Marco de trabajo basado en DevOps para optimizar el desarrollo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

Cabrera Córdova, Anthony Rolando

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Trabajo de titulación, previo a la obtención del título de Ingeniero de Software

Ing. Jácome Guerrero, Patricio Santiago, PhD

17 de julio de 2023

Latacunga

Reporte de Verificación de Contenidos

Document Information

Analyzed document	TESIS CABRERA ANTHONY.pdf (D172048344)
Submitted	2023-07-13 14:06:00
Submitted by	Juan Carlos Altamirano
Submitter email	jc.altamiranoc@uta.edu.ec
Similarity	0%
Analysis address	jc.altamiranoc.uta@analysis.urkund.com

Sources included in the report

W	URL: https://www.linkedin.com/pulse/what-difference-between-methodology-framework-dafir-jacob/ Fetched: 2023-07-13 14:07:00	1
W	URL: https://doi.org/10.1007/978-3-319-18612-2_19 Fetched: 2023-07-13 14:07:00	1
W	URL: http://revistaespirales.com/index.php/es/article/view/269 Fetched: 2023-07-13 14:07:00	1
W	URL: https://books.google.com.ec/books?hl=es&lr=&id=jC59DwAAQBAJ&oi=fnd&pg=PP1&dq=gitlab&ots=9RztpR... Fetched: 2023-07-13 14:07:00	1
W	URL: https://medium.com/taptuit/the-eight-phases-of-a-devops-pipeline-fda53ec9bba Fetched: 2023-07-13 14:07:00	1

.....

Ing. Jácome Guerrero, Patricio Santiago, PhD

C.C.: 1001689791

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Certificación

Certifico que el trabajo de titulación, **"Marco de trabajo basado en DevOps para optimizar el desarrollo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE"** fue realizado por el señor **Cabrera Córdova, Anthony Rolando**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Latacunga, 17 de julio de 2023

.....

Ing. Jácome Guerrero, Patricio Santiago, PhD

C.C.: 1001689791

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Responsabilidad de Autoría

Yo, **Cabrera Córdova, Anthony Rolando**, con cédula de ciudadanía n° 1718104027, declaro que el contenido, ideas y criterios del trabajo de titulación: "**Marco de trabajo basado en DevOps para optimizar el desarrollo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE**" es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 17 de julio de 2023

.....

Cabrera Córdova, Anthony Rolando

C.C.: 1718104027

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Autorización de Publicación

Yo, **Cabrera Córdova, Anthony Rolando**, con cédula de ciudadanía n° 1718104027, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Marco de trabajo basado en DevOps para optimizar el desarrollo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE**", en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Latacunga, 17 de julio de 2023

.....

Cabrera Córdova, Anthony Rolando

C.C.: 1718104027

Dedicatoria

Dedico este proyecto y título a mis abuelos que no entendían lo que hacía, pero sabían que lo haría bien. Además, se lo dedico a mis padres, tíos y familiares que no estudiaron una carrera universitaria por limitaciones de acceso o económicas.

Se lo dedico también a los estudiantes de todos los niveles, los adolescentes, los escolares, las personas menores de 20 años que han pensado en mejores condiciones de vida. Sí se puede.

Finalmente, mis logros están dedicados a las personas que me conocerán y utilizarán esta investigación como herramienta para conocer algo nuevo.

Agradecimiento

Agradezco a Dios por acompañarme durante toda mi vida y darme su abrigo cuando tuve que vivir lejos de mi familia.

A mis padres les doy las gracias por esforzarse tanto en educar a sus hijos y apuntar siempre alto junto a nosotros con nuestras aspiraciones de vida.

A mis hermanos les agradezco por ser mis primeros amigos, con quienes aprendí a resolver problemas y pasar buenos momentos.

A mis abuelos les debo la motivación para tener una vida de sacrificio y bendecida con una bella familia. Que así sea.

A mis docentes y compañeros de la Universidad les agradezco por ser mis acompañantes en un largo viaje de exploración y nuevos conocimientos como lo fue mi carrera.

A mi tutor, el PhD. Santiago Jácome, le doy las gracias por su paciencia y buena voluntad para ayudarme en este último paso.

Finalmente, agradezco a la Universidad de las Fuerzas Armadas ESPE Sede Latacunga por haberme acogido como una casa. Espero haber aportado positivamente a su visión como institución de Educación Superior referente en el país.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

El presente trabajo de titulación presenta un marco de trabajo basado en DevOps que optimiza el desarrollo de software para la gestión de historias clínicas para el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE, ESPE SALUD. Su desarrollo se dividió en tres partes: la primera de ellas es la elaboración de un marco teórico que contextualice la problemática detrás del proyecto, así como las herramientas técnicas y metodológicas que aportarán a la construcción de la propuesta; en segundo lugar se encuentra el diseño de marco de trabajo basado en DevOps que en su presentación brinda una explicación de cómo ser adaptado a través de la herramienta GitLab; finalmente, la última etapa trata de la implementación del marco de trabajo propuesto dentro del proyecto ESPE SALUD y la evaluación de los indicadores a través de la medición de resultados obtenidos. El enfoque de la investigación está en la aplicación de prácticas ágiles de desarrollo relacionadas a los principios de DevOps a través de herramientas tecnológicas que permiten que DevOps se cumpla. Se explicó cada una de las fases del ciclo de desarrollo de software que aportan en la adopción de DevOps dentro de un sistema de información, en este caso dentro del proyecto ESPE SALUD.

Palabras clave: DevOps, marco de trabajo, historias clínicas.

Abstract

This degree work presents a framework based on DevOps that optimizes the development of software for the management of medical records for the Integrated Health System of the University of the Armed Forces ESPE, ESPE SALUD. Its development was divided into three parts: the first one is the elaboration of a theoretical framework that contextualizes the problems behind the project, as well as the technical and methodological tools that will contribute to the construction of the proposal; secondly, there is the design of the framework based on DevOps that in its presentation provides an explanation of how to be adapted through the GitLab tool; finally, the last stage deals with the implementation of the proposed framework within the ESPE SALUD project and the evaluation of the indicators through the measurement of results obtained. The focus of the research is on the application of agile development practices related to DevOps principles through technological tools that allow DevOps to be fulfilled. Each of the phases of the software development cycle that contribute to the adoption of DevOps within an information system, in this case within the ESPE SALUD project, was explained.

Keywords: DevOps, framework, medical records.

Capítulo I

Introducción al Proyecto

Antecedentes

En la medicina hospitalaria e institucional existen elementos indispensables que permiten su normal desarrollo, entre ellos la Historia Clínica. Los Sistemas de Información dedicados a Historias Clínicas gestionan digitalmente la información de los pacientes, aportando a la continuidad del cuidado médico y asegurando la confidencialidad de sus contenidos. (Silvestre et al., 2019)

Hoy por hoy es fundamental que los profesionales que desarrollan Sistemas de Gestión de Historias Clínicas centren sus esfuerzos en la entrega de software de calidad a sus clientes, así lo afirma Hüttermann (Hüttermann, 2012).

El mismo autor (Hüttermann, 2012) manifiesta que en el ciclo del software es normal encontrar conflictos entre el desarrollo y las operaciones. Por un lado, el equipo de desarrollo trabaja en cambios que esperan desplegar en el menor tiempo a producción, por otro, el de operaciones busca estabilidad y los cambios constantes no son una práctica que recomiendan.

Siguiendo esta problemática, se encuentra la investigación de Ebert, Gallardo, Hernantes y Serrano (Ebert et al., 2016) que señalan al DevOps como la manera eficiente para integrar el desarrollo y las operaciones a través de una conexión eficiente y fluida.

DevOps viene de la combinación de dos palabras del idioma inglés: *development* (desarrollo) y *operations* (operaciones). Este paradigma involucra aspectos como: colaboración, automatización del despliegue, testing y entrega continua. (Lwakatare et al., 2015)

Ante la poca integración de los procesos del ciclo de software en los sistemas de gestión de historias clínicas, la academia está en la obligación de proponer metodologías o marcos de

trabajo ágiles que involucren paradigmas como el DevOps, de forma que su aplicación aporte a conseguir cambios en el menor tiempo y sin poner en juego la calidad del desarrollo de software.

El presente proyecto de investigación propone definir un Marco de trabajo basado en DevOps para optimizar el desarrollo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

Planteamiento y Formulación del Problema

La Universidad de las Fuerzas Armadas ESPE se encuentra en un proceso de digitalización de Historias Clínicas según lo afirma el Dr. Bolívar Llamuca, médico general y ocupacional de la Institución (comunicación personal, 12 de octubre de 2020). El proyecto consta de un sistema web para la gestión de historias clínicas que será utilizado en las sedes por el personal de la Universidad.

Sin embargo, en el proyecto, el ciclo de vida del software no ha sido direccionado por un marco de trabajo estructurado, así como el equipo de trabajo se ha visto disminuido debido a la desvinculación de programadores y tutores (comunicación personal, 12 de octubre de 2020).

Por otro lado, la pandemia mundial del virus SARS-CoV-2 ha producido estrictas medidas de confinamiento y distanciamiento que no han permitido al equipo de trabajo continuar con su labor presencial en las instalaciones del centro educativo (El Universo, 2020).

Mientras se desarrollan los requerimientos pendientes y aparecen otros nuevos, el equipo de desarrollo de software debe estar en la capacidad de convertir en el menor tiempo cada requisito en funcionalidades, incorporando nuevas metodologías de desarrollo.

Basándonos en esta problemática se formula la siguiente pregunta: ¿cómo optimizar el desarrollo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE?

Justificación e Importancia

Al día de hoy, el proyecto para la gestión de historias clínicas del Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE se encuentra en su etapa de pruebas. Según lo afirma el médico de la institución, las pruebas del personal médico con el sistema están planificadas para diciembre (comunicación personal, 12 de octubre de 2020).

Mientras el sistema está siendo probado, aparecen nuevos requerimientos y correcciones en las funcionalidades que retrasan la puesta en producción. Según el Dr. Llamuca (comunicación personal, 12 de octubre de 2020), el no existir una metodología o marco de trabajo es un factor determinante en la demora de cada iteración del proyecto.

Por otro lado, las técnicas para el desarrollo ágil han sido un tema de gran interés dentro y fuera de la academia. Los autores (Hemon et al., 2020) sostienen que, aunque los enfoques de desarrollo de software ágil se han hecho cada vez más frecuentes, muchas organizaciones no han conseguido disminuir los tiempos al desplegar sus cambios en producción dentro de sus sistemas, en gran parte debido a las diferentes funciones departamentales que operan en solitario. En un esfuerzo por eliminar esta falta de integración, las empresas han optado por el DevOps. A medida que la digitalización continúa, las empresas implementan cada vez más DevOps.

Levantamiento de Información

Para el levantamiento de información se elaboró una encuesta dirigida a los miembros del equipo de trabajo del proyecto ESPE SALUD, conformado por tres estudiantes de la Carrera de Ingeniería de Software y el médico de la institución. Esta tiene el fin de medir cuantitativamente el nivel de satisfacción de los miembros del equipo en cuanto a diferentes parámetros como: tareas, organización y horas de trabajo.

Las opciones de repuestas a las preguntas fueron definidas con los valores de 1 a 5. Donde 1 representa “Nunca” y 5 “Siempre”. Las preguntas se detallan a continuación:

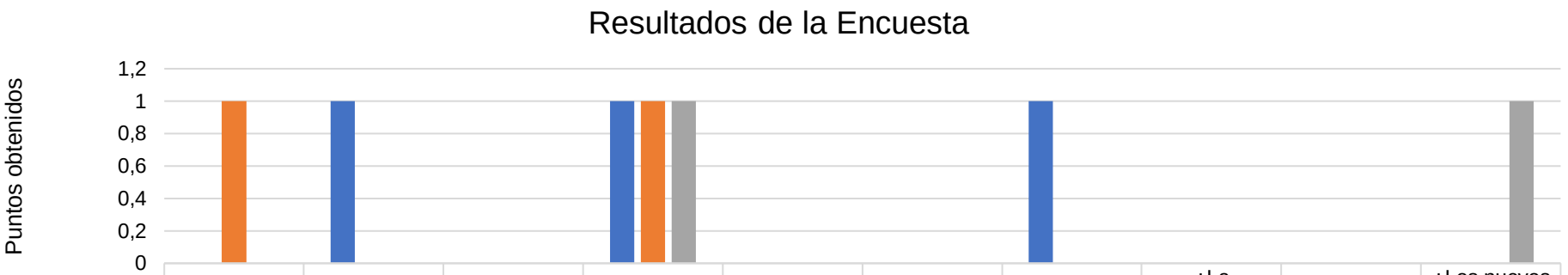
- ¿Se está utilizando herramientas para la administración y control del proyecto?
- ¿Los nuevos requerimientos del sistema son explicados y debidamente documentados?
- ¿Existe una convención para nombrar a las versiones sobre las que trabaja?
- ¿Los cambios al código del sistema están siendo versionados?
- ¿Existe un mecanismo para aprobar o rechazar cambios?
- ¿El tiempo y los logs al generar una nueva versión del sistema (release) está siendo monitoreado?
- ¿Existe un mecanismo para solicitar y definir responsables para cambios en el código?
- ¿La publicación de los cambios en el servidor de pruebas/producción se realiza de forma automática?
- ¿Puede usted recuperar cualquier versión anterior del sistema?
- ¿Los nuevos programadores que se unen al proyecto son capaces de configurar su entorno de trabajo local de forma ágil?

En la Tabla 1 se muestran los resultados de la encuesta.

Resultados.

Tabla 1

Tabla de resultados a partir de la encuesta de levantamiento de información.



	¿Se está utilizando herramientas para la administración y control del proyecto?	¿Los nuevos requerimientos del sistema son explicados y debidamente documentados?	¿Existe una convención para nombrar a las versiones sobre las que trabaja?	¿Los cambios al código del sistema están siendo versionados?	¿Existe un mecanismo para aprobar o rechazar cambios?	¿El tiempo y los logs al generar una nueva versión del sistema (release) está siendo monitoreado?	¿Existe un mecanismo para solicitar y definir responsables para cambios en el código?	¿La publicación de los cambios en el servidor de pruebas/producción se realiza de forma automática?	¿Puede usted recuperar cualquier versión anterior del sistema?	¿Los nuevos programadores que se unen al proyecto son capaces de configurar su entorno de trabajo local de forma ágil?
Encuestado 1	0	1	0	1	0	0	1	0	0	0
Encuestado 2	1	0	0	1	0	0	0	0	0	0
Encuestado 3	0	0	0	1	0	0	0	0	0	1

Nota. Tabla de resultados del levantamiento de información. Los 1 quieren decir “sí”, los 0 significan “no”.

Análisis. En la encuesta participaron cuatro personas, la primera de ellas cumple con el rol de usuario, la persona que provee los nuevos requerimientos del proyecto y además tiene un rol como “administrador” del proyecto que revisa los avances semanales.

Por otro lado, tres programadores participaron de la encuesta registrando sus respuestas.

Los promedios de los puntajes entre 0 a 1 en las respuestas fueron:

a) Primera pregunta: 0,33 puntos.

b) Segunda pregunta: 0,33 puntos.

c) Tercera pregunta: 0 puntos.

d) Cuarta pregunta: 1,5 puntos.

e) Quinta pregunta: 0 puntos.

f) Sexta pregunta: 0 puntos.

g) Séptima pregunta: 0,33 puntos.

h) Octava pregunta: 0 puntos.

i) Novena pregunta: 0 puntos.

j) Décima pregunta: 0,33 puntos.

En base a estos promedios se debe destacar el escaso esfuerzo que se hace a nivel de monitoreo, además pasa la mismo con la integración y el despliegue continuo de los nuevos cambios del software. Inclusive a la hora de presentarse nuevos requerimientos, estos

no cumplen con una suficiente documentación que sirva para estimar o valorar los requerimientos en la etapa de análisis.

DevOps es una práctica que combina agilidad y rigor en equipos multifuncionales. La agilidad es un método que posibilita reaccionar rápidamente a requerimientos cambiantes del usuario en un ciclo iterativo. (Wiedemann et al., 2020)

Se necesita de una metodología de trabajo que ofrezca las directrices para saber cómo trabajar durante los ciclos iterativos. Atendiendo los requerimientos que aparecen debido a las nuevas necesidades de los usuarios finales del Sistema Integrado de Salud.

En virtud de lo mencionado, se ha decidido proponer un marco de trabajo con las mejores prácticas de DevOps para optimizar el desarrollo de software para el proyecto de gestión de historias clínicas del Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE, garantizando el desarrollo, entrega y despliegue continuo del software.

Objetivos

Objetivo General

Definir un Marco de trabajo basado en DevOps para optimizar el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

Objetivos Específicos

- Realizar el estudio bibliográfico sobre marcos de trabajo de desarrollo de software y DevOps, con el fin de conocer el estado de la literatura en estas ramas de la Ingeniería de Software.
- Desarrollar un marco de trabajo basado en DevOps que sea una propuesta transversal y aplicable.
- Implementar el marco de trabajo en el desarrollo del sistema para la gestión de historias clínicas del Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.
- Evaluar el marco de trabajo implementado, a través de los indicadores de la variable dependiente realizando entrevistas individuales al equipo de desarrollo del proyecto para la gestión de historias clínicas.

Metas

Definición de un Marco de trabajo basado en DevOps para optimizar el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

Hipótesis

Si se define un marco de trabajo basado en DevOps entonces se optimiza el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

Variables de la Investigación

Variable dependiente

Definición de un marco de trabajo basado en DevOps.

Variable independiente

Se optimiza el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el departamento médico de la Universidad de las Fuerzas Armadas ESPE.

Indicadores

- 70% de adaptación y aceptación del marco de trabajo de parte de los miembros del equipo de trabajo.

Capítulo II

Marco Teórico

Gestión Ágil de Proyectos de Software

Metodología de Gestión de Proyectos de Software

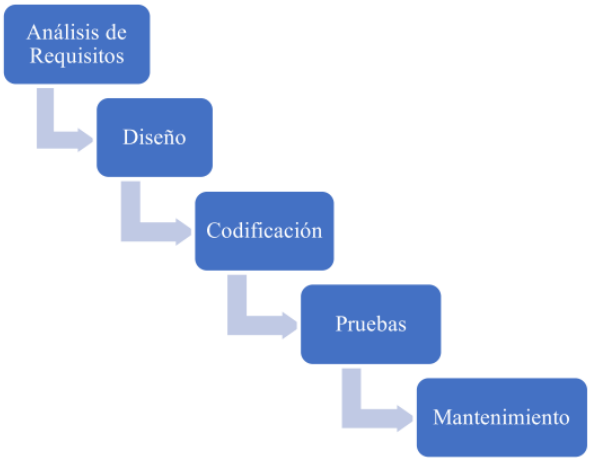
Los autores (Molina et al., 2018) afirman que en la década de los 50, el desarrollo de software era dirigido por la tarea de codificar, dejando en segundo plano el comprender y recoger las necesidades de los usuarios, los mismos que a menudo no quedaban satisfechos con la calidad del producto final. Entiéndase calidad del software como el resultado de la evaluación de diferentes parámetros como son la velocidad, flexibilidad, seguridad, usabilidad, escalabilidad, entre muchos otros.

Es así como nacían las metodologías de desarrollo de software. Estas hacen uso de diversas herramientas, técnicas, métodos y modelos de desarrollo de software.

Las primeras en aparecer fueron las metodologías de desarrollo tradicionales basadas en el modelo cascada o “Waterfall” en inglés. El modelo cascada se encuentra representado en la Figura 1. El autor Pressman (Pressman, 2010) explica que en ese entonces se planteó poner en orden a la creación de software una vez que la demanda creció exponencialmente. El modelo cascada consta de cinco etapas secuenciales sin forma de iterar o retroceder.

Figura 1

Fases del modelo cascada



Nota. El gráfico muestra las fases del modelo de desarrollo de software cascada. Tomado de los autores Molina et al. (Molina et al., 2018)

Por otro lado, en 2001 nace la organización “The Agile Alliance” en Estados Unidos, una sociedad sin ánimos de lucro dedicada a promover conceptos sobre el desarrollo ágil de software y ayudar a adoptar a las empresas esas prácticas (José H. Canós, 2012).

Generalmente las metodologías tradiciones vienen cargadas de burocracia. Los procesos ágiles en las metodologías fueron propuestas para ahorrar tiempo y dinero. (Garayar Velásquez, 2018) Es así como aparecieron varias, entre ellas: Scrum y Extreme Programming (XP).

Diferencias entre Marco de Trabajo y Metodología

En un contexto general, un marco de trabajo o *framework* es un grupo de prácticas, conceptos y técnicas para definir una solución general que pueda servir para un determinado grupo de problemas (Galindo Haro, 2010).

El autor (Jacob, 2016) hace un contraste entre ambas. Un marco de trabajo provee una guía o marco de acción en el que podemos basarnos a la hora de trabajar. Por otro lado, una metodología es un proceso sistemático que por la búsqueda de los resultados deseados no da espacio para innovar o ser creativos.

En la metodología se siguen firmemente los procedimientos establecidos que entregan resultados constantes. En el marco de trabajo en cambio, hay flexibilidad, pero eso implica un grado de ambigüedad ya que no existen caminos exactos para resolver un problema.

Gestión Ágil de Proyectos (AgilePM)

Basado en los aportes de los autores (Tomek & Kalinichuk, 2015), la teoría Agile se encarga de la comprensión de la secuencia de tareas a realizar, asumiendo que una especificación detallada de diseño puede ser impulsada con la colaboración del contratante, los diseñadores e inversionistas. Visto desde el punto de vista de las ciencias económicas o computacionales, la metodología Agile requiere dos tipos de planes: a largo plazo (realización completa del proyecto) y la a corto plazo (cada iteración).

