van S. M, F. H.-G., L. Maciaszek, y L. Maciaszek, Eds., SciTePress, 2020, pp. 228–233. [En línea]. Disponible en: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85091794103&partnerID=40&md5=9220d3db45831f50a86e4c6ca7941448
[41] N. Herbst *et al.*, “Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges”, *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 3, núm. 4, 2018, doi: 10.1145/3236332.
[42] D. Taibi, V. Lenarduzzi, y C. Pahl, “Continuous architecting with microservices and DevOps: A systematic mapping study”, *8th International Conference on Cloud Computing and Services Science, CLOSER 2018*, vol. 1073. Springer Verlag, Tampere University, Tampere, Finland, pp. 126–151, 2019. doi: 10.1007/978-3-030-29193-8_7.
[43] J. Cito, G. Mazlami, P. Leitner, y A. C. M. SIGSOFT, “TemPerf: Temporal correlation between performance metrics and source code”, en *2nd International Workshop on Quality-Aware DevOps, QUDOS 2016, co-located with the International Symposium on Software Testing and*

Analysis, ISSTA 2016, D. Ardagna, F. Willnecker, van H. A, y G. Casale, Eds., Association for Computing Machinery, Inc, 2016, pp. 46–47. doi: 10.1145/2945408.2945420.
[44] H. Huijgens *et al.*, "Strong agile metrics: Mining log data to determine predictive power of software metrics for continuous delivery teams", en *11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2017*, A. Zisman, E. Bodden, W. Schafer, y van D. A, Eds., Association for Computing Machinery, 2017, pp. 866–871. doi: 10.1145/3106237.3117779.
[45] D. Sun, M. Fu, L. Zhu, G. Li, y Q. Lu, "Non-Intrusive Anomaly Detection with Streaming Performance Metrics and Logs for DevOps in Public Clouds: A Case Study in AWS", *IEEE Trans Emerg Top Comput*, vol. 4, núm. 2, pp. 278–289, 2016, doi: 10.1109/TETC.2016.2520883.
[46] M. Farshchi, J.-G. Schneider, I. Weber, y J. Grundy, "Metric selection and anomaly detection for cloud operations using log and metric correlation analysis", *Journal of Systems and Software*, vol. 137, pp. 531–549, 2018, doi: 10.1016/j.jss.2017.03.012.
[47] not-for-profit organization, "ISBSG", https://www.isbsg.org/.
[48] S. S. Samarawickrama, I. Perera, y C.-S. E. I. M. I. T. (MIT); N. S. Foundation, "Continuous scrum: A framework to enhance scrum with DevOps", en *17th International Conference on Advances in ICT for Emerging Regions, ICTer 2017*, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 19–25. doi: 10.1109/ICTER.2017.8257808.
[49] microsoft, "Visual Studio Code", https://code.visualstudio.com/.
[50] Guido van Rossum., "python", https://www.python.org/.
[51] "GitHub", https://github.com/.
[52] microsoft, "Windows Azure", https://azure.microsoft.com/es-mx.

ANEXOS

En este apartado el lector encontrará información y datos adicionales que son relevantes para la comprensión del contenido principal.

1 Sprint 1

1.1 Historias de usuario estimadas para el Sprint 1.

Las HU son una explicación general e informal de una función de software escrita desde la perspectiva del usuario final, las siguientes historias de usuario tienen por objetivo graficar los procesos de planeación Tabla 7, lanzamiento Tabla 8 y operación Tabla 9 correspondientes al Sprint 1, en ellas se puede ver los criterios de aceptación, tareas y actividades correspondientes para lograr su objetivo, por lo que se establece el siguiente formato [10] [11].

El método de estimación de puntos de historia por talla de camiseta permite asignar rápidamente niveles de esfuerzo a las historias de usuario. Cada tamaño representa un rango de complejidad y esfuerzo relativo para completar la historia de usuario. Por ejemplo, una historia clasificada como XS puede ser muy simple y requerir poco esfuerzo, mientras que una historia clasificada como XL puede ser muy compleja y requerir mucho más trabajo [10].

Tabla 7 Historia de usuario para graficar el proceso de planeación

Épica: Tablero de control para la gestión de proyectos DevOps					
Historia de Usuario: 02GráficarProcesoPlaneación					
ID: HU02	Prioridad: Alta	Tamaño: G	Tipo: Funcional	Técnico: Software	Iteración: 2
Narrativa de la Historia de Usuario					
En el proceso de planeación debe medir n el esfuerzo por puntos de historia COMO: Administrador de proyectos de software bajo el enfoque DevOps QUIERO: Que el tablero de control tenga la capacidad de visualizar el esfuerzo por puntos de historia completados en una gráfica, PARA: Dar seguimiento a las horas dedicadas a cada tarea en el desarrollo de software					
Notas					
* La suma de los puntos de historia serán las historias de usuario planeadas para un sprint * por ejemplo: B 25 puntos de historia estimados para liberar en el primer sprint					
Criterios de Aceptación					
· La herramienta debe recibir como entrada las horas planeadas de un proyecto en específico · El formado de la fecha de entrada será el siguiente DD-MM-AAAA hh: mm: ss. · La herramienta debe mostrar como salida la planeación real vs la planeación estimada del proyecto · Cada gráfica del proceso de planeación estará asociada a un proyecto · Las horas dedicadas a cada tarea deben marcarse en el sistema					
Tareas					
1. Definir roles, responsabilidades y autoridades para la planificación del proyecto 1.1. Diseñar y codificar el módulo correspondiente al proceso de planeación 2.Gestionar los filtros por proyecto (fecha y número de proyecto) 3.Mostrar la gráfica que permita gestionar los proyectos					

Tabla 8 Historia de usuario para graficar el proceso de lanzamiento

Épica: Tablero de control para la gestión de proyectos DevOps					
Historia de Usuario: 06GráficarProcesoLanzamiento					
ID: HU06	Prioridad: Media	Tamaño: M	Tipo: Funcional	Técnico: Software	Iteración: 2
Narrativa de la Historia de Usuario					
El proceso de Lanzamiento debe medir el tiempo de despliegue. COMO: Administrador de proyectos de software bajo el enfoque DevOps QUIERO: Que el Tablero de control permita visualizar en una gráfica el proceso de lanzamiento PARA: Dar seguimiento a la estimación del tiempo en el proceso de lanzamiento.					
Notas					
* Un tiempo de despliegue corto indica una entrega rápida y eficiente del software					
Criterios de Aceptación					
1. La herramienta debe recibir los datos de las horas estimadas de un proyecto en específico en el proceso de lanzamiento 2. Mostrar las horas estimadas vs las reales del proyecto en una gráfica 3. Cada gráfica del proceso de lanzamiento estará asociada a un proyecto					
Tareas					
1. Diseñar y codificar el módulo que corresponde al proceso de lanzamiento 2.Desplegar en el servidor el módulo de lanzamiento 2. Hacer pruebas unitarias al módulo de lanzamiento 3. Monitorear el desempeño del módulo de lanzamiento					

