APLICACIÓN WEB PARA GESTIONAR INCIDENCIAS EN CI/CD EN UN ENTORNO DEVOPS

WEB APPLICATION TO MANAGE CI/CD INCIDENTS IN A DEVOPS ENVIRONMENT

Aldonso Becerra Sánchez
Universidad Autónoma de Zacatecas, México
a7donso@uaz.edu.mx

Felipe de Jesús Delgado Troncoso
Universidad Autónoma de Zacatecas, México
felipe.deltro@gmail.com

Gustavo Zepeda Valles
Universidad Autónoma de Zacatecas, México
gzepeda@uaz.edu.mx

René Ulises González Arroyo
Universidad Autónoma de Zacatecas, México
rene_ulises@uaz.edu.mx

Santiago Esparza Guerrero
Universidad Autónoma de Zacatecas, México
chago@uaz.edu.mx

Nancy Delgado Salazar
Universidad Autónoma de Zacatecas, México
nancydesal@uaz.edu.mx

Recepción: 15/noviembre/2024 **Aceptación:** 19/mayo/2025

Resumen

Actualmente muchas empresas de desarrollo de software trabajan con la integración del concepto Continuous Quality, sincronizado con DevOps para monitorear la no existencia de alguna violación a la calidad. Sin embargo, al momento de implementar herramientas de CI/CD suelen ocurrir muchas incidencias y gasto de tiempo al corregir este tipo errores, ya que no se tiene un lugar céntrico donde se puedan consultar las soluciones correspondientes. Este trabajo propone el desarrollo de una plataforma para categorizar y almacenar estas incidencias, permitiendose crear bibliotecas especializadas. Con esto se logra coadyuvar a

disminuir los factores implicados en la afectación de la calidad calidad no prevista. Para desarrollar la propuesta, se emplea la metodología de prototipo, la cual gira alrededor de iteraciones continuas, produciendo una aplicación que no solo optimiza el proceso, sino que también promueve la adopción de mejores prácticas y fomenta un entorno colaborativo en la resolución de problemas.

Palabras Clave: Calidad continua, Control de incidencias, DevOps, Desarrollo de software.

Abstract

Currently, many software development companies work with the integration of the Continuous Quality concept, synchronized with DevOps to monitor the non-existence of any quality violation. However, when implementing CI/CD tools, many incidents and time is wasted when correcting these types of errors, since there is no central place where the corresponding solutions can be consulted. This work proposes the development of a platform to categorize and store these incidents, allowing the creation of specialized libraries. This helps to reduce the factors involved in affecting unforeseen quality. To develop the proposal, the prototype methodology is used, which revolves around continuous iterations, producing an application that not only optimizes the process, but also promotes the adoption of best practices and fosters a collaborative environment in problem solving.

Keywords: *Continuous quality, DevOps, Incident control, Software development.*

1. Introducción

En la construcción de cualquier tipo de software es indispensable el uso de metodologías y herramientas para llevar un control de tiempo, dinero y esfuerzo. Al desarrollar software con metodologías ágiles, las empresas llevan un control más riguroso de los productos generados; esto ha demostrado ahorrar tiempo en el proceso, mejorando la comunicación entre los clientes y los programadores, a diferencia de las metodologías tradicionales. En este sentido, si se comete un error en una etapa de la construcción del software, este no será resuelto hasta llegar a etapas avanzadas, retrasando la construcción del software; estos errores, también

conocidos como incidencias, pueden afectar a largo plazo en el resultado. La falta de cooperación, y los problemas que surgen entre el equipo de operaciones y el equipo de desarrollo, ha ganerado la evolución de la manera de crear software. Para estos proyectos, con base en las metodologías ágiles, nace la disciplina Development and Operations (DevOps, Desarrollo y Operaciones). DevOps es una cultura basada en mejorar la comunicación entre los dos departamentos (Figura 1), reduciendo tiempos muertos y agilizando procesos; esto permite liberar productos de software de forma rápida y frecuente a los clientes [Toh, 2019]. DevOps enfatiza el aprendizaje y la mejora continua mediante la transmisión de comentarios desde producción hasta el desarrollo, mejorando el tiempo de ciclo [Huttermann, 2012]. DevOps implica aspectos de cultura, automatización, medición y uso compartido [Smeds, 2015], [Purohit, 2020].

Fuente: [Felipe, 2022]

Figura 1 Flujo de trabajo de DevOps.