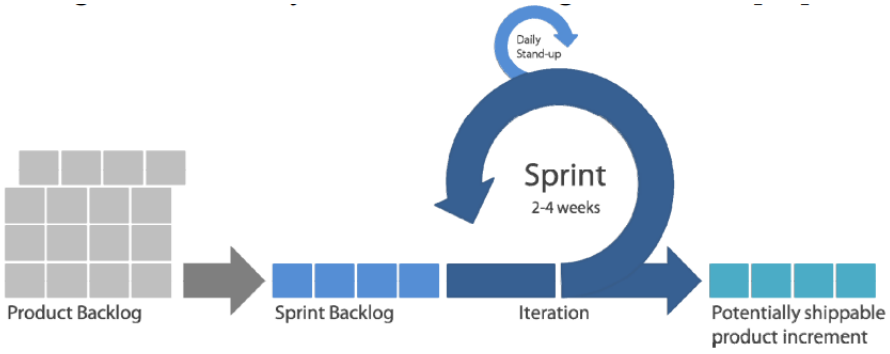
Herramientas Ágiles en el Mercado

Scrum. En este caso Scrum, representado en la Figura 2, trata de un marco de trabajo o por su traducción al inglés “framework”. Determina la manera de trabajar entre los diferentes

miembros del equipo para que se consigan resultados en el menor tiempo con una colaboración eficaz (Molina et al., 2018).

Figura 2

Representación gráfica de Scrum Framework



Nota. Representación del marco de trabajo Spring Framework, Tomado del artículo del autor (Hammad & Inayat, 2019).

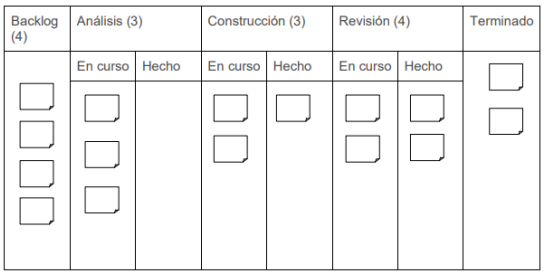
La administración del tiempo se considera la pieza fundamental de Scrum. Para su control se define una cantidad específica de tiempo para desarrollar cada una de las actividades en los proyectos, de esta forma las miembros del equipo Scrum respetan los tiempos y además desarrollan la capacidad para reaccionar rápidamente a los cambios.

Kanban. Su traducción del japonés es “tarjetas visuales”. Se trata de un panel con columnas y tarjetas que se desplazan entre columnas, simulando la evolución de un proceso durante el tiempo, desde su inicio hasta su fin (Gracia et al., 2016).

En la Figura 3 se observa la apariencia de un tablero Kanban. Hoy en día existen populares herramientas que permiten crear tableros Kanban y gestionar proyectos en internet, entre ellos está Trello[1].

Figura 3

Tablero Kanban



Nota. La figura muestra un bosquejo de un tablero Kanban. Tomado del autor (Gracia et al., 2016).

Entre las reglas de la aplicación de metodología Kanban están:

- Visualizar los estados
- Limitar el trabajo en progreso

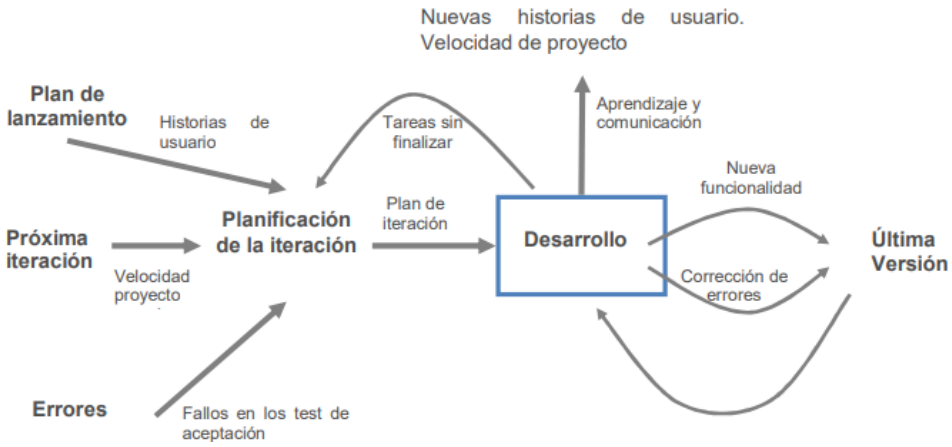
[1] Trello: Herramienta web de libre acceso proveída por la empresa Atlassian.

- Medir los flujos de trabajo

Extreme Programming. Es una metodología de desarrollo popular, creada por Kent Beck, uno de los principales propulsores del manifiesto ágil (Gracia et al., 2016). En la Figura 4 se observa el ciclo de trabajo donde destacan los roles de trabajo y las tareas o historias de usuario.

Figura 4

Representación gráfica Extreme Programming



Nota. Representación de las acciones y fases en la metodología Extreme Programming. Tomado del autor (Gracia et al., 2016).

Management 3.0. Para la autora (Melnik, 2019), es un modelo de administración flexible y adaptativo. El modelo define las siguientes características conceptuales:

- Determinar los límites.
- Desarrollo de competencias.
- Empoderar a los equipos.
- Crecer la estructura.

- Agregar energía a las personas.
- Mejorar la situación en la organización.

Fundamentos de DevOps

Transformación Digital y Productos Digitales

Desde la aparición de internet, las soluciones tecnológicas que han llegado al mercado han logrado simplificar los problemas de los usuarios y mejorar su usabilidad y experiencia.

En el desarrollo de software la evolución ha sido amplia. En los últimos años las soluciones que han destacado es la capacidad de utilizar el internet para desplegar servidores y servicios a través del *Cloud Computing.* Incluso soluciones como el *DevOps* ahora permiten que los requerimientos de un sistema se conviertan rápidamente en funcionalidades publicadas en producción, esto a través de procesos automatizados de integración, entrega y despliegue.

Fundamentos del Enfoque DevOps

En el 2009, Patrick Debois acuñaría el término DevOps en un evento tecnológico de nombre “DevOps Days” en Bélgica (Farías Alejandro, 2017). Muchos autores lo definen como un movimiento, otros como una filosofía. DevOps es una metodología de desarrollo de software especializada en la entrega continua.

El término “DevOps” es la unión de dos palabras del idioma inglés, por un lado “development” que se traduce como “desarrollo” y hace referencia a las personas encargadas de construir la solución software y “operations” que quiere decir “operaciones”. Las operaciones en el desarrollo se refieren al despliegue y monitoreo del software en ambiente de producción. En la Figura 5 se grafica lo mencionado, agregando el “aseguramiento de la calidad” como otro apartado fundamental del DevOps.

Figura 5

Representación gráfica de DevOps y los elementos que lo forman

Nota. Esta figura agrupa los elementos que forman parte de DevOps.

Para Velásquez (2018), DevOps es un movimiento ágil que se centra en la comunicación y colaboración del equipo de desarrollo y el equipo de operaciones, en donde tanto el equipo de operaciones y el equipo de desarrollo tienen la misma meta a colaborar.

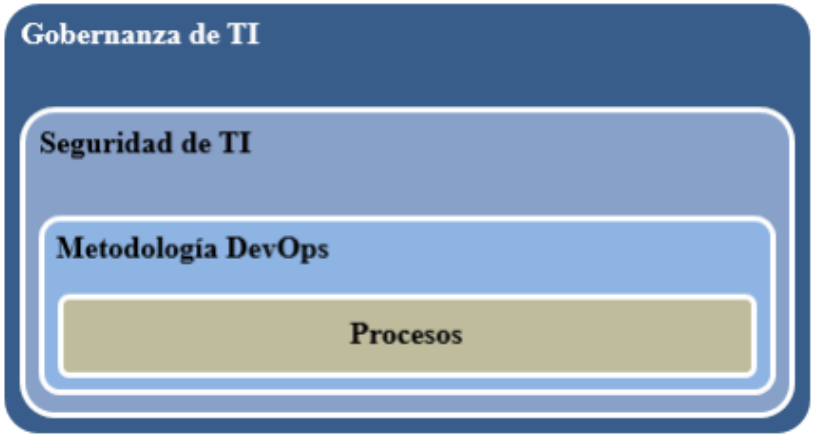
Gestión de la Implementación de DevOps en la Empresa

En la actualidad, adoptar DevOps en los procesos significa un cambio cultural, ya que aboga por la comunicación, colaboración continua y automatización de los procesos (Quintero Parra & Daza Benjumea, 2017).

A nivel nacional, empresas como Corporación Favorita han llevado a cabo esfuerzos para conseguir una reingeniería de sus procesos de TI basada en DevOps. Según Estrella y Landázuri (Landázuri & Estrella, 2019) el resultado logró implementar herramientas de control y administración en cada fase del desarrollo y operación de la empresa. En la Figura 6 se muestra la arquitectura de modelo de sus procesos una vez que se implementó DevOps.

Figura 6

Arquitectura de modelo de los procesos de Corporación Favorita.



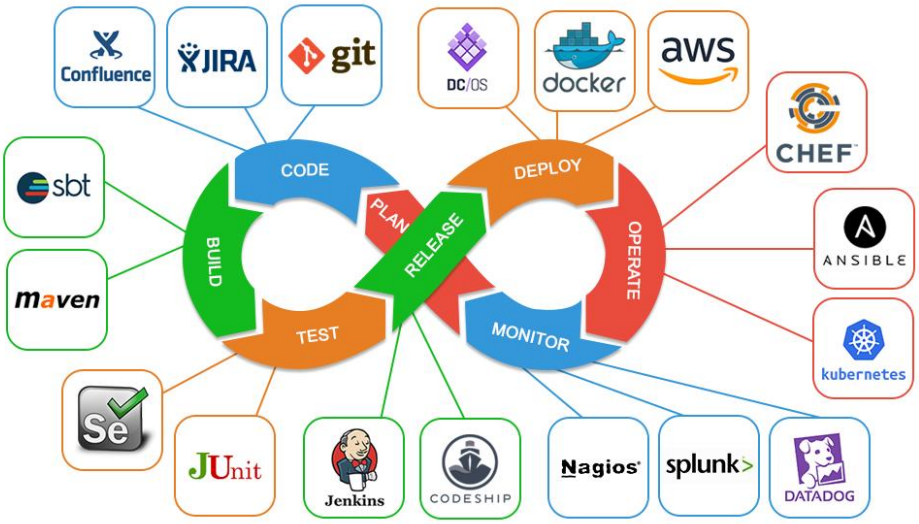
Nota. Esta figura explica la implicación de los procesos de TI en una compañía con DevOps. Obtenido de los autores (Landázuri & Estrella, 2019).

DevOps Toolchain

DevOps Toolchain se traduce como “cadena de herramientas de DevOps”. Este término hace referencia a todos los softwares que facilitan la implementación de DevOps, véase la Figura 7. Estas herramientas cubren el proceso desde la fase de planificación hasta la fase de monitoreo.

Figura 7

Representación del Ciclo DevOps y la variedad de herramientas software para cada fase.



Nota. El gráfico detalla softwares propietarios y de código libre que se pueden ocupar en las diferentes etapas de DevOps. Tomado del portal (Landázuri & Estrella, 2019).

Es amplia la gama de herramientas disponibles, sin embargo, es necesario recordar que lo importante es entender el enfoque de cada una de las fases, después con las herramientas que estén a disposición se puede conseguir el fin de implementar DevOps dentro de un sistema de información.

Una forma didáctica de entender el DevOps Toolchain es mediante la Tabla Periódica de DevOps de la compañía DigitalAI (Digital.ai, 2020). Esta representación gráfica muestra docenas de herramientas en todas las áreas de DevOps. La tabla considera 17 categorías a través de 5 tipos de licencias, desde código abierto hasta software propietario.

En la Figura 8 se aprecia la representación de la Tabla Periódica de DevOps. La herramienta en línea también permite conocer opciones relacionadas entre las que se escoja.

Además, se puede definir un conjunto de herramientas para crear un *"Pipeline"* proceso que involucra todo el ciclo de DevOps.

Figura 8

Tabla de representación las herramientas DevOps del mercado.

Nota. Herramientas DevOps ordenadas según la etapa del ciclo de desarrollo de software en la que son utilizadas. Tomado del portal https://digital.ai/periodic-table-of-devops-tools y de los autores (Landázuri & Estrella, 2019).

A continuación, listaremos las categorías de la Tabla Periódica y las herramientas que se destacan dentro de ellas según la fuente (Digital.ai, 2020):

- Reportes
    - ***Instana:*** Aplicación de reportes con características de Inteligencia Artificial.
    - ***Datadog:*** Plataforma de monitoreo ara equipos de desarrollo y operaciones.

- Administrador de paquetes
    - ***Jfrog Artifactory:*** Repositorio para almacenar paquetes y artefactos
    - ***Sonatype Nexus:*** Plataforma que ofrece control total sobre los riesgos y vulnerabilidades de las aplicaciones.
- Nube
    - ***AWS:*** Plataforma de la empresa Amazon. Ofrece los servicios de PaaS, IaaS y SaaS.
    - ***Apprenda:*** Ofrece el servicio de PaaS.
- Colaboración
    - ***Slack:*** Aplicación que permite la comunicación textual a través de canales.
    - ***Microsoft Teams:*** Evolución de Skype. Permite comunicación textual y videoconferencias.
- Configuración de la automatización
    - ***Red Hat Ansible:*** Plataforma para la configuración de aplicaciones.
    - ***Terraform:*** Herramienta para construir y lanzar infraestructura de forma segura.
- Contenedores
    - ***Docker:*** Plataforma que encapsula el software dentro de contenedores.
    - ***Kubernetes:*** Manipula contenedores para ser publicados o dados de baja. También se conoce como un orquestador de contenedores.

- Integración continua
    - ***Jenkins:*** Herramienta para la integración desarrollada en Java.
    - ***Azure DevOps Code:*** Provee servicios para planificar e integrar código.
- Bases de datos
    - ***Liquibase:*** Librería para detectar los cambios sobre una base de datos.
    - ***Delphix:*** Facilita la actualización y escalamiento de bases de datos relacionales.
- Despliegue
    - ***Azure DevOps Pipeline:*** Provee servicios para desplegar software.
    - ***UrbanCode Deploy:*** Herramienta para despliegues automáticos.
- Planificación ágil empresarial
    - ***Jira Align:*** Aplicación basada en Kanban e integrada con Git.
    - ***Targetprocess:*** Permite concentrarse en las tareas prioritarias y agendar el trabajo.
- Seguimiento de problemas
    - ***Jira:*** Dispone de foros por cada incidencia que se cree.
    - ***Trello:*** Utiliza la metodología Kanban y es de libre acceso.
- Administrador de lanzamientos
    - ***AWS CodePipeline:*** Aplicación que construye y despliega código.

    - ***CloudBees Flow:*** Plataforma orquestadora de aplicaciones.

- Seguridad
    - ***OWASP ZAP:*** Aplicación gratuita capaz de encontrar vulnerabilidades.
    - ***Aqua Security:*** Detecta los problemas que pueden ocurrir en el paso del entorno de desarrollo al de producción.

- Serverless
    - ***AWS Lambda:*** Permite ejecutar código sin tener un servidor.
    - ***Heroku:*** Aloja aplicaciones en la nube y es capaz de desplegarlas.

- Administrador de código
    - ***Git:*** Sistema de control de versiones para sistemas operativos.
    - ***GitHub:*** Sistema de control de versiones en internet.

- Pruebas
    - ***Selenium:*** Es un framework para las pruebas de software.
    - ***JUnit:*** Framework para pruebas unitarias desarrollado en Java.

- Administrador del valor e impacto
    - ***Tasktop:*** Integra todas las herramientas utilizadas en el ciclo DevOps.
    - ***GitLab:*** Ofrece una suite completa de herramientas que cubre el proceso de DevOps.

Contenedores y Computación Serverless

Los contenedores y la computación serverless nacen como la solución para implementar sistemas de información en la nube, sin la necesidad de que ese servicio implique poner a trabajar a un servidor dedicado en la empresa o en las granjas de servidores de las empresas proveedoras de infraestructura como código.

Infraestructura Como Código

Se refiere a la automatización del despliegue de aplicaciones sobre servidores. Estos servidores son totalmente configurables desde código (Banchoff & Fernandez, 2019). También es conocida como *"IaaS"* gracias a su traducción al inglés *"Infrastructure as a service"*. Esta solución elimina del esquema de roles a los administradores de servidores que había en el pasado ya que ahora los servicios *"IaaS"* son proveídos por empresas en la nube. En la Figura 9 se muestra una representación de cómo el código es el responsable de levantar servidores.

Figura 9

Representación gráfica la Infraestructura como código

Nota. La figura muestra una representación de cómo se puede habilitar infraestructura desde código. Tomado del portal (Combe et al., 2016).

Entre los proveedores destacados del mercado se encuentra la empresa Amazon con Amazon Web Services (AWS), Google con su servicio Google Cloud Platform (GCP) y la empresa DigitalOcean con su servicio de tecnología en la nube.

Contenedores

Los contenedores de código han surgido de la necesidad de encapsular varias partes de software en un solo lugar, tal como funcionan los contenedores internacionales de envío y entrega de bienes materiales. En la Figura 10 se muestra un ejemplo.

El artículo (Merkel, 2014) propone “imagina que puedes empaquetar una aplicación con todas sus dependencias fácilmente y ejecutar, probar y desplegar”. El concepto de contenedores sería el nacimiento de piezas fundamentales del DevOps como son la compilación, entrega y despliegue continuo.

Figura 10

Relación de contenedores con contenedores computacionales

Nota. Representación de la independencia y autonomía de los contenedores computacionales. Tomado del portal (Combe et al., 2016).

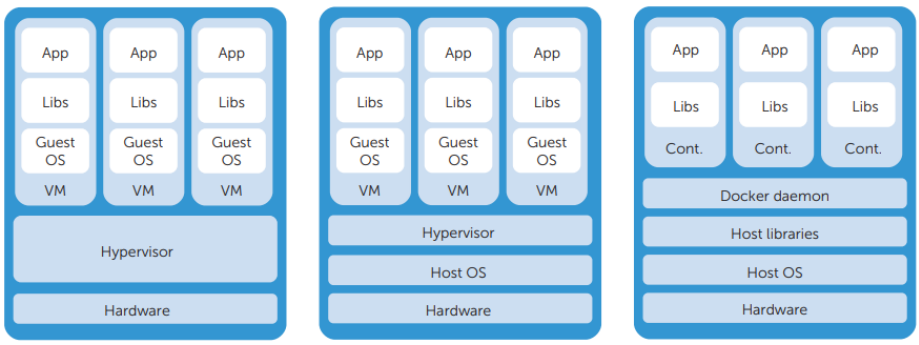
Los contenedores se encargarán de que todo se junte y funcione. Cuando uno se cae, existe otro con las mismas configuraciones, listo para dar contingente. La tecnología Docker fue la creadora de los Contenedores.

Docker

Es una solución informática nacida en el 2013 gracias al auge del Cloud Computing. Según Combe et al.(Combe et al., 2016), es la solución para la virtualización ligera de software a través de contenedores. En la Figura 11 se muestra una comparación entre ambientes de producción normales y uno con Docker (ambiente C). Es una forma de entender como Docker simplifica la virtualización (VM) y ayuda a desplegar ambientes de producción de forma rápida.

Figura 11

Comparación entre la estructura de máquinas virtualizadas y Docker.



Nota. La figura muestra la diferencia entre el sistema de archivos de máquinas virtuales a cómo lo hace Docker. Tomado de los autores Combe et al.(Combe et al., 2016).

Orquestación de Contenedores

Se conoce con este nombre a la capacidad a la tecnología capaz de gestionar un amplio número de contenedores que prestan un servicio desplegado en diferentes servidores y equipos (Nogués García, 2018).

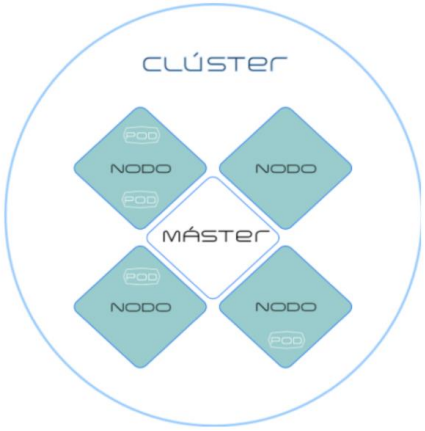
Para muestra, controlar el despliegue de un grupo de contenedores a la vez que se ejecutan en equipos en la nube y local es posible a través de orquestadores de contenedores. Existen opciones bastante populares en el mercado para realizar esta tarea, entre ellas las principales: Docker Swarm y Kubernetes.

Kubernetes. Según el autor (Nogués García, 2018), Kubernetes es una plataforma creada por Google y ahora liberada para que evolucione con los aportes de la comunidad de desarrollo. Se creó para orquestar contenedores Docker y gestionar sus ciclos de vida.

Su forma de trabajar es distribuyendo los contenedores en *"Pods"* y el conjunto de estos forman un nodo. Luego, al tener un grupo de nodos, se crea un *"clúster"* de contenedores. En la Figura 12 se encuentra una representación de lo antes mencionado.

Figura 12

Arquitectura de Kubernetes



Nota. Representación de los nodos máster e hijos de un clúster de contenedores. Tomado del autor (Nogués García, 2018).

Herramientas de Computación Serverless

AWS Lambda. Alternativa del grupo de herramientas AWS para crear aplicaciones sin necesidad de un servidor. Según su sitio web (Amazon, n.d.), AWS Lambda permite ejecutar código sin administrar servidores. La forma en la que la empresa mide el uso es a través de las consultas o la interacción que se tienen con el servicio en cuestión. A esto se le llama una "carga de trabajo".

Se puede ejecutar código de cualquier lenguaje y de todo tipo. La herramienta permite cargar el código desde un archivo comprimido .Zip. La compilación y verificación de código es una tarea que realiza Lambda.

Google Cloud Functions. Google lo llama "*FaaS*" o "*Function as a Service*", su traducción es función como servicio, es la apuesta de la compañía para ejecutar código en la nube y ofrecerlo de forma masiva a través de internet.

Su sitio web (Google Cloud, n.d.) asegura que el escalamiento es automático, cuenta con seguridad integrada y no habrá necesidad de actualizar servidores.

Azure WebJobs. Es la solución de la compañía Microsoft para ejecutar código en la nube sin necesidad de un servidor. Tiene la capacidad de planificar acciones y configurar eventos que ocurran cada cierto tiempo.

De esta forma la compañía provisionó a su Suite de herramientas Azure con una solución de tipo función como servicio.