Tabla 9 Historia de usuario para graficar el proceso de operación

Épica: Tablero de control para la gestión de proyectos DevOps					
Historia de Usuario: 08GráficarProcesoOperación					
ID: HU08	Prioridad: Media	Tamaño: CH	Tipo: Funcional	Técnico: Software	Iteración: 2
Narrativa de la Historia de Usuario					
El proceso de operación debe medir el Tiempo medio de detección (MTTD) COMO: Administrador de proyectos de software bajo el enfoque DevOps QUIERO: Que el Tablero de control permita visualizar en una gráfica el proceso de operación PARA: Dar seguimiento al tiempo de respuesta a incidencias en el proceso de operación.					
Notas a Conversación					
* Mostrar de manera gráfica el estado general de las actividades relacionadas con este proceso * El tiempo que tarda tu equipo en descubrir una incidencia.					
Criterios de Aceptación					
1. La herramienta debe recibir los datos del tiempo en el que se resuelve una incidencia o falla de un proyecto en específico en el proceso de operación 2. Mostrar el tiempo de atención de incidencias del proyecto en una gráfica 3. Cada gráfica del proceso de operación estará asociada a un proyecto					
Tareas					
1. diseñar y codificar el módulo que corresponde al proceso de operación 2.desplegar en el servidor el módulo de operación 2. hacer pruebas unitarias al modulo 3. monitorear el desempeño del módulo de operación					

1.2 Product Backlog para Sprint 1

El Product Backlog es una lista dinámica y prioritaria de todas las funcionalidades y requisitos necesarios para un producto de software. Es una parte la metodología ágil, especialmente en el marco de Scrum, contiene elementos ordenados por prioridad, con los elementos más importantes o de mayor valor en la parte superior y los menos importantes en la parte inferior, las tareas se agregan, se eliminan o se modifican según las necesidades del negocio, el feedback de los usuarios y otros factores. El Product Backlog proporciona transparencia y visibilidad sobre el trabajo pendiente en el proyecto. Todos los miembros del equipo tienen acceso y pueden ver qué se está trabajando actualmente y qué se planea para el futuro [10] [11].

La Figura 13 muestra el Product Backlog correspondiente al Sprint 1 del Tablero de control DevOps

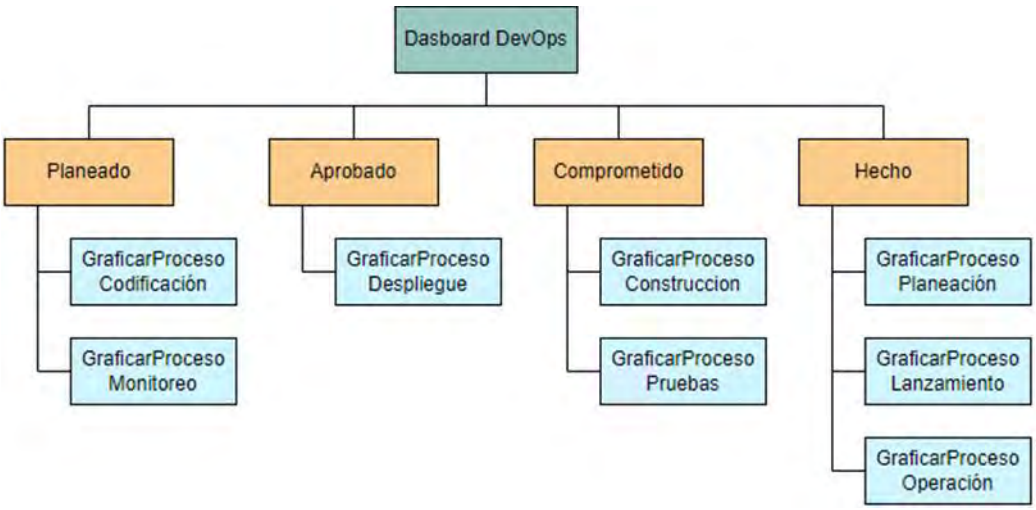

Figura 13 Product Backlog para el tablero de control DevOps correspondiente al Sprint 1

Durante la reunión de Sprint Planning, el equipo selecciona elementos del Product Backlog que planea completar durante el próximo Sprint. Estos elementos se mueven del Product Backlog al Sprint Backlog, que es la lista de trabajo específica para el Sprint en curso [11].

1.3 Sprint Backlog 1

El Sprint Backlog muestra una lista de tareas identificadas que hacen visible todo el trabajo necesario para alcanzar el compromiso correspondiente al primer Sprint, donde se da alta prioridad al desarrollo de los procesos de planeación, lanzamiento y operación, la Figura 14 muestra las tareas realizadas en color gris, en color verde las tareas que están en proceso y en color blanco aquellas tareas que están por hacer [48].

