UNIVERSIDAD SAN PEDRO

FACULTAD DE INGENIERÍA

PROGRAMA DE ESTUDIOS DE INGENIERIA INFORMATICA Y DE SISTEMAS

Flujo de trabajo para automatizar el control de calidad de aplicaciones web bajo el enfoque Devops, en la empresa SingLabz Solutions S.A.C, Lima 2018

Tesis para obtener el título de Ingeniero en Informática y de Sistemas

Autor

Marchan Marquina Eleana Teresa

Asesor

Carrasco Alvarado, Wilmer

Chimbote– Perú

2020

ÍNDICE

Palabras claves

Tema	Aplicaciones web
Especialidad	Ingeniería de software

Keywords

Theme	Web applications
Specialty	Software engineering

Línea de investigación Concytec

Área **Sub Área** **Disciplina**	Ingeniería y Tecnología Ingeniería Eléctrica, Electrónica e Informática. Automatización y Sistemas de Control

“Flujo de trabajo para automatizar el control de la calidad de aplicaciones web bajo el enfoque DevOps, en la Empresa SingLabz Solutions S.A.C., Lima 2018”

Resumen

El presente proyecto tuvo como propósito desarrollar un flujo de trabajo automatizado para el control de calidad de aplicación web bajo el enfoque DevOps, en la empresa **SingLabz Solutions S.A.C**, para el establecimiento de los requerimientos se tomó en cuenta la necesidad de la empresa de entregar software de calidad de manera rápida, que haya pasado como mínimo por pruebas unitarias y funcionales de manera automatizada.

Asimismo, la presente investigación es de carácter descriptivo y diseño no experimental de corte transversal, y de acuerdo con la orientación es de tipo tecnológica. Para el análisis y diseño del flujo automatizado para el control de calidad de aplicaciones web, se aplicó la metodología Scrum. Para la implementación de las fases de integración y entrega continúa se utilizó como servidor de integración Jenkins, para la creación de los ambientes de prueba, se utilizó Docker y Selenium Grid y como repositorio para la gestión del versionado de los aplicativos se utilizó Gitlab.

El desarrollo del sistema permitió facilitar el control de la calidad de aplicaciones web, dar a conocer la cultura DevOps, promover un clima de colaboración entre los equipos de desarrollo, QA y operaciones, la adopción de nuevas tecnologías y la mejora en la calidad de vida de los trabajadores de la empresa.

Abstract

The purpose of this project was to develop an automated workflow for the quality control of the web application under the DevOps approach, in the company SingLabz Solutions SAC, for the establishment of the requirements, the need of the company to deliver software was taken into account of quality quickly, that has passed at least unit and functional tests in an automated way.

Likewise, this research is of a descriptive nature and a non-experimental cross-sectional design, and according to the orientation it is of a technological nature. For the analysis and design of the automated flow for the quality control of web applications, the Scrum methodology was applied. For the implementation of the integration and continuous delivery phases, Jenkins was used as the integration server, for the creation of the test environments, Docker and Selenium Grid was used, and Gitlab was used as a repository for managing the versioning of the applications.

The development of the system included facilitating the quality control of web applications, publicizing the DevOps culture, promoting a climate of collaboration between the development, QA and operations teams, the adoption of new technologies and the improvement in the quality of life of the company's workers.

1. Introducción

Durante la revisión bibliográfica, realizado por la autora; se han abordado los trabajos de mayor relación con los propósitos de la presente investigación, los mismos que se detallan a continuación:

Se revisó la tesis de Farias (2017) en la tesis para el grado de titulación: "*Definición de un ambiente de construcción de aplicaciones empresariales a través de Devops, Microservicios y Contenedores*". El estudio tuvo como objetivo, describir las herramientas y conocimientos necesarios para la definición de un ambiente de construcción de aplicaciones empresariales a través de DevOps, el trabajo de investigación hace uso de Scrum como metodología ágil para la fase de la planificación del ambiente DevOps y para la implementación de las fases de desarrollo utiliza el enfoque DevOps. El estudio hace énfasis en la adopción de métodos ágiles y en la descripción de los procesos de integración, entrega y despliegue continuos a través de la automatización de procesos y en el uso de una arquitectura de microservicios que facilite el desarrollo de ambientes de construcción de aplicaciones más productivos. En conclusión, la implementación del enfoque de DevOps implica un cambio cultural y un compromiso de las organizaciones, no todo se trata de herramientas y automatización, es importante la colaboración entre los equipos de desarrollo y operaciones.

Por otro lado, Tasato (2013) en la tesis para el grado de titulación: *"Desarrollo de una infraestructura de software para realizar pruebas automatizadas de sistemas de información desarrollados en lenguaje cobol en el contexto bancario"*. El proyecto tuvo como objetivo automatizar las pruebas de regresión encargadas de probar los componentes que interactúan con el componente cambiado, lo que conlleva a realizar las mismas pruebas reiteradas veces por algún cambio, dichas pruebas son realizadas por el área de certificación de un banco, el autor describe que el área se ha convertido en un cuello de botella para la institución debido a la lentitud de las pruebas realizadas de manera manual. El proyecto se basa en la NTP-ISO/IEC 29110-5-1-2 es una traducción de la norma internacional ISO/IEC 29110-5-1-2 [NTP, 2012].

El estudio fue acerca del desarrollo de una infraestructura que permita automatizar las pruebas de calidad de software realizadas a los sistemas bancarios, debido al impacto en un cambio sobre alguna variable o funcionalidad, las pruebas a realizar deben ser

rigurosas, con el fin de asegurar que no se haya afectado una funcionalidad. En conclusión, es importante tener definido los casos de prueba para poder obtener el resultado deseado, es importante promover en las organizaciones la cultura de la calidad que incentive la creación de nuevas ideas y la adquisición de nuevos conocimientos.

Así mismo Belalcázar (2017) en la tesis de doctorado: "*Arquitectura de un data center con herramientas devops*", el estudio tuvo como objetivo efectivizar el uso de DevOps a través de la entrega de servicios estratégicos para la organización, permitiendo alinear la ejecución de DevOps con la estrategia del negocio. Con el fin de alcanzar los objetivos. Se usó como modelo de alineamiento estratégico SAM-RS y como metodología Lean Startup, *s*e describió la propuesta de un modelo para la administración de un Data Center, a través del estudio de marcos de trabajo de COBIT (Control Objectives for Information and related Technology) y SAM (Strategic Alignment Model. SAM) e ITIL (IT Infrastructure Library, biblioteca de infraestructura de TI), estos modelos de alineación estratégica brindan un marco de referencia que buscan definir un alineamiento estratégico entre los recursos de TI y la organización, relacionándolos con el enfoque DevOps. En conclusión, se puede integrar DevOps con marcos de trabajo como COBIT, ITIL, SAM-RS para obtener resultados beneficiosos para las organizaciones.

También se revisó a Céspedes (2017) en la tesis de maestría: *"Propuesta de implementación del proceso de soporte de aplicaciones de seguridad de la información para que sea brindado por InfoSec de Intel Costa Rica",* el estudio tuvo como objetivo diseñar una propuesta de implementación de los procesos de ITIL, apoyados en el enfoque DevOps y los principios Agile, se describió una propuesta para el servicio de soporte de aplicaciones a través de la adopción de marcos de trabajo como ITIL, DevOps y Agile. El estudio hace un análisis del estado del servicio de soporte de aplicaciones que aseguran la seguridad de la información en la empresa, Se usó como metodología ITIL 2011, DevOps, Agile a través del uso de Scrum. En conclusión, es necesario el compromiso de la empresa para poder contar con las herramientas necesarias para implementar los marcos de trabajo propuestos, el autor destaca también el beneficio del uso de listas de verificación para conocer el grado de conocimiento que existe en la empresa con respecto a los marcos de referencia como ITIL, DevOps y Agile.

Por otra parte, Santos (2016) en la tesis de maestría: *"Management del aseguramiento de la calidad en desarrollos de software de telecomunicaciones"*, El estudio tuvo como objetivo analizar el nivel de las pruebas de calidad de software en la empresa Telecom y probar la hipótesis acerca de que la empresa debe cambiar su manera de realizar el control de calidad, se describió la importancia del testing en el desarrollo de software, los diferentes tipos y niveles de prueba y la adopción de nuevas metodologías de desarrollo que involucren el testing desde el inicio del desarrollo y no al final de la cadena de software. No se precisa una metodología clara. En conclusión, se propone un cambio en la manera en la que se desarrolla software en la empresa se recomienda adoptar metodologías agiles.

La presente investigación, tiene su sustento desde el punto de vista científico, porque busca generar una base de conocimientos a través de la aplicación de metodologías, herramientas y experiencias que permitan diseñar e implementar un flujo de trabajo automatizado para el control de calidad de las aplicaciones web desarrolladas por la empresa SingLabz Solutions S.A.C, que permita ejecutar la mayor cantidad de pruebas en poco tiempo, evitar la repetición de tareas que no aportan valor, mejorar el trabajo de integración de los componentes del software; de esta manera la empresa podrá atender de manera más eficaz la inclusión de una nueva funcionalidad o la corrección de una incidencia reportada. Esto asegurará un flujo constante de entrega de software y la detección temprana de errores.

Desde el punto de vista social busca mejorar el clima de trabajo de los equipos de desarrollo y operaciones TI, al eliminar el pensamiento de trabajo en equipos separados por un trabajo más colaborativo, el aseguramiento de la calidad será adoptada no solo por el equipo de QA sino por los demás equipos de la empresa, de esta manera se promueve la adopción de nuevas tecnologías; todo esto mejorará la calidad de vida de los trabajadores, ya que podrán estar seguros, que el software que se encuentra en producción ha sido probado de manera adecuada y se encuentra monitoreado constantemente.

La problemática encontrada con respecto al control de la calidad de los aplicativos web es debido a que no se contaba con un proceso automatizado que les permita realizar las pruebas necesarias que cubran en un nivel deseado la calidad de los aplicativos web, se encontró que los aplicativos son probados en ambientes de prueba de pre producción, que

no cuentan con las mismas características de producción, además las pruebas realizadas a los aplicativos son solo pruebas de verificación manuales, descuidando la revisión de la calidad del código fuente escrito. No existía una gestión de los builds y releases producidos por el equipo, por lo que era casi imposible volver a una versión anterior estable de un aplicativo, esto sumado al aumento en la demanda de aplicativos webs, hace que el equipo tenga que desarrollar más aplicaciones en poco tiempo. Por lo tanto, se deduce que se hace necesario la creación de un flujo de trabajo para automatizar el control de calidad de aplicaciones web con una metodología ágil como DevOps y así reducir el número de incidencias encontradas y aumentar la productividad del equipo de desarrollo. Ante tal realidad y para resolver esta problemática, la autora se planteó la siguiente interrogante:

¿Cómo desarrollar un Flujo de trabajo para automatizar el control de la calidad de aplicaciones web bajo el enfoque Devops, en la Empresa SingLabz Solutions S.A.C.?