Planificación de Proyectos DevOps y Cloud Computing

Fases de DevOps

Planificación. En esta fase se debaten y redactan los requerimientos. Se crean las tareas del proyecto.

Codificación. Creación y desarrollo del código según la tarea. Existen varias herramientas para la creación de código o IDE[2] que permiten la programación.

[2] IDE: Integrated Development Environment, en su traducción al español, Entorno Integrado de Desarrollo.

Compilación. En esta fase se realiza el análisis del código en un estado estático para verificar que no existen errores. Además, se realizan las pruebas unitarias que tienen la capacidad de notificar cuál de las pruebas no se han cumplido hasta el momento.

Pruebas. Esta fase se ejecuta antes del despliegue al servidor y se encarga de automatizar las pruebas para que, una vez respondan de forma positiva, automáticamente se de paso al despliegue de la aplicación.

Despliegue. Esta fase consiste en la configuración del proyecto para que el despliegue sea automático.

Operación. Se encarga de que los sistemas funcionen de forma constante y se encuentren siempre en operación. Normalmente se suelen usar herramientas para recibir retroalimentación por parte de los usuarios activos del sistema.

Monitoreo. Esta fase se detectan fallos y se observan los fallos que hayan ocurrido a la hora de la complicación, pruebas y despliegue del sistema. Se puede hacer un seguimiento de la disponibilidad de cada una de las versiones con fines estadísticos para los interesados.

Definición de KPI's para DevOps

Los autores (Bertolino et al., 2020) aseguran que normalmente el rendimiento y la seguridad son los KPIs más considerados a la hora de hablar de DevOps. Ellos consideran que la fiabilidad del software debe entrar entre el grupo de indicadores claves.

- Rendimiento
- Seguridad
- Fiabilidad del software

Por otro lado, la misma investigación asegura que se pueden considerar factores como: frecuencia de despliegue, tiempo requerido para cambios, tiempo tomado para restaurar un servicio, promedio de fallos en los cambios.

Políticas de Seguridad y Requisitos

La seguridad es uno de los factores pendientes de DevOps. Varios análisis apuntan a este apartado como una de sus debilidades.

La investigación (Castro Sánchez, 2020) determina que mientras más rápido se despliegan los sistemas a raíz de DevOps, más garantías se tienen que otorgar para una segura publicación de los sistemas.

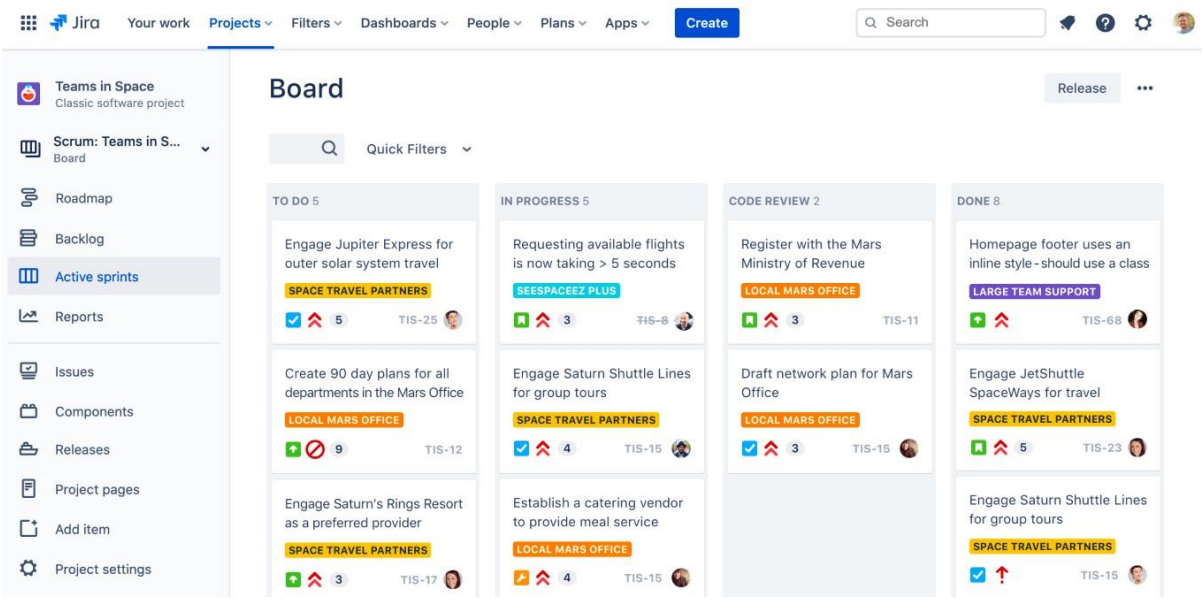
Herramientas para Planificación

Jira. Es una aplicación creada por la empresa Atlassian que permite la planificación y control de proyectos de desarrollo de software. Utiliza la metodología Kanban para manipular tarjetas a través de columnas

En la Figura 13 se observa una captura de su ejecución.

Figura 13

Pizarra de trabajo en Jira



Nota. Captura de pantalla de una pizarra de trabajo ágil en Jira. Tomado de Atlassian *https://www.atlassian.com/es/software/jira*.

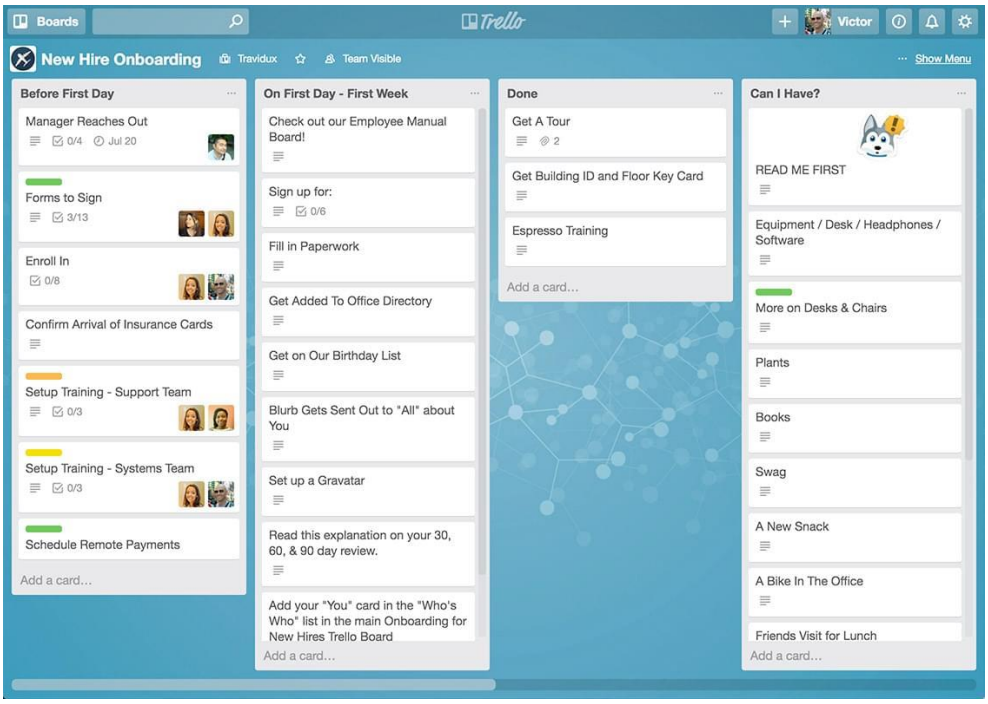
Trello. Es la alternativa gratuita de la empresa Atlassian para la planificación. Funciona de forma parecida a Jira, pero con un diseño sencillo y menos funcionalidades.

Su enfoque no es directamente para proyectos de desarrollo de software solamente, sino que invitan a todo tipo de personas a que la utilicen.

En la Figura 14 se muestra un ejemplo de uso de Trello.

Figura 14

Pizarra de trabajo en Trello

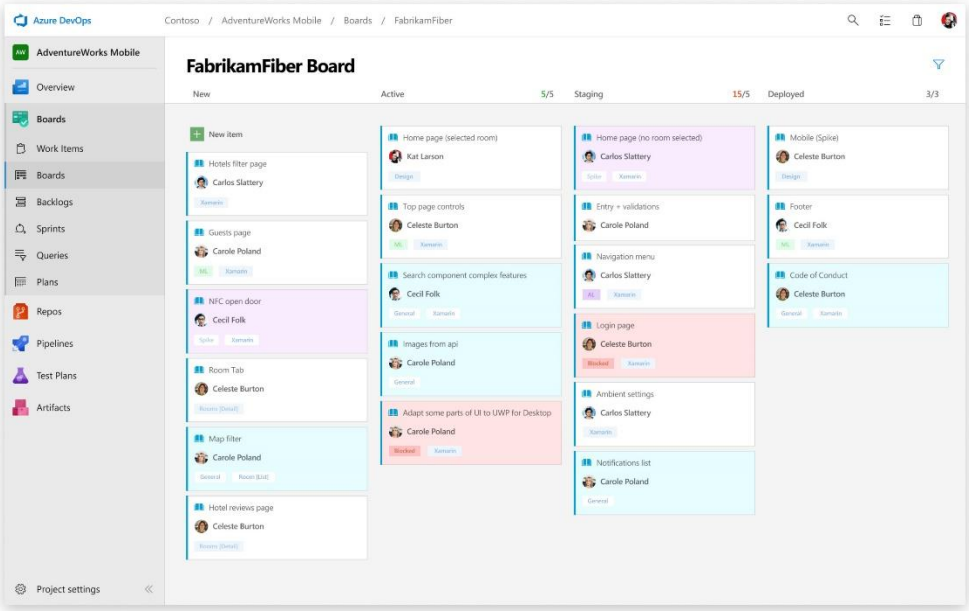


Nota. Captura de pantalla de una pizarra de trabajo ágil en Trello. Tomado de Trello *https://crjdesign.co.uk/*.

Azure DevOps Boards. Es la apuesta de la empresa Microsoft a la primera fase del ciclo DevOps, la planificación. Es parte de la suite de herramientas disponibles dentro Azure DevOps. Es competencia directa de Jira, pero para diferenciarse, tiene la capacidad de levantar infraestructura como código y publicar aplicaciones cumpliendo con el ciclo de DevOps. En la Figura 15 se muestra una demostración de uso de Azure Boards.

Figura 15

Pizarra de trabajo en Azure DevOps Board



Nota. Captura de pantalla de una pizarra de trabajo ágil en Azure DevOps. Tomado de Microsoft *https://azure.microsoft.com/es-es/services/devops/boards/*.

Integración Continua

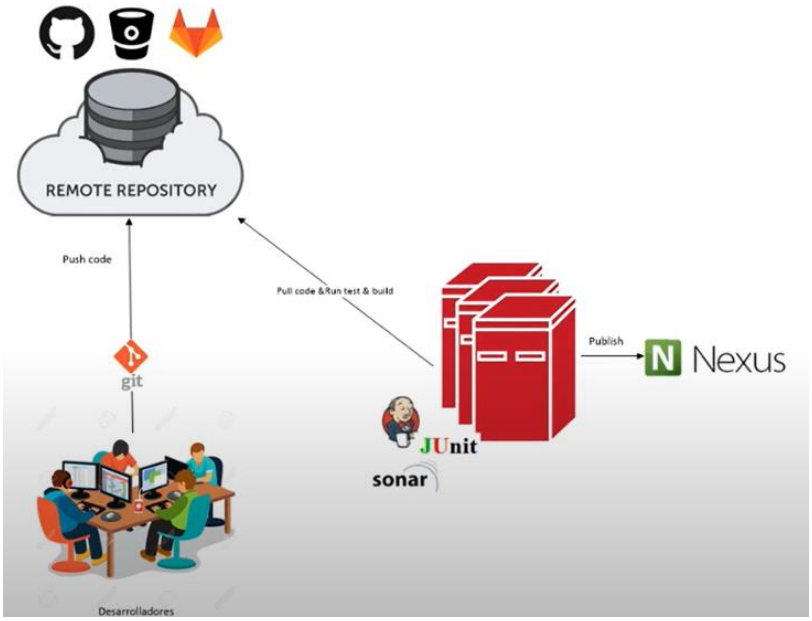
Consiste en la combinación de cada uno de los cambios que realizan sobre la base de código los miembros del equipo. Esta combinación o integración es ágil, fácil de recuperar o corregir si ha habido errores frutos de su uso. En la Figura 16 se observa una configuración de integración continua.

Como lo señalan los autores (Salamon et al., 2019), cada integración es automática y tiene el fin de detectar los errores de integración lo antes posible. De esta manera se reducen

los problemas de integración del código e incremente la velocidad a la que se desarrolla el software.

Figura 16

Esquema de integración continua



Nota. Representación de un proceso de integración continua. Tomado del contenido multimedia *https://www.youtube.com/watch?v=0Pwgq3WQNYs*.

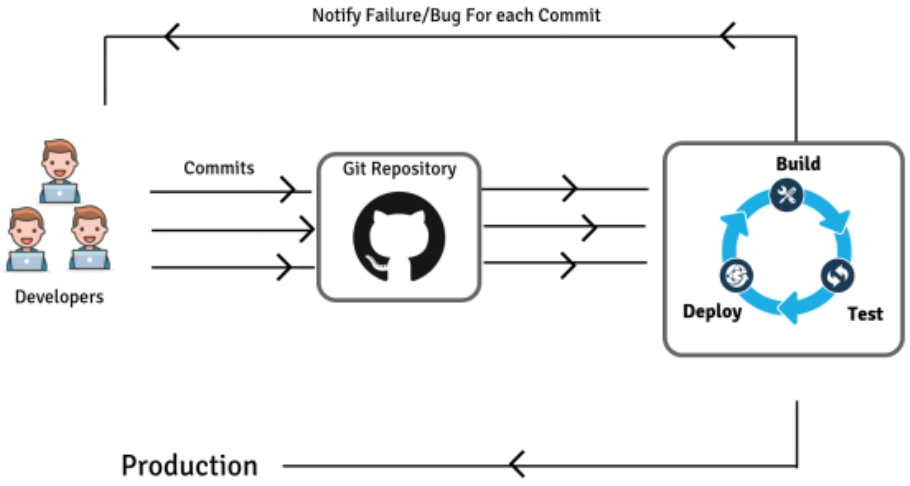
Herramientas Utilizadas Para la Integración Continua

GitHub. Plataforma de Microsoft para albergar proyectos de código. Tiene una versión gratuita y otra versión para empresas.

No se ha quedado solo con ser un repositorio de código, sino que ha implementado nuevas funcionalidades como sus conocidos *"Actions"* son eventos capaces de ejecutar instrucciones de compilación, pruebas y despliegue de código. En la Figura 17 se observa a GitHub en medio de un proceso de DevOps.

Figura 17

Representación gráfica del uso de GitHub dentro de un proceso DevOps



Nota. Ciclo de integración continua con GitHub. Tomado de QaTouch *https://www.qatouch.com/blog/continuous-integration-with-jenkins-and-GitHub/.*

GitLab. GitLab, al igual que GitHub, es una plataforma de alojamiento de aplicaciones y de seguimiento de problemas basada en la tecnología Git (O'Grady, 2018). El mismo autor asegura que la plataforma fue lanzada en 2011 y sus funciones han ido avanzando durante los años. La versión Community se encuentra disponible de forma gratuita, mientras que GitLab EE (Enterprise Edition) es el plan empresarial que ofrece la compañía.

Es necesario comprender el uso de la tecnología Git. Git es una tecnología que ofrece el control absoluto de los cambios y nuevas versiones del código fuente (Dompablo Tobar et al., 2018). En una circunstancia normal, los cambios en el código se los realiza localmente y después pasa a ser publicado en un servidor remoto de versionamiento, allí es cuando entra GitLab.

Los conceptos básicos de Git son:

1. Ramas *(Branches)*: permite duplicar el código fuente sin necesidad que esto afecte al resto del código del equipo de trabajo.

2. *Master*: es la rama donde se encuentra el código en su versión final. Normalmente las ramas se crean a partir de Master.

3. *Commit*: es como se llama al cambio realizado sobre el código fuente. Un *commit* consta de un nombre, fecha y hora, responsable del cambio y las modificaciones realizadas. Se puede retroceder entre *commits* para llegar a una versión anterior.

4. *Pull request:* es la manera de confirmar la integración entre el código. Al terminar un feature es necesario integrarlo con el resto del sistema. Si el código nuevo es correcto y aceptado por el manager, entonces se confirma el *Pull request.*

Desde hace 4 años, GitLab cuenta con disponibilidad para configurar DevOps dentro de los proyectos. De esta manera, cualquier nuevo cambio que llega al código se construye y si las pruebas son positivas, entonces se pasa a generar un nuevo release. A esto se le llama un Pipeline[3].

Jenkins. Para el autor (Vadapalli, 2018), Jenkins es una herramienta basada en la web hecha en Java. Es utilizada para la construcción continua, las pruebas, el despliegue y

[3] Pipeline: es el conjunto completo de procesos que se ejecutan cuando activa un trabajo de un proyecto DevOps.

además se integra con herramientas de construcción de software como Maven o el repositorio de un proyecto sobre Git.

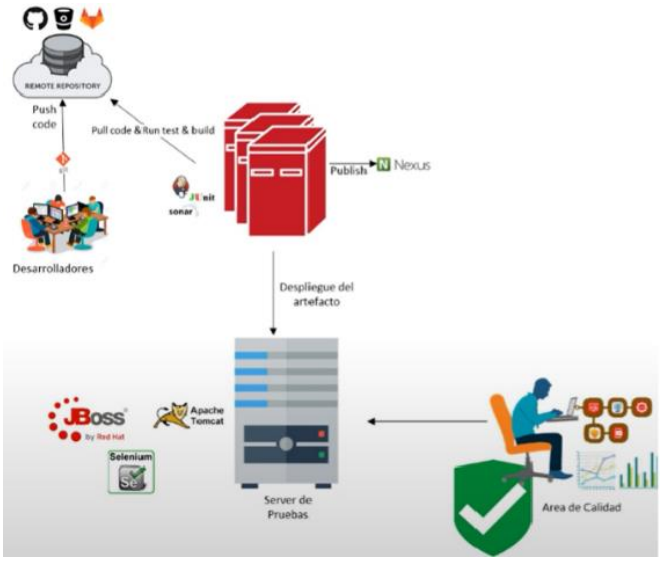
Entrega Continua

En la actualidad, pensar que se deben hacer ejecuciones manuales para compilar, probar y desplegar software en las diferentes plataformas móviles y web resulta una tarea complicada. DevOps ha querido transformar esta problemática a través de la Entrega Continua del software.

El autor (Iñiguez Sánchez, 2017) define a la entrega continua como una disciplina donde el software se construye todo el tiempo de manera que puede ser publicado a los servidores de pruebas y producción en cualquier momento. Señala, además, que a raíz de DevOps nació un proceso llamado “Pipeline de entrega continua de software”, véase la Figura 18, que se encarga de encontrar con facilidad y de forma rápida los fallos de la programación, errores regresivos y errores al integrar los componentes software.

Figura 18

Representación gráfica de un Pipeline para la entrega continua de software



Nota. Representación de un proceso de entrega continua. Tomado del contenido multimedia https://www.youtube.com/watch?v=0Pwgq3WQNYs.

PDCA: Calidad del Producto Digital

El ciclo PDCA es también llamado ciclo Deming. Aparece de las palabras en inglés *plan, do, check y act;* planear, hacer, chequear y actuar, respectivamente. Este ciclo ayuda a incrementar la productividad en diferentes procesos dentro de una organización (Aparicio Britto & Choy Balboa, 2020).

Es conocido además como el ciclo de Mejora Continua. Puede ser utilizada en cualquier disciplina, no necesariamente en desarrollo de software. Los autores (Aparicio Britto & Choy Balboa, 2020) determinan 4 pasos necesarios para cumplir con el ciclo, ellos son:

1. Definir un plan de mejora continua.
2. Realizar las correcciones.

3. Chequear los resultados obtenidos.

4. Normalizar los progresos logrados.

Pruebas Estáticas

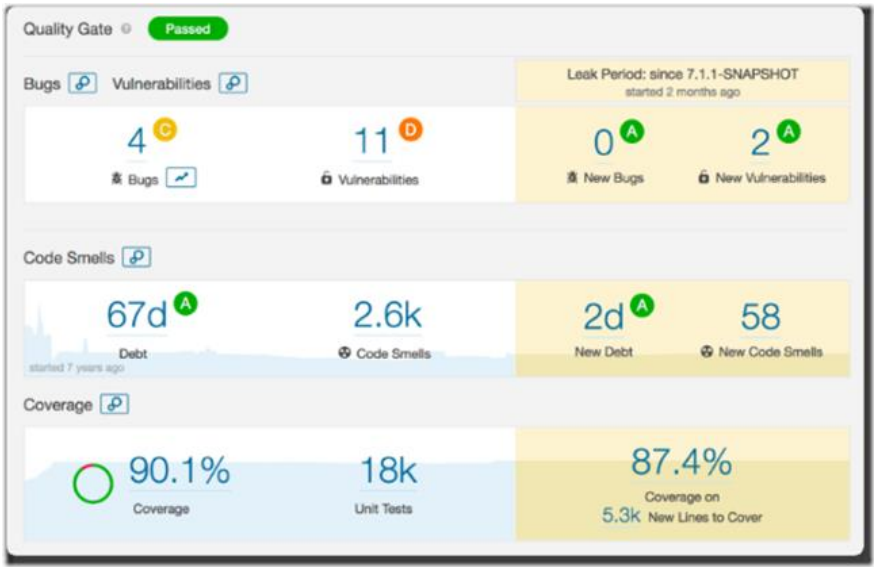
Las pruebas estáticas desde inicios de siglo y su proceso ha sido especificado por el autor (Sommerville, 2006). En la actualidad, existen herramientas digitales capaces de automatizar las revisiones de código e inspeccionar de forma que se puedan notificar los potenciales Bugs e incidencias encontradas.

Entre las herramientas disponibles está SonarQube. En la Figura 19 se muestra un reporte de pruebas generado por SonarQube y el resultado del reporte es “PASSED” que quiere decir “APROBADO”. De esta forma el Pipeline puede continuar a la siguiente fase de despliegue continuo.

SonarQube es una herramienta de tipo propietario que aún se mantiene vigente luego de varios años de haber sido publicada.