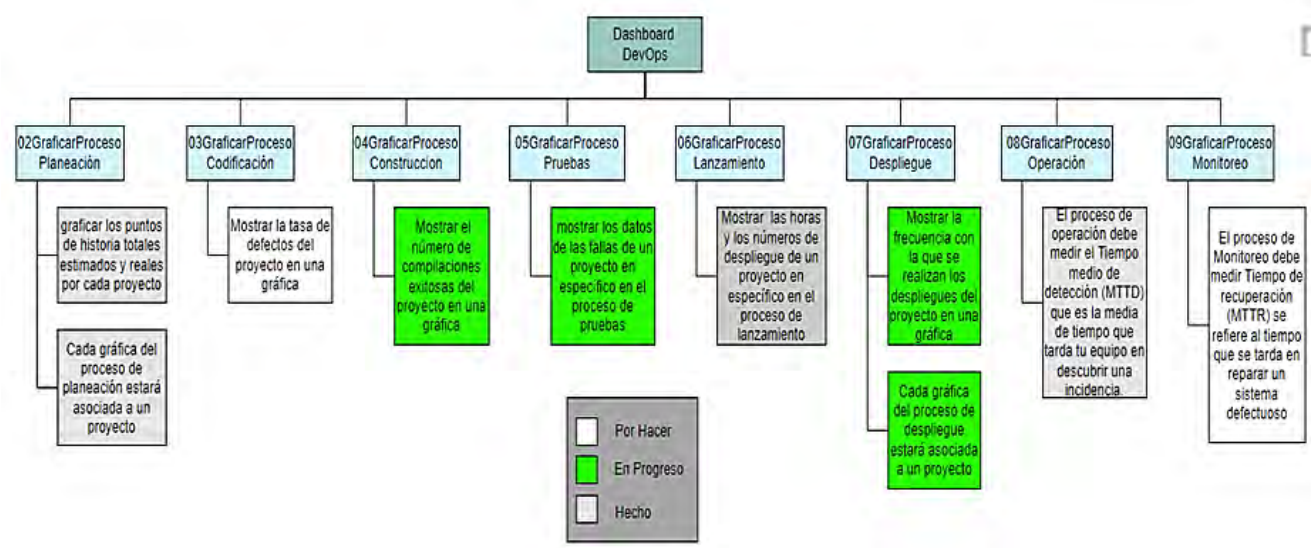

Figura 14 Sprint Backlog 1 para el tablero de control DevOps

2 Sprint 2

2.1 Historias de usuario estimadas para el Sprint 2

Las siguientes historias de usuario tienen por objetivo la gráfica los procesos de construcción Tabla 10, pruebas Tabla 11 y despliegue Tabla 12 correspondientes al tablero de control DevOps para el Sprint 2, en ellas se puede ver los criterios de aceptación y tareas correspondientes para lograr su objetivo.

Tabla 10 Historia de usuario para graficar el proceso de construcción

<table>
<tr><td colspan="6">Épica: Tablero de control para la gestión de proyectos DevOps</td></tr>
<tr><td colspan="6">Historia de Usuario: 04GráficarProcesoConstruccion</td></tr>
<tr><td>ID:
HU04</td><td>Prioridad: Media</td><td>Tamaño: G</td><td>Tipo: Funcional</td><td>Técnico:
Software</td><td>Iteración: 3</td></tr>
<tr><td colspan="6">Narrativa de la Historia de Usuario</td></tr>
<tr><td colspan="6">En el proceso de Construcción se debe medir la tasa compilaciones de código exitosas

COMO: Administrador de proyectos de software bajo el enfoque DevOps
QUIERO: Que el tablero de control me permita visualizar en una gráfica el número de compilaciones exitosas para proceso de Construcción
PARA: Dar seguimiento a los registros de compilaciones en el proceso de construcción.</td></tr>
<tr><td colspan="6">Notas</td></tr>
<tr><td colspan="6">* La métrica calcula las compilaciones exitosas entre las no exitosas
Por ejemplo
De 10 compilaciones totales 6 fueron exitosas y 4 no lo fueron por lo que tendremos una tasa del 60% de compilaciones exitosas en el registro del proyecto</td></tr>
<tr><td colspan="6">Criterios de Aceptación</td></tr>
<tr><td colspan="6">1. La herramienta debe recibir los datos de las compilaciones exitosas y no exitosas de un proyecto en específico
2. Mostrar el número de compilaciones exitosas del proyecto en una gráfica
3. Cada gráfica del proceso de construcción estará asociada a un proyecto</td></tr>
<tr><td colspan="6">Tareas</td></tr>
<tr><td colspan="6">1. Diseñar y codificar el módulo que corresponde al proceso de construcción
2.Desplegar en el servidor el módulo de construcción
2. Hacer pruebas unitarias al módulo de construcción
3. Monitorear el desempeño del módulo de construcción</td></tr>
</table>

Tabla 11 Historia de usuario para graficar el proceso de pruebas

Épica: Tablero de control para la gestión de proyectos DevOps					
Historia de Usuario: 05GráficarProcesoPruebas					
ID: HU05	Prioridad: Media	Tamaño: M	Tipo: Funcional	Técnico: Software	Iteración: 3
Narrativa de la Historia de Usuario					
El proceso de Pruebas debe medir la tasa de éxito de pruebas COMO: Administrador de proyectos de software bajo el enfoque DevOps QUIERO: Que el Tablero de control permita visualizar en una gráfica el proceso de Pruebas PARA: detectar si algunas partes del software es más probable que incluyan errores					
Notas					
* Con esta métrica se puede visualizar el porcentaje de pruebas que se ejecutan correctamente sin errores					
Criterios de Aceptación					
1. La herramienta debe recibir los datos de las fallas de un proyecto en específico en el proceso de pruebas 2. Mostrar el número de fallas encontradas del proyecto en una gráfica 3. Cada gráfica del proceso de pruebas estará asociada a un proyecto					
Tareas					
1. Diseñar y codificar el módulo que corresponde al proceso de pruebas 2.Desplegar en el servidor el módulo de pruebas 2. Hacer pruebas unitarias al módulo de pruebas 3. Monitorear el desempeño del módulo de pruebas					

Tabla 12 Historia de usuario para graficar el proceso de despliegue

Épica: Tablero de control para la gestión de proyectos DevOps					
Historia de Usuario: 07GráficarProcesoDespliegue					
ID: HU07	Prioridad: Media	Tamaño: CH	Tipo: Funcional	Técnico: Software	Iteración: 3
Narrativa de la Historia de Usuario					
El proceso de Despliegue debe medir la frecuencia de despliegue COMO: Administrador de proyectos de software bajo el enfoque DevOps QUIERO: Que el Tablero de control permita visualizar en una gráfica el proceso de despliegue PARA: Dar seguimiento a las horas destinadas en el proceso de despliegue.					
Notas a Conversación					
* Mostrar de manera gráfica el estado general de las actividades relacionadas con este proceso * Tiempo necesario para completar una sola implementación hasta su llegada a producción					
Criterios de Aceptación					
1. La herramienta debe recibir los datos de la frecuencia con la que se hacen los despliegues de un proyecto en específico en el proceso de despliegue 2. Mostrar la frecuencia con la que se realizan los despliegues del proyecto en una gráfica 3. Cada gráfica del proceso de despliegue estará asociada a un proyecto					
Tareas					
1. Diseñar y codificar el módulo que corresponde al proceso de despliegue 2.Desplegar en el servidor el módulo de despliegue 2. Hacer pruebas unitarias al módulo de despliegue 3. Monitorear el desempeño del módulo de despliegue					

2.2 Product Backlog para Sprint 2

La Figura 15 muestra el Product backlog para el Sprint 2 donde se muestran las historias de usuario hechas y las planeadas para el próximo Sprint 3