Para la implementación del flujo automatizado, se ha considerado la operacionalización previa de las variables que permitan una mejor descripción y aplicación de la metodología de diseño utilizada, que a continuación procedo a describir:

¿Qué es DevOps?

El término DevOps fue formalizado por Patrick Debois, procede de fusionar los equipos de Desarrollo y Operaciones con el objetivo de entregar valor frecuentemente y garantizar la continuidad de las operaciones en la organización, de manera iterativa e incremental, el fin primario de DevOps es promover el trabajo coordinado de ambos equipos.

Por otro lado Tony Stafford (2017) afirma que, al hablar de DevOps, se piensa solo en herramientas, sin embargo, se necesita mucho más, debemos entender que DevOps no es algo que se pueda comprar. Es una nueva metodología de trabajo que deberá adoptar la organización y que tomará tiempo en crecer. El primer paso para tener éxito con DevOps es la cultura: crear equipos multifuncionales guiados por la empatía, la transparencia, el respeto y la alineación para producir el mismo resultado.

Cultura DevOps

La cultura DevOps se basa en el modelo CAMS (Culture, Automation, Measurement, Sharing), inventado por Damon Edwards y John Willis. A continuación, se detalla que significa cada aspecto de la cultura DevOps.

Culture, la cultura es la parte más importante del movimiento DevOps, debemos involucrar a todas las partes involucradas en el desarrollo del producto o servicio desde el área técnica hasta las áreas encargadas del descubrimiento, diseño y definición del producto solo así lograremos tener una visión transversal en la que se basa DevOps, para esto debemos enfocarnos en estos 6 principios: Acción orientada en el cliente, promueve una cultura abierta que reduce la burocracia y se enfoca en la innovación, además requiere abrirse a la experimentación y aplicar técnicas que nos ayuden a descubrir lo que realmente necesita el cliente, otro aspecto es crear con el fin en mente, para esto se necesita no perder de vista como es que finalmente llega al usuario final el producto o servicio que estamos construyendo y el impacto que tendrá, para aplicar este principio se necesita la participación de todos los involucrados eliminando la mentalidad de departamento o silo, se requiere de una mentalidad abierta a la experimentación y de conceptos como el producto mínimo viable (MVP, Minimum Viable Product), el siguiente aspecto es la responsabilidad End-To-End (extremo a extremo), la

responsabilidad de un participante en la creación de un producto o servicio no acaba cuando acaba su participación sino que abarca todas las etapas de la vida del producto, por ejemplo la responsabilidad de la continuidad del producto no es solo de operaciones sino compartida por todos los que participaron en su concepción e implementación, el siguiente aspecto son equipos autónomos y multifuncionales la responsabilidad compartida no puede ser alcanzada por medio de silos o departamentos separados habituales en las organizaciones sino por equipos que cuenten con personas de todas las disciplinas involucradas, se busca que los involucrados se sientan no solo responsables de su trabajo sino del resultado en conjunto, otro aspecto es la mejora continua, se centra en actividades como incrementar el flujo, reducir el desperdicio, inspeccionar y adaptarse para esto se requiere contar con datos que permitan medir el proceso y sus resultados, por último automatizar todo lo que se pueda, la automatización permite eliminar tareas repetitivas liberando a las personas para actividades que generen más valor como la innovación, los beneficios que se obtienen a la larga compensan la inversión requerida para implementar la automatización. Alonso Álvarez (2020).

Automation, la automatización es uno de los aspectos más visible de DevOps y que más impacto genera, cuando automatizamos reemplazamos las tareas manuales generalmente repetitivas por software que las ejecute, entre las tareas a automatizar se tienen las pruebas, la integración, el despliegue o la provisión de infraestructura, entre los beneficios encontramos: la reducción de costes, al automatizar las tareas repetitivas como las pruebas de regresión que ayudan a comprobar la integridad del software ante un cambio, suponen un ahorro en tiempo y costo sin embargo es importante elegir que se automatiza para que el coste obtenido sea menor que al de las tareas manuales; otro beneficio es la calidad, las pruebas de regresión automatizadas pueden realizarse a diario para anticipar la detección de errores y mejorar la calidad del software; el siguiente beneficio es la liberación de recursos, el esfuerzo invertido en las pruebas manuales puede ahora enfocarse más en el valor, en la innovación y experimentación, otro beneficio es que brinda una solución más ingenieril que permite sistematizar y estandarizar los procesos de revisión del software y por último la automatización de la emulación de equipos físicos en elementos virtuales gestionados a través de código gracias al Cloud Computing.

Por otra parte, Mike Cohn (2009) dice que una estrategia de automatización de pruebas efectiva requiere la automatización de pruebas en tres niveles diferentes, como se muestra en la pirámide de automatización de pruebas.

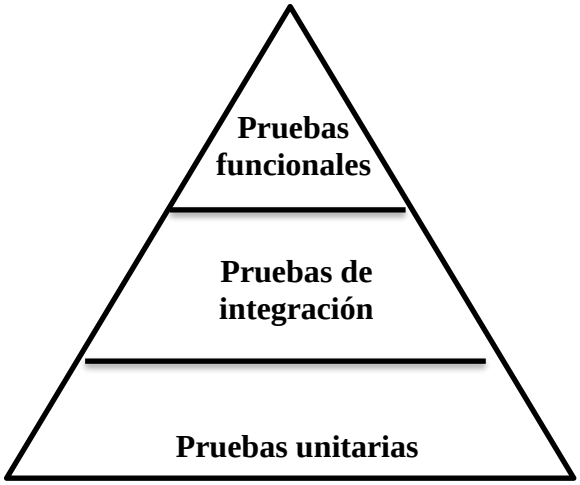

Figura 01: Pirámide de automatización de pruebas
Fuente: https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test automation-pyramid

Las pruebas unitarias deben ser la base de una sólida estrategia de automatización de pruebas y, como tal, representan la parte más grande de la pirámide. Las pruebas automatizadas de la interfaz de usuario se ubican en la parte superior de la pirámide de automatización de pruebas porque son frágiles, costosas y requieren mucho esfuerzo debido a los cambios constantes que sufren las interfaces de cara al usuario.

Por lo general las pruebas automatizadas no incluyen la capa de los servicios, perdiendo la oportunidad de validar esa capa intermedia que existe entre las pruebas unitarias y las de usuario. Centrándose en las pruebas de la interfaz de usuario que resultan costosas y fáciles de romper por algún cambio en las interfaces del usuario.

Asi mismo Mike Wacker (2015) afirma de manera similar que Google sugiere una división de: 70/20/10: 70% pruebas unitarias, 20% pruebas de integración y 10% pruebas funcionales.

Measurement, las medidas son la clave, porque las decisiones deben tomarse sobre hechos, datos, no opiniones. Podríamos medir bien el tiempo de respuesta del sistema, lo que permite saber, por ejemplo, si el último cambio realizado en el sistema ha mejorado el rendimiento o, por el contrario, lo ha degradado.

Por lo tanto, en el lado del desarrollador, es muy importante que proporcionen servicios de monitoreo a bordo en las aplicaciones provistas. A partir de estas medidas, los KPI (indicadores clave de rendimiento) se pueden establecer para responder preguntas importantes, como:

¿Cuántos usuarios se han registrado hoy?

¿Cuáles son los ingresos de hoy?

¿Cuáles son los costos operativos?

¿Cuál es el número de boletos abiertos en Call Center hoy?, etc. Hay muchas plataformas de instrumentación para producir gráficos como Graphite, Grafana.

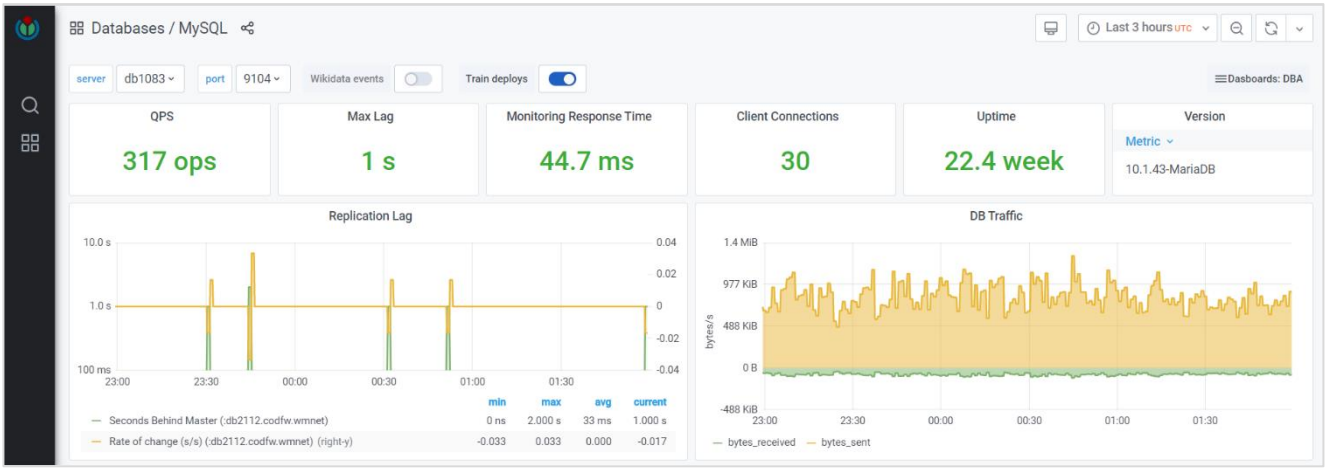

Figura 02: Wikipedia metrics

Fuente: https://grafana.wikimedia.org/d/000000273/mysql?orgId=1

Sharing, compartir el último componente de CAMS, tiene tres componentes: la visibilidad, es lo que permite a todos ver el progreso de otras partes de la organización. Concretamente, la visibilidad permite saber si el trabajo de un equipo puede presentar un problema a otro equipo a través de la retroalimentación temprana. Otro componente la transparencia, es lo que permite a todos trabajar hacia un objetivo común, por eso hicimos lo que hicimos, en la práctica la falta de transparencia puede llevar a una desalineación entre los equipos y llevar a desarrollos inapropiados, por ejemplo. El último componente la transferencia de conocimiento tiene como objetivo, evitar restricciones en la organización, promover la inteligencia colectiva. Para entender mejor que es una restricción, tomemos un ejemplo, a menudo en las organizaciones solo una persona tiene conocimientos técnicos y cuando esta persona se va de vacaciones, todo el equipo se queda atascado. Esto se llama una restricción. Para evitar esto, la solución es compartir el conocimiento entre las personas. Tomemos un ejemplo concreto, el despliegue de una aplicación. Si solo una persona sabe cómo desplegar una aplicación, cuando esta no esté, el equipo ya no podrá desplegar la aplicación. Hay mucho camino para poner en práctica, para el intercambio de conocimientos se pueden adoptar algunas prácticas como los Stand-ups diarios y retrospectivas. En el lado del desarrollo, la documentación en general y el código bien documentado son formas de compartir el conocimiento. A la luz de este modelo, podemos ver que la automatización y la entrega continúa son solo una pequeña parte de DevOps. DevOps no es solo herramientas, es mucho más, es una cultura.