Por otra parte, Continuous Integration (CI, Integración Continua) es una fase de DevOps que controla el código frecuentemente integrado, siendo verificado automáticamente para ingresar código fuente limpio al ya existente. Toda esta filosofía de trabajo permite realizar entregas de productos de software más rápidas y confiables [Ståhl, 2014]. En este mismo sentido se tiene otra fase de DevOps, Continuous Delivery (CD, Entrega Continua), enfocada sobre la labor del equipo de operaciones. Con ello se busca generar entregas iterativas en un ciclo corto para garantizar que el software siempre se encuentre en un estado de lanzamiento, evitando afectar los cambios continuos. Al implementar CD se permite que los equipos de desarrollo y operaciones trabajen en la iteración paralelamente [Chen, 2015]. Estudios realizados, para conocer el comportamiento al implementar las prácticas de DevOps [Toh, 2019], relatan que los problemas más discutidos entre sus encuestados son la falta de cultura en el equipo; el control y conocimiento de

las herramientas y hardware, el control de los recursos, y por último, la no existencia de un proceso limpio para implementar DevOps o una guía concreta. Por ello se puede deducir que la implementación de DevOps en una empresa u organización no es una tarea fácil; es así como se originan muchos cambios internos, tecnológicos, estructurales y de proceso.

Al implementar DevOps se observa que la mayoría de los encuestados concuerdan en considerar esta disciplina destacable, principalmente por sus ventajas más importantes [Toh, 2019]; entre las cuales se pueden incluir: los equipos de trabajo tienen más comunicación y calidad cuando se realiza un proyecto, la calidad del software aumenta y el costo de desarrollo y el tiempo se ven disminuidos. Las ventajas más a resaltar implican la existencia de más funciones implementadas y lanzamientos más frecuentes [Riungu-Kalliosaari, 2016]. También se menciona que DevOps ayuda en la automatización, con ello reduciendo el esfuerzo requerido para configuraciones manuales, ayudando a las empresas y organizaciones a realizar lanzamientos de software con tanta frecuencia como sea posible.

Al manejar la calidad en DevOps, existe algo conocido como Continuous Quality (CQ, Calidad Continua), con lo cual se refiere al enfoque encargado del control de la calidad de CI/CD. De esta manera se coloca a la par, la calidad con CI/CD, ayudando a solucionar problemas más tempranamente, contribuyendo al no aumento de la deuda técnica [Quality Clouds, 2020]. Al mejorar la calidad del software, las empresas pierden menos tiempo de desarrollo y menos dinero por cada proyecto que desarrollan. La calidad es uno de los factores más relevantes, por tanto, si no se trata adecuadamente, puede afectar a las empresas en cada uno de sus proyectos. Existen algunas formas de llevar una buena calidad en CI/CD, una de ellas es como lo maneja "Quality Clouds", el cual es un gestor de software de control de calidad. Se integra la Calidad Continua (CQ) con herramientas y con el apoyo de un "Gerente de Calidad" para almacenar y verificar las violaciones de las mejores prácticas en cada etapa. Estos pueden seleccionarse tanto de una biblioteca preexistente, como personalizarse según sus necesidades individuales. En el momento de desarrollar el software, cada paso se está monitoreando para dar una respuesta en tiempo real a los cambios que se están realizando. De esta

manera se detectan errores tempranos para ser corregidos a la brevedad antes de pasar a cualquier otra etapa del desarrollo; ya sea que estén recién codificados o heredados. Estas correcciones incrementales reforzarán la calidad del software.

Al momento de realizar verificaciones de la calidad, todas las fases pertenecientes de CI/CD utilizan varias herramientas durante las fases de desarrollo, compilación y de lanzamiento para realizar revisiones de la calidad. Existen muchas herramientas excelentes que se integran bien en esta parte del proceso, incluidas Jenkins, Bamboo, BitBucket, GitLab, Copado y CircleCI [Quality Clouds, 2020].

En este sentido, un miembro del equipo de Quality Clouds, menciona que uno sus clientes, con una configuración de DevOps madura, comenzó a implementar un enfoque de calidad continua. En los primeros 6 meses, ya han visto un aumento del 15% en su velocidad de desarrollo. Con menos deuda técnica acumulada, los primeros indicios muestran un prometedor impacto en el desarrollo de la plataforma en el futuro.

Cuando se desarrolla software con metodologías ágiles, y en un entorno como DevOps, se agiliza el trabajo de construcción de los proyectos por medio de las herramientas de automatización; por ende, el costo de desarrollo es menor. Actualmente algunas empresas trabajan con la integración de herramientas que utilizan el concepto de Continuous Quality; las cual se sincronizan con DevOps para verificar y monitorear en cada paso la no existencia de alguna violación a la calidad. Sin embargo, al momento de implementar herramientas de CI/CD, suelen ocurrir muchas incidencias y gasto de tiempo no planeados al corregir este tipo errores. Esto sucede debido a la carencia de un lugar céntrico donde se puedan consultar las soluciones a estos errores, y se invierte mucho tiempo en la búsqueda en diferentes páginas de internet.