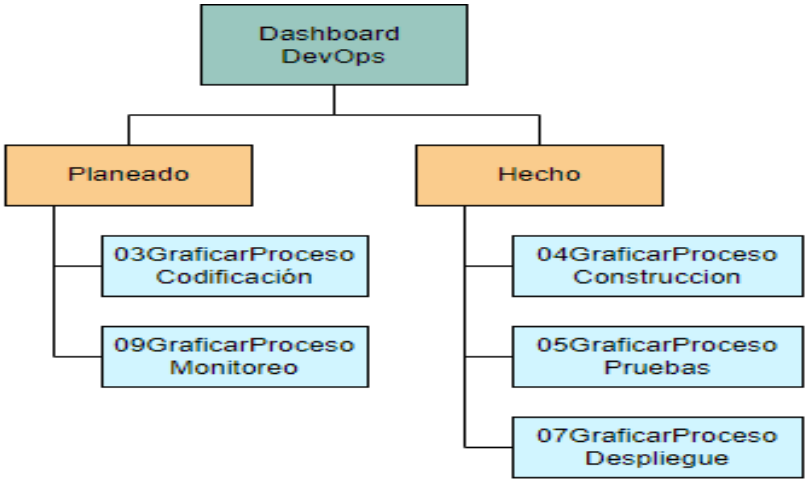

Figura 15 Product Backlog Sprint 2

2.3 Sprint Backlog 2

La figura 16 muestra la lista de tareas comprometidas en el segundo Sprint donde se da prioridad al desarrollo de los procesos de construcción, pruebas y despliegue del Tablero de control DevOps, las tareas hechas se muestran en color gris, las que están en proceso en color verde.

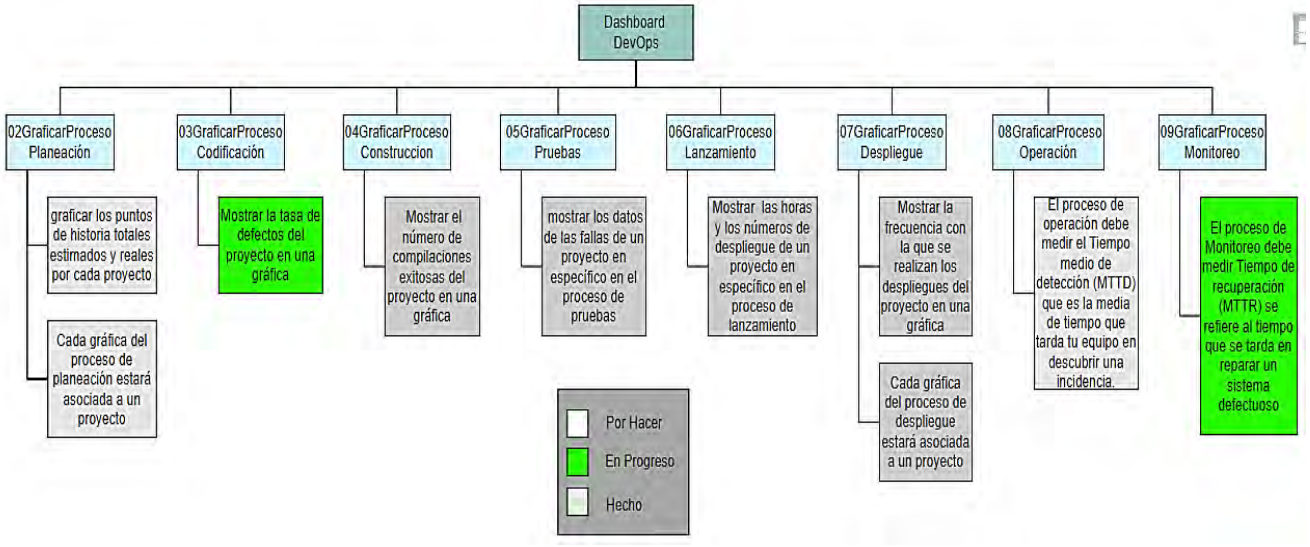

Figura 16 Sprint backlog 2

3 Sprint 3

3.1 Historias de usuario estimadas para el Sprint 3

Las siguientes historias de usuario tienen como objetivo graficar los procesos de codificación y monitoreo del Tablero de control DevOps correspondientes al Sprint 3, en ellas se puede ver los criterios de aceptación y tareas correspondientes para lograr su objetivo.

Tabla 13 Historia de usuario para graficar el proceso de codificación

<table>
<tr><td colspan="6">Épica: Tablero de control para la gestión de proyectos DevOps</td></tr>
<tr><td colspan="6">Historia de Usuario: 03GráficarProcesoCodificación</td></tr>
<tr><td>ID: HU03</td><td>Prioridad: Media</td><td>Tamaño: G</td><td>Tipo: Funcional</td><td>Técnico: Software</td><td>Iteración: 4</td></tr>
<tr><td colspan="6">Narrativa de la Historia de Usuario</td></tr>
<tr><td colspan="6">En el proceso de Codificación se debe medir la tasa de defectos.

COMO: Administrador de proyectos de software bajo el enfoque DevOps
QUIERO: Que el tablero de control me permita visualizar en una gráfica la tasa de defectos
PARA: Dar seguimiento a los defectos encontrados.</td></tr>
<tr><td colspan="6">Notas</td></tr>
<tr><td colspan="6">*La tasa de defectos es la suma de defectos encontrados durante el proceso de desarrollo de un proyecto DevOps

* por ejemplo
Documentación comentarios, mensajes
Sintaxis ortografía, puntuación, errores tipográficos, formatos de instrucción
Asignación declaración, nombres duplicados, alcance, límites
Interfaces llamadas de procedimiento y referencia, E/S, formatos de usuario
Comprobación mensajes de error, comprobaciones inadecuadas
Datos estructura, contenido
Función lógica, punteros, bucles, recursividad, computación, defectos de función
Sistema configuración, temporización, memoria
Ambiente diseño, compilación, prueba u otros problemas del sistema de soporte</td></tr>
<tr><td colspan="6">Criterios de Aceptación</td></tr>
<tr><td colspan="6">1. La herramienta debe recibir el número y tipo de defectos encontrados durante el proceso desarrollo de un proyecto en específico
2. Mostrar la tasa de defectos del proyecto en una gráfica
3. Cada gráfica del proceso de codificación estará asociada a un proyecto</td></tr>
<tr><td colspan="6">Tareas</td></tr>
<tr><td colspan="6">1. Diseñar y codificar el módulo que corresponde al proceso de codificación
2.Desplegar en el servidor el módulo de codificación
3.Hacer pruebas unitarias al módulo de codificación
4.Monitorear el desempeño del módulo de codificación</td></tr>
</table>