Fases de DevOps

DevOps se divide en 6 fases iterativas, no se debe caer en el error de pensar en el ciclo de desarrollo en cascada, las cuales detallo a continuación:

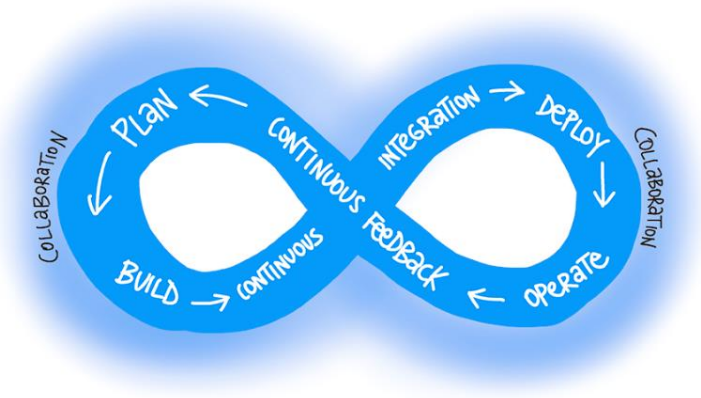

Figura 03: Fases de DevOps

En la fase de planificación, haciendo uso de Scrum se obtienen a través de historias de usuarios los requerimientos y actividades a realizar para implementar las fases de integración, entrega, despliegue y monitoreo continúo.

La fase de integración continúa alienta a los desarrolladores a compartir sus pruebas de código fusionando sus cambios en un repositorio de control de versiones compartido después de la finalización de pequeñas tareas. Mediante un script se activa un flujo de compilación automático para descargar el código más reciente del repositorio compartido, compilar, probar y validar la rama principal completa.

En la fase de entrega continúa, se automatiza todo el proceso de publicación de software. Cada revisión efectuada activa un proceso automatizado que ejecuta las pruebas de caja negra y almacena en un repositorio una versión del aplicativo. Cuando se implementa de manera adecuada la entrega continúa, los desarrolladores dispondrán siempre de un artefacto listo para ser desplegado en producción que se ha sometido a un proceso de pruebas estandarizado.

En la fase de despliegue continuo, el código es desplegado en el entorno de producción. Aquí nos aseguramos de que el código esté implementado correctamente en todos los servidores. Pallavi Poojary (2016).

DevOps promueve la automatización de los despliegues por medio de herramientas y scripts, con el objetivo de que todo el proceso se resuelva con un botón de aprobación o, idealmente, la activación de una característica. Entre cada entorno de despliegue, hay que tener muy en cuenta la administración del contexto (crear, configurar y destruir entornos), realizar y superar las pruebas específicas de cada uno (como pueden ser pruebas de rendimiento, resiliencia, funcionales, de seguridad o de UX); y administrar la gestión de la configuración de acuerdo con las complejas necesidades de los diferentes contextos de despliegue. Lo más crítico y dificultoso en esta fase, más que conocida y adoptada en el entorno IT, es la llegada del concepto Cloud con sus capacidades de Infraestructura como código, que fuerza un cambio en el paradigma de la gestión de la infraestructura. Que pasa de una gestión de recursos finitos a una gestión basada en una optimización permanente de costes. Juan Quijano (2018).

Infraestructura como código (Iac), es la gestión de la infraestructura (redes, máquinas virtuales, equilibradores de carga y topología de conexión), es una práctica clave de DevOps y se usa junto con la entrega continúa, los equipos realizan cambios en la descripción del entorno y la versión del modelo de configuración, que generalmente se encuentra en formatos de código bien documentados, como yaml. El proceso de lanzamiento ejecuta el modelo para configurar los entornos de destino. Si el equipo necesita hacer cambios, editan la fuente, no el objetivo. Infraestructura como código permite a los equipos de DevOps probar aplicaciones en entornos similares a la producción al principio del ciclo de desarrollo.

En la Fase de Monitoreo continuo, última fase de un proceso DevOps, es dónde se definirán las medidas que estaré monitorizando para controlar el estado de salud de las aplicaciones y su infraestructura, siendo esto el histórico de las mediciones durante un periodo de tiempo, que muestran la evolución del sistema. Juan Quijano (2018)

Uno de los objetivos del monitoreo es lograr una alta disponibilidad al minimizar el tiempo de detección y el tiempo de mitigación (TTD, TTM). En otras palabras, tan pronto como surjan problemas de rendimiento y otros problemas, los datos en caliente del diagnóstico sobre los problemas se transmitirán a los equipos de desarrollo a través de un monitoreo automatizado, eso es TTD. Los equipos de DevOps actúan sobre la información para mitigar los problemas lo más rápido posible, de modo que los usuarios ya no se vean afectados, eso es TTM. Los tiempos de resolución se miden y los equipos trabajan para mejorar con el tiempo. Después de la mitigación, los equipos trabajan en cómo solucionar problemas en la causa raíz para que no vuelvan a ocurrir. Ese tiempo se mide como TTR, tiempo de solución. Un segundo objetivo de la supervisión es habilitar el aprendizaje validado mediante el seguimiento del uso. El concepto central de aprendizaje validado es que cada implementación es una oportunidad para seguir los resultados experimentales que respaldan o disminuyen las hipótesis que llevaron al despliegue. Sam Guckenheimer (2018).

Servidor de integración continua: Jenkins

Es una aplicación gratuita open-source creado por Kohsuke Kawaguchi, basado en el proyecto Hudson. Jenkins se basa en la programación de tareas, cada tarea hace uso de plugins que hacen posible la integración de Jenkins con GIT, Docker, phpUnit, etc.

Las tareas por programar van desde descargar el código fuente del sistema de control de versiones que manejemos como GIT, ejecutar las pruebas unitarias y si todo va bien el equipo contará con un build que podrá ser guardado en un repositorio de artefactos, mediante una tarea programada. Con Jenkins se puede programar tareas desde el inicio hasta el despliegue de la versión final del software en producción. Para integrar las tareas en un flujo automatizado se debe construir un pipeline en Jenkins, escrito en Groovy denominado Jenkins File. A través de la ejecución del pipeline podremos visualizar que tarea es la que está causando problemas.

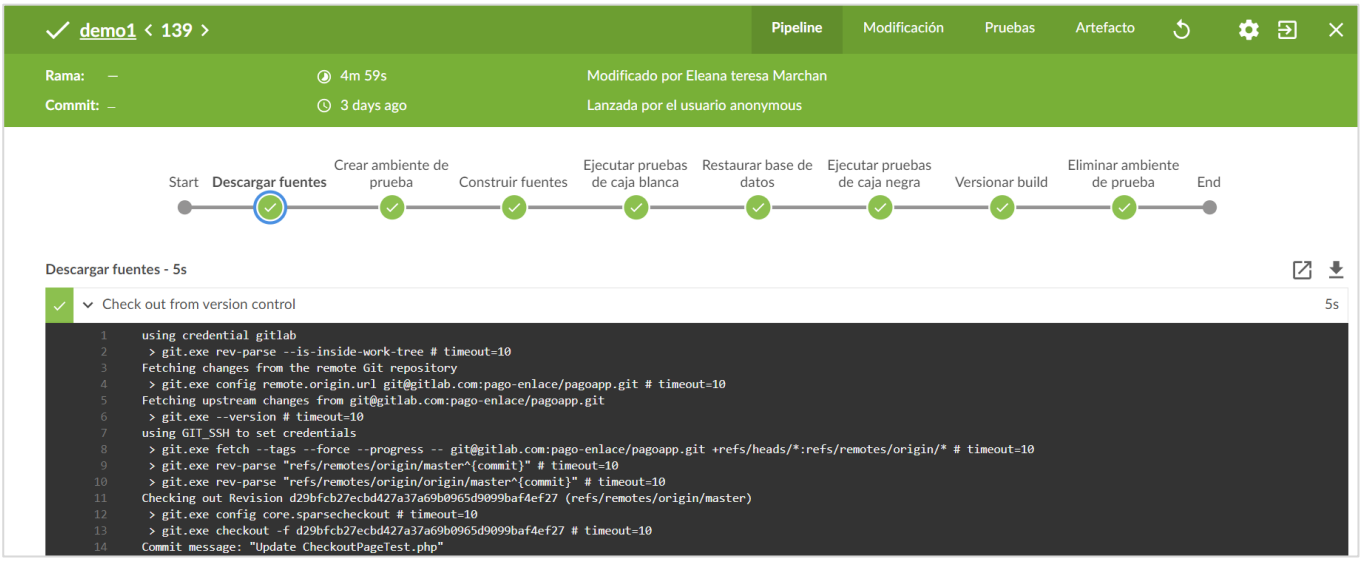

Figura 04: Pipeline integración continúa

Aplicación web

Una aplicación web es un programa informático que utiliza navegadores web y tecnología web para realizar tareas a través de Internet, las aplicaciones web usan una combinación de scripts del lado del servidor (PHP y ASP) para manejar el almacenamiento y recuperación de información, y scripts del lado del cliente (JavaScript y HTML) para presentar información a los usuarios. Ndegwa (2016)

Calidad de software

(Ian Sommerville 2011) afirma que la calidad de software es un proceso subjetivo en el que el equipo de gestión de calidad tiene que usar su juicio para decidir si se logró un nivel aceptable de calidad, debe considerar si el software se ajusta o no a su propósito pretendido. La calidad del software no solo se trata de si la funcionalidad de este se implementó correctamente, si no también depende de atributos no funcionales del sistema entre ellos la confiabilidad, usabilidad, eficiencia y mantenibilidad del software. Por lo general se considera que los atributos de confiabilidad son los atributos de calidad más importantes de un sistema. También es significativo el rendimiento del software. Los usuarios rechazaran el software que sea demasiado lento. La calidad del software es la suma de: calidad de procesos + calidad de producto + calidad de las personas.