Figura 19

Reporte de pruebas de SonarQube



Nota. Captura de pantalla de un reporte de SonarQube. Tomado del portal (Iñiguez Sánchez, 2017).

Pruebas Dinámicas

Las pruebas dinámicas se realizan mediante código ejecutado. Según los autores (Tuya et al., 2007), entre el abanico de pruebas dinámicas están las pruebas de caja blanca o estructurales, que utilizan el código fuente y su estructura para seleccionar casos de prueba y por otro lado están las de caja negra o funcionales, que considera únicamente las entradas y salidas.

La investigación ha establecido una clasificación de cuatro pruebas dinámicas para ser analizadas, ellas son:

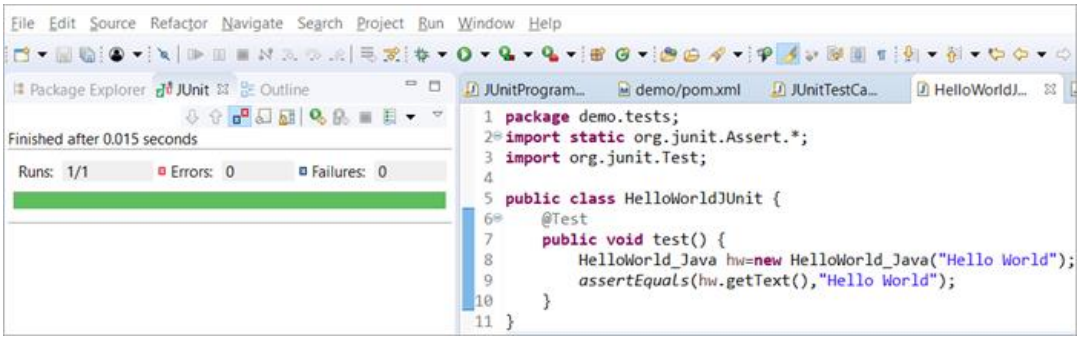
Pruebas de Unidad

Es la primera forma de encontrar errores en el código ejecutado (Tuya et al., 2007). Prueban módulos considerados la unidad más pequeña de diseño. Estos módulos pueden ser una función o un método en una clase. Los autores especifican que las pruebas son ejecutadas por los mismos programadores que construyen el módulo.

Una de las herramientas populares para el desarrollo de pruebas unitarias en lenguaje Java es JUnit. La herramienta se describe como un framework para escribir pruebas unitarias repetibles. En la Figura 20 se observa la ejecución de una prueba unitaria con JUnit en el IDE Eclipse.

Figura 20

Prueba unitaria en lenguaje Java con JUnit



Nota. Captura de pantalla de una prueba unitaria. Tomado del portal (https://myservername.com/).

Pruebas de Integración

Estas pruebas utilizan los módulos antes probados y su funcionalidad es encontrar problemas que pueden aparecer en la interacción entre módulos. Es un error pensar que, si los módulos han sido exitosamente probados, entonces el funcionamiento colectivo de ellos también será exitoso. Aquí se utilizan las pruebas de integración.

Un ejemplo de prueba de integración es el validar una pantalla de inicio de sesión en una aplicación web. El software Selenium ofrece la opción de simular la interacción de un usuario en un navegador de internet y de esta manera realizar pruebas automatizadas (*Selenium*, n.d.). Siguiendo el ejemplo, se necesita validar la entrada de texto en los campos correo electrónico y contraseña, además probar el módulo del botón “iniciar sesión”. A esto se le llama una prueba de integración de módulos. En la Figura 21 se muestra un ejemplo de uso de Selenium.

Figura 21

Ejemplo de prueba de integración con elementos web usando Selenium y Java

```
WebElement element = driver.findElement(By.name(name));

element.sendKeys("mystery magic");
element.submit();

(new WebDriverWait(driver, 10)).until(
    new ExpectedCondition<Boolean>() {
        public Boolean apply(WebDriver d) {
            return d.getTitle()
            .toLowerCase().startsWith("mystery");
        }
    }
);

System.out.println(driver.getTitle());
```

Nota. Extracto de una prueba de integración. Tomado del portal (https://www.slideshare.net/).

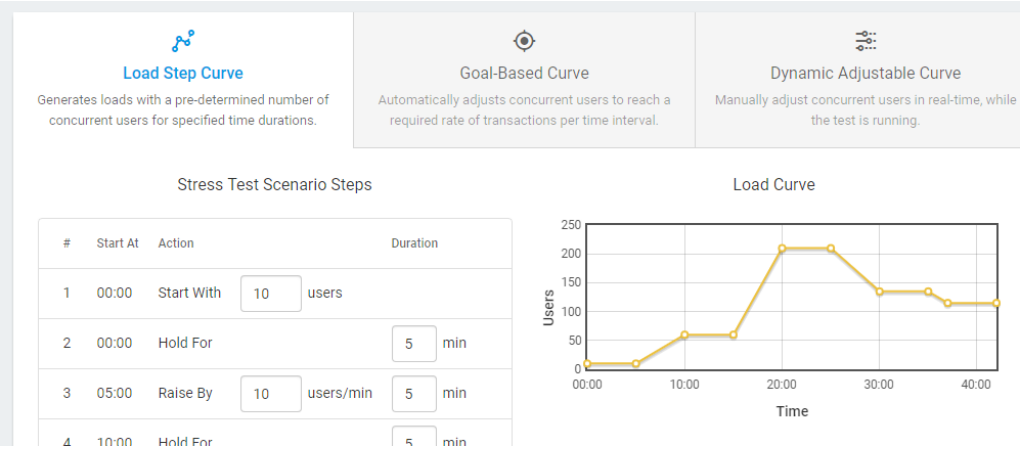
Pruebas de Rendimiento

Las pruebas de rendimiento han sido creadas para asegurar que el software procese la carga de trabajo esperada. Normalmente, su ejecución consiste en preparar una secuencia de pruebas en las que la carga va aumentando hasta que el software no pueda responder (Sommerville, 2006).

Necesariamente las pruebas de rendimiento deben estresar al sistema, por esta razón se las conoce como pruebas de estrés. De forma común para tareas como esta se utiliza la herramienta JMeter. JMeter ha sido desarrollado por la empresa fundación Apache y tiene la capacidad para simular múltiples usuarios virtuales interactuando en simultaneo en un mismo servidor web. En la Figura 22 hay un ejemplo de prueba con JMeter.

Figura 22

Prueba de rendimiento de un script JMeter con la herramienta LoadView



Nota. Captura de pantalla de una prueba de rendimiento con JMeter. Tomado de https://www.loadview-testing.com

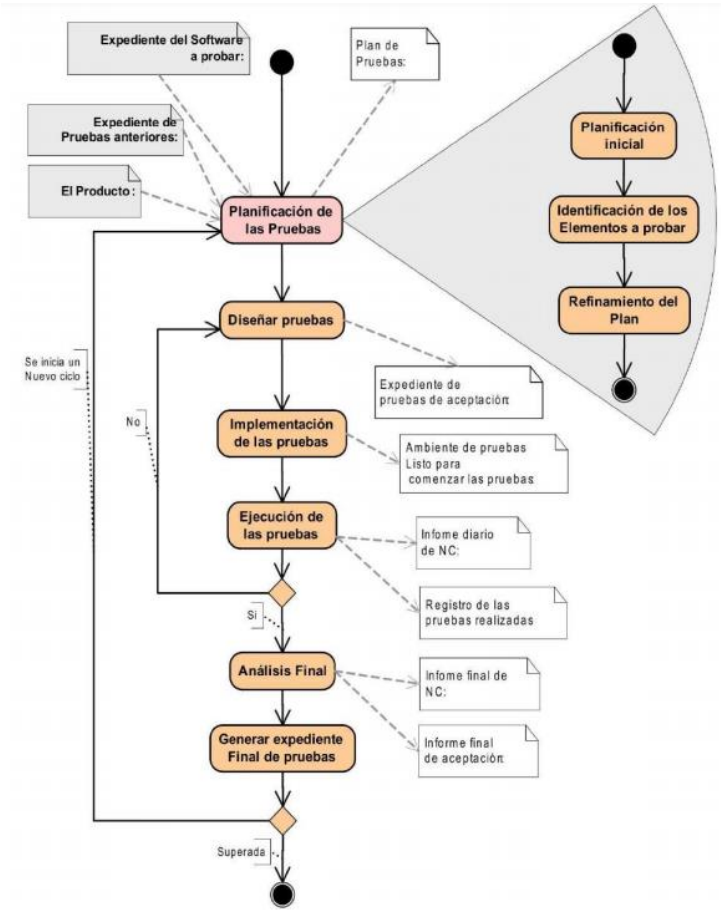
Pruebas de Aceptación

Según los autores (Pardo Matos & Febles Estrada, 2014), las pruebas de aceptación buscan determinar si el software cumple o no los con los criterios de aceptación del cliente final. Se realiza un proceso formal donde el cliente acepta o rechaza de un módulo, componente o sistema.

En la Figura 23 se muestra un flujo de trabajo de prueba de aceptación (Fernandez Prezl et al., 2007).

Figura 23

Propuesta de flujo de trabajo para pruebas de aceptación



Nota. La figura explica los pasos a desarrollar en una prueba de aceptación de software. Tomado de la investigación (Fernandez Prezl et al., 2007).

Despliegue Continuo

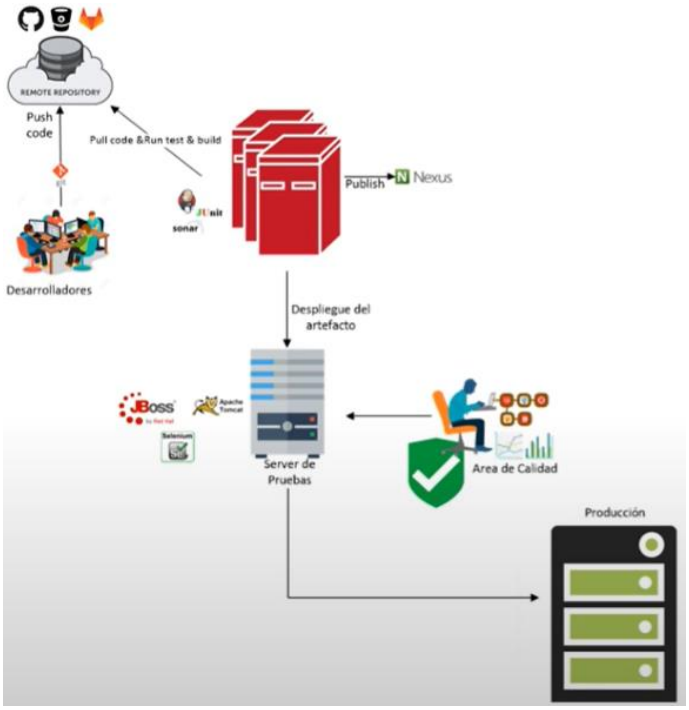
Es el proceso por el cual los cambios son implementados en producción de forma automática. En este punto el artefacto a ser desplegado ya ha sido validado y ha pasado por las pruebas. El cliente ya puede observar los cambios solicitados.

Normalmente, un nuevo despliegue se acompaña con un nombre de versión o release y sus datos históricos como fecha, hora y duración de la ejecución del Pipeline son guardados

en un histórico para los futuros reportes. La Figura 24 describe una demostración de un flujo de despliegue continuo.

Figura 24

Flujo de trabajo de Despliegue Continuo



Nota. Representación de un proceso de despliegue continuo. Tomado del contenido multimedia https://www.youtube.com/watch?v=0Pwgq3WQNYs.

Coordinación del Lanzamiento de los Productos Software

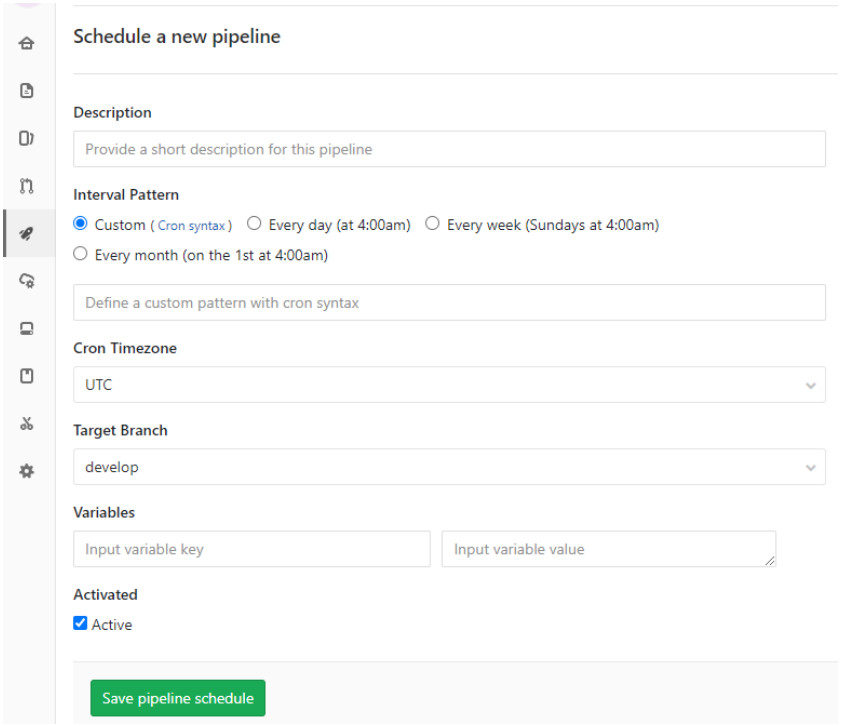
Una de las etapas finales del ciclo DevOps es el despliegue o en su traducción al inglés "*deploy*" del software. Durante el Pipeline los sistemas están construido dentro de los contenedores y listos para ser publicados.

Las herramientas para el despliegue como Jenkins o GitLab ofrecen la oportunidad de programar publicaciones de las aplicaciones que han sido correctamente construidas. En la

Figura 25 se muestra un ejemplo de configuración para programar un lanzamiento dentro de la herramienta GitLab.

Figura 25

Configuración de un despliegue programado en GitLab



Nota. Captura de pantalla de la creación de un Pipeline en GitLab. Capturada por el autor: Cabrera Córdova, Anthony Rolando.

Detección de Errores y Mejoras

Durante la ejecución y compilación de los Pipelines, la manera de observar su una ejecución es o no correcta, es a través del resultado final que puede ser exitoso o no exitoso.

Si fue exitoso, quiere decir que no presento ningún tipo de error durante su ejecución. Pero si hubiese problemas, entonces las novedades se encontraron dentro del Log de ejecución de la herramienta que se está utilizando.

Herramientas Para el Despliegue Continuo

AWS CodeDeploy. Servicio de Amazon que automatiza implementaciones de aplicaciones en diferentes infraestructuras, Facilita la rápida actualización de los servicios a través de sus cambios.

Entre los beneficios que la herramienta agrega (Amazon, n.d.-a), están:

1. Reducir tiempos de inactividad.
2. Automatizar los despliegues.
3. Herramienta intuitiva y centralizada

IBM Urbancode. Herramienta parecida a AWS CodeDeploy, en este caso de la empresa IBM (IBM, n.d.). Entre sus principales características están:

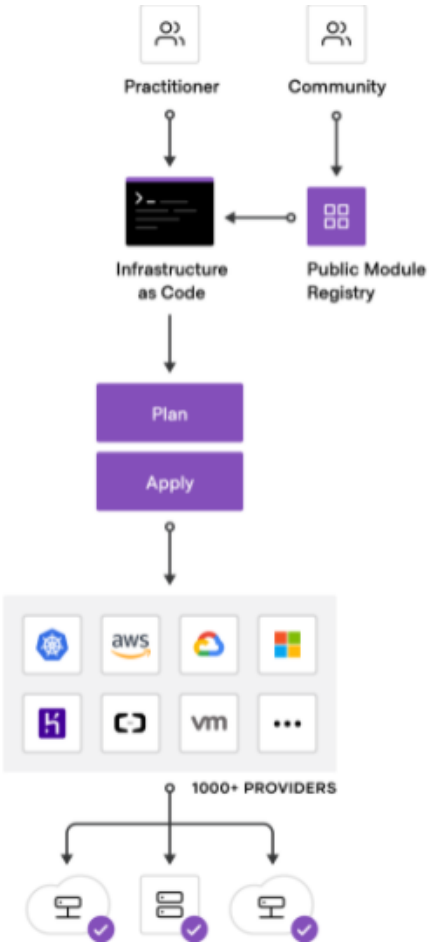
1. Velocidad al desplegar.
2. Automatizar cada proceso.
3. Posibilidad para escalar la aplicación en cualquier momento.

Terraform. Uno de los servicios en cuanto a configuración de software se trata. Terraform fue creado por Hashi Corp. Se utiliza a través de un CLI y es capaz de levantar infraestructura en segundos.

En la Figura 26 se la vinculación de Terraform al proceso de despliegue de los sistemas y su capacidad de trabajar con cualquier servicio en la nube de publicación de código.

Figura 26

Uso de Terraform como infraestructura como código



Nota. Representación de la funcionalidad de la herramienta de infraestructura como código, Terraform. Tomado de Terraform https://www.terraform.io.

Monitoreo Continuo y Seguridad en la Nube

Monitoreo del Software en Producción

La implementación del software se encuentra permanentemente reportando su estado a través de los Logs. En este caso, es posible identificar los Logs de la aplicación a través de

los contenedores que han levantado la solución y han comentado si ha aparecido alguna novedad a la hora de la compilación pruebas o despliegue.

Para el monitoreo, es necesario tomar en cuata los KPIs que fueron señalados con anterioridad y usando una de las herramientas disponibles en la web, medir cada uno de los resultados.

Métricas de Producción

Los valores a medir son amplios, entre ellos:

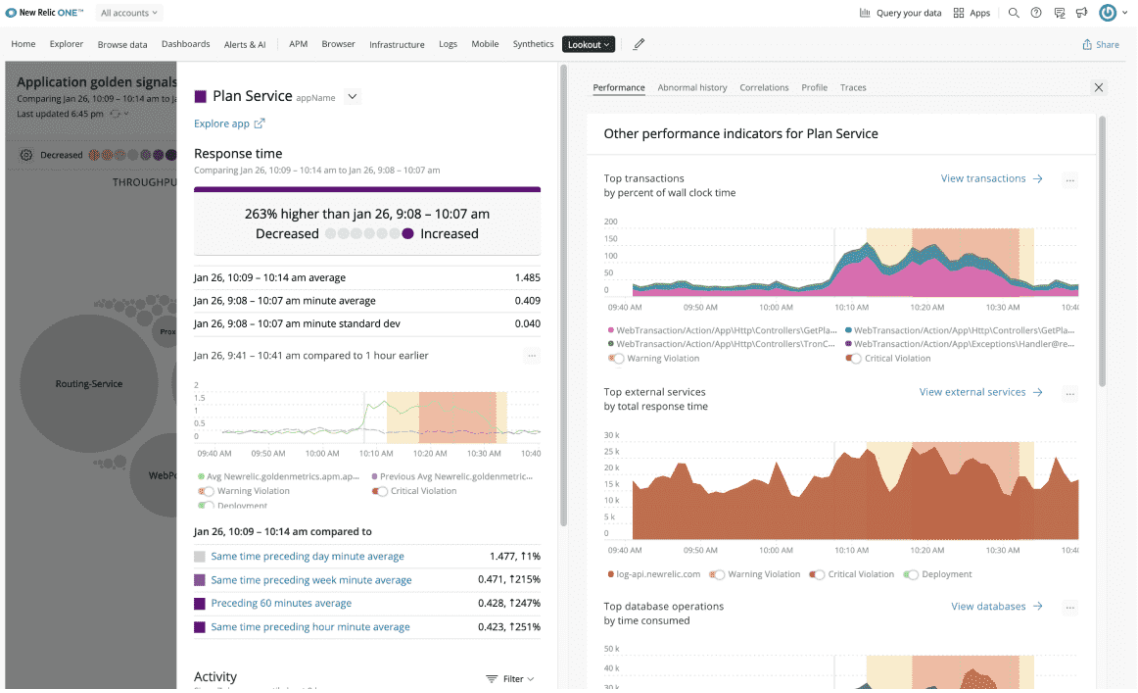
1. Total de Pipelines ejecutados.
2. Total de Pipelines exitosos.
3. Total de Pipelines con falla.
4. Tiempo promedio de despliegue.
5. Frecuencia de despliegue.
6. Tiempo promedio entre cambios.

Herramientas para Monitoreo en Producción

New Relic. Plataforma capaz de monitorear las aplicaciones lanzadas en servidores en la nube. Ofrece una amplia gama de reportes. En la Figura 27 se puede observar un caso de uso de la plataforma.

Figura 27

Reporte de New Relic

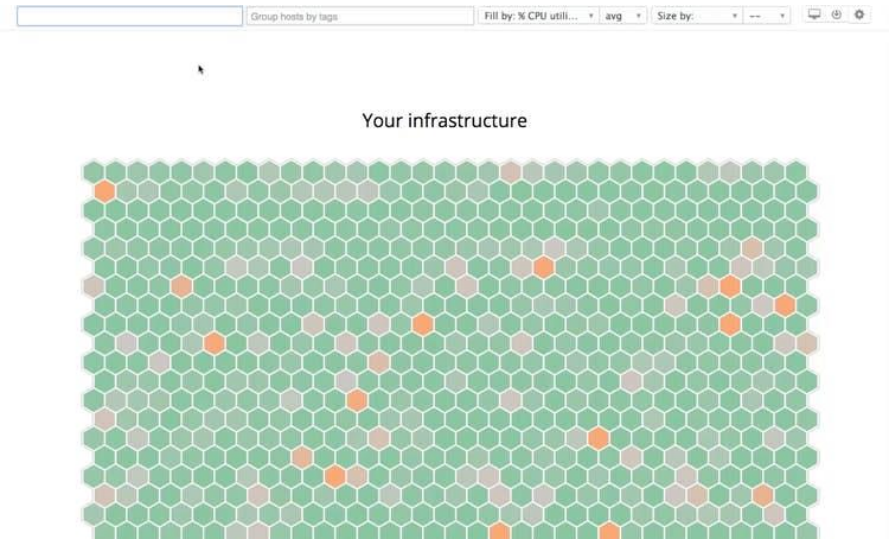


Nota. Captura de pantalla de un reporte en New Relic. Tomado de New Relic https://newrelic.com

Datadog. Se define como un "*Cloud monitoring as a service*" es una aplicación bastante conocida en el mercado. En la Figura 28 se muestra un ejemplo de resumen en una imagen tipo colmena de las veces que se ha desplegado y qué tan exitosas o no han sido estas salidas a producción.