Tabla 14 Historia de usuario para graficar el proceso de monitoreo

Épica: Tablero de control para la gestión de proyectos DevOps					
Historia de Usuario: 09GráficarProcesoMonitoreo					
ID: HU09	Prioridad: Media	Tamaño: CH	Tipo: Funcional	Técnico: Software	Iteración: 4
Narrativa de la Historia de Usuario					
El proceso de Monitoreo debe medir Tiempo de recuperación (MTTR) COMO: Administrador de proyectos de software bajo el enfoque DevOps QUIERO: Que el Tablero de control permita visualizar en una gráfica el proceso de monitoreo PARA: Dar seguimiento al tiempo que se invierte en atender la falla en el proceso de monitoreo.					
Notas					
* Mostrar de manera gráfica el estado general de las actividades relacionadas con este proceso * Mide la facilidad en que un equipo puede repararse					
Criterios de Aceptación					
1. La herramienta debe recibir los datos del tiempo en el que se resuelve una incidencia o falla de un proyecto en específico en el proceso de monitoreo 2. Mostrar el tiempo de atención de incidencias del proyecto en una gráfica 3. Cada gráfica del proceso de monitoreo estará asociada a un proyecto					
Tareas					
1. Diseñar y codificar el módulo que corresponde al proceso de monitoreo 2.Desplegar en el servidor el módulo de monitoreo 3.Hacer pruebas unitarias al módulo de monitoreo 4.Monitorear el desempeño del módulo de monitoreo					

3.2 Product Backlog para Sprint 3

La Figura 17 muestra en el Product Backlog las historias de usuario correspondientes al Sprint 3 del Tablero de control DevOps.

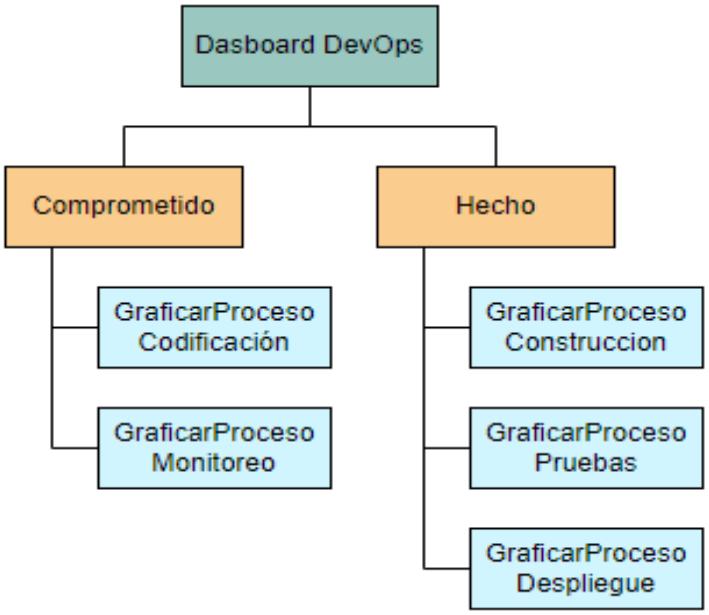

Figura 17 Product Backlog del Sprint 3

La Figura 18 lista las tareas correspondientes al tercer Sprint donde se da prioridad al desarrollo de los procesos de codificación y monitoreo para el Tablero de control DevOps, las tareas que están hechas se muestran en color gris, las que están en proceso en color verde.

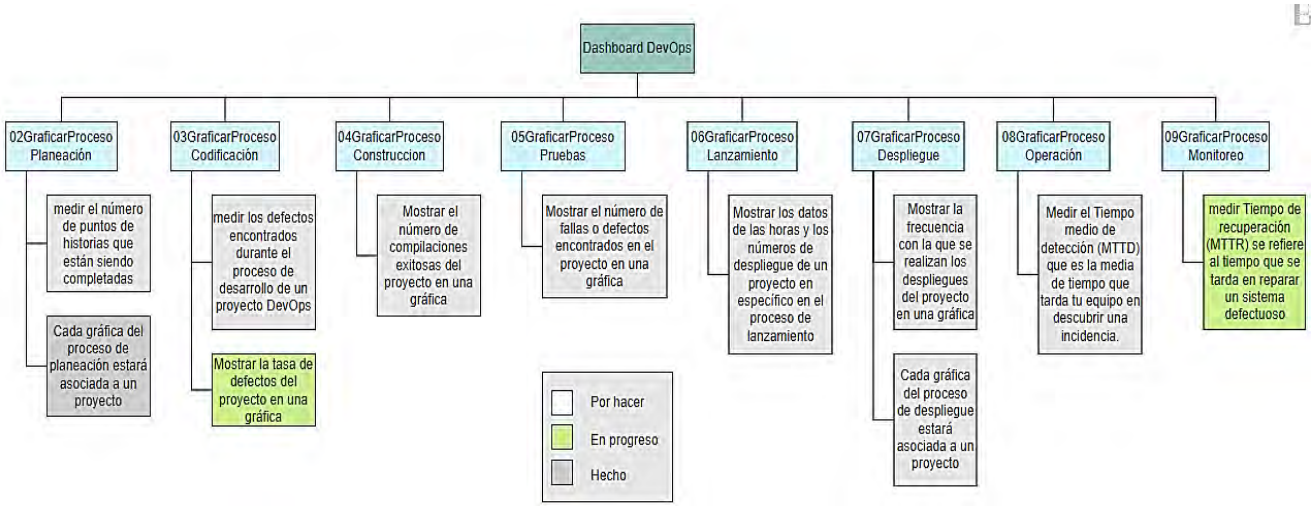

Figura 18 Sprint backlog 3

4 Codificación

4.1 Herramientas utilizadas para la codificación

Para la codificación del Tablero de Control de DevOps, se emplea la herramienta Visual Studio Code en su versión 1.88 compatible con el sistema operativo Windows_NT x64. Se utiliza este entorno de desarrollo integrado (IDE)[49]
El lenguaje de programación seleccionado para este proyecto es Python,[50] en su versión 3.10.5. Python es conocido por su simplicidad, legibilidad y su robusta biblioteca estándar. La figura 19 muestra un ejemplo de cómo se estructura el proyecto.