Para la detección de incidencias, y su posible solución, existen varias herramientas de software, las cuales ayudan a visualizar y monitorear los servicios y servidores, tales son: Grafana, New Relic, DataDog, Sentry, entre otros. De igual forma existen foros, páginas web y comunidades, así como StackOverFlow, donde se discuten y comparten incidencias y posibles soluciones. Hasta donde se tiene conocimiento, no existe algún software para la gestión y almacenamiento de incidencias con sus

posibles soluciones, donde todas las incidencias se encuentren agrupadas en un solo lugar y sea sencillo realizar una búsqueda o un filtro. Es por eso que el propósito del presente trabajo es proponer un desarrollo de una aplicación de software para almacenar, consultar y resolver las incidencias más frecuentes en CI/CD en un entorno DevOps. El proyecto tiene como fin implementar una mejora en el desarrollo de sistemas de software basada en CQ.

Cuando se habla de la calidad en el software, es normal hablar de estándares y normas que ayudan a mejorar esta. Una de las normas más comunes y conocidas es la ISO 9001, centrada en todos los elementos de administración de calidad que impliquen tener un sistema efectivo para administrar y mejorar productos y servicios de manera general o constante. Esta norma ayuda a las empresas a tener una certificación donde se demuestra que de manera formal cuentan con calidad en sus productos de software. Tal que, estos son verificados, llevando un control de ellos en los ciclos de vida de cada software; análisis de requerimientos, diseño del software, desarrollo del software, pruebas y validaciones, y, por último, pruebas y mantenimiento [Natarajan, 2017].

Es habitual utilizar la calidad continua como un enfoque sistemático para encontrar y corregir defectos de software durante todas las fases del ciclo de desarrollo de software; ahí es donde se contribuye a reducir el riesgo de vulnerabilidades de seguridad y defectos de software (errores). Esto ayuda a los desarrolladores a encontrar y solucionar problemas lo antes posible en el ciclo de desarrollo. Al utilizar este enfoque, las organizaciones reducen el tiempo perdido en el retrabajo, aumentan la eficiencia y la satisfacción de sus clientes. Por tanto, es un proceso continuo que realiza mejoras continuas de sus procesos. DevOps y CI/CD son utilizadas para asegurar que la calidad sea constante y se pueda operar a un ritmo adecuado para estar al día con las demandas de los consumidores del software. Existen varias herramientas para el control de la calidad del software, una de ellas llamada ConQAT (Continuous Quality Assessment Toolkit), la cual puede monitorear varios proyectos al mismo tiempo [Deissenboeck, 2008]. En el ámbito de software existen diferentes modelos para permitir un proceso de mejora continua con las implementaciones que se realizan. Cada uno de los modelos cuentan con

características y una estructura. Existen empresas que implementan estos modelos con el fin de certificar y garantizar sus productos y proceso realizados. Por otro lado, se menciona [Scalone, 2006] que los modelos de calidad implican documentos que integran la mayor parte de las mejores prácticas. También proponen temas de administración en los que cada organización debe hacer énfasis, además de integrar diferentes prácticas dirigidas a los procesos clave para permitir medir los avances en calidad. La estructura de los modelos de calidad de software generalmente está formada por criterios que son evaluados métricamente; incluyendo diversos factores de calidad, obteniendo así una evaluación del software desde una vista general hasta algo muy específico de este.

Cuando se desarrolla un software, la calidad debe estar presente en todo momento (calidad y modelos a nivel proceso), siendo programada desde el inicio del proyecto. Posteriormente se debe seguir llevando el control y supervisión de cada fase del ciclo de vida del producto para que la calidad sea inherente al producto, así mitigando los riesgos y ofreciendo seguimiento continuo. Si se siguen ciertos factores y reglas, el nivel puede ser óptimo para así cumplir los factores de calidad. De igual forma, si se dejan algunos de estos factores en las etapas de vida del software, este puede llegar a no concretarse u obtener fallas en el producto desarrollado [Callejas-Cuervo, 2017]. En este sentido se tiene a Bootstrap, el cual es un modelo de trabajo centrado en mejorar los procesos para reducir tiempos e incrementar la calidad [Herrera, 2012], [Callejas-Cuervo, 2017]. Otros esquemas que coadyuvan también el área del desarrollo y la calidad como procesos, implican el uso de Personal Software Process (PSP) [Vargas, 2010], Team Software Process (TSP) [Mondragón, 2011] y la ISO/IEC 15504 [Callejas-Cuervo, 2017].