Figura 28

Reporte de Datadog



Nota. Captura de pantalla de un reporte de Datadog. Tomado de Datadog https://www.datadoghq.com

Herramientas de Seguridad en la Nube

Cloud Custodium. Servicio capaz de definir reglas que definan la infraestructura en la nube. Utiliza un archivo con formato YAML para realizar su configuración. Las incidencias encontradas son reportadas junto a las métricas.

Cartography. Se utiliza como una herramienta para probar diferentes escenarios de aplicación de los sistemas desplegados. Es particularmente buena para revelar dependencias que sean un factor de inseguridad para los sistemas en producción.

Retroalimentación Continua

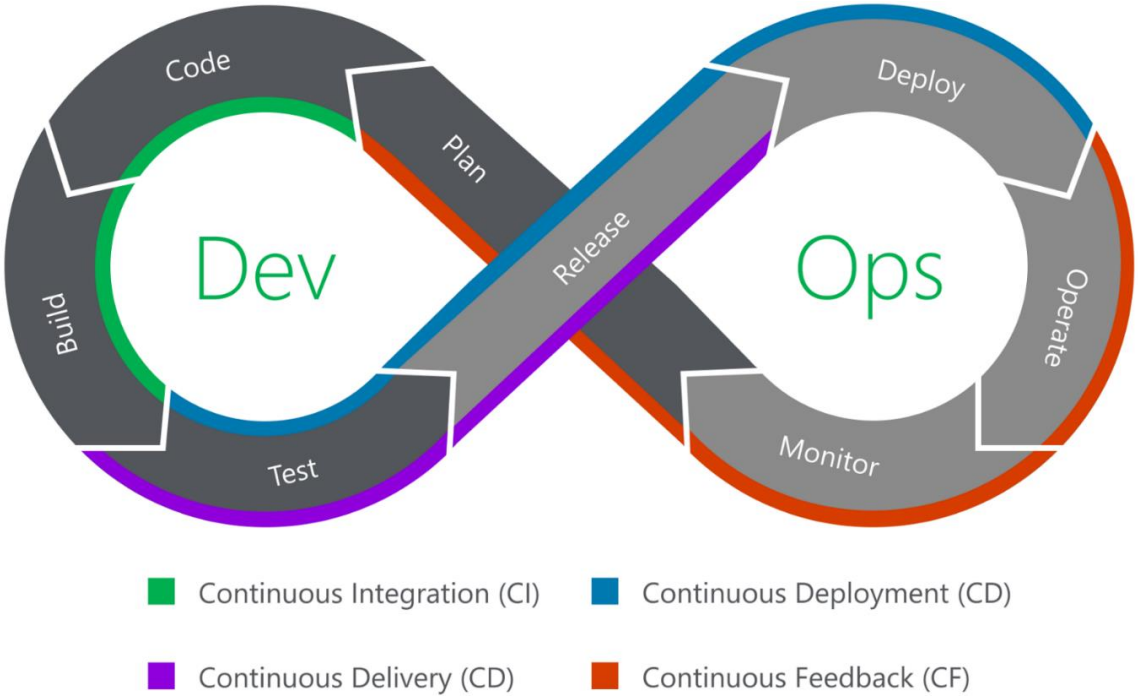
También conocida como Continuous Feedback (CF), es la fase que une los datos obtenidos a través de las operaciones y monitoreo del software junto a la planificación de los nuevos requerimientos y metas de desarrollo. Según el autor (Pennington, 2019), si la

retroalimentación continua se cumple a cabalidad, se implementarán nuevos cambios de forma acelerada gracias a estadísticas y comentarios obtenidos luego de interactuar con stakeholders y usuarios del sistema.

En la *Figura 29* se encuentra representada el ciclo de DevOps antes visto y clasificado con cada una de las etapas ágiles de esta práctica: integración continua (CI), entrega continua (CD), despliegue continuo (CD) y retroalimentación continua (CF). De la misma forma, en la *Tabla XX* se plantea un esquema de resumen de cada una de las etapas del ciclo de DevOps agregando las herramientas disponibles en la web para su uso.

Figura 29

Ciclo integrado de DevOps clasificado por etapas



Nota. Representación del ciclo DevOps y las etapas ágiles en la que las fases del desarrollo de software intervienen. Tomado del autor (Pennington, 2019).

Plantilla Resumen de los Procesos de un Marco de Trabajo DevOps

Tabla 2

Resumen de los procesos de un marco de trabajo DevOps

Fase DevOps	Fase	Objetivo	Actividades	Productos entregables	Herramientas
RC	Planificación	Identificar las tareas a desarrollar durante el ciclo de trabajo.	Redactar las tareas a través de una herramienta de planificación.	Reporte de tareas planificadas para el ciclo de trabajo.	Asana, Trello, Jira, Pivotal Tracker.
IC	Codificación	Crear y versionar el código fuente por cada tarea.	Desarrollar el código fuente y versionar cada cambio a través de una herramienta de versionamiento.	Nuevas versiones del código en el repositorio de control de versiones.	Git, Notepad++, Bitbucket, Maven, Gradle, Npm.
IC	Compilación	Ejecutar el código fuente y crear la versión ejecutable/compilada del software.	Configurar y ejecutar el archivo de compilación del proyecto.	Archivos compilados del software y recepción de errores de compilación.	Lenguajes de programación, Docker, servidores en la nube.
DC **EC**	Pruebas	Proporcionar información objetiva e independiente sobre la calidad del producto	Ejecutar las pruebas estáticas, unitarias, de integración, de rendimiento y aceptación.	Reporte de éxito o fracaso de las pruebas.	JUnit, Selenium, JMeter, Sonarqube.
DC **EC**	Lanzamiento	Preparar el software compilado para ser desplegado en producción.	Se genera un nuevo lanzamiento cuando los hitos del sistema se hayan cumplido.	Versión de lanzamiento del sistema lista para producción.	AWS Codepipeline, Cloudbees Flow.
DC	Despliegue	Publicar la aplicación compilada en un servidor.	La aplicación compilada se copia en el servidor web destinado al funcionamiento del sistema.	Se puede acceder al sistema desplegado a través de la dirección IP del servidor en cuestión.	Ansible, Puppet, Terraform.

Fase DevOps	Fase	Objetivo	Actividades	Productos entregables	Herramientas
RC	Operar	Asegurar el correcto funcionamiento del sistema implementado.	Informar sobre las estadísticas de uso del sistema y reportar novedades.	Retroalimentación del uso de usuarios y stakeholders del sistema.	Kubernetes, Ansible, Chef.
RC	Monitoreo	Reportar el estado de compilaciones, pruebas y despliegues del sistema.	Se visualizan estadísticas sobre despliegues exitosos y no exitosos. Número de errores e incidencias encontradas al compilar.	Gráficas de reportes. Notificaciones de advertencia.	New Relic, Datadog, Instana, Bugsnag, OverOps.
Integración continua (IC)		Mejorar la calidad del software a través de la detección rápida de errores.	Combina el código fuente en un repositorio central de forma periódica.	Creación de nuevas versiones del código fuente llamadas "commit".	Jenkins, Azure DevOps, Circle CI, GitLab CI.
Entrega continua (EC)		Compilar los cambios del código fuente en un servidor.	Compilación automática del software en servidores de pruebas.	Compilación en servidor de pruebas y generación del reporte de la compilación.	Argo, Azure DevOps, GitLab CD, Harness.
Despliegue continuo (DC)		Desplegar el software compilado al servidor de producción sin intervención humana.	Publicación automática de la aplicación en servidores.	Nueva versión o *"release"* del software en producción. Recepción de reporte al desplegar.	Octopus deploy, Spinnaker, Argo y GitLab.
Retroalimentación continua (RC)		Comunicar los comentarios, mejoras y retroalimentación al equipo.	Redactar nuevas incidencias y envío de reportes sobre el uso y monitoreo del sistema.	Informe de nuevos cambios, funcionalidades y errores.	Slack, Discord, Telegram.

Nota. Esta tabla resume las fases del desarrollo de software con la fase DevOps a la que corresponde, detallando sus principales características y herramientas.

Estado Actual del Uso de DevOps en la Industria

Gracias al reporte 2021 del estado del uso de DevOps (Contents, n.d.) se socializan las siguientes afirmaciones sobre DevOps y la industria:

1. El 90% de los encuestados que utilizan ampliamente DevOps han reportado mejoras en como su equipo ejecuta las tareas repetitivas.

2. El 97% de encuestados que utilizan ampliamente DevOps están de acuerdo que la automatización mejora la calidad de su trabajo.

3. El 62% de las organizaciones que utilizan parcialmente DevOps reportan incremento en sus niveles de automatización.

Además, han aparecido interesantes novedades en cuanto al uso del Cloud Computing:

1. Todas las empresas que utilizan servidores en la nube, pero la mayoría los utiliza de una forma poco eficiente. De todos modos, los equipos DevOps lo hacen bien.

2. Las organizaciones no deben considerarse como altamente competitivas solo por usar la nube y automatización.

Capítulo III

Propuesta de Marco de Trabajo

Desafíos en la Gestión de Historias Clínicas en la Institución Patrocinadora

Usuarios

Son seis áreas de salud (medicina general y enfermería, odontología, laboratorio clínico, recaudación, fisioterapia y medicina ocupacional) y una de admisión las que forman parte del Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas. Los usuarios en cuestión corresponden a los siguientes roles:

1. Médico general
2. Trabajador Social
3. Enfermería
4. Encargado de Admisión

Necesidades

Como fruto de reuniones con los usuarios expertos del Sistema Integrado de Salud de la Institución se ha llegado a resumir las principales necesidades que cada área presenta.

En primer lugar, los médicos generales expresan la necesidad de registrar tanto a pacientes internos como externos a la Universidad de las Fuerzas Armadas ESPE, para garantizar una atención médica integral a un público amplio. Además, requieren acceder a los datos de los pacientes internos registrados en el sistema "Banner" de la universidad, con el fin de mantener la información actualizada y evitar duplicidades.

En cuanto a la gestión de perfiles de pacientes externos, los médicos generales desean tener la capacidad de abrir expedientes médicos y gestionar los antecedentes

personales de cada paciente. Esto permitiría obtener información relevante sobre enfermedades previas, así como prevenir enfermedades crónicas o hereditarias. Asimismo, necesitan gestionar datos específicos de la historia clínica, como los signos vitales, hospitalizaciones y estudios complementarios, para realizar diagnósticos precisos.

Otra necesidad identificada es la emisión de certificados médicos para estudiantes y empleados que asistan al consultorio médico, con el propósito de mantener un registro de atenciones y certificados. Además, desean contar con la posibilidad de enviar mensajes vía correo electrónico para evaluar la satisfacción del cliente.

Los usuarios administrativos del sistema web ESPE SALUD tienen la necesidad de gestionar los datos de los usuarios, incluyendo roles y status, con el objetivo de controlar eficazmente el acceso a los registros y evitar vulnerabilidades. Asimismo, desean visualizar la documentación versionada y filtrada por fecha de los cambios realizados en el sistema, así como gestionar los datos de los catálogos para garantizar la escalabilidad del sistema.

En cuanto a los usuarios estadísticos, su necesidad radica en la posibilidad de descargar reportes consolidados de las atenciones médicas, filtrados por año, mes y día, para llevar un control estadístico de los datos. Además, requieren obtener reportes y gráficos estadísticos relacionados con las atenciones médicas registradas en las historias clínicas, pudiendo filtrarlos por dispensarios y por usuario.

Es fundamental que el sistema implementado en la Universidad de las Fuerzas Armadas ESPE cumpla con estándares internacionales reconocidos en el ámbito de la salud. Entre estos estándares se encuentran HL7 (Health Level Seven) e ICD 10 (International Classification of Diseases, 10th Revision).

La adopción de HL7 permitirá establecer una comunicación eficiente y precisa entre los diferentes sistemas de información médica, facilitando la interoperabilidad y el intercambio de datos entre distintos actores y organizaciones de salud. Y, por otro lado, la utilización del estándar ICD 10 en el sistema de gestión de historias clínicas asegurará la codificación uniforme y precisa de los diagnósticos y procedimientos médicos.

Evaluación de la Plataforma Actual

No existe una sola plataforma de gestión de historias clínicas. Existen diferencias significativas entre las sedes de la Institución en términos de sistemas informáticos utilizados. En la Sede Latacunga, se lleva a cabo la gestión de las historias clínicas mediante hojas de cálculo de Excel, lo cual puede resultar limitado en términos de funcionalidad y dificultar la integridad y seguridad de los datos.

Por otro lado, la Sede Matriz cuenta con un software desarrollado hace seis años por la empresa ESPE INNOVATIVA. Durante ese tiempo de uso, es importante realizar una evaluación exhaustiva para determinar su eficacia actual y la capacidad para satisfacer las necesidades actuales de gestión de historias clínicas.

Frente a estas diferencias, el sistema web ESPE SALUD viene a ser el sistema informático que funcionará de forma transversal a través de las sedes de la Institución, garantizando buenas prácticas en su desarrollo a través de un marco de trabajo DevOps que aporte positivamente al desarrollo.

Propuesta del Marco de Trabajo

Nombre

Marco de Trabajo DevOps para el Desarrollo de Software, de aquí en adelante referenciado como MTDDS.

Objetivo

Definir un marco de trabajo basado en DevOps que proponga una guía para el desarrollo, entrega, despliegue y monitoreo continuo del sistema de historias clínicas del Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE, permitiendo una ágil evolución frente a los nuevos requerimientos.

Características

El propósito de la propuesta de marco de trabajo MTDDS, es ser una guía para la correcta adopción de DevOps a la medida de un sistema de historias clínicas. Entre ellos, los puntos del ciclo al que hemos establecido énfasis son cuatro: Integración Continua, Entrega Continua, Despliegue Continuo y Retroalimentación Continua. El autor (Farcic, 2016) asegura que los equipos inician con una integración continua y luego continúan lentamente hacia la entrega y el despliegue.

Entre las características del marco de trabajo están:

Ser una guía de qué se debe hacer, pero no del cómo.

1. Fácil de aplicar y entender.
2. Estar apegado a prácticas modernas de uso de la técnica DevOps.
3. De bajo costo económico para su adopción.

Roles y Responsabilidades

Dentro del MTSSD, se definen roles y responsabilidades específicas para cada fase del ciclo de desarrollo. Estos roles son desempeñados por el Desarrollador de Software y los Administradores del Proyecto, quienes son responsables de llevar a cabo las actividades correspondientes en cada etapa.

Desarrollador de Software. tiene la responsabilidad de realizar la codificación y el desarrollo del software, siguiendo las pautas y estándares establecidos. Además, se encarga de la implementación de pruebas unitarias y la corrección de posibles errores o fallos identificados durante el proceso.

Una vez que el requerimiento ha sido versionado, el desarrollador publica los cambios en el repositorio de código de GitLab donde debe configurar el proyecto para que utilice la funcionalidad de Integración, Entrega y Despliegue Continua de la herramienta. Una vez hecha la configuración se debe ejecutar y publicar el sistema.

Administradores del Proyecto. son los encargados de la gestión y coordinación general del proyecto. Su responsabilidad abarca la planificación, asignación de tareas, seguimiento del progreso y la garantía de que se cumplan los plazos establecidos. También se encargan de recopilar y documentar los requisitos del sistema, así como de generar los entregables correspondientes a cada fase, tales como listas de requerimientos, informes de pruebas, reportes de despliegue y rendimiento, entre otros.

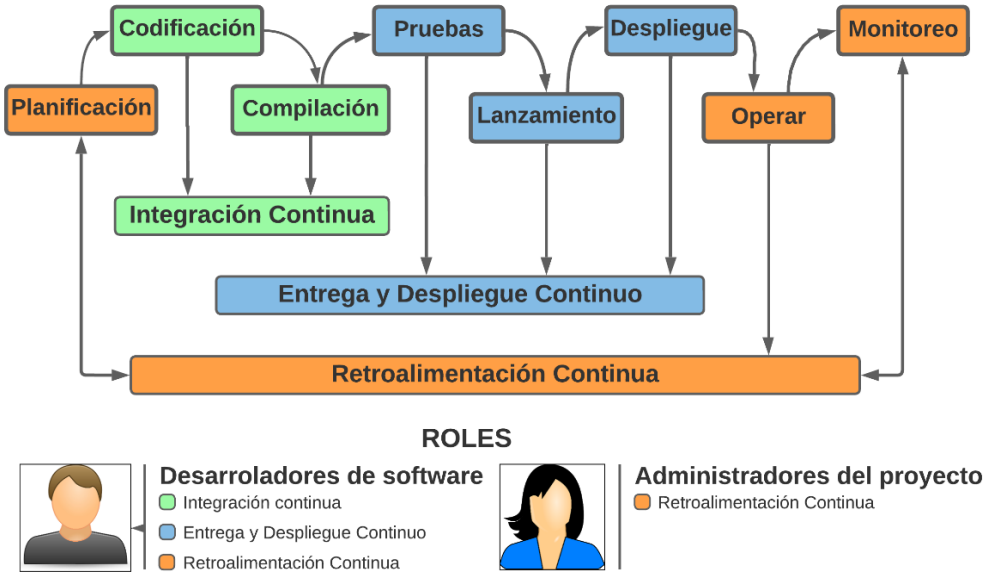
Además, esta persona tiene acceso al repositorio de código en GitLab del proyecto, donde creará nuevos requerimientos utilizando la opción “Pizarra” que se encuentra en el apartado “Issues”.

Flujo de trabajo

MTDDS define fases con un propósito particular en cada una de ellas. La Figura 30 muestra una vista general de las etapas que forman parte del MTDDS, agregando además el rol responsable de cada una de las fases.

Figura 30

Flujo de trabajo del marco de Marco de Trabajo DevOps para el Desarrollo de Software – MTDDS



Nota. Representación del marco de trabajo propuesto. Basado en la investigación de Farías Alejandro (Farías Alejandro, 2017)

A continuación, se describen los elementos que forman parte del marco de trabajo separando cada uno según su categoría DevOps:

Tabla 3

Descripción de los elementos del MTDDS

Integración continua		
Id	**Fase**	**Descripción**
1	Codificación	Consiste en la programación de los requerimientos del sistema y su posterior versionamiento. Se desarrolla el código fuente cumpliendo con la planificación.

Id	Fase	Descripción
2	Compilación	El código fuente es ejecutado y se crea una versión ejecutable del software.
Entrega y despliegue continuo		
Id	**Fase**	**Descripción**
3	Pruebas	Proporciona la información sobre la calidad del código que se escribe, además si la nueva versión del código cumple con las pruebas unitarias programadas. Entre las pruebas que se ejecutan están pruebas estáticas, dinámicas, de integración, rendimiento y aceptación.
4	Lanzamiento	Prepara el software compilado que está listo para ser desplegado en producción. Normalmente el nuevo lanzamiento se realiza cuando se hayan cumplidos los hitos o metas.
5	Despliegue	Publicación de la aplicación en el servidor de desarrollo o producción.
Retroalimentación continua		
Id	**Fase**	**Descripción**
6	Operación	Se asegura el correcto funcionamiento del sistema desplegado en los servidores.
7	Monitoreo	Visualización de los reportes sobre pruebas, compilaciones, despliegues y rendimiento de la aplicación publicada. Es posible identificar caídas a través del monitoreo del sistema.
8	Planificación	Se identificarán las tareas a desarrollar durante el ciclo de trabajo.

Nota. Esta tabla explica en lo que consisten las fases del marco de trabajo propuesto.

Entregables, Indicadores y Herramientas

Cada etapa del MTDDS define entregables, indicadores y herramientas del proceso. En la Tabla 4 se detallan cada uno, según su fase.

Tabla 4

Descripción de los entregables del MTDDS

Integración continua				
Id	**Fase**	**Entregables**	**Herramienta**	**Indicadores**
1	Codificación	Código fuente, documentación técnica.	GitLab.	Líneas de código escritas, tasa de reutilización de código.
2	Compilación	Artefactos compilados en imágenes de Docker.	Docker y GitLab.	Tiempo de compilación, número de errores de compilación.
Entrega y despliegue continuo				
Id	**Fase**	**Entregables**	**Herramienta**	**Indicadores**
3	Pruebas	Casos de prueba, informes de pruebas.	Imagen Docker de JUnit.	Número de defectos encontrados y corregidos.
4	Lanzamiento	Notas de lanzamiento, Pipeline de CI/CD de integración de cambios.	GitLab y Docker.	Tiempo de lanzamiento, cantidad de incidencias en el lanzamiento.

Id	Fase	Entregables	Herramienta	Indicadores
5	Despliegue	Aplicación desplegada en el entorno de producción, documentación de despliegue.	GitLab, Docker, Kubernetes y máquina virtual en AWS.	Tiempo de despliegue, tasa de éxito en el despliegue.
Retroalimentación continua				
Id	**Fase**	**Entregables**	**Herramienta**	**Indicadores**
6	Operación	Notificación de incidencias de mantenimiento y soporte continuo.	Kubernetes y máquina virtual en AWS.	Tiempo de resolución de incidencias, disponibilidad del sistema, satisfacción del usuario.
7	Monitoreo	Informes de monitoreo, registros de rendimiento y uso.	Grafana, Kubernetes y máquina virtual en AWS.	Tiempo de respuesta del sistema, utilización de recursos, cumplimiento de acuerdos de nivel de servicio.
8	Planificación	Plan de proyecto, definición de requerimientos, cronograma de actividades.	Pizarra de GitLab.	Tiempo de planificación, grado de cumplimiento de los requerimientos, desviación en el cronograma.