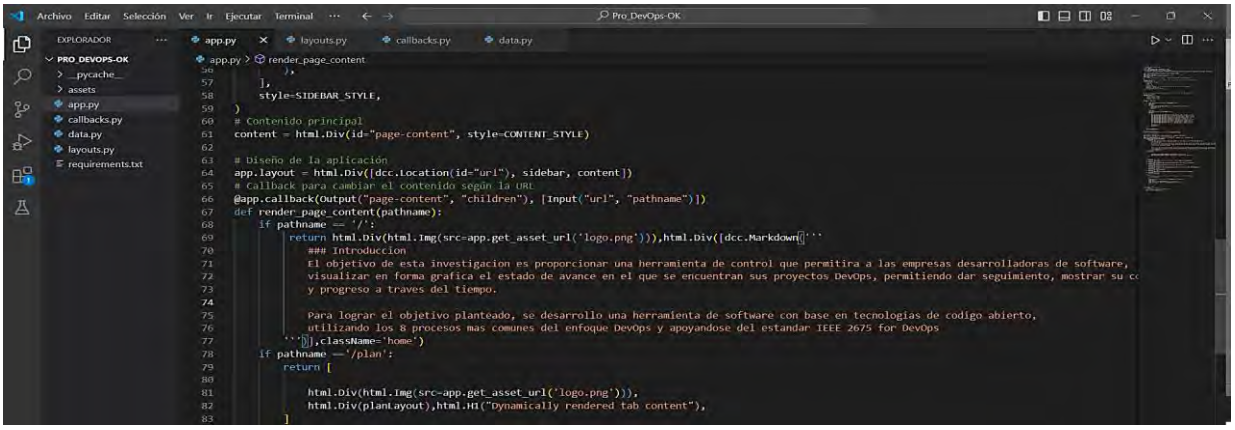

Figura 19 Estructura del tablero de control en Python

Como repositorio de versionamiento de código, se utiliza GitHub, una plataforma de desarrollo colaborativo que aloja proyectos utilizando el sistema de control de versiones Git, la Figura 20 muestra el repositorio del código en GitHub, donde se puede acceder al código fuente correspondiente [51].

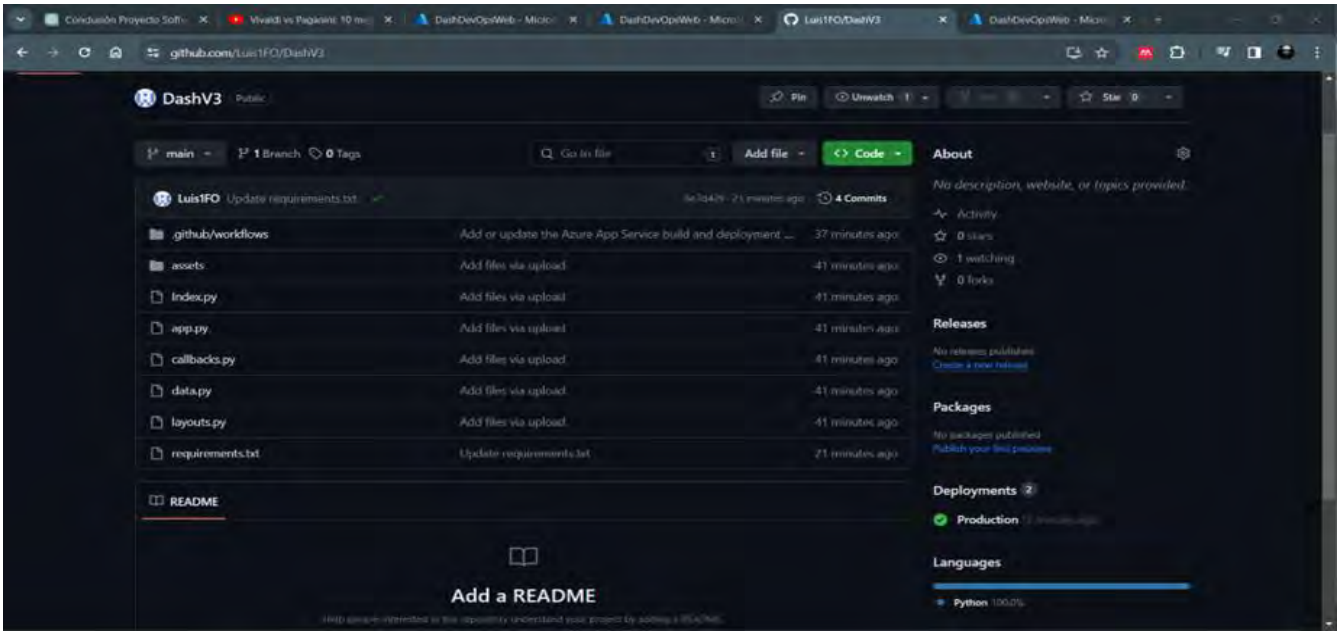

Figura 20 Repositorio del Tablero de control DevOps

5 Construcción

Como parte de la construcción y desarrollo automatizado del Tablero de control DevOps los Pipelines ilustran el comportamiento en la integración continua, la Figura 21 y 22 muestran las fases de construcción y despliegue.