La clasificación de los modelos de calidad de software se fundamenta en el enfoque de evaluación, ya sea en el nivel de proceso, producto o calidad de uso. El modelo de calidad de producto se centra en especificar y evaluar el cumplimiento de criterios del producto tal como fueron solicitados, aplicando medidas internas y, si es necesario, externas [Bevan, 2010]. En este contexto, normas y estándares han definido la calidad del producto en tres tipos: interna, externa y en uso [Callejas-Cuervo, 2017], [Rodríguez, 2016].El objetivo principal de este enfoque es verificar

que las características definidas por el cliente durante la etapa de análisis sean cumplidas completamente y satisfagan sus requisitos. Bajo este enfoque, se puede mencionar a FURPS [Soto, 2015], Boehm [Velazco, 2016], [Callejas-Cuervo, 2017], McCall [Khosravi, 2004], la ISO 9126 [Ango, 2014}, GILB [Khosravi, 2004] o la ISO 25000 [Alfonso, 2012].

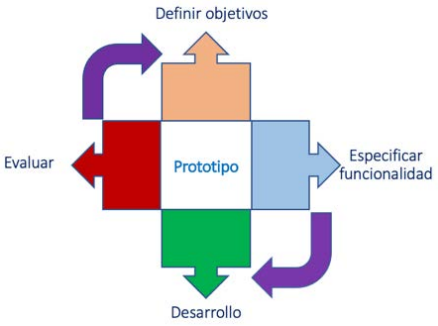
Se puede decir entonces que la calidad es uno de los factores que afectan los procesos de desarrollo de software en sus distintas etapas. Una organización puede implementar la automatización de este tipo de procedimientos de varias formas. La automatización de DevOps está respaldada por diferentes diseños; algunos de los cuales incluyen el uso de la nube para el almacenamiento de datos de gran tamaño, el uso de correo electrónico basado en la nube y servicios de registro. Incluso, todo esto utiliza una herramienta de monitoreo en tiempo real. El uso de SaaS (Software as a Service) e IaaS (Infrastructure as a Service) también es capaz de soportar la automatización de DevOps [Mishra, 2020].

En el ámbito de la cultura DevOps, las empresas y organizaciones suelen utilizar herramientas para la automatización, así de esta forma ahorra tiempo y dinero en posibles errores cometidos si estos trabajos se realizaran de manera manual. El manejo de la calidad está presente en cada etapa, donde se utiliza SonarQube para el control de la calidad [SonarSource, 2024], así como Omnia [Miglierina, 2017].

2. Métodos

Para el desarrollo de la propuesta, se empleó la metodología de prototipo (Figura 2) que es un método de desarrollo de sistemas, donde se prueba y retrabaja según sea necesario hasta obtener un resultado aceptable como punto de partida [Sommerville, 2010}, [Weitzenfeld, 2007].

Las fases de esta metodología incluyen: definición de objetivos; funcionalidad; diseño, desarrollo y evaluación del prototipo. Un prototipo es una implementación de muestra, el cual está limitado únicamente a las funcionalidades principales del sistema. Funciona a manera de prueba y error (girando en las etapas) entre los desarrolladores y usuarios, para que estos últimos prueben las funcionalidades antes de que sean implementadas por completo.



Fuente: elaboración propia

Figura 2 Metodología de Prototipo.

Objetivos del prototipo

El objetivo principal es diseñar, desarrollar e implementar un sistema de software donde se permitirá recabar las incidencias más frecuentes cometidos por desarrolladores en CI/CD en un entorno DevOps. Posteriormente se les podrá dar una solución, y serán visibles por otros desarrolladores que podrán agregar comentarios y aportaciones a estas incidencias; esto permite tener una visualización gráfica de las etapas y herramientas que tienden a generar más incidencias. Los objetivos del prototipo son los siguientes:

- El sistema debe ser ingresado por medio de usuario y contraseña.
- El sistema de software podrá ser accedido desde cualquier parte.
- Deberá ser capaz de permitir registrar, modificar, eliminar y mostrar los errores que se cometen en el desarrollo.
- El sistema permitirá agregar comentarios en las incidencias reportadas.
- El sistema mostrará gráficas con las incidencias reportadas.
- Se podrán generar reportes de los errores registrados.