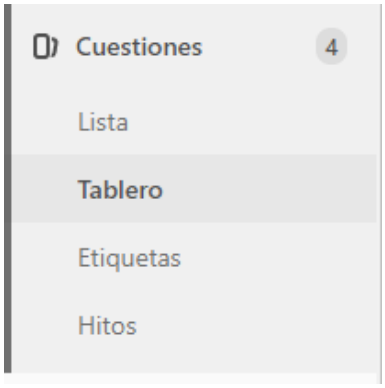
Nota. Esta tabla detalle los entregables e indicadores del marco de trabajo propuesto.

Configuración del Marco de Trabajo

Planificación. Dentro de GitLab, se puede redactar las tareas en la herramienta "Tablero" del apartado "Cuestiones" tal como se muestra en la Figura 31.

Figura 31

Opción de Tablero de GitLab

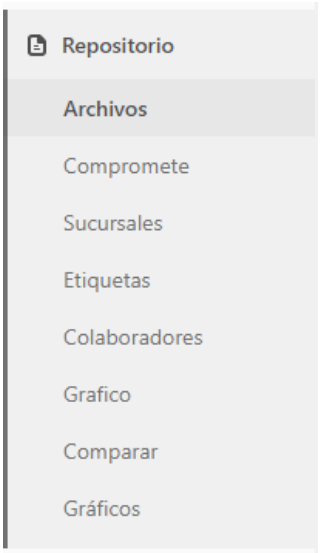


Nota. Captura de pantalla del botón para acceder a la opción de "Tablero" de GitLab. Tomado de GitLab.com

Codificación. Fuera del software que se utilice para programar lo que es de interés para DevOps es el versionamiento de los cambios. Dentro de GitLab, las versiones del código se encuentran dentro de la opción "Archivo" del repositorio del proyecto, así se muestra en la Figura 32.

Figura 32

Opción de Repositorio de GitLab



Nota. Captura de pantalla del botón para acceder a la opción de “Archivos” de GitLab. Tomado de GitLab.com

Compilación. Dentro de GitLab, la compilación se configura dentro de un archivo de instrucciones en formato YAML[4]. El archivo en cuestión se encuentra dentro de los ficheros del proyecto y lleva el nombre de “GitLab-ci.yml” entre las instrucciones que alberga, en la Figura 33 se muestra un ejemplo de configuración.

[4] YAML: Yet Another Markup Language, lenguaje que sirve para serializar datos y es capaz de ser interpretado por humanos.

Figura 33

Instrucción de compilación dentro de un archivo YAML

```yaml
build_app:
  stage: build_and_test
  tags:
    - production
  script:
    - npm run build --prod
  after_script:
    - cp $PROJECT_PATH/data/nginx/nginx.conf $APP_OUTPUT_PATH
    - cp $PROJECT_PATH/Dockerfile $APP_OUTPUT_PATH
  artifacts:
    name: "front-end-app"
    expire_in: 2h
    paths:
      - $APP_OUTPUT_PATH
```

Nota. Captura de pantalla del archivo de configuración para un Pipeline en GitLab. Tomado de GitLab.com

Pruebas. Dentro de GitLab, las pruebas al igual que la compilación, se configuran dentro de un archivo de instrucciones en formato YAML, en la Figura 34 se muestra un ejemplo de configuración de pruebas.

Figura 34

Configuración para la ejecución de pruebas automáticas de un proyecto

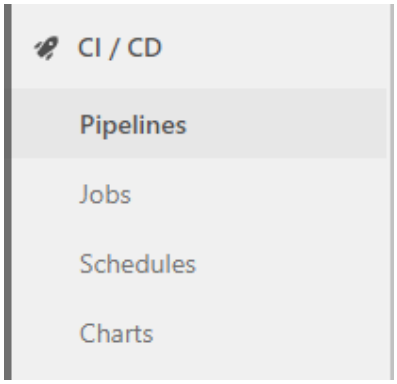
```yaml
# Actually Disabled
test_app:
  stage: build_and_test
  tags:
     - production
  variables:
     CHROME_BIN: google-chrome
  before_script:
     - apt-get update && apt-get install -y apt-transport-https
     - wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add -
     - sh -c 'echo "deb https://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.list.d/google.list'
     - apt-get update && apt-get install -y google-chrome-stable
  script:
     - npm run test:ci
  coverage: '/Statements.*?(\d+(?:\.\d+)?)%/'

  # Avoids all pipeline artifacts to be fetched
  dependencies: []
```

Nota. Captura de pantalla de la configuración de pruebas automáticas en GitLab. Tomado de GitLab.com

Despliegue. Dentro de GitLab, la configuración de despliegue se encuentra dentro de la opción “Pipeline” del menú CI/CD, así se muestra en la Figura 35.

Figura 35

Opción de Repositorio de GitLab



Nota. Captura de pantalla del botón para acceder a la opción de “Pipelines” de GitLab. Tomado de GitLab.com

Monitoreo. Dentro de GitLab, el acceso a las métricas se encuentra dentro de la opción “Métricas” del menú “Operaciones”, así se muestra en la Figura 36.

Figura 36

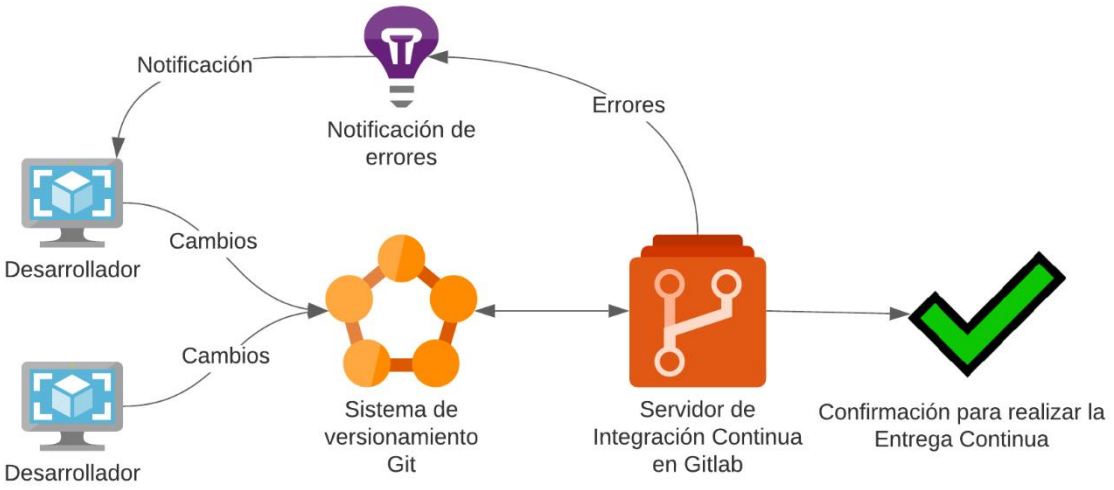
Acceso a las métricas del proyecto

Nota. Captura de pantalla del botón para acceder a la opción de “Métricas” de GitLab. Tomado de GitLab.com

Integración Continua. En la Figura 37, se representa la forma en la que se utiliza la Integración Continua dentro de la propuesta de MTDSS. Se puede interpretar que es un primer paso dentro del proceso de entrega y despliegue continuo del software. Si no existen errores a la hora de versionar e integrar los cambios, entonces GitLab hace una confirmación de que las nuevas modificaciones han sido integradas con éxito.

Figura 37

Representación gráfica de la Integración Continua del marco de trabajo



Nota. Esta figura representa la integración continua del marco de trabajo. Basado en la investigación de Salamon et al. (2019).

Entrega Continua. En el caso del marco de trabajo propuesto, la herramienta GitLab cuenta con un proceso automatizado para configurar la entrega continua dentro de un sistema informático. Estas herramientas se encuentran en el menú “CI/CD” de cualquier proyecto y tiene la capacidad de crear Pipelines a raíz de las ramas o versiones que tenga el proyecto.

Una vez elegida la rama en la que se ejecutará el Pipeline, GitLab analiza la ubicación del archivo de configuración llamado “.GitLab-ci.yml”. Dentro de él se especifican las

versiones, variables de entorno, imágenes de contenedores, sentencias de compilación y de pruebas.

Despliegue continuo. Para el marco de trabajo, al igual que con las dos fases anteriores, se utilizará GitLab para generar el despliegue continuo. GitLab (GitLab, n.d.) recomienda que el diclo de DevOps que ellos proveen se pueda implmenetar en cualquier proyecto.

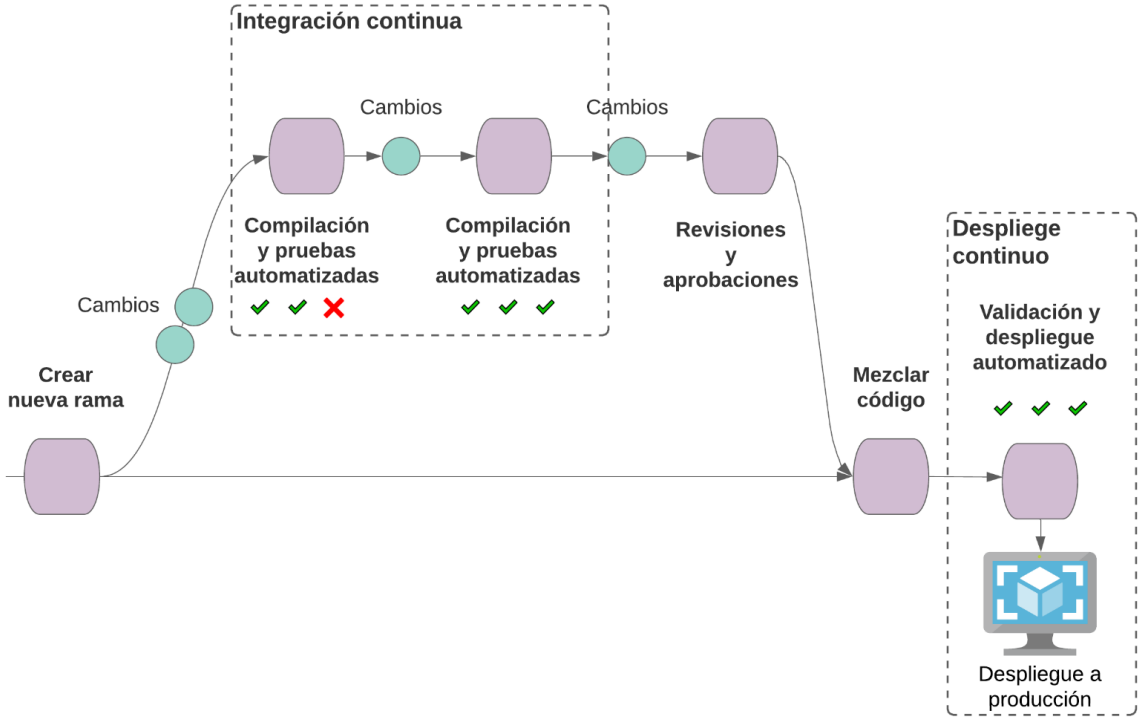
Al enviar una nueva versión hacia GitLab, se enciente un activador (trigger[5]) capaz de ejecutar las instrucciones de compilación y pruebas configuradas previamente. Después, es posible desplegar estos nuevos cambios en los ambientes de producción.

En la Figura 38 se muestra una representación gráfica del Pipeline de despliegue continuo de GitLab.

[5] Triggers: Son acciones que representan una nueva acción para GitLab, entre ella, hacer un “Push” de la nueva versión de código usando Git.

Figura 38

Relación entre la Integración Continua y el Despliegue Continuo en GitLab



Nota. Representación gráfica de un Pipeline desde la codificación hasta el despliegue continuo en GitLab. Basado en GitLab. (GitLab, n.d.)

Retroalimentación Continua. Esta última fase de centra en la comunicación del equipo de trabajo. Existen varias herramientas que aportan a la comunicación de un grupo de personas en internet, empezando por la empresa Facebook y sus servicios de mensajería: Messenger, Instagram y WhatsApp.

El marco de trabajo propone el uso del servicio Discord. Una aplicación multiplataforma de uso gratuito. El autor (Morris, 2020) ha hecho una guía completa de cómo usar la herramienta. En su libro explica que Discord es una tendencia mundial en el mundo de la comunicación VOIP (*Voice over IP*), usado ampliamente por comunidades de video jugadores en interne

Matriz resumen del Marco de Trabajo

Tabla 4

Cuadro de resumen del MTDDS

Fase DevOps	Fase	Descripción	Entregables	Indicadores	Herramientas
RC	**Planificación**	Se identificarán las tareas a desarrollar durante el ciclo de trabajo.	Plan de proyecto, definición de requerimientos, cronograma de actividades.	Tiempo de planificación, grado de cumplimiento de los requerimientos, desviación en el cronograma.	GitLab.
IC	**Codificación**	Se desarrolla el código fuente cumpliendo con la planificación.	Código fuente, documentación técnica.	Líneas de código escritas, tasa de reutilización de código.	GitLab.
IC	**Compilación**	El código fuente es ejecutado y se crea una versión ejecutable del software.	Artefactos compilados en imágenes de Docker.	Tiempo de compilación, número de errores de compilación.	Docker y GitLab.
DC EC	**Pruebas**	Información sobre la calidad del código que se escribe.	Casos de prueba, informes de pruebas.	Número de defectos encontrados y corregidos.	Imagen Docker de JUnit
DC EC	**Lanzamiento**	Preparar el software compilado para ser desplegado en producción.	Notas de lanzamiento, Pipeline de CI/CD de integración de cambios.	Tiempo de lanzamiento, cantidad de incidencias en el lanzamiento.	GitLab y Docker
DC	**Despliegue**	Publicación de la aplicación en el servidor de desarrollo o producción.	Aplicación desplegada en el entorno de producción, documentación de despliegue.	Tiempo de despliegue, tasa de éxito en el despliegue.	GitLab, Docker, Kubernetes y máquina virtual en AWS.

Fase DevOps	Fase	Descripción	Entregables	Indicadores	Herramientas
RC	**Operar**	Asegurar el correcto funcionamiento del sistema implementado.	Notificación de incidencias de mantenimiento y soporte continuo.	Tiempo de resolución de incidencias, disponibilidad del sistema, satisfacción del usuario.	Kubernetes y máquina virtual en AWS.
RC	**Monitoreo**	Visualización de los reportes sobre pruebas, compilaciones, despliegues y rendimiento.	Informes de monitoreo, registros de rendimiento y uso.	Tiempo de respuesta del sistema, utilización de recursos, cumplimiento de acuerdos de nivel de servicio.	Grafana, Kubernetes y máquina virtual en AWS.
Integración continua (IC)		Mejorar la calidad del software a través de la detección rápida de errores	Notificación de compilación exitosa o reporte de fallos. Generación de artefacto en forma de contenedor.	Número de commits enviados a **master** sin incidencias.	GitLab Packages.
Entrega continua (EC)		Compilar los cambios del código fuente en un servidor	Notificación de ejecución de Pipeline exitosa o reporte de fallos.	Reporte de ejecución de pruebas.	GitLab CI.
Despliegue continuo (DC)		Desplegar el software construído al servidor de producción sin intervención humana.	Publicación automática de la aplicación en servidor.	Nueva versión o *"release"* del software en producción.	GitLab CI.
Retroalimentación continua (RC)		Comunicar los comentarios, mejoras y retroalimentación al equipo.	Redactar nuevas incidencias y envío de reportes sobre el uso y monitoreo del sistema.	Informe y solicitudes de nuevos cambios, funcionalidades y errores.	Discord.

Nota. La tabla explica las etapas del marco de trabajo propuesto, así como sus entregables, indicadores y herramientas recomendadas a utilizar.

Capítulo IV

Implementación del Marco de Trabajo en el Sistema de Gestión de Historias Clínicas ESPE SALUD

El siguiente capítulo habla de la implementación del Marco de Trabajo propuesto en el sistema para la gestión de historias clínicas ESPE SALUD (https://salud.espe.edu.ec).

Definición de responsables según su rol

Administrador del proyecto:

- Dr. Bolívar Llamuca, Médico General de la Universidad de las Fuerzas Armadas ESPE Sede Latacunga.

Desarrolladores de software:

- Luis Pillaga
- Diego Maigualca
- Anthony Cabrera

Estudiantes egresados de la carrera de Ingeniería de Software.

Fases de DevOps implementadas

Durante el recorrido por cada una de las fases en las que se implementó DevOps en el sistema ESPE SALUD, se detallará la configuración DevOps en los dos proyectos que dan vida a este software:

1. ESPE SALUD Cliente.
2. ESPE SALUD Servidor.

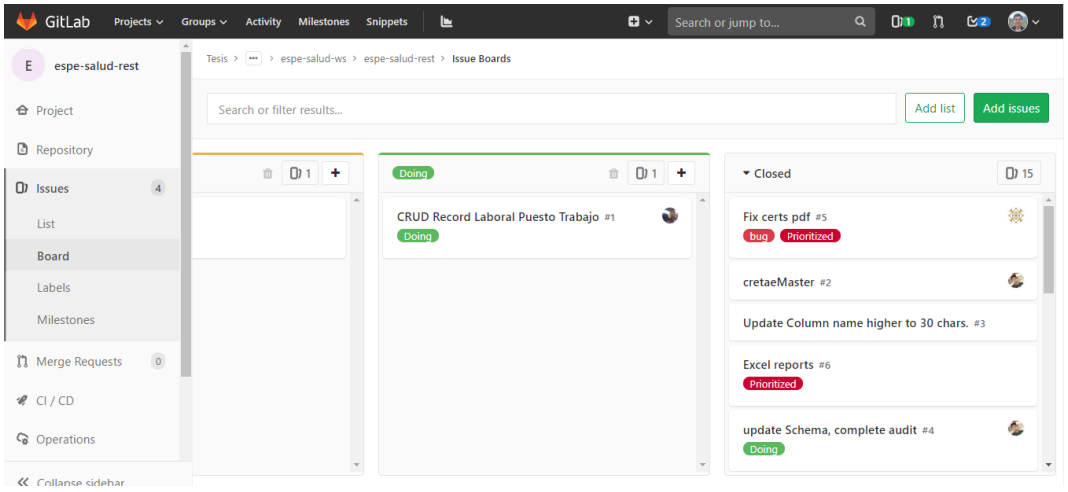
Planificación

Los requerimientos fueron listados como Issues dentro de GitLab. Se utilizó la opción de tablero para diferenciar las tareas según su prioridad y el estado de ejecución de las mismas.

En la Figura 39 se indica el gráfico de la aplicación Cliente y en la Figura 40 el de la aplicación Servidor.

Figura 39

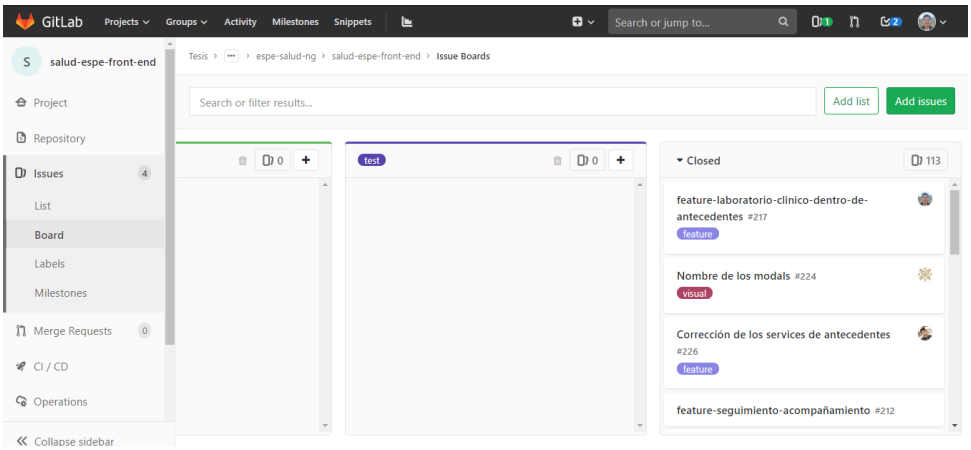
Tablero de tareas del proyecto ESPE SALUD Servidor en GitLab



Nota. Captura del tablero de tareas de los proyectos ESPE SALUD en GitLab.espe.edu.ec.

Figura 40

Tablero de tareas del proyecto ESPE SALUD Cliente en GitLab



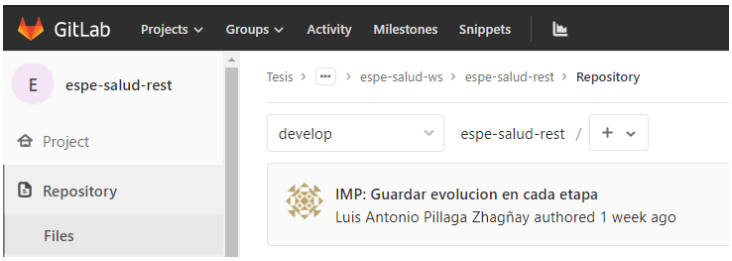
Nota. Captura del tablero de tareas de los proyectos ESPE SALUD en GitLab.espe.edu.ec.

Codificación

Se creó un repositorio de versionamiento para el código del sistema ESPE SALUD en cada uno de sus proyectos. Para la aplicación Servidor se creó un repositorio llamado “espe-salud-rest” (Figura 41) y para la aplicación cliente se creó otro llamado “salud-espe-front-end” (Figura 42).

Figura 41

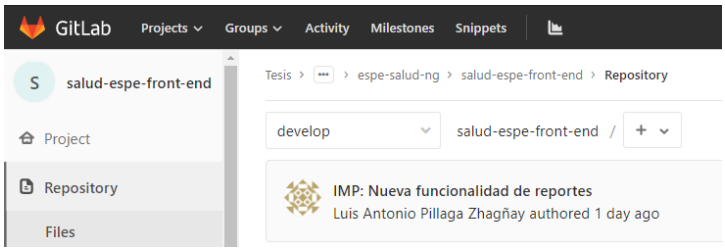
Cabecera del repositorio ESPE SALUD Servidor en GitLab



Nota. Captura de pantalla del proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 42

Cabecera del repositorio ESPE SALUD Cliente en GitLab



Nota. Captura de pantalla del proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

Compilación

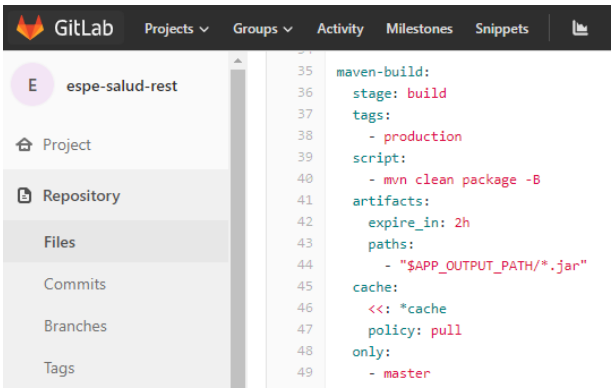
Las tecnologías en las que están desarrollados los proyectos de ESPE SALUD son: Java por el lado del Servidor y JavaScript por el lado del Cliente.