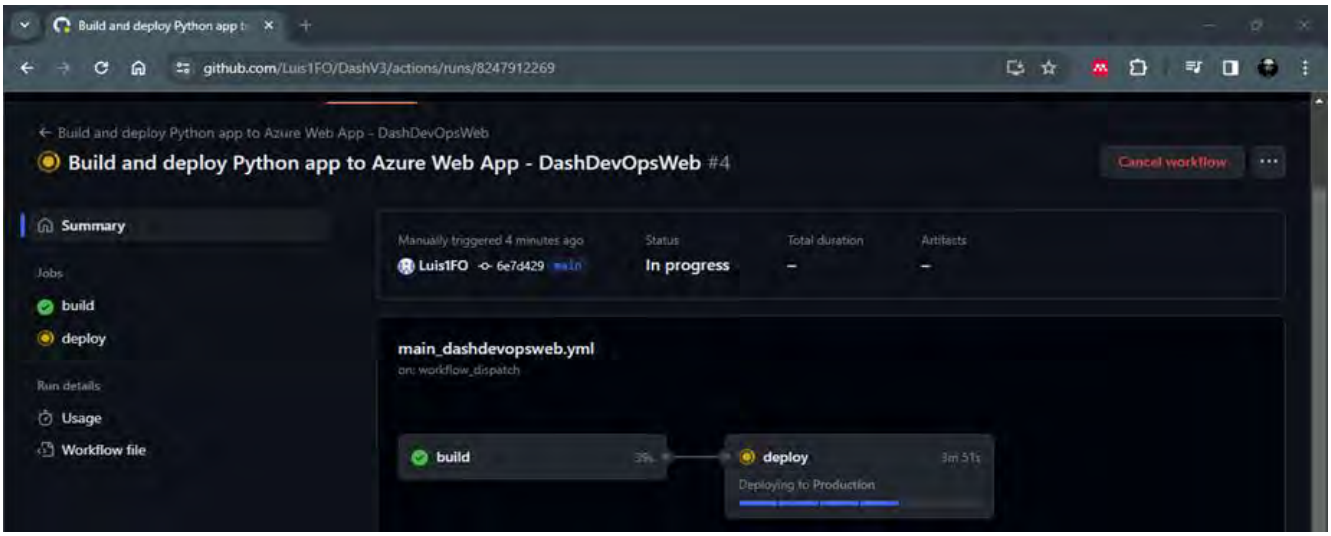

Figura 21 Pipeline del Tablero de control DevOps

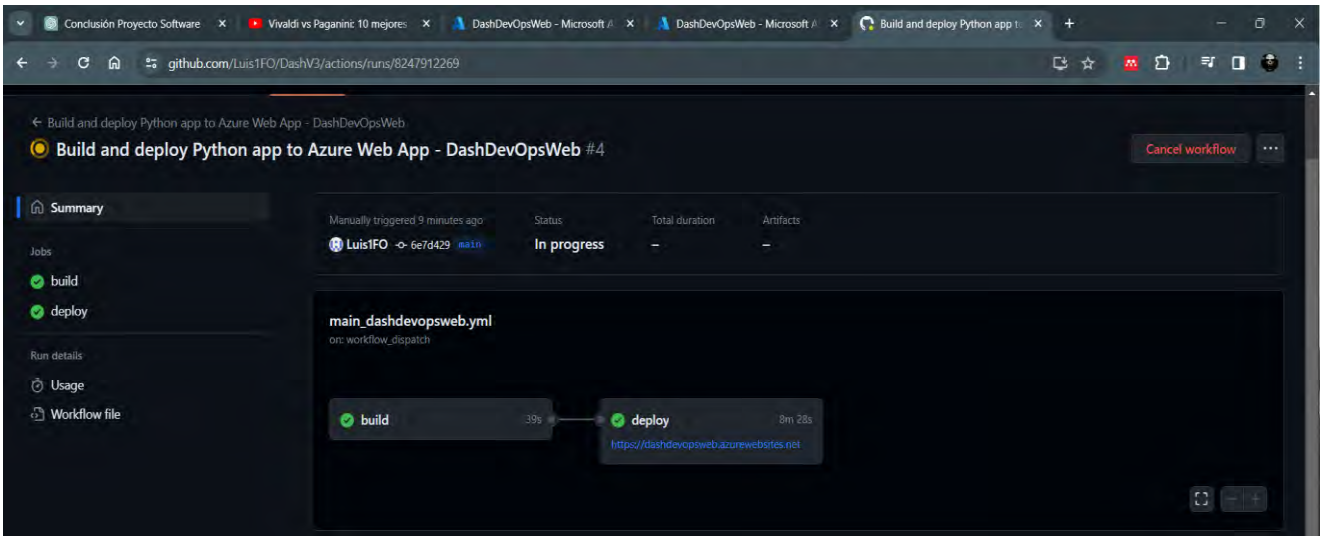

Figura 22 Despliegue del Tablero de control DevOps

6 Lanzamiento

6.1 Integración continua

Una ventaja de tener la aplicación con el código almacenado en un repositorio de GitHub, es que los desarrolladores puedan impulsar cambios de código todos los días, varias veces al día, la Figura 16 muestra el registro de esos cambios.

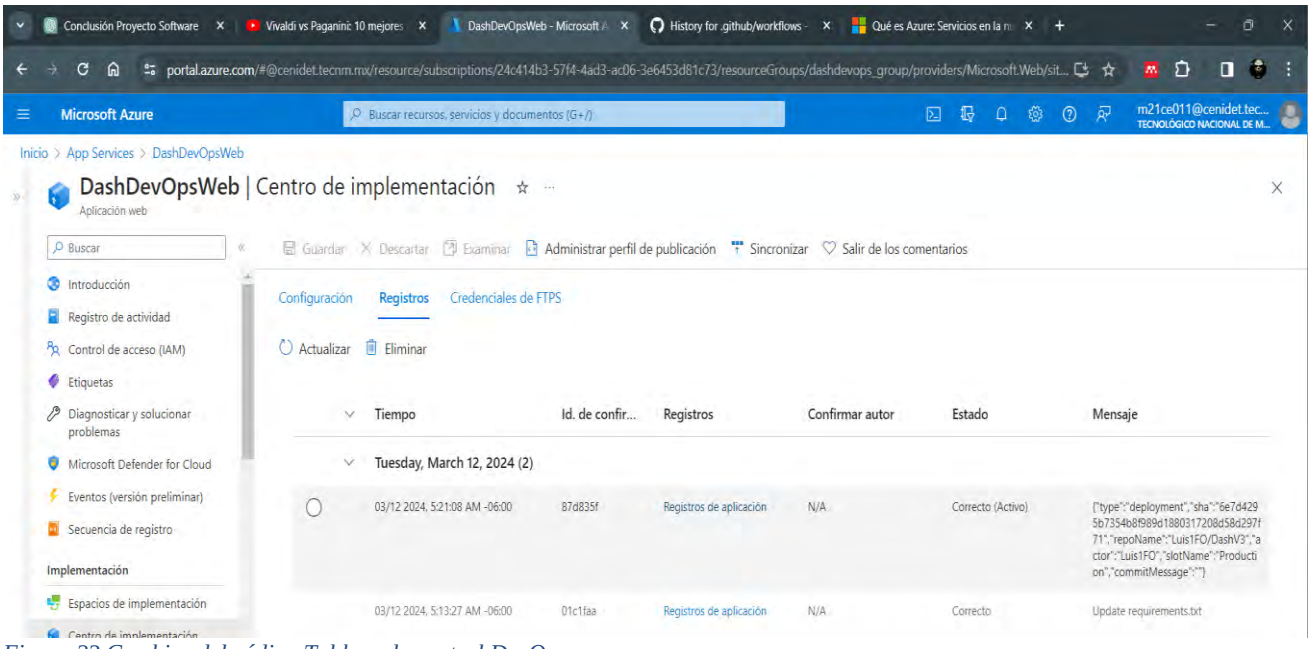

Figura 23 Cambios del código Tablero de control DevOps

Microsoft Azure, es una plataforma en la nube que ofrece una amplia gama de servicios de computación, almacenamiento, redes, bases de datos, etc. Permiten a las organizaciones construir, implementar y administrar aplicaciones y servicios a través de Internet, las figuras 24 y 25 muestran la consola de Windows Azure desde donde podemos configurar los detalles y se realiza el despliegue del Tablero de control DevOps [52].