Pruebas de software

(Ian Sommerville 2011) afirma que las pruebas intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el software, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o la información de atributos no funcionales del programa. El proceso tiende dos metas:

- Demostrar al desarrollador y al cliente que el software cumple con los requerimientos.
- Encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación.

Las pruebas se consideran parte de un proceso más amplio de verificación y validación del software. Aunque ambas no son lo mismo, se confunden con frecuencia. La finalidad de la verificación es comprobar que el software cumpla con su funcionalidad y con los requerimientos no funcionales establecidos. La meta de la validación es un proceso más general, la validación es esencial pues las especificaciones de requerimientos no siempre reflejan los deseos o las necesidades reales de los clientes y usuarios del sistema. Las pruebas se realizan en tres niveles de granulación:

Pruebas de unidad, donde se ponen a prueba unidades de programa o clases de objetos individuales. Las pruebas de unidad deben enfocarse en comprobar la funcionalidad de objetos o métodos. Pruebas del componente, donde muchas unidades individuales se integran para crear componentes compuestos. La prueba de componentes debe enfocarse en probar interfaces del componente. Pruebas del sistema, donde algunos o todos los componentes en un sistema se integran y el sistema se prueba como un todo. Las pruebas del sistema deben enfocarse en poner a prueba las interacciones de los componentes.

Scrum

Es un proceso de gestión que reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. La gerencia y los equipos de Scrum trabajan juntos alrededor de requisitos y tecnologías para entregar productos funcionando de manera incremental usando el empirismo. Es un marco de trabajo simple que promueve

la colaboración en los equipos para lograr desarrollar productos complejos. (Joel Francia 2017).

La investigación tiene un alcance de carácter descriptivo, por lo que no es necesario plantear una hipótesis debido a que no intenta correlacionar o explicar casualidad de variables y el objetivo a alcanzar está claro. Por tal razón se considera una hipótesis implícita.

Para la ejecución del presente proyecto, me he trazado como objetivo general; Implementar un flujo de trabajo automatizado para el control de calidad de aplicaciones web bajo el enfoque DevOps para la empresa SingLabz Solutions S.A.C; y como objetivos específicos: analizar el proceso actual del control de calidad de las aplicaciones web desarrolladas en la empresa SingLabz Solutions S.A.C., para determinar las pruebas a implementar, métricas de calidad de software y herramientas; otro objetivo específico es diseñar el flujo de trabajo automatizado bajo el enfoque DevOps para obtener los requerimientos, tareas y herramientas que permitan automatizar el control de calidad de las aplicaciones web; el último objetivo específico es implementar un flujo de trabajo automatizado para el control de la calidad de aplicaciones web en la empresa SingLabz Solutions S.A.C. utilizando Docker, Git, Jenkins, Selenium.

2. Metodología

El presente proyecto de investigación es de carácter descriptivo; porque la recopilación de datos obtenidos por instrumentos de investigación nos ha permitido observar, conocer y describir la situación en la que se encontraba la empresa y sus requerimientos para el diseño e implementación del flujo de trabajo de automatizado para el control de calidad de aplicaciones web.

El diseño de la investigación es no experimental de corte transversal por que los datos fueron tomados en una sola vez utilizando los instrumentos de recolección de datos. El proyecto de acuerdo con la orientación es de tipo tecnológica porque se aplicaron los procesos correspondientes a un sistema informativo que se orientó a la solución de la problemática percibida, se utilizaron los conocimientos obtenidos en las investigaciones y en la práctica.

Por otro lado, la población involucrada para la presente investigación fueron los trabajadores de la Empresa SingLabz Solutions S.A.C, que son en cantidad de 15 personas entre desarrolladores (5), de calidad (4), Base de datos (3) y operaciones TI (3) mientras que la muestra fue tomada de forma intencional y estuvo representada por los desarrolladores, testers directamente que están determinados por un numero de doce (12) personas.

Tabla 01: Técnicas e instrumentos de recolección de datos

Técnicas	Instrumentos	Ámbito
Observación	Guía de Observación	A los procesos de desarrollo y pruebas de software en la empresa SingLabz Solutions.
Encuesta	Cuestionario	Preguntas a personal que labora en la empresa y desarrollan los aplicativos web.
Análisis Documentario	Guía de Registro	Texto, tesis, revistas y estudios previos

3. Resultados

Para el primer objetivo específico de la investigación que comprende el análisis del proceso actual para el control de calidad de las aplicaciones web, se verificó los siguientes aspectos:

Uso de software de control de versiones

Se verificó que el equipo hacia uso de herramientas ftp como FileZilla, se recomienda usar una herramienta de control de versiones de código y que cada desarrollador trabaje sus cambios en una rama aparte de la rama master, el objetivo es que la rama master cuente con una versión estable del código fuente, lo que a su vez facilitará la integración continúa. Asi mismo el equipo se compromete a realizar integraciones de sus avances a la rama principal de manera regular.

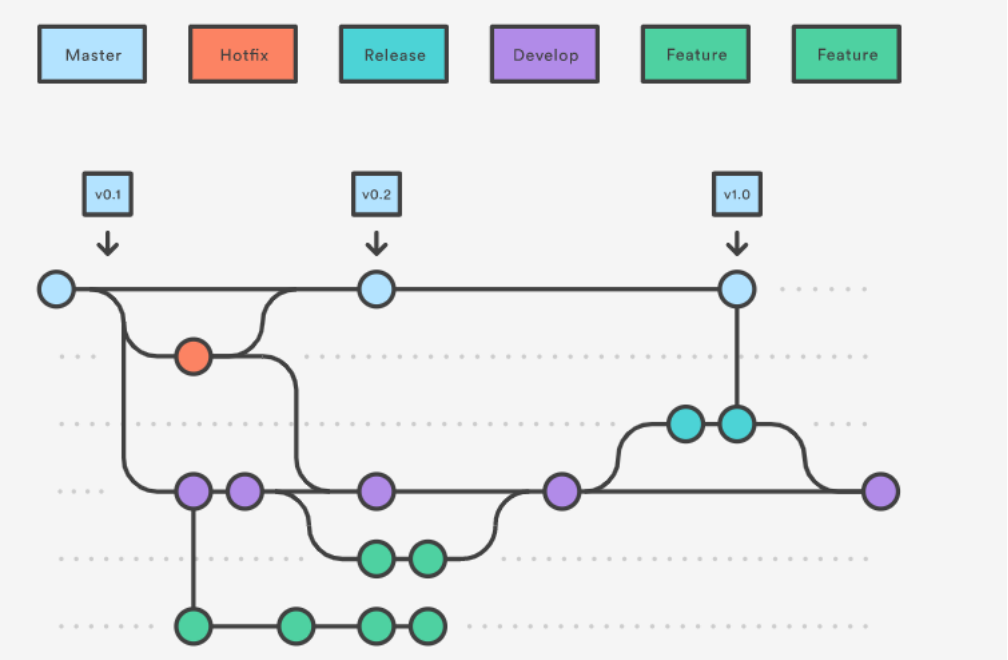

Figura 05: Gitflow
Fuente: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Código fuente cubierto por pruebas unitarias

Se encontró que el código fuente no cuenta con pruebas unitarias que permitan validar los métodos que componen el proyecto.

Resultados:

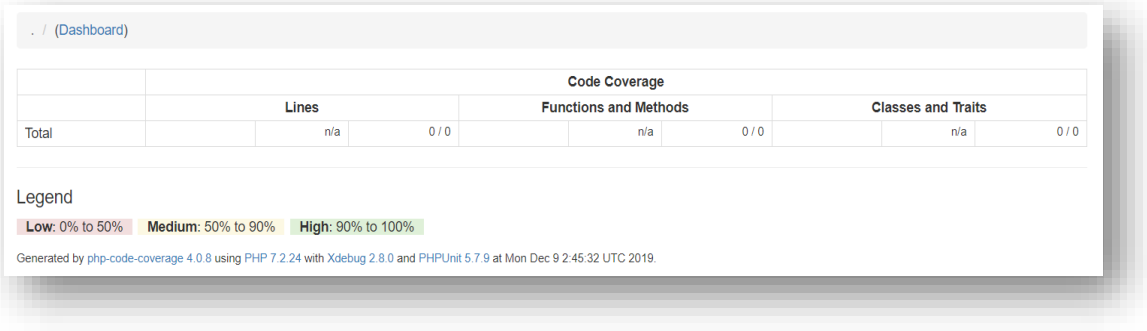

Figura 06: Reporte de porcentaje de código coberturado por pruebas unitarias

Código con alto grado de acoplamiento

Se encontró inconvenientes para escribir pruebas unitarias debido a que el código fuente contaba con un alto grado de acoplamiento, no permitía escribir las pruebas unitarias. Ante esto se tuvo que refactorizar el código fuente para probar los módulos de manera aislada.

Características de ambientes de desarrollo y QA, diferentes al ambiente de producción

Se verificó que el desarrollo y las pruebas se realizaban en un ambiente que no contaba con las características del ambiente de producción, donde finalmente se iba a desplegar el aplicativo:

Tabla 02: Comparación de ambientes de desarrollo

Servidor de pruebas			**Servidor de producción**	
Herramientas	Versión		Herramientas	Versión
Php	5.7		php	7.2
Mysql	5.0.27		Mysql	8.0

Para la implementación del segundo objetivo específico que comprende el diseño del flujo de trabajo automatizado para el control de calidad de aplicaciones web bajo el enfoque DevOps, se aplicó la fase de Planificación de las metodologías DevOps y SCRUM para obtener los requerimientos funcionales y no funcionales del flujo automatizado DevOps a través de las historias de usuarios.

a. Roles

Tabla 03: Roles del equipo SCRUM

Rol	**Responsable**
Product Owner	Robin Ortiz Gutiérrez
Scrum master	Carlos Sing Ramos
Equipo de desarrollo	Carlos Armando Álvarez Chumbiauca Juan Carlos Damián Odar

b. Reunión de planificación

Durante las reuniones de planificación se acordó debido a la importancia en la protección de los datos de las tarjetas de pago de los usuarios finales, automatizar los escenarios de prueba de la historia de usuario realizar pago, a continuación, se listan los escenarios de prueba a automatizar y se presenta el diseño del flujo DevOps.