Funcionalidad del prototipo

Para el desarrollo de este sistema, funcionalidades esperadas a implementar:

- Administrar errores: recopilar, modificar, eliminar y mostrar documentación de errores.
- Generar reportes de errores. El administrador podrá realizar todas las actividades del desarrollador, además podrá eliminar o modificar cualquier

incidencia en el sistema. El desarrollador realizará las mismas acciones que el invitado, y podrá capturar incidencias nuevas y podrá modificarlas. Igualmente podrá agregar comentarios en las incidencias; mientras que el invitado podrá ver todas las incidencias registradas y sus soluciones. De igual forma podrá ver las gráficas y generar reportes de las incidencias.

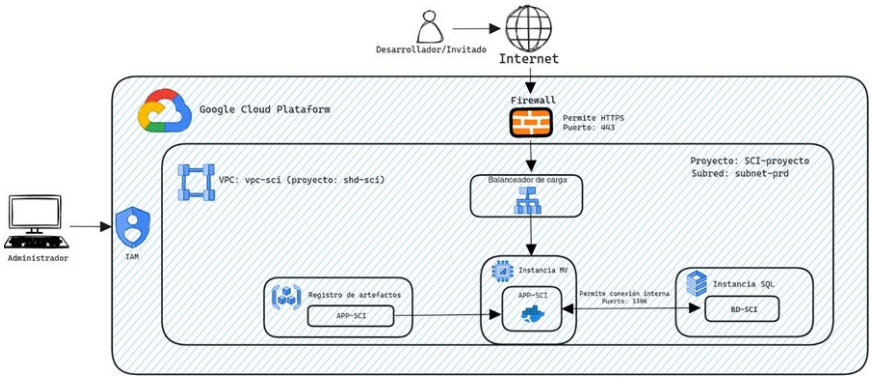
Desarrollo del prototipo

En cuanto al esquema propuesto, las siguientes tecnologías fueron seleccionadas para el desarrollo de los módulos del sistema:

- Python (back-end)
- Django (back y front)
- Mysql (base de datos)
- Bootstrap (front-end)

Arquitectura

En la Figura 3 se diagrama la arquitectura del sistema creado, la cuál utiliza varios servicios y características de GCP (Google Cloud Platform) para proporcionar una aplicación escalable y segura.



Fuente: elaboración propia

Figura 3 Componentes de la arquitectura propuesta.

El uso de un balanceador de carga garantiza una distribución uniforme de la carga de trabajo; mientras que las políticas de firewall y autenticación IAM (Identity and Access Management) aseguran que solo el tráfico y los usuarios autorizados

puedan interactuar con la aplicación y sus recursos. Los elementos sobresalientes del esquema de arquitectura son:

- Usuario: un desarrollador que accede a la aplicación a través de internet.
- Internet: medio a través del cual el desarrollador se conecta a la aplicación.
- Firewall: configurado para permitir tráfico HTTPS en el puerto 443.
- Load balancer: distribuye el tráfico entrante entre las instancias de la APP.
- VM instance: una instancia de máquina virtual donde se ejecuta la aplicación (SCI-APP). Esta instancia tiene Docker instalado para contener la aplicación.
- Instance SQL: una instancia de base de datos MySQL (SCI-DB) para almacenar los datos de la aplicación con conexiones en el puerto 3306.
- Artifact registry: repositorio para almacenar los artefactos (por ejemplo, imágenes Docker), conteniendo la imagen de la aplicación (SCI-APP).
- IAM - Auth (Administrador): el administrador utiliza la autenticación IAM para gestionar y desplegar la aplicación y sus componentes.
- VPC (vpc-sci): red privada virtual específica del proyecto (proyecto: shd-sci, subnet: subnet-prd).

Seguridad en el software desarrollado

En los proyectos de software la seguridad es un aspecto importate a considerar, teniendo en cuenta en este proyecto:

- Protección CSRF: Cross-Site Request Forgery es un proceso donde un atacante engaña a un usuario autenticado para que ejecute acciones no deseadas en una aplicación web. Por ello se agregó en los formularios y vistas basadas en clases esta validación contra CSRF, la cual genera un token que se invita mediante un método POST.
- Protección contra inyecciones SQL: la inyección SQL es un proceso donde un atacante inserta o manipula consultas SQL maliciosas para ejecutar acciones no autorizadas en la base de datos de una aplicación. El ORM de Django se emplea para interactuar con la base de datos en lugar de escribir consultas SQL manualmente. El ORM de Django escapa automáticamente los valores, previniendo inyecciones SQL. En cuanto a los principios de

privilegios mínimos, se configuran las cuentas de la base de datos con los menores privilegios necesarios para la aplicación. Implementar estas prácticas es crucial para prevenir inyecciones SQL y asegurar la integridad y seguridad de tu base de datos. Django ofrece herramientas integradas para facilitar el desarrollo seguro.