Para la compilación local del proyecto Java, se necesita de un motor de dependencias llamado Maven en el que se especifican cada uno de los módulos del sistema, las versiones de sus dependencias y lenguaje de programación. Con el proyecto JavaScript ocurre lo mismo, en este caso el nombre del archivo es package.json.

Para asegurar que la compilación se realizará de forma automática, fue necesario configurar dos archivos dentro de cada uno de los proyectos. Se trata del archivo ".GitLab-ci.yml". GitLab detecta el archivo y obedece cada una de las instrucciones que se digitaron dentro de él. En la Figura 43 se muestra un extracto de código donde se indica la versión de Maven para el proyecto Cliente, así como en la Figura 44 se muestra la configuración del proyecto JavaScript.

Figura 43

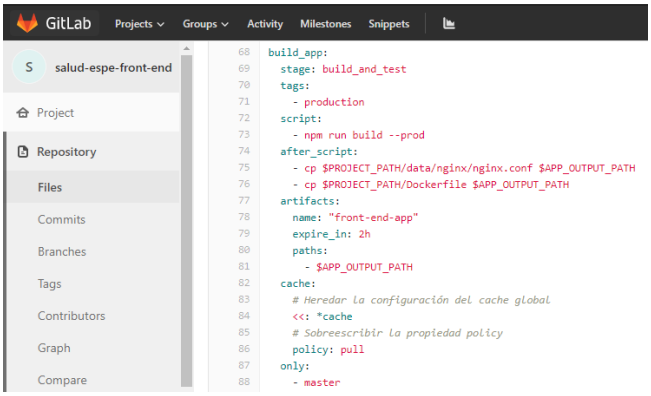
Archivo .GitLab-ci.yml de ESPE SALUD Servidor en GitLab



Nota. Captura de pantalla del archivo de configuración DevOps del proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 44

Archivo .GitLab-ci.yml de ESPE SALUD Cliente en GitLab



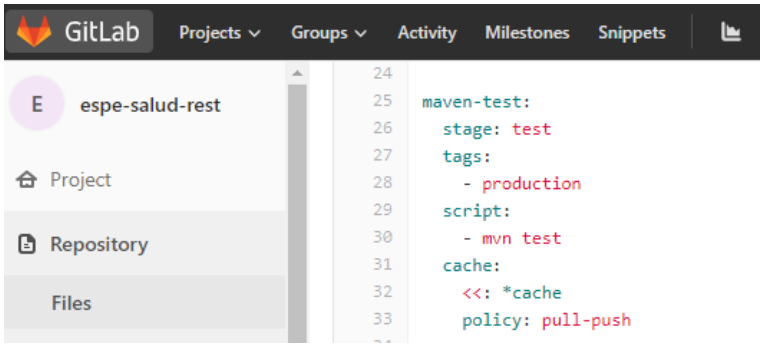
Nota. Captura de pantalla del archivo de configuración DevOps del proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

Pruebas

De la misma forma que en la compilación los archivos de prueba del sistema ESPE SALUD se ejecutan de forma automática con las instrucciones de las Figuras 45 y 46 para Servidor y Cliente respectivamente.

Figura 45.

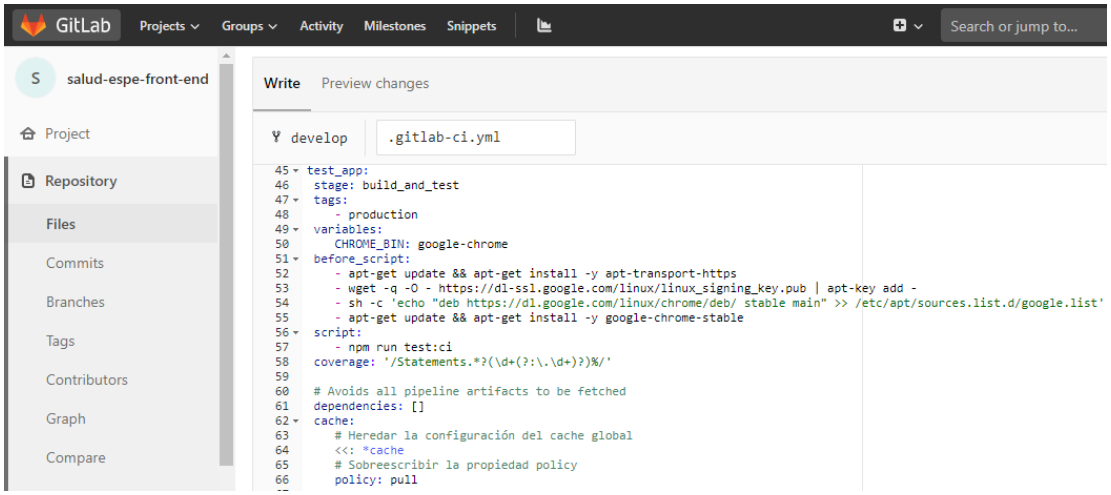
Configuración para la ejecución de pruebas en el archivo .GitLab-ci.yml de ESPE SALUD Servidor en GitLab



Nota. Captura de pantalla de la configuración de una prueba en el proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 46.

Configuración para la ejecución de pruebas en el archivo .GitLab-ci.yml de ESPE SALUD Cliente en GitLab



Nota. Captura de pantalla de la configuración de una prueba en el proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

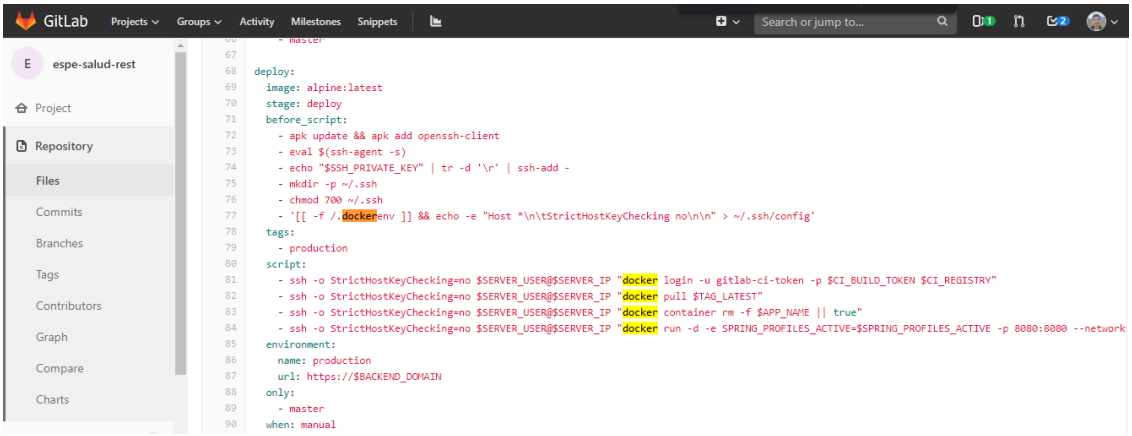
Despliegue

Para desplegar ambas soluciones de forma local es necesario ejecutar una sentencia “build” y “run” en ambos lados.

Para automatizar este proceso, se definieron instrucciones de “deploy” en el archivo “.GitLab-ci.yml”. En la Figura 47 y 48 se muestran las instrucciones definidas para las aplicaciones Servidor y Cliente respectivamente.

Figura 47

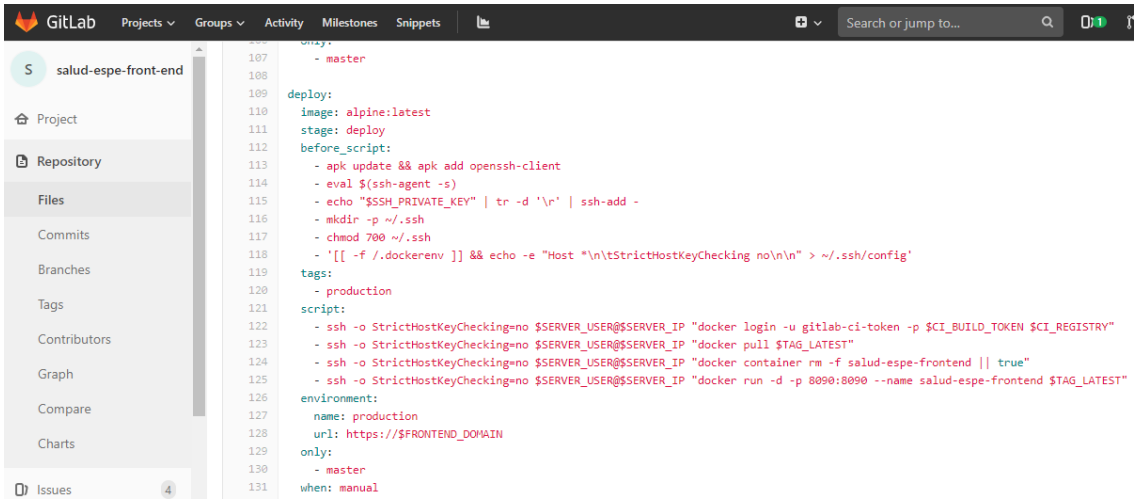
Configuración para el despliegue automático en el archivo .GitLab-ci.yml de ESPE SALUD Servidor en GitLab



Nota. Captura de pantalla de la configuración para el despliegue automático del proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec

Figura 48

Configuración para el despliegue automático en el archivo .GitLab-ci.yml de ESPE SALUD Cliente en GitLab



Nota. Captura de pantalla de la configuración para el despliegue automático del proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

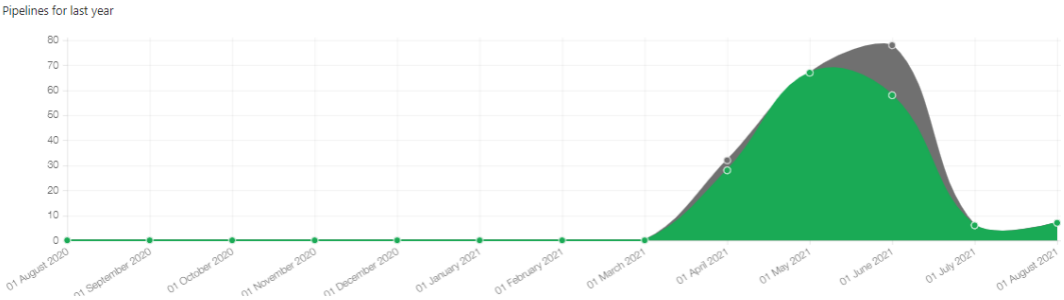
Monitoreo

El monitoreo realizado sobre el sistema ESPE SALUD consiste de las métricas en las que los Pipelines de ejecución se han realizado de forma exitosa y los que por alguna razón han fallado al ejecutarse.

En las Figuras 49 y 50 se muestran de color verde los Pipelines que no presentaron problemas y se convirtieron en un nuevo release de las aplicaciones. En color gris está la totalidad de Pipelines. Las gráficas son de la aplicación Servidor y de la aplicación Cliente respectivamente.

Figura 49

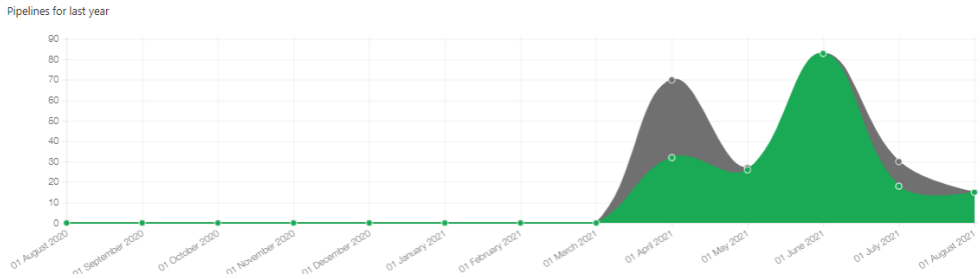
Monitoreo de Pipelines de ESPE SALUD Servidor en GitLab



Nota. Los Pipelines exitosos son los coloreados con verde. Los grises son los que fallaron al ejecutarse en el proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 50

Monitoreo de Pipelines de ESPE SALUD Servidor en GitLab



Nota. Los Pipelines exitosos son los coloreados con verde. Los grises son los que fallaron al ejecutarse en el proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

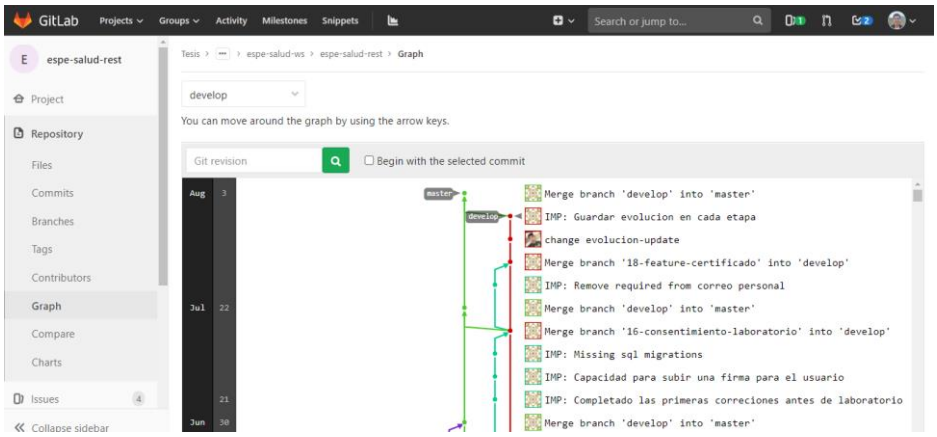
Como se nota en ambos gráficos, la implementación de DevOps empezó desde marzo de 2021, mientras que antes de esa fecha no existían esfuerzos por implementar despliegue continuo dentro del proyecto.

Integración Continua

La herramienta de control de versiones Git ha sido suficiente para llevar un correcto control de los cambios y la integración de esos avances entre los desarrolladores. En evidencia de esto se encuentra la Figura 51 de la aplicación Servidor y la Figura 52 de la aplicación Cliente que indican los flujos y dirección de las integraciones del código hasta el 10 de agosto de 2021.

Figura 51

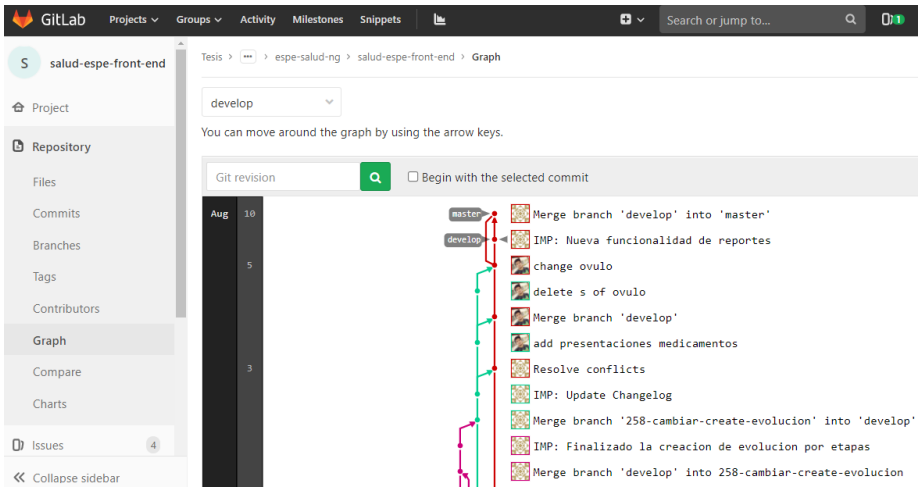
Monitoreo de la integración continua de ESPE SALUD Servidor en GitLab



Nota. Captura de pantalla de la integración continua de los cambios del proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 52

Monitoreo de la integración continua de ESPE SALUD Cliente en GitLab



Nota. Captura de pantalla de la integración continua de los cambios del proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

En ambos casos, la integración converge encima de dos ramas de Git: develop y master. Esto es una demostración de las convenciones a la hora de programar que se han utilizado. La última versión de las aplicaciones se encuentra sobre la rama master.

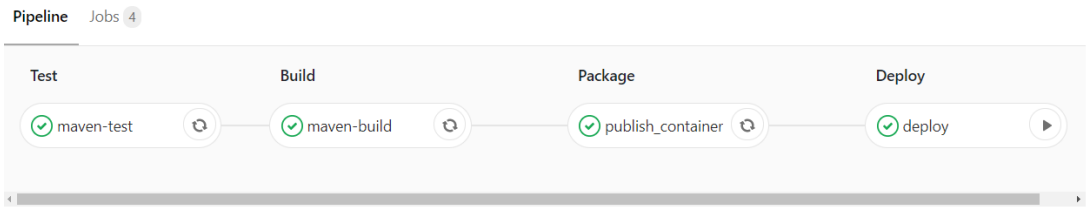
Entrega continua

La entrega continua representa al resultado de la ejecución de las configuraciones para construir y probar el sistema ESPE SALUD. Tal como lo señalamos en los puntos de Compilación, Pruebas y Despliegue.

Para ejecutar estas instrucciones una después de otra, se genera un Pipeline a través del menú “CI/CD” de GitLab. Al entrar hace falta elegir una de las ramas de Git en el proyecto para ejecutar su archivo de instrucciones. En la Figura 53 se muestran los procesos que se ejecutaron de manera exitosa en unos de los Pipelines en la aplicación Servidor, así mismo la Figura 54 muestra un proceso similar en la aplicación Cliente.

Figura 53

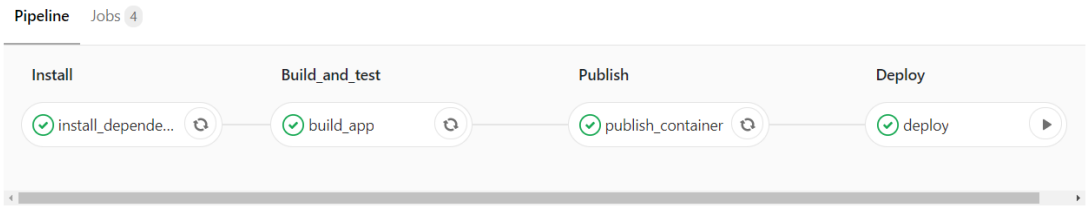
Monitoreo de la entrega continua de ESPE SALUD Servidor en GitLab



Nota. Captura de pantalla de las etapas para la entrega continua en el Pipeline del proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 54

Monitoreo de la entrega continua de ESPE SALUD Cliente en GitLab



Nota. Captura de pantalla de las etapas para la entrega continua en el Pipeline del proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

Es necesario reiterar que los Pipeline ejecutan las instrucciones definidas en el archivo ".GitLab-ci.yml". Si las instrucciones no han encontrados errores en la compilación de los proyectos, entonces se marca con un visto y se las etiqueta como "pasadas" (passed).

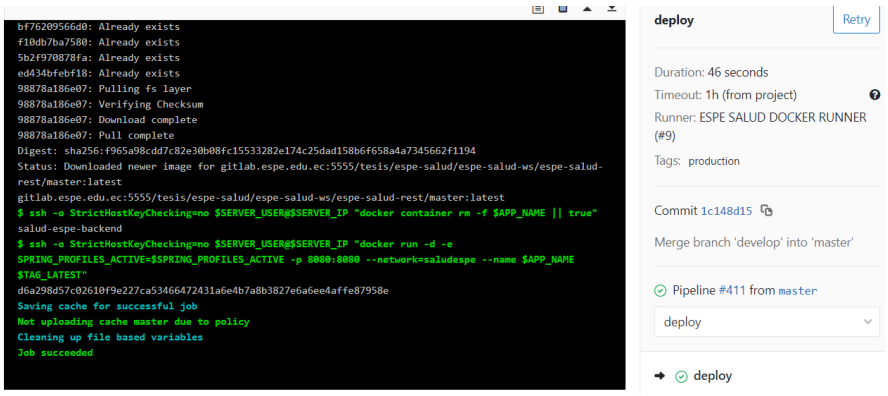
Despliegue continuo

El despliegue continuo ocurre cuando la entrega continua termina y las aplicaciones están listas para ser publicadas. Se obtiene los archivos compilados de ambos proyectos y se los libera en el servidor de producción en el que se está trabajando.

En el caso de ESPE SALUD se ha utilizada servidores Linux con sistema operativo CentOS 8. En la Figura 55 se muestra una evidencia de las instrucciones que se ejecutaron para liberar el nuevo release del proyecto Servidor, así mismo con el Figura 56 el release de la aplicación Cliente.

Figura 55

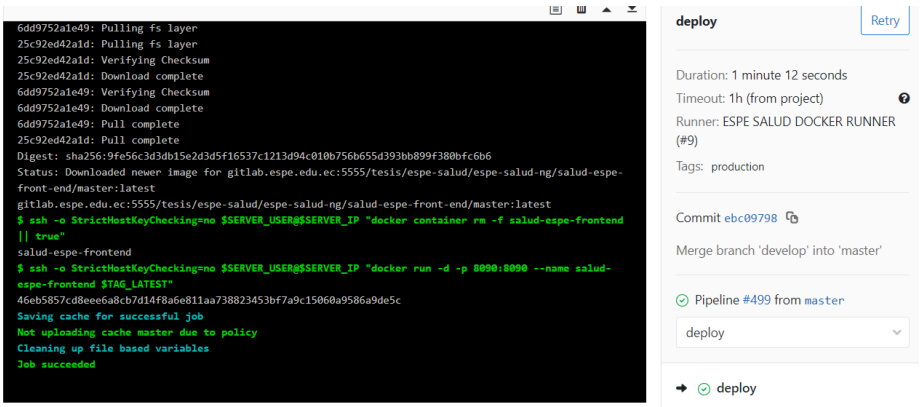
Logs del proceso de despliegue continuo de ESPE SALUD Servidor en GitLab



Nota. Captura de la finalización del despliegue continuo en el proyecto Servidor ESPE SALUD en GitLab.espe.edu.ec.