Tabla 04: Escenarios de prueba

Escenarios de prueba
Tarjeta correcta
Tarjeta perdida
Tarjeta no cuenta con fondos
Operación denegada
Código de verificación incorrecto
Emisor de tarjeta no disponible
Error de procesamiento

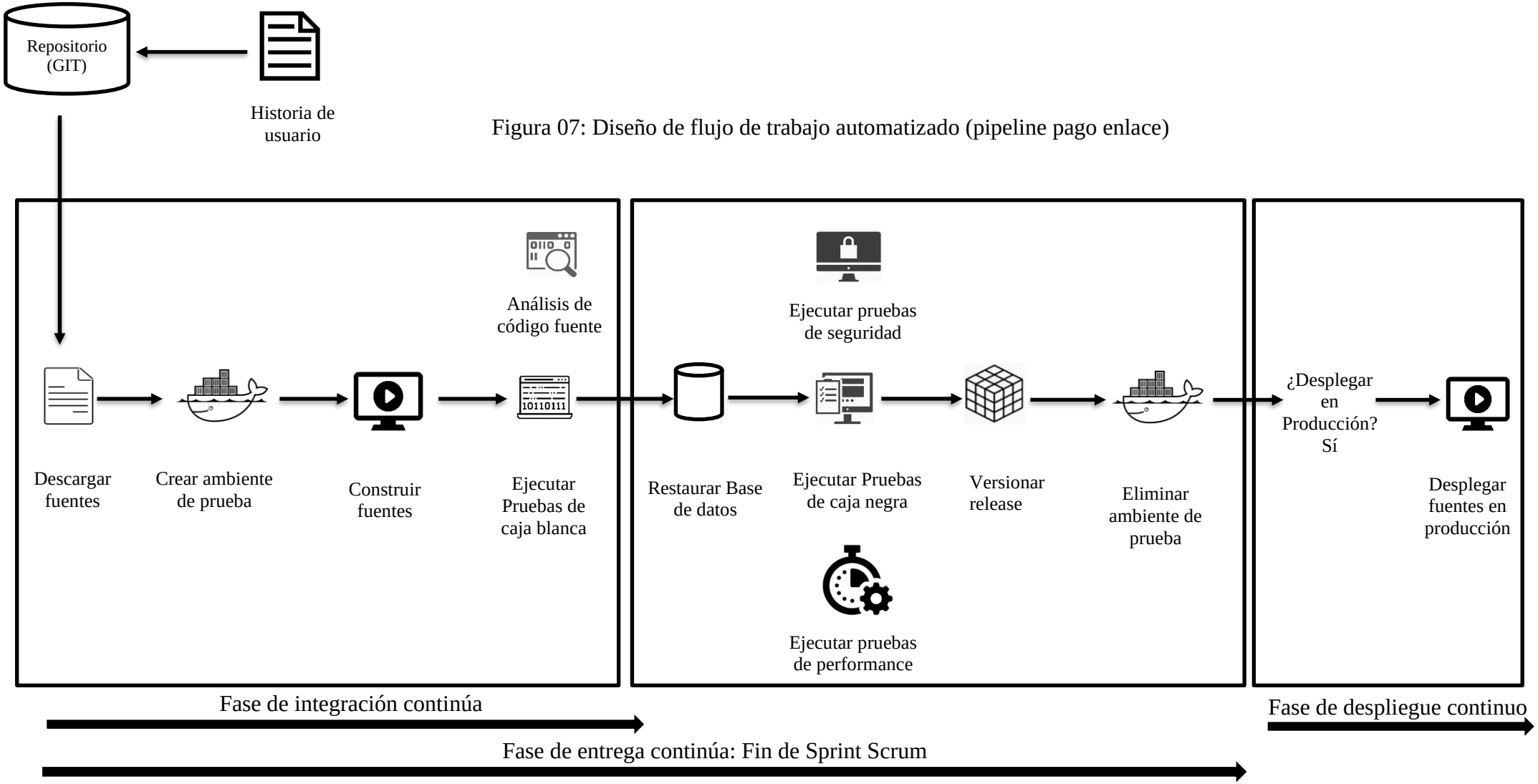

Figura 07: Diseño de flujo de trabajo automatizado (pipeline pago enlace)

Tabla 05: Product Backlog

	Fase DevOps	Historia de usuario	Prioridad	Tiempo estimado	Tareas
1	Integración continúa	Descargar fuentes			• Crear stage denominado: Descargar fuentes. • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
2	Integración continúa	Crear ambiente de prueba			• Crear stage denominado: Crear ambiente de prueba. • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
3	Integración continúa	Construir fuentes			• Crear stage denominado: Construir fuentes. • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
4	Integración continúa	Ejecutar pruebas de caja blanca			• Crear stage denominado “Ejecutar pruebas de caja blanca”. • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
5	Entrega continúa	Restaurar base de datos			• Crear stage para ejecutar scripts de base de datos que permita la restauración de una copia de base de datos. • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
6	Entrega continúa	Ejecutar pruebas de caja negra.			• Crear stage denominado “Ejecutar pruebas de caja negra”. • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
7	Entrega continúa	Versionar y publicar release en repositorio			• Crear stage denominada “versionado de release”. • Ejecutar pipeline.

			• Verificar resultados en Dashboard de Jenkins.
8	Entrega continúa	Eliminar ambiente de prueba	• Crear stage denominada "Eliminar ambiente de prueba". • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins.
9	Despliegue continuo	Desplegar release en producción.	• Crear etapa denominada "desplegar fuentes en producción" • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins. • Verificar el ingreso a página de inicio de aplicativo desplegado.
10	Monitoreo continuo	Monitoreo continuo.	• Crear etapa denominada "ejecutar Monitoreo y publicar reporte". • Ejecutar pipeline. • Verificar resultados en Dashboard de Jenkins. • Verificar resultados de reporte de Monitoreo.

c. Historias de usuario

Tabla 06: Descargar fuentes

Historia de Usuario	
Numero: 1	**Usuario:** Desarrollador
Nombre historia: Descargar fuentes	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Media
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Generar un stage en Jenkins que permita automatizar la descarga de las fuentes desde gitlab.com, hacia la carpeta de trabajo del pipeline creado en Jenkins.	
Precondición: 1.- Las fuentes del proyecto deben estar subidas en un gestor de versionado de código.	

2.- Los desarrolladores deben mantener actualizada la rama master.
Postcondición: Las fuentes del proyecto se encuentran disponibles en Jenkins.
Observaciones: Es necesario que el equipo de desarrollo se acostumbre a desplegar continuamente en un sistema de control de versiones.

Tabla 07: Crear ambiente de prueba

Historia de Usuario	
Numero: 2	**Usuario:** Desarrollador
Nombre historia: Crear ambiente de prueba	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 2	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá generar un stage en Jenkins que permita automatizar la creación de 3 ambientes de prueba que permitan desplegar las fuentes del aplicativo, base de datos y un contenedor con Selenium Grid que permita gestionar los navegadores web como Firefox y Chrome, a utilizar en las pruebas de caja negra.	
Precondición: 1.- El Stage que automatiza la descarga de las fuentes debe haberse ejecutado exitosamente. 2.- Se debe contar con un script de base de datos que permita crear una base de datos. 3.- En el archivo Docker file se debe especificar la versión de la imagen de php que se desea descargar, además de especificar las librerías necesarias a instalar en los contenedores de prueba: - Xdebug - Mbstring - Zip - Gd - pdo pdo_mysql 3.- El archivo Docker composer debe contar con:	

- La configuración de las credenciales de acceso y el puerto de conexión a la base de datos. - La imagen de MYSQL a descargar. - El puerto y la imagen de los navegadores web en los que se desea probar.
Postcondición: Los ambientes de prueba han sido creados y se encuentran disponibles, se puede comprobar ejecutando el comando docker ps por consola.
Observaciones:

Tabla 08: Construir fuentes

Historia de Usuario	
Numero: 3	**Usuario:** Desarrollador
Nombre historia: Construir fuentes	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá implementar un stage en Jenkins que permita automatizar la descarga e instalación de las librerías que necesita el aplicativo para funcionar.	
Precondición: 1.- El stage que automatiza la creación de los ambientes de prueba debe haberse ejecutado exitosamente. 2.- Las librerías serán tomadas de lo configurado en el archivo composer.json en el parámetro require: - php: "^5.6 || ^7.0, - zendframework/zend-component-installer": "^1.0 || ^2.1, - zendframework/zend-mvc": "^3.1.1, - zfcampus/zf-development-mode": "^3.2, - doctrine/doctrine-orm-module": "^2.1, - zendframework/zend-mvc-console": "^1.2, - culqi/culqi-php": "^1.3, - zendframework/zend-session": "^2.9, - zendframework/zend-validator": "^2.12,	

- zendframework/zend-mail": "^2.10, - php-webdriver/webdriver": "^1.7 3.- y en el parámetro require-dev: - phpunit/phpunit": "^8.5, - zendframework/zend-test": "^3.3, - phpoffice/phpspreadsheet": "^1.10, - dawood/phpscreenrecorder": "^1.4, - php-mock/php-mock-phpunit": "^2.5
Postcondición: El aplicativo cuenta con las librerías instaladas y actualizadas necesarias para poder ejecutar las pruebas unitarias y funcionales.
Observaciones:

Tabla 09: Ejecutar pruebas de caja blanca

Historia de Usuario	
Numero: 4	**Usuario:** Desarrollador
Nombre historia: Ejecutar pruebas de caja blanca	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá implementar un stage en Jenkins que permita automatizar la ejecución de las pruebas de caja blanca y generar un reporte de cobertura en html.	
Precondición: 1.- El stage que automatiza la construcción de las fuentes debe haberse ejecutado exitosamente. 2.- Se debe contar con la librería PHPUNIT instalada	
Postcondición: Las pruebas de caja blanca han sido ejecutadas, el reporte de cobertura en formato html ha sido generado y se encuentra disponible.	
Observaciones:	

Tabla 10: Restaurar base de datos

Historia de Usuario	
Numero: 5	**Usuario:** Desarrollador
Nombre historia: Restaurar base de datos	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá implementar un stage en Jenkins que permita automatizar la creación de una base de datos de prueba y la inserción de data de prueba.	
Precondición: 1.- El stage que automatiza la ejecución de las pruebas de caja blanca debe haberse ejecutado exitosamente. 2.- Se debe contar con un ambiente de prueba disponible que tenga MYSQL instalado y configurado.	
Postcondición: Se cuenta con una base de datos de prueba disponible para poder ejecutar las pruebas de caja negra.	
Observaciones:	

Tabla 11: Ejecutar pruebas de caja negra.

Historia de Usuario	
Numero: 6	**Usuario:** Desarrollador
Nombre historia: Ejecutar pruebas de caja negra.	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá implementar un stage en Jenkins que permita automatizar la ejecución de las pruebas de caja negra.	
Precondición:	

1.- El stage que automatiza la restauración de la base de datos debe haberse ejecutado exitosamente. 2.- Contar con una base de datos prueba. 3.- Se debe contar con un ambiente de prueba disponible que cuente con los navegadores web Chrome y Firefox habilitados. 4.- Se deberá contar con un listado de las pruebas a ejecutar en formato excel, este archivo será leído de manera automática y deberá considerar casos de prueba para verificar un resultado exitoso hasta las excepciones que se necesitan controlar.
Postcondición: Las pruebas de caja negra han sido ejecutadas.
Observaciones:

Tabla 12: Versionar y publicar release en repositorio.