- Protección contra XSS en Django: la inyección de scripts entre sitios (XSS) es un ataque donde un atacante inserta scripts maliciosos en páginas web vistas por otros usuarios. Estos scripts pueden robar datos sensibles, manipular el contenido de la página o realizar acciones no autorizadas en nombre del usuario. Implementar estas prácticas es crucial para prevenir ataques XSS y asegurar la integridad y seguridad de una aplicación web. Django proporciona herramientas que facilitan el desarrollo seguro:
    - ✓ Escapado de datos: escapa todos los datos proporcionados por el usuario antes de mostrarlos en la página web. Django escapa automáticamente los datos en sus plantillas, pero es importante asegurarse de que esto se aplique en todo momento.
    - ✓ Uso de plantillas seguras: usa el sistema de plantillas de Django, escapando automáticamente los datos para prevenir la ejecución de scripts maliciosos.
    - ✓ Validación de entradas: valida y sanitiza todas las entradas del usuario antes de procesarlas o mostrarlas. Esto incluye eliminar o codificar caracteres especiales propensos a ser utilizados en ataques XSS.
    - ✓ HTTPOnly y Secure Cookies: configura las cookies con las opciones HttpOnly y Secure para evitar su accesibilidad desde JavaScript, asegurando que solo se envíen a través de conexiones HTTPS.

Flujo de la arquitectura

El mecanismo de flujo de datos a seguir por el sistema propuesto se define como:

- Acceso de los usuarios: el usuario accede a la aplicación desde su dispositivo a través de Internet. La solicitud llega al Firewall, permitiendo el tráfico HTTPS en el puerto 443.

- Distribución de la solicitud: la solicitud es recibida por el Load Balancer, distribuyendo la carga de trabajo entre las instancias de la aplicación.
- Ejecución de la aplicación: el Load Balancer redirige la solicitud a una instancia específica de la VM, donde se ejecuta la aplicación (SCI-APP) en un contenedor Docker. La aplicación puede necesitar acceder a la base de datos para recuperar o almacenar información.
- Acceso a la base de datos: la aplicación (SCI-APP) se conecta internamente a la Instance SQL en el puerto 3306 para realizar operaciones de datos.
- Administración y despliegue: el administrador gestiona y despliega la aplicación y sus componentes usando las credenciales de IAM. Los artefactos de la aplicación, como las imágenes de Docker, se almacenan en el Artifact Registry y se despliegan en las instancias de VM.

Evaluación del prototipo

Para la parte de seguimiento de las fases de la metodología de Prototipo, la parte de la evaluación se definió con el cumplimiento y validación de los objetivos planteados en diversas iteraciones, para la iteración 1 se completó el proceso de ingreso por medio de usuario y contraseña. En la iteración 2 se desarrolló la funcionalidad de acceso remoto y funcional por medio de vínculo web; mientras que en la iteración 3 se generó el módulo de registrar, modificar, eliminar y mostrar los errores cometidos en el desarrollo. En tanto que en la iteración 4 se definió la funcionalidad de agregar comentarios en las incidencias reportadas; prosiguiendo en la iteración 5 con el procedimiento de mostrar gráficas con las incidencias reportadas. Finalmente, en la iteración 6 se generaron los reportes de los errores registrados.

3. Resultados

En la Figura 4a se puede apreciar la página de inicio de sesión, donde lo desarrolladores pueden ingresar al sistema por medio de su correo electrónico y su contraseña; de igual forma se puede ingresar al sistema como invitado sin la necesidad de tener una cuenta. Mientras que en la Figura 4b se puede apreciar la

página de registro donde se podrá crear un usuario por medio de un correo electrónico para poder interactuar con el sistema. Mientras que la Figura 4c permite la recuperación de contraseña correspondiente. Después de este proceso, se puede verificar su página de home (Figura 5), donde se puede navegar a las distintas opciones del sistema.

a) Login de usuario. *b) Registro de nuevo usuario.*

c) Recuperación de contraseña.

Fuente: elaboración propia

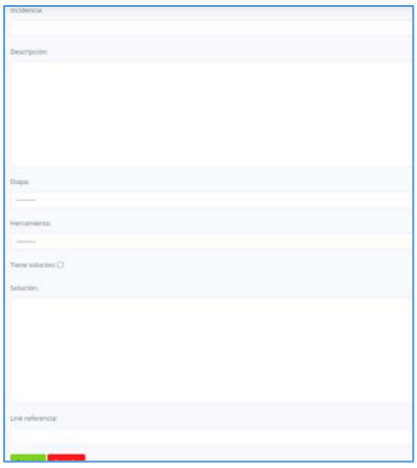
Figura 4 Ventanas de inicio de usuarios.