Figura 56

Logs del proceso de despliegue continuo de ESPE SALUD Cliente en GitLab



Nota. Captura de la finalización del despliegue continuo en el proyecto Cliente ESPE SALUD en GitLab.espe.edu.ec.

Las instrucciones que se visualizan son bastante parecidas ya que las tecnologías y dependencias se encuentran definidas en contenedores usando Docker. Se han creado contenedores basados en Maven y Node.js[6] respectivamente para cada una de las aplicaciones.

De esta forma se personalizar la manera en la que el programador prefiere que su aplicación se ejecute. Controlando cada uno de los pasos y creando aplicaciones que se distribuyen a través de contenedores independientes al servidor o sistema operativo donde se aloje, su comportamiento será el mismo gracias a Docker.

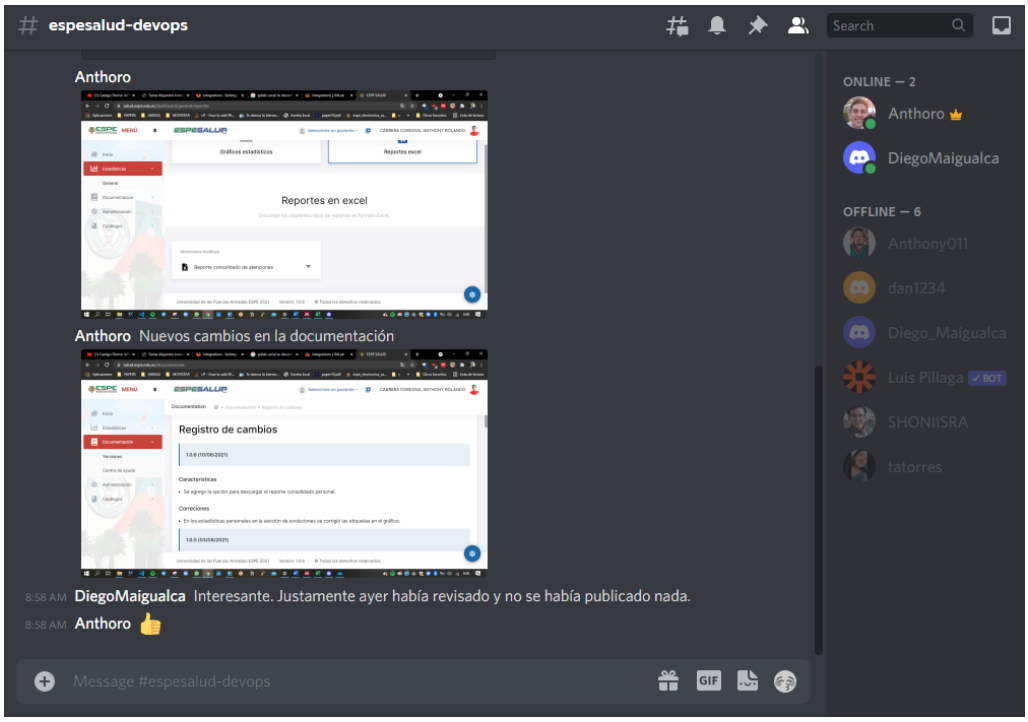
Retroalimentación continua

El proceso de comunicación del equipo se desarrolló a través de la herramienta Discord tal como la propuesta recomienda. Discord permitió enviar mensajes, tener reuniones de videollamada y compartir cualquier tipo de archivo de manera rápida y bastante amigable gracias a su interfaz de usuario *(UI).* En la Figura 57 se evidencia una de las conversaciones del equipo de trabajo.

[6] Node.js: Servidor que permite la ejecución de aplicaciones JavaScript.

Figura 57

Captura de conversaciones del equipo de trabajo en Discord



Nota. Captura del canal de comunicación del equipo de desarrollo en Discord.

Evidencia del despliegue

Una vez realizada el despliegue el sistema se encuentra disponible a través de la URL https://salud.espe.edu.ec/.

En la Figura 58 se evidencia la página de inicio del sistema ESPE SALUD. De la misma forma en la Figura 59 se encuentra la funcionalidad de Usuarios y finalmente la Figura 60 está la funcionalidad de Laboratorio Clínico de un paciente. Esto como vista preliminar del sistema ESPE SALUD.

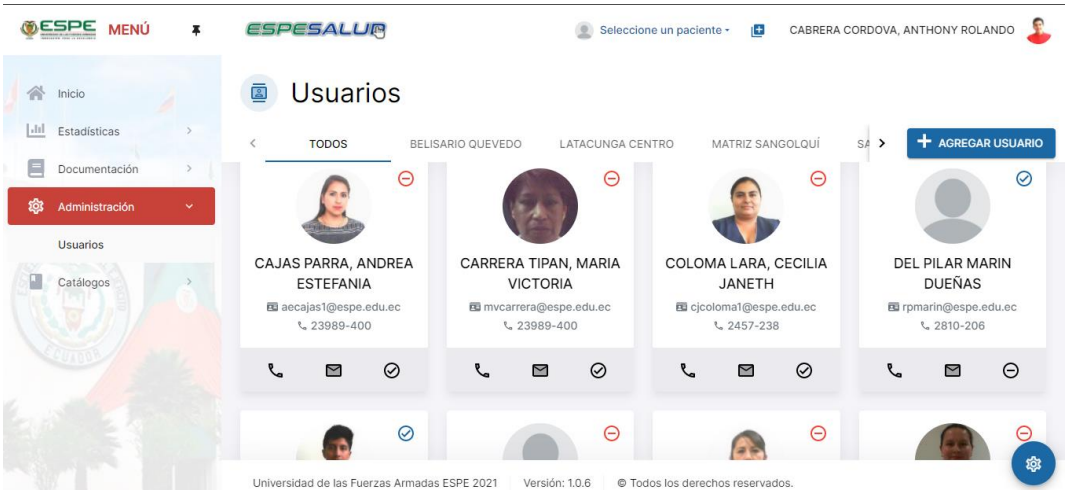
Figura 58

Pantalla de inicio de ESPE SALUD

Nota. Captura de la pantalla principal del proyecto ESPE SALUD.

Figura 59

Módulo de Usuarios de ESPE SALUD



Nota. Captura de pantalla del módulo de usuarios del proyecto ESPE SALUD.

Figura 60

Módulo de Laboratorio Clínico de ESPE SALUD

Nota. Captura de pantalla del módulo de laboratorio clínico del proyecto ESPE SALUD.

Capítulo V

Evaluación del Marco de Trabajo

Caracterización de Evaluadores

La población que participará son personas involucradas en el desarrollo de software, las cuales conocen del Sistema ESPE SALUD ya que participaron activamente de su desarrollo.

Propuesta

Está orientada en la hipótesis planteada y en las variables dependientes e independientes, identificadas, las mismas que se describen en la Tabla 5.

Tabla 5

Cuadro de resumen de la hipótesis de investigación

Hipótesis	Variables	Atributos	Valor
Si se define un marco de trabajo basado en DevOps entonces se optimiza el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.	(Dependiente) Definición de un marco de trabajo basado en DevOps.	Aceptación de la funcionalidad.	Aceptación de al menos 70%.
	(Independiente) Se optimiza el desarrollo de software entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el departamento médico de la Universidad de las Fuerzas Armadas ESPE.		

Nota. Tabla de resumen de la hipótesis y resultado de su aceptación.

Artefactos de Evaluación

Toda la propuesta va a ser evaluada mediante el análisis del atributo de aceptación.

Para la evaluación de la aceptación las personas que prueben el marco de trabajo deben implementar la propuesta en el sistema ESPE SALUD, respetando cada uno de las etapas y procesos indicados en el capítulo 4.

Finalmente se interpretará como el marco de trabajo aporta positivamente al desarrollo de software a través de encuestas de evaluación.

Procedimiento

Primero, el grupo de desarrolladores y administrador del equipo se comprometen a utilizar el marco de trabajo durante el desarrollo del sistema ESPE SALUD.

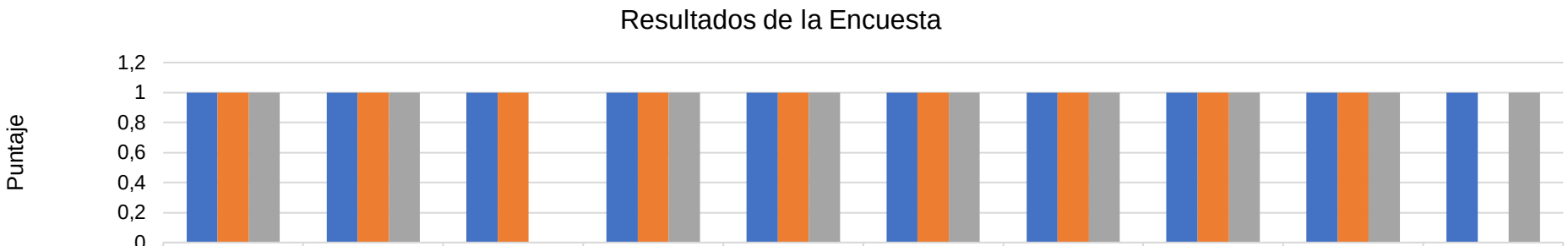
Después, cuando ya se haya utilizado el marco de trabajo por al menos 90 días, se efectuó una reunión de trabajo y finalmente todos deben responder la encuesta formulada. La encuesta en cuestión tiene las preguntas que se les hizo a los programadores al inicio de la investigación, son cuestionamientos que admiten dos respuestas: sí (1) y no (0).

Resultados

Como resultados de las encuestas contestadas por los desarrolladores, que se observan en el Anexo 01, se tienen los resultados de la Figura 37. Las preguntas realizadas fueron las mismas que se hicieron en un inicio en el Capítulo I en la fase de Antecedentes sobre el estado actual del sistema ESPE SALUD.

Tabla 6

Tabla de resultados a partir de la encuesta de resultados



	¿Se está utilizando herramientas para la administración y control del proyecto?	¿Los nuevos requerimientos del sistema son explicados y debidamente documentados?	¿Existe una convención para nombrar a las versiones sobre las que trabaja?	¿Los cambios al código del sistema están siendo versionados?	¿Existe un mecanismo para aprobar o rechazar cambios?	¿El tiempo y los logs al generar una nueva versión del sistema (release) está siendo monitoreado?	¿Existe un mecanismo para solicitar y definir responsables para cambios en el código?	¿La publicación de los cambios en el servidor de pruebas/producción se realiza de forma automática?	¿Puede usted recuperar cualquier versión anterior del sistema?	¿Los nuevos programadores que se unen al proyecto son capaces de configurar su entorno de trabajo local de forma ágil?
Encuestado 1	1	1	1	1	1	1	1	1	1	1
Encuestado 2	1	1	1	1	1	1	1	1	1	0
Encuestado 3	1	1	0	1	1	1	1	1	1	1

Nota. La tabla detalla los resultados de la encuesta de evaluación posterior a la implementación del marco de trabajo. Los 1 quieren decir “sí”, los 0 significan “no”.

Después del análisis de las encuetas aplicadas se tiene que el grado de aceptación de funcionalidad por parte de los participantes es del 96,6%.

Como resultado se tiene que la propuesta es aceptada, la aplicación del marco de trabajo optimiza el desarrollo del proyecto “ESPE SALUD” el cual se encuentra en este momento desplegado en la infraestructura de la institución: http://salud.espe.edu.ec/.

Viabilidad

A raíz de los resultados se infiere: que el marco de trabajo optimiza el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE. Haciendo que los procesos estén documentados, los requerimientos se atiendan ágilmente y la integración, entrega y despliegue se realicen de forma automática.

Capítulo VI:

Conclusiones y Recomendaciones

Conclusiones

1. Se definió un marco de trabajo basado en DevOps, que optimizó el desarrollo, entrega, despliegue y monitoreo continuo de software para la gestión de historias clínicas en el Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

2. Se realizó el estudio bibliográfico sobre marcos de trabajo de desarrollo de software y DevOps, con el fin de conocer el estado de la literatura en estas ramas de la Ingeniería de Software.

3. Se desarrolló un marco de trabajo basado en DevOps que sea una propuesta transversal y aplicable.

4. Se implementó el marco de trabajo en el desarrollo del sistema para la gestión de historias clínicas del Sistema Integrado de Salud de la Universidad de las Fuerzas Armadas ESPE.

5. Se validó el marco de trabajo implementado, a través de los indicadores de la variable dependiente realizando entrevistas individuales al equipo de desarrollo del proyecto para la gestión de historias clínicas.

6. La implementación de DevOps garantizó que los procesos, ambientes de desarrollo y servidores estén configurados para generar nuevas versiones de producción en el menor tiempo.

7. El marco de trabajo propuesto, permite a las personas e investigadores el empezar con la implementación de soluciones DevOps de una forma interactiva ya que la propuesta hace una introducción de las herramientas disponibles y la forma de iniciar con el uso de Gitlab.

Recomendaciones

1. Aprovechar las bondades que aporta DevOps para continuar con la optimización del desarrollo de software de gestión de historias clínicas, así como el desarrollo de todo tipo de sistemas de información.

2. Formalizar los flujos de trabajo que utilizamos cada día. Estos pueden convertirse en parte de la literatura disponible y serán de gran utilidad para el resto de investigadores.

3. Utilizar las nuevas tecnologías de DevOps para generar soluciones a distintos problemas que se tienen día tras día con la implementación y publicación de sistemas.

4. Implementar soluciones nuevas a los sistemas de instituciones como la Universidad de las Fuerzas Armadas ESPE. Además de ser un ejemplo de trabajo de titulación es una demostración de la transferencia de tecnología que ocurre dentro de las aulas.

5. Adoptar una cultura DevOps dentro de una organización implica un cambio a nivel comunicacional de y de la organización en general, por este motivo es necesario brindar la capacitación necesaria a los miembros del equipo de trabajo.

6. Migrar hacia un enfoque DevOps implica establecer un solo mecanismo para programar, probar y publicar de forma continua. Las anteriores prácticas al uso de DevOps se deben olvidar.

Bibliografía

Amazon. (n.d.-a). *AWS CodeDeploy | Implementación de software automatizada*. Retrieved August 18, 2021, from https://aws.amazon.com/es/codedeploy/

Amazon. (n.d.-b). *AWS Lambda - Gestión de recursos informáticos*. Retrieved August 18, 2021, from https://aws.amazon.com/es/lambda/

Aparicio Britto, K. B., & Choy Balboa, A. L. (2020). Aplicación del ciclo PDCA para mejorar la calidad del servicio de atención al usuario de Comdata Group, 2018-2019. *Repositorio Institucional - UTP*.

Banchoff, D., & Fernandez, M. (2019). *Infraestructura como Código Caso de estudio: Cientópolis*.

Bertolino, A., Angelis, G. De, Guerriero, A., Miranda, B., Pietrantuono, R., & Russo, S. (2020). DevOpRET: Continuous reliability testing in DevOps. *Journal of Software: Evolution and Process*, e2298. https://doi.org/10.1002/SMR.2298

Castro Sánchez, J. E. (2020). *DevSecOps: Implementación de seguridad en DevOps a través de herramientas open source*.

Combe, T., Martin, A., & Di Pietro, R. (2016). To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing*, *3*(5), 54–62. https://doi.org/10.1109/MCC.2016.100

Contents. (n.d.). *State of DevOps Report 2021*.

Digital.ai. (2020). *Periodic Table of DevOps Tools*. https://digital.ai/periodic-table-of-devops-tools

Dompablo Tobar, J., Ángel, M., & Rincón, M. (2018). *DevOps para automatización de Gitlab en alta disponibilidad*.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94–100. https://doi.org/10.1109/MS.2016.68

El Universo. (2020, March 17). *Por coronavirus, universidades de Ecuador suspenden clases presenciales, que pasan a ser virtuales*.

Farcic, Viktor. (2016). The DevOps 2.0 toolkit : automating the continuous deployment pipeline with containerized microservices. *CreateSpace Independent Publishing Platform*.

Farías Alejandro, I. K. (2017). *Definición de un ambiente de construcción de aplicaciones empresariales a través de Devops, Microservicios y Contenedores*.

Fernandez Prezl, Y., Lucia, R., Delgado2, C., Lucy, C., Aguila2, C., Gonzlez Jorrfn3, M., Molina3, A., & Hernandez, V. (2007). Pruebas de aceptaci6n para un software con la presencia de una entidad certificadora de la calidad Testing of the client's acceptance with the participation of a company guaranteeing the quality. *AGOSTO*, *1*(3), 84–95.

Galindo Haro, J. M. (2010). Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE Resumen. In *Universitat Oberta de Catalunya*.

Garayar Velásquez, D. (2018). Desarrollo de una aplicación web basado en la metodología scrum - devops para la gestión de contratación del servicio docente

de la UGEL Andahuaylas - región Apurímac 2018. *Universidad Nacional José María Arguedas*.

GitLab. (n.d.). *CI/CD concepts*. Retrieved August 11, 2021, from https://docs.gitlab.com/ee/ci/introduction/

Google Cloud. (n.d.). *Cloud Functions*. Retrieved August 18, 2021, from https://cloud.google.com/functions

Gracia, R., Consultor, P., Cristina, A., & Troncho, D. (2016). *Gestión de proyectos ágiles*.

Hammad, M., & Inayat, I. (2019). Integrating risk management in scrum framework. *Proceedings - 2018 International Conference on Frontiers of Information Technology, FIT 2018*, 158–163. https://doi.org/10.1109/FIT.2018.00035

Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020). From Agile to DevOps: Smart Skills and Collaborations. *Information Systems Frontiers*, *22*(4), 927–945. https://doi.org/10.1007/s10796-019-09905-1

Hüttermann, M. (2012). *DevOps for Developers - - Google Libros*. Apress. https://books.google.com.ec/books?hl=es&lr=&id=JfUAkB8AA7EC&oi=fnd&pg=PR3&dq=devops&ots=wonnc5ytlP&sig=iNqmOTLX4rErYT5EWXRQ666RdnA&redir_esc=y#v=onepage&q=devops&f=false

IBM. (n.d.). *UrbanCode*. Retrieved August 18, 2021, from https://www.ibm.com/cloud/urbancode

Iñiguez Sánchez, L. A. (2017). *Arquitectura tecnológica para la entrega continua de software con despliegue en contenedores*.

Jacob, D. (2016, October 5). *What is The Difference Between Methodology and Framework?* LinkedIn. https://www.linkedin.com/pulse/what-difference-between-methodology-framework-dafir-jacob/

José H. Canós, M. C. P. P. L. (2012). *Métodologías Ágiles en el Desarrollo de Software*.

Landázuri, C. J., & Estrella, E. (2019). *Diseño de un modelo de gobernabilidad y gestión de TI para el área de desarrollo de proyectos de software de corporación favorita, basado en la metodología DevOps*.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of devOps. *Lecture Notes in Business Information Processing*, *212*, 212–217. https://doi.org/10.1007/978-3-319-18612-2_19

Melnik, V. (2019). *AGILE-MANAGEMENT 3.0 CONCEPT AS A FACTOR OF TECHNOLOGICAL PROGRESS DEVELOPMENT IN THE DIGITAL SOCIETY*.

Merkel, D. (2014). *Docker: Lightweight Linux Containers for Consistent Development and Deployment*.

Molina, B., Vite, H., & Dávila, J. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales Revista Multidisciplinaria de Investigación*, *2*(17). http://revistaespirales.com/index.php/es/article/view/269

Morris, T. (2020). *Discord for Dummies*. John Wiley & Sons.

Nogués García, J. (2018). *Orquestación de contenedores con Kubernetes*.

O'Grady, A. (2018, November 30). *GitLab Quick Start Guide: Migrate to GitLab for all your repository*. Packt Publishing Ltd.

https://books.google.com.ec/books?hl=es&lr=&id=jC59DwAAQBAJ&oi=fnd&pg=PP1&dq=gitlab&ots=9RztpRwwPF&sig=7Datslk1iw-TeRQsX4qYuW7yk9k&redir_esc=y#v=onepage&q=gitlab&f=false

Pardo Matos, J. M., & Febles Estrada, A. (2014). *Proceso de Pruebas de Aceptación de Software*.

Pennington, J. (2019, July 18). *The Eight Phases of a DevOps Pipeline | by JakobTheDev | Taptu | Medium*. https://medium.com/taptuit/the-eight-phases-of-a-devops-pipeline-fda53ec9bba

Pressman, R. S. (2010). *Ingeniería del software : un enfoque práctico* (7ma ed.). McGraw-Hill.

Quintero Parra, C. M., & Daza Benjumea, M. J. (2017). *Desarrollo de software bajo paradigma DevOps: Prototipo "PlaceHealth."*

Salamon, A., Maller, P., Boggio, A., Mira, N., Perez, S., & Coenda, F. (2019). *La Integración Continua Aplicada en el Desarrollo de Software en el Ámbito Científico-Técnico*.

Selenium. (n.d.). Retrieved August 30, 2021, from https://www.selenium.dev/

Silvestre, L. A. C., Bravo, M., M, G., & J, V. (2019). *Propuesta de una Aplicación Web para la administración y manejo del historial clínico electrónico (HCE) en el sector salud, utilizando el estándar HL7 para la interoperabilidad.*

Sommerville, I. (2006). Software Engineering, 9th Edition. *Addison-Wesley*.

Tomek, R., & Kalinichuk, S. (2015). Agile PM and BIM: A Hybrid Scheduling Approach for a Technological Construction Project. *Procedia Engineering*, *123*, 557–564. https://doi.org/10.1016/J.PROENG.2015.10.108

Tuya, J., Ramos Román, I., & Dolado Cosín, J. (2007). *Técnicas cuantitativas para la gestión en la ingeniería del software*.

Vadapalli, S. (2018). *DevOps : Dive into the core DevOps strategies, rapid learning solution*. 134.

Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2020). *Transforming Disciplined IT Functions* (pp. 293–306). https://doi.org/10.4018/978-1-7998-4165-4.ch015

Anexos