Historia de Usuario	
Numero: 7	**Usuario:** Desarrollador
Nombre historia: Versionar y publicar release en repositorio	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá implementar un stage en Jenkins que permita automatizar el versionado y publicación del aplicativo luego de haber pasado de manera exitosa las pruebas de caja blanca y negra.	
Precondición: 1.- El stage que automatiza la ejecución de las pruebas de caja negra debe haberse ejecutado exitosamente.	
Postcondición: El aplicativo ha sido versionado y se encuentra disponible en el sistema de gestión de versiones.	
Observaciones:	

Tabla 13: Eliminar ambiente de prueba

Historia de Usuario	
Numero: 7	**Usuario:** Desarrollador
Nombre historia: Eliminar ambiente de prueba	
Prioridad en el negocio: Media	**Importancia del desarrollo:** Alta
Iteración asignada: 1	**Modulo:** Cliente
Programador responsable: Carlos Álvarez Chumbiauca	
Descripción: Se deberá implementar un stage en Jenkins que permita automatizar la eliminación de los ambientes de prueba creados, con la finalidad de iniciar las pruebas en un ambiente de prueba nuevo.	
Precondición: 1.- El stage que automatiza el versionado y registro de release de aplicativo en gitlab.com debe haberse ejecutado exitosamente.	
Postcondición: Los ambientes de pruebas creados han sido eliminados.	
Observaciones:	

d. Entregas funcionales

El equipo de desarrollo se comprometió a entregar en 2 sprints la implementación de las fases de integración y entrega continua del flujo DevOps:

Tabla 14: Detalle de sprint1

Sprint 1	
Objetivo: Implementar la fase DevOps de Integración continúa	
Fecha de inicio: 13/08/2018 - Fecha de fin: 21/08/2018	
Historias de Usuario	**Tiempo estimado**
Descargar fuentes	1 día
Crear ambiente de prueba	1 día
Construir fuentes	3 días
Ejecutar pruebas de caja blanca.	3 días
Total, de días del sprint	8 días

Tabla 15: Detalle de sprint2

Sprint 2	
Objetivo: Implementar la fase DevOps de Entrega continúa	
Fecha de inicio: 22/08/2018 - Fecha de fin: 30/08/2018	
Historias de Usuario	**Tiempo estimado**
Restaurar base de datos	1 día
Ejecutar pruebas de caja negra	3 días
Versionar release	1 día
Eliminar ambiente de prueba	0.5 día
Total, de días del sprint	5.5 días

Para el desarrollo del tercer objetivo específico de la investigación que comprende la implementación del flujo de trabajo automatizado bajo el enfoque DevOps se desarrollaron las fases de integración y entrega continúa de DevOps, se utilizó java groovy como lenguaje de scripting para construir el pipeline de las etapas del flujo automatizado y como servidor de integración continúa Jenkins, los ambientes de prueba

han sido creados usando Docker, PHP Unit como Framework para la implementación de las pruebas de caja blanca y Selenium para la implementación de las pruebas de caja negra.

Resultados de Sprint1

Implementación de Fase DevOps de integración continúa, en esta fase se descargan las fuentes desde la rama master del proyecto, para ejecutar de manera automatizada pruebas de caja blanca, al finalizar esta fase se contará con una potencial versión del código fuente que puede ser desplegado en producción.

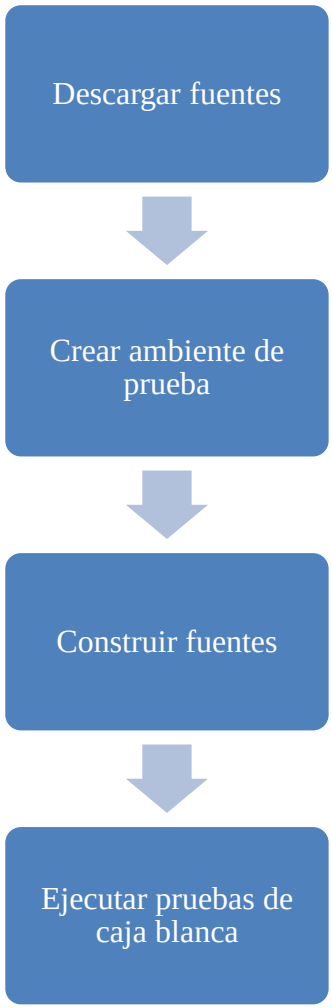

Figura 08: Actividades de fase de integración continúa DevOps

Creación de pipeline: Pago enlace

Stage1: Descargar fuentes

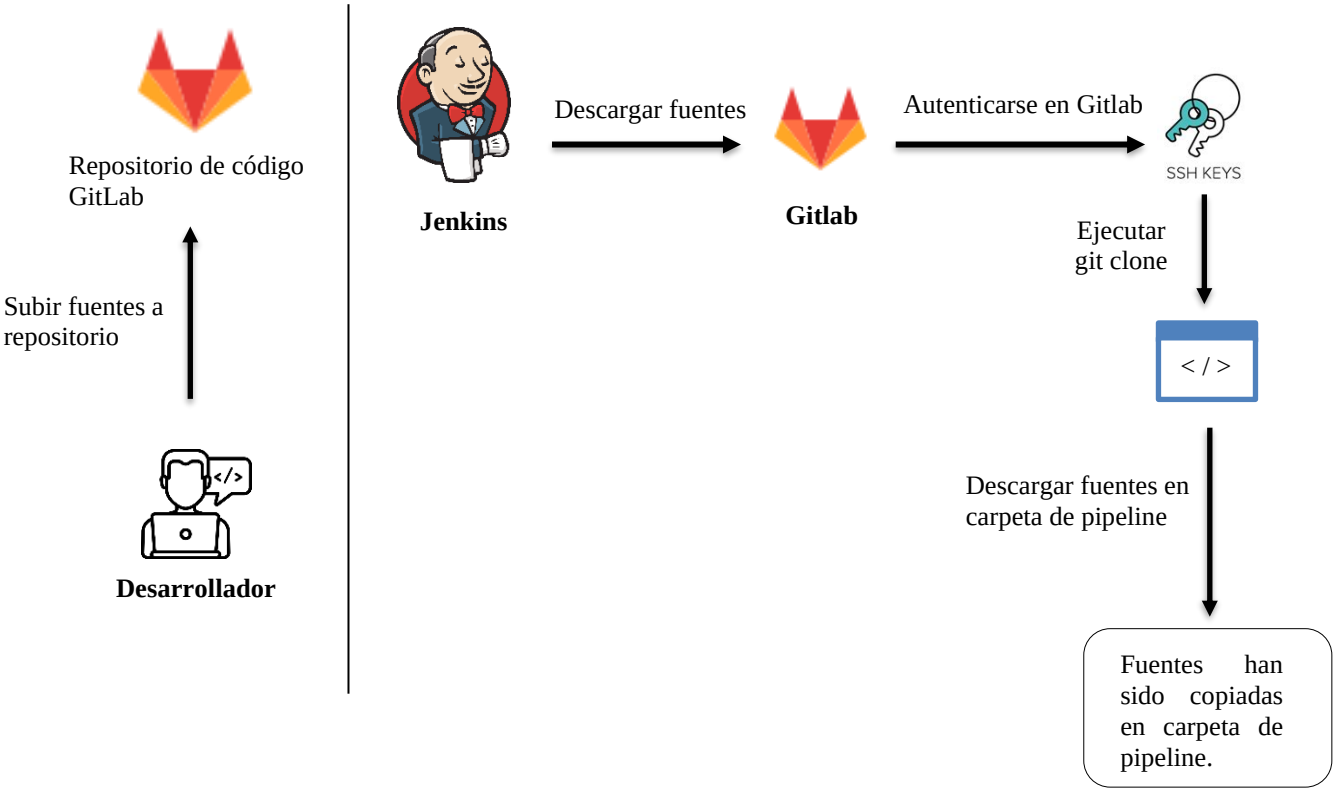

Figura 09: Diagrama de flujo de stage descargar fuentes

Script de automatización:

```
stage('Descargar fuentes'){

    checkout([$class: 'GitSCM',
    branches: [[name: '*/master']],
    doGenerateSubmoduleConfigurations: false,
    extensions: [],
    submoduleCfg: [],
    userRemoteConfigs:
        [[
            credentialsId: 'gitlab',
            url: 'git@gitlab.com:pago-enlace/pagoapp.git'
        ]]
    ])
}
```

Figura 10: Script de automatización de descarga de fuentes

Resultado de ejecución:

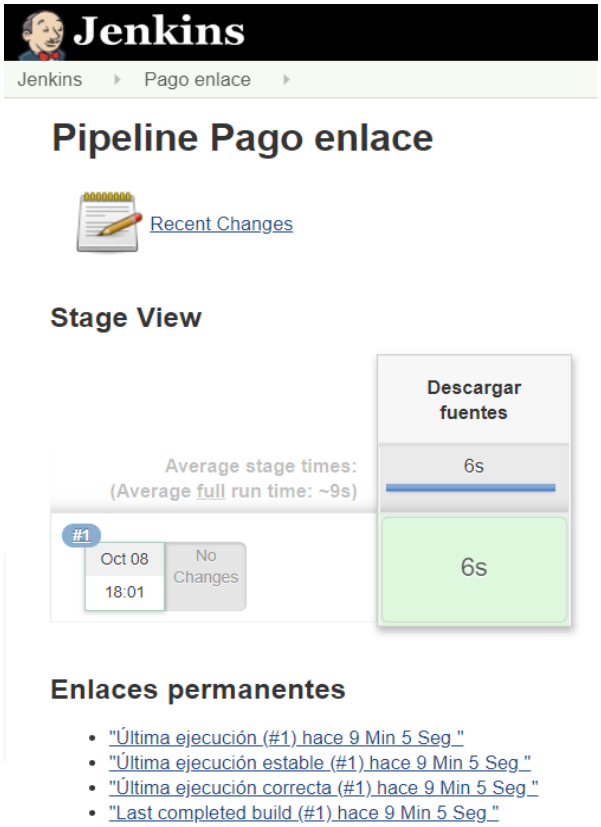

Figura 11: Ejecución de stage descargar fuentes

Descargar fuentes - 14s

Check out from version control

```
using credential gitlab
Cloning the remote Git repository
Cloning repository git@gitlab.com:pago-enlace/pagoapp.git
 > git.exe init C:\Program Files (x86)\Jenkins\workspace\Pipeline Pago Enlace # timeout=10
Fetching upstream changes from git@gitlab.com:pago-enlace/pagoapp.git
 > git.exe --version # timeout=10
using GIT_SSH to set credentials
 > git.exe fetch --tags --force --progress -- git@gitlab.com:pago-enlace/pagoapp.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe config remote.origin.url git@gitlab.com:pago-enlace/pagoapp.git # timeout=10
 > git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe config remote.origin.url git@gitlab.com:pago-enlace/pagoapp.git # timeout=10
Fetching upstream changes from git@gitlab.com:pago-enlace/pagoapp.git
using GIT_SSH to set credentials
 > git.exe fetch --tags --force --progress -- git@gitlab.com:pago-enlace/pagoapp.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
 > git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision d29bfcb27ecbd427a37a69b0965d9099baf4ef27 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f d29bfcb27ecbd427a37a69b0965d9099baf4ef27 # timeout=10
Commit message: "Update CheckoutPageTest.php"
First time build. Skipping changelog.
```

Figura 12: Log de ejecución

Las fuentes del proyecto han sido descargadas correctamente y se encuentran disponibles en la carpeta del pipeline Pago enlace de Jenkins.

Figura 13: Carpeta de proyecto descargado desde GitLab

Stage2: Crear ambientes de prueba

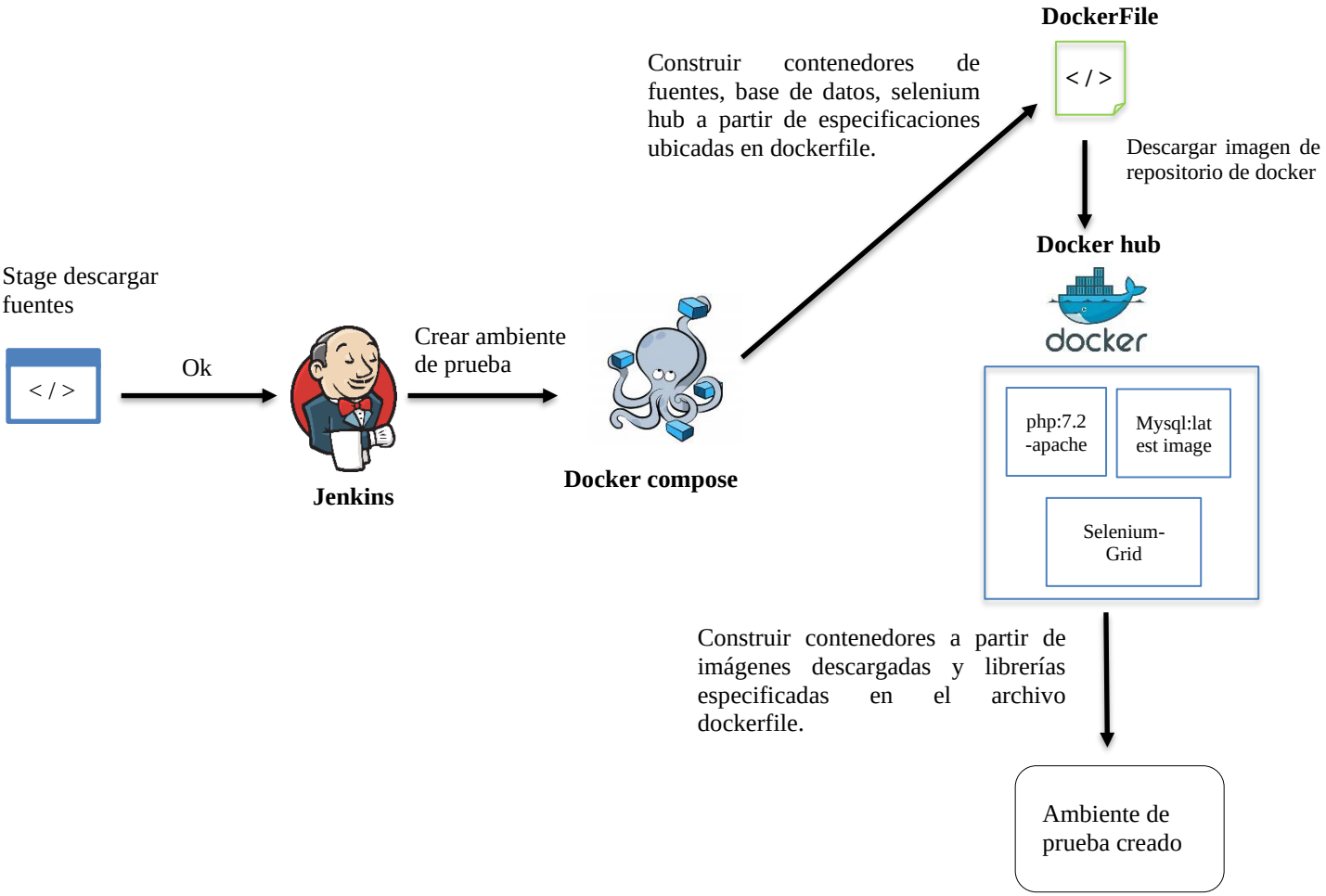

Figura 14: Diagrama de flujo de stage crear ambiente de prueba

Script de automatización:

```
stage('Crear ambiente de prueba'){

    bat label: 'docker', script: 'docker-compose up -d --build'
}
```

Figura 15: Script de automatización de creación de ambiente de prueba

Resultado de la ejecución

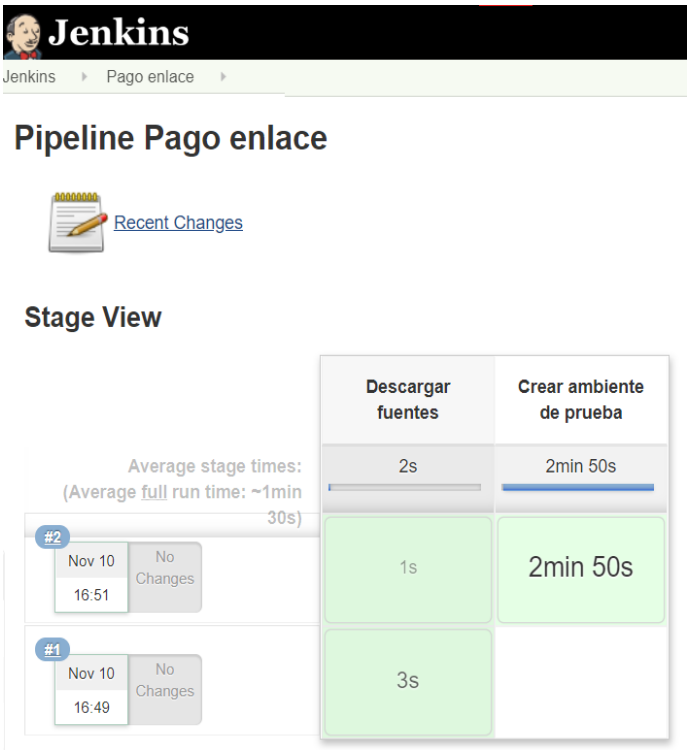

Figura 16: Ejecución de stage de creación de ambiente de prueba

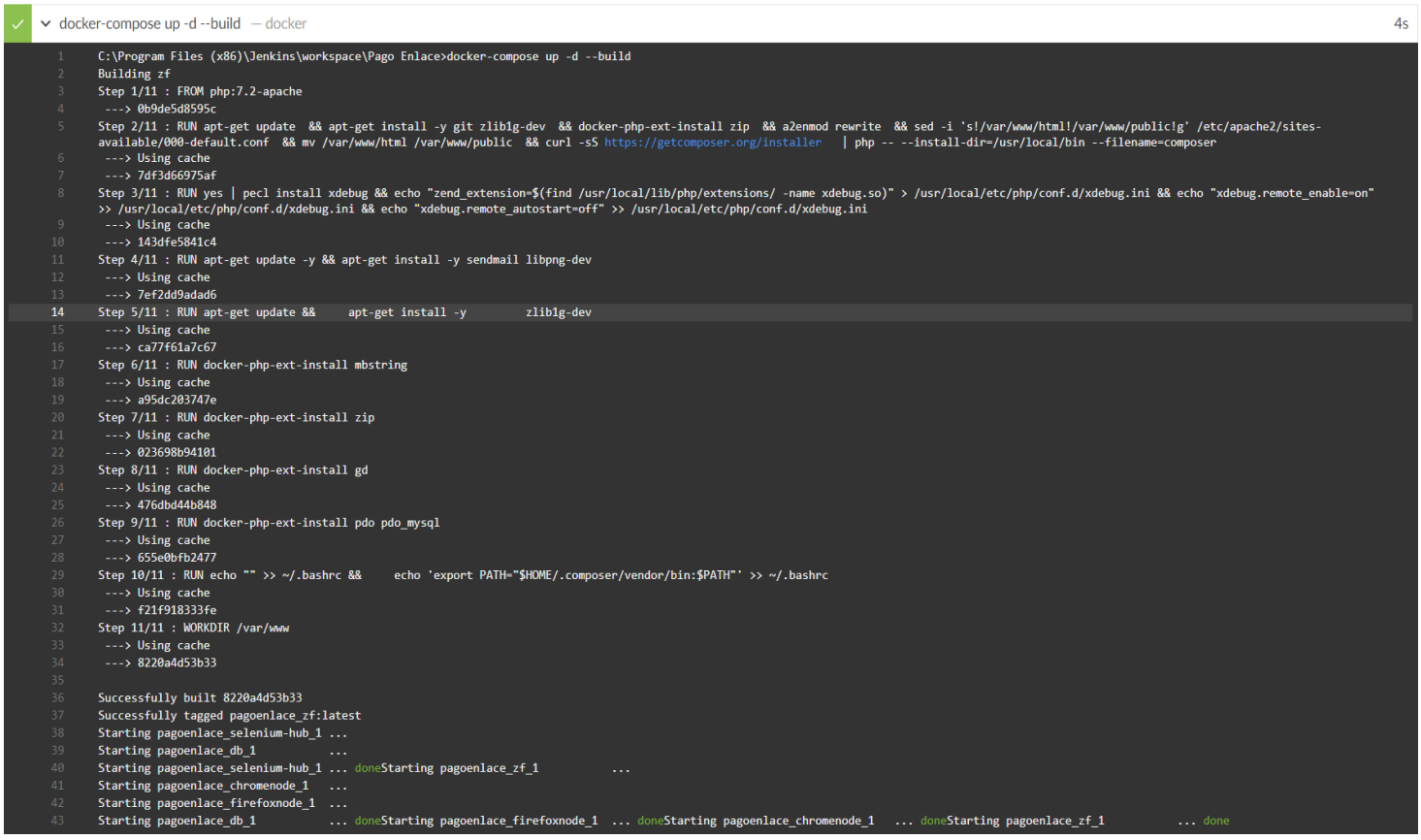

Figura 17: Log de ejecución

La construcción del ambiente de prueba comprende la creación de contenedores donde se alojarán las fuentes, base de datos y Selenium Grid que permita gestionar la automatización de las pruebas funcionales en los navegadores de Firefox y Chrome, la configuración de los contenedores se encuentra especificadas en los archivos docker-compose.yml y dockerfile del proyecto. En estos archivos se configuran las imágenes a utilizar de PHP, MYSQL, Selenium HUB, librerías, el puerto por donde se tendrá acceso al aplicativo, entre otras configuraciones. De esta manera es sencillo configurar un ambiente de prueba que cuente con las mismas características del ambiente de producción.