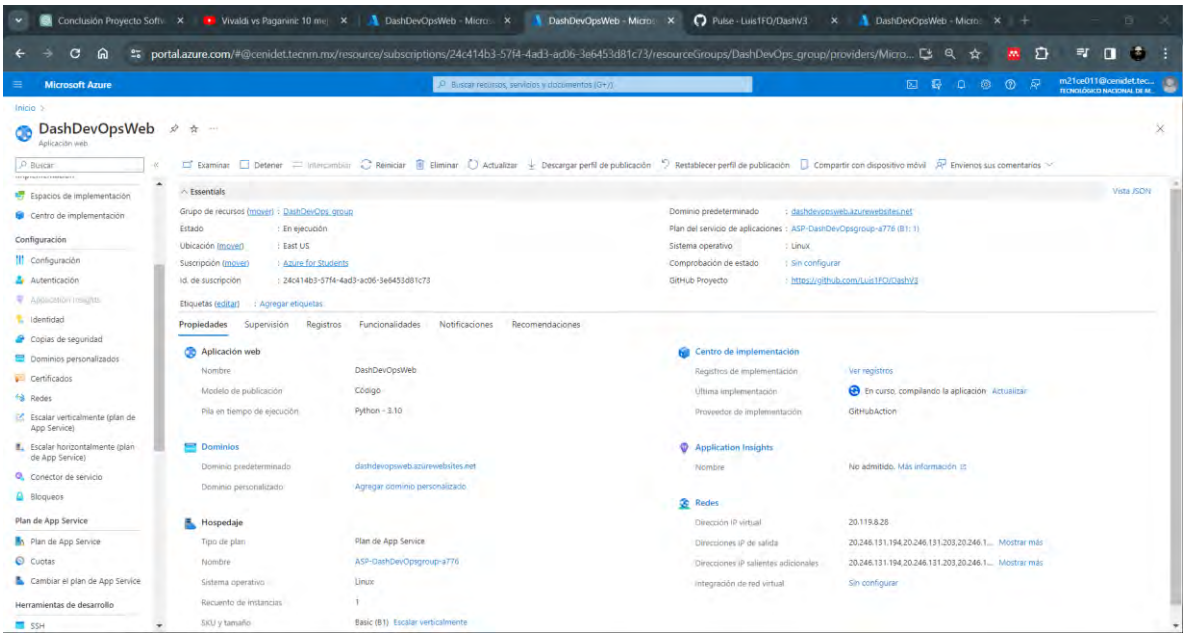

Figura 24 Consola de configuración Windows Azure

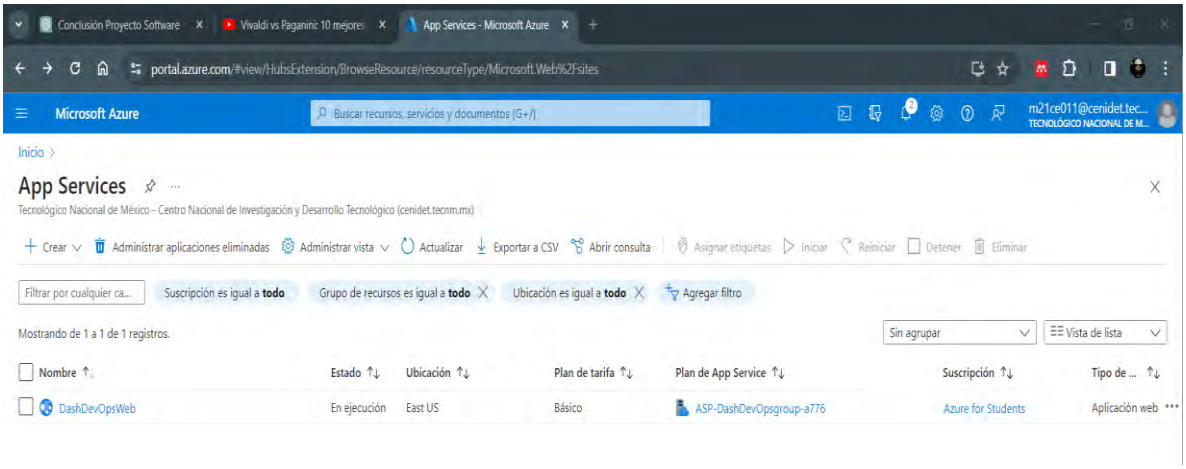

Figura 25 Estatus general de la aplicación

7 Evaluación del tablero de control DevOps como proyecto en esta Investigación

En esta investigación, el tablero de control con métricas DevOps fue evaluado como un proyecto, la evaluación se centró en el cumplimiento de los objetivos planteados.

Objetivos

- Establecer indicadores de desempeño para las prácticas DevOps.
- Establecer métricas para el proceso de cada práctica.
- Dar seguimiento a las actividades de los procesos mediante los resultados de las métricas e indicadores propuestos.

Metodología de Evaluación

Identificación y definición de las prácticas clave de DevOps y definición de métricas e indicadores específicos, el desglose de cada práctica en sus componentes y etapas, identificación de métricas específicas para cada proceso y la revisión de la utilidad de las métricas seleccionadas.

Tabla 15 Evaluación para el tablero de control DevOps

Fase DevOps	Métrica aplicable	Fórmula	Esta tesis
Planificación	Esfuerzo por puntos de historia	Se establecen mediante un consenso	45 puntos de historia totales
Codificación	Tasa de defectos	TD = Númdefectos / líneas-código X100	TD = 15 defectos / 5 kloc x100= 30%
Construcción	Tasa de compilaciones de código exitosas	TCE = NúmTotalCompiExito / NumTotalCompilaciones X100	TCE = 18 compilaciones exitosas / 20 intentos de compilación X100 =90%
Pruebas	Tasa de éxito de pruebas	TasaExt=NúmTotalPrueExito / NúmTotalPruebas X100	TasaExt= 15 pruebas exitosas / 20 pruebas ejecutadas X 100 =75%
Lanzamiento	Tiempo promedio de despliegue	TD = TiempoTotal / NúmTotalLanzamientos	TD = 43800 minutos / 15 lanzamientos = 2,920 min o 48 hrs Por lanzamiento
Despliegue	Frecuencia de despliegue	FD= NúmeroTotalDespliegues / TiempoTotal	FD = 6 despliegues / 6 meses = 1 despliegue por mes
Operación	Tiempo medio de detección (MTTD)	MTTD= Td / Ne	MTTD= 20+30+10+40+20 =120 min / 5 errores = 24 minutos por detección
Monitoreo	Tiempo medio de recuperación (MTTR)	MTTR= Td / Nr	MTTR = 30+40+20=90 min / 3 recuperaciones = 30 minutos por recuperación