Fuente: elaboración propia

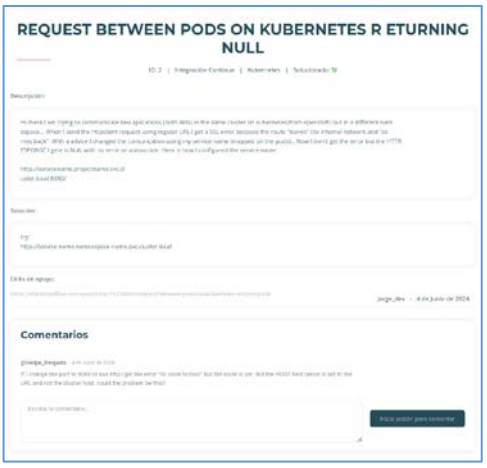
Figura 5 Página principal del sistema

Una vez ingresado al sistema, se puede generar una nueva incidencia (Figura 6a), donde el usuario registrado podrá agregar un error y su posible solución. Una vez

dada de alta la icidencia, se puede tener acceso a mostrar la especificación del registro proporcionado (Figura 6b), donde se tiene una incidencia, sus detalles y comentarios que pueden realizar otros usuarios registrados.



a) Agregar nueva incidencia.



b) Detalle de incidencia.

Fuente: elaboración propia

Figura 6 Ventanas de registro de nueva incidencia y sus características.

En la Figura 7 se puede apreciar que se proporcionan tres tipos de gráficas para determinar las herramientas y etapas donde se registran más incidencias, y por ende donde es más propenso a pasar más tiempo de desarrollo. En Figura 7, la de barras, para incidencias registradas en herramientas específicas de CI/CD; de pastel, para incidencias etapas del proceso de desarrollo; mientras que también se muestra la de radar, para etapa y herramienta.

Fuente: elaboración propia

Figura 7 Listado de incidencias y su búsqueda.

En la Figura 8 se muestran las ventanas de registro de incidencias y características. Donde además se muestran herramientas de listado y búsqueda de las incidencias más relevantes reportadas en el sistema. Página donde se pueden crear reportes de incidencias por medio de filtros y combinaciones de estos mismos para búsquedas más rápidas y concretas.

a) Listado de incidencias registradas.

b) Búsqueda de incidencias.

Fuente: elaboración propia

Figura 8 Ventanas de registro de nueva incidencia y sus características.

4. Discusión

Este trabajo está regido bajo el régimen de calidad a nivel de proceso y a nivel de producto, además de guiarse por los lineamientos de Calidad Continua de DevOps. El resultado de la propuesta permite ayudar a solucionar el problema de la falta de calidad que aún no se toma en cuenta en muchos de los desarrollos de sistemas de software. En este trabajo se planteó generar una herramienta para recopilar los errores más frecuentes en CI/CD con una plataforma web, para posteriormente almacenar y consultar estos cuando sea necesario. Con ello se verifica la calidad y se revisará constantemente lo que anteriormente no se tomaba en cuenta.

Al realizar este software se pretende incrementar la calidad en las empresas que utilizan CI/CD en un entorno DevOps; con ello apoyar a reducir el retrabajo, reducción de tiempos al encontrar incidencias, el costo de sus proyectos y el esfuerzo, además de ayudar a los desarrolladores encargados de estas áreas.

5. Conclusiones

Es bien conocido que en la actualidad se disponen de diversas empresas de desarrollo de software laborando con un flujo de trabajo integrado. Dicho proceso, implica diversas utilerías para aplicar el paradigma del concepto de Continuous Quality, las cual van de la mano con el proceso de DevOps. Este flujo de datos y acciones se emplea para verificar y monitorear en cada instante el apego a la calidad de los productos. Sin embargo, en el instante que se empiezan a emplear herramientas de CI/CD, suelen suceder errores/incidencias e inversiones de tiempo no planeadas para subsanar este tipo de situaciones. Esto se deriva de la ausencia de un repositorio central para almacenar y consultar las soluciones a estos hechos. Esto saca a la luz la necesidad de generar una herramienta que pueda categorizar y almacenar los errores más frecuentes. En este trabajo se llevó a cabo una propuesta para crear bibliotecas que ayuden a la calidad, y estas así mismo a disminuir los factores ya antes mencionados que se pierden con la falta de calidad no prevista. En el proceso de desarrollo se utilizó la metodología de prototipo, la cual se encamina bajo un flujo de iteraciones continuas con el fin de lograr el producto deseado.