Docker-compose.yml

```
version: '3.3'

services:
  zf:
    build: .
    ports:
      - "8090:80"
    links:
      - selenium-hub
    volumes:
      - .:/var/www
    environment:
      MYSQL_ROOT_PASSWORD: pass777
      MYSQL_DATABASE: pago_app

  selenium-hub:
    image: selenium/hub
    #    container_name: selenium-hub
    ports:
      - "4444:4444"
    environment:
      GRID_MAX_SESSION: 16
      GRID_BROWSER_TIMEOUT: 3000
      GRID_TIMEOUT: 3000

  chromenode:
    image: selenium/node-chrome-debug
    ports:
      - 5901:5900
    volumes:
      - /dev/shm:/dev/shm
    depends_on:
      - selenium-hub
    environment:
      - HUB_PORT_4444_TCP_ADDR=selenium-hub
      - HUB_PORT_4444_TCP_PORT=4444
      - NODE_MAX_SESSION=4
      - NODE_MAX_INSTANCES=4
```

```
  firefoxnode:
    image: selenium/node-firefox-debug
    ports:
      - 5900:5900
    volumes:
      - /dev/shm:/dev/shm
    depends_on:
      - selenium-hub
    environment:
      - HUB_PORT_4444_TCP_ADDR=selenium-hub
      - HUB_PORT_4444_TCP_PORT=4444
      - NODE_MAX_SESSION=4
      - NODE_MAX_INSTANCES=4

  db:
    image: mysql:5.7.31
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: pass777
      MYSQL_DATABASE: pago_app
    volumes:
      - ./mysql/data:/var/lib/mysql:delegated
      - ./mysql:/etc/mysql/conf.d
    ports:
      - "3306:3306"
volumes:
  pagodb:
```

Figura 18: docker-compose.yml

Dockerfile

```
FROM php:7.2-apache

RUN apt-get update \
 && apt-get install -y git zlib1g-dev \
 && docker-php-ext-install zip \
 && a2enmod rewrite \
 && sed -i 's!/var/www/html!/var/www/public!g'
/etc/apache2/sites-available/000-default.conf \
 && mv /var/www/html /var/www/public \
 && curl -sS https://getcomposer.org/installer \
  | php -- --install-dir=/usr/local/bin --filename=composer

RUN yes | pecl install xdebug \
&& echo "zend_extension=$(find /usr/local/lib/php/extensions/
-name xdebug.so)" > /usr/local/etc/php/conf.d/xdebug.ini \
&& echo "xdebug.remote_enable=on" >>
/usr/local/etc/php/conf.d/xdebug.ini \
&& echo "xdebug.remote_autostart=off" >>
/usr/local/etc/php/conf.d/xdebug.ini

RUN apt-get update -y && apt-get install -y sendmail libpng-
dev
RUN apt-get update && \
```

```
    apt-get install -y \
        zlib1g-dev

RUN docker-php-ext-install mbstring
RUN docker-php-ext-install zip
RUN docker-php-ext-install gd
RUN docker-php-ext-install pdo pdo_mysql

# Export composer vendor path
RUN echo "" >> ~/.bashrc && \
    echo 'export PATH="$HOME/.composer/vendor/bin:$PATH"' >>
~/.bashrc

WORKDIR /var/www
```

Figura 19: dockerfile

Stage3: Construir fuentes

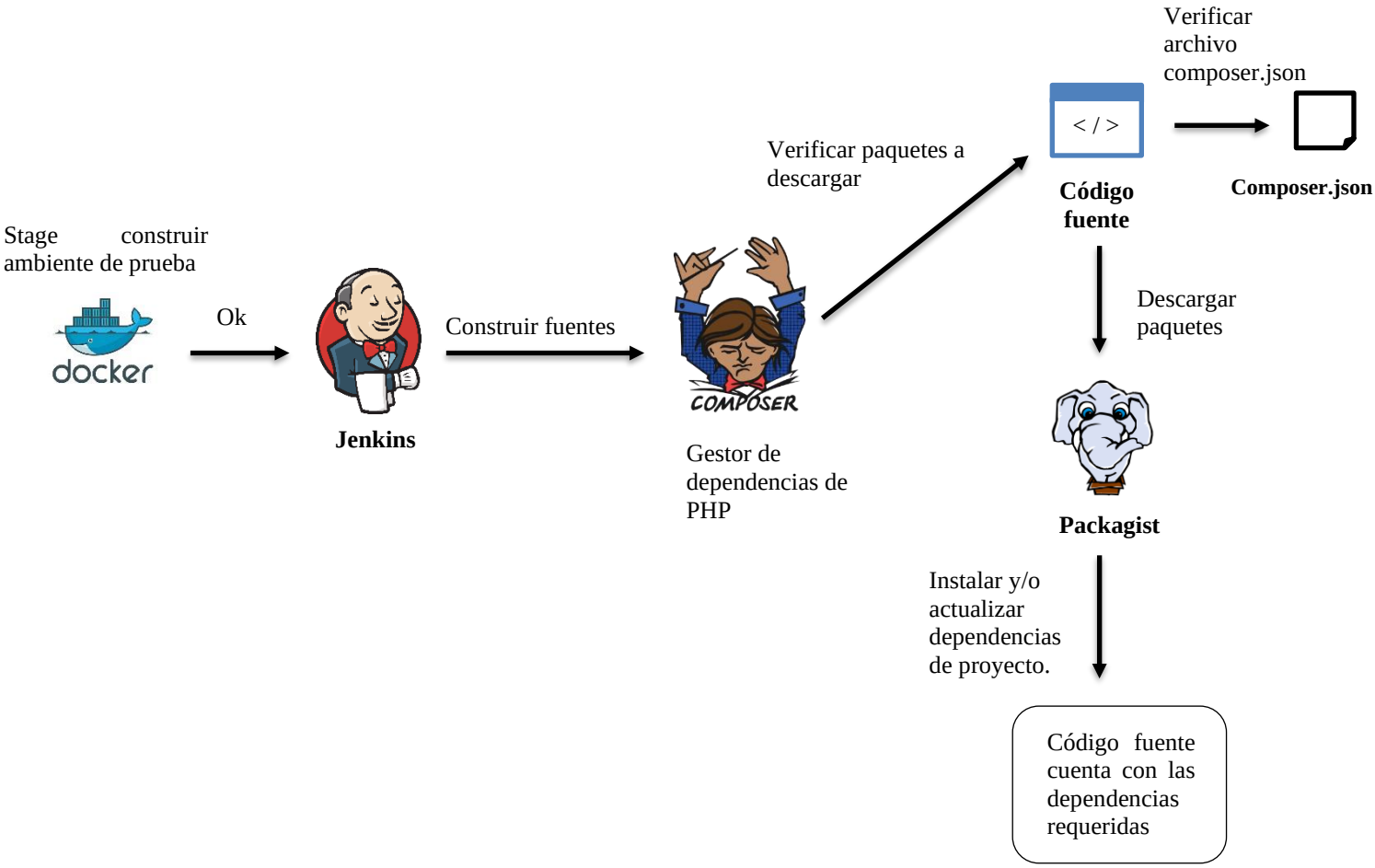

Figura 20: Diagrama de flujo de stage construir fuentes

Script de automatización:

```
stage('Construir fuentes'){
  bat label: 'docker', script: 'docker-compose run zf composer install'
}
```

Figura 21: Script de automatización de construcción de fuentes

Resultado de ejecución

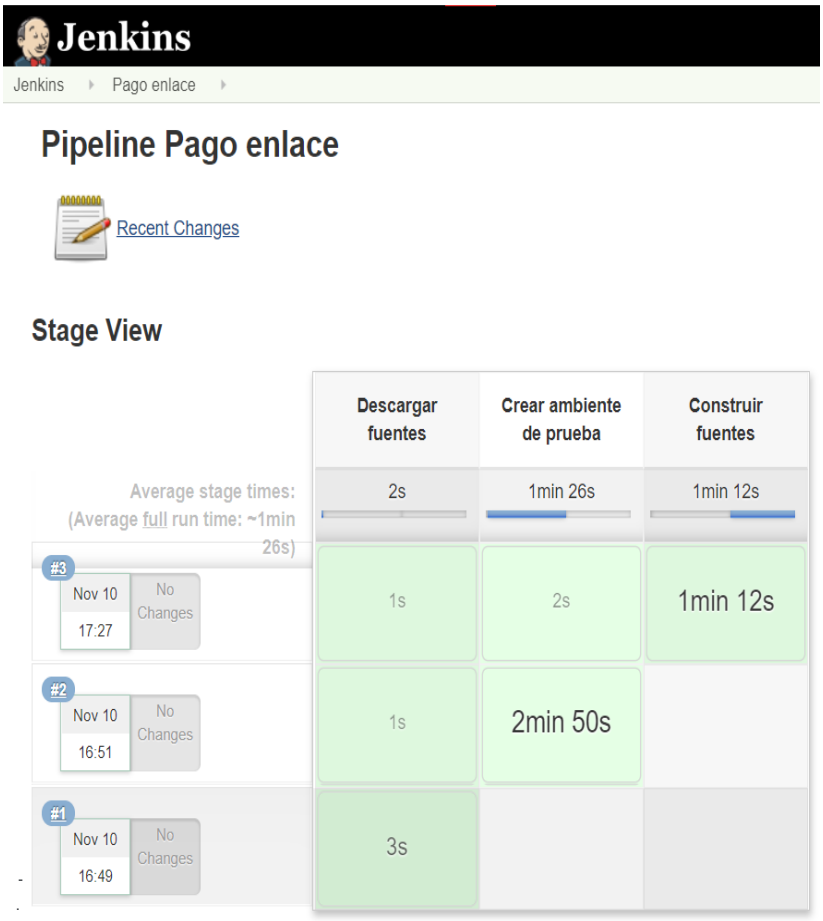

Figura 22: Ejecución de stage de construcción de fuentes