Como trabajo futuro se planea generar un módulo de aprendizaje supervisado, empleando machine learning, tal que se permita predecir incidencias futuras, así como los tipos y soluciones posibles de ellas.

6. Bibliografía y Referencias

[1] Quality Clouds. Continuous quality - the missing CI/CD ingredient. 2020.

[2] Alfonso, P.L. Revisión de modelos para evaluar la calidad de productos web. experimentación en portales bancarios del nea. Tesis de maestría, Universidad Nacional de La Plata, Argentina, 2012.

[3] Ango, L.F. Evaluación de sistemas. Tesis de maestría, Pontificia Universidad Catolica del Ecuador, Ibarra, 2014.

[4] Bevan, N. Los nuevos modelos de iso para la calidad y la calidad en uso del software. In: Calidad del producto y proceso software. Editorial Ra-Ma, España, 5-75, 2010.

[5] Callejas-Cuervo, M., Alarcón-Aldana, A. C., Álvarez- Carreño, A. M. Modelos de calidad del software, un estado del arte. Entramado Vol. 13, No. 1, 236-250, 2017.

[6] Chen, L. Continuous delivery: Huge benefits, but challenges too. IEEE Software, Vol. 32, No. 2, 50-54, 2015.

[7] Deissenboeck, F., Wagner, S., Pizka, M., Hummel, B., Juergens, E., Mas, B. Tool Support for Continuous Quality Control. IEEE Software, Vol. 25, No. 5, 60-67, 2008.

[8] Felipe, A. M., Núñez, F. DevOps: un vistazo rápido. Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla, Vol 10, No. 19, 35-40, 2022.

[9] Herrera, A.: Bootstrap. Procesos Software, 2012.

[10] Huttermann, M. DevOps for Developers. Apress, Berkeley, CA, 2012.

[11] Khosravi, K., Guéhéneuc, Y.G. A quality model for design patterns. In: German Industry Standard, 2004.

[12] Miglierina, M., Tamburri, D.A. Towards omnia: A monitoring factory for quality-aware devops. Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering Companion, abril, 145-150, 2017.

[13] Mishra, A. Otaiwi, Z. Devops and software quality: A systematic mapping. Computer Science Review, Vol. 38, 100308, 2020.

[14] Mondragón, O. Integrando TSP y CMMI Lo mejor de dos mundos. Software Guru, Vol. 50, 2011.

[15] Natarajan, D. Implementing qms with erp software. In ISO 9001 Quality Management Systems,129-137, 2017.

[16] Purohit, K. Executing devops ci/cd, reduce in manual dependency. International Journal of Scientific Development and Research, Vol. 5, No. 9, 511-515, 2020.

[17] Riungu-Kalliosaari, L., M¬®akinen S., Lwakatare, L. H., Tiihonen, J. Devops adoption benefits and challenges in practice: A case study. International Conference on Product-Focused Software Process Improvement, 2016.

[18] Rodríguez, M. Evaluation of software product functional suitability: A case study. Software Quality Professional Magazine, Vol. 18, No. 3, 2016.

[19] Scalone, F.: Estudio comparativo de los modelos y estándares de calidad del software. Tesis de maestría, Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, 2006.

[20] Smeds, J., Nybom, K., Porres, I. Devops: A definition and perceived adoption impediments. International Conference on Agile Software Development, Vol 212, 166-177, 2015.

[21] Sommerville, I.: Software Engineering. 8 edn. Addison-Wesley, England, 2010.

[22] SonarSource: SonarQube 10.6 Documentation. Disponible en: https://docs.sonarsource.com/ sonarqube/latest/, 2024.

[23] Soto, J. R. Actividad 2.2: Cuadro comparativo de modelos para evaluar la calidad del software (módulo: evaluación de la calidad de la tecnología educativa). En: ISO 69, 2015.

[24] Ståhl, D., Bosch, J. Modeling continuous integration practice differences in indus- try software development. Journal of Systems and Software, Vol. 87, 48-59, 2014.

[25] Vargas, F., Soto, D. Introduciendo psp (proceso personal de software) en el aula. Revista Colombiana de Tecnologías de Avanzada, Vol. 2, No. 16, 2010.

[26] Velazco, A. Modelo en espiral, introducción Boehm, 2016.

[27] Weitzenfeld, A., Guardati, S.: Capítulo 12: Ingeniería de software: el proceso para el desarrollo de software, Introducción a la Computación. CENGAGE Learning, Mexico, 2007.

[28] Toh, M. Z., Sahibuddin, S., Mahrin, M. N. Adoption issues in devops from the perspective of continuous delivery pipeline. Proceedings of the 2019 8th International Conference on Software and Computer Applications, 173-177, 2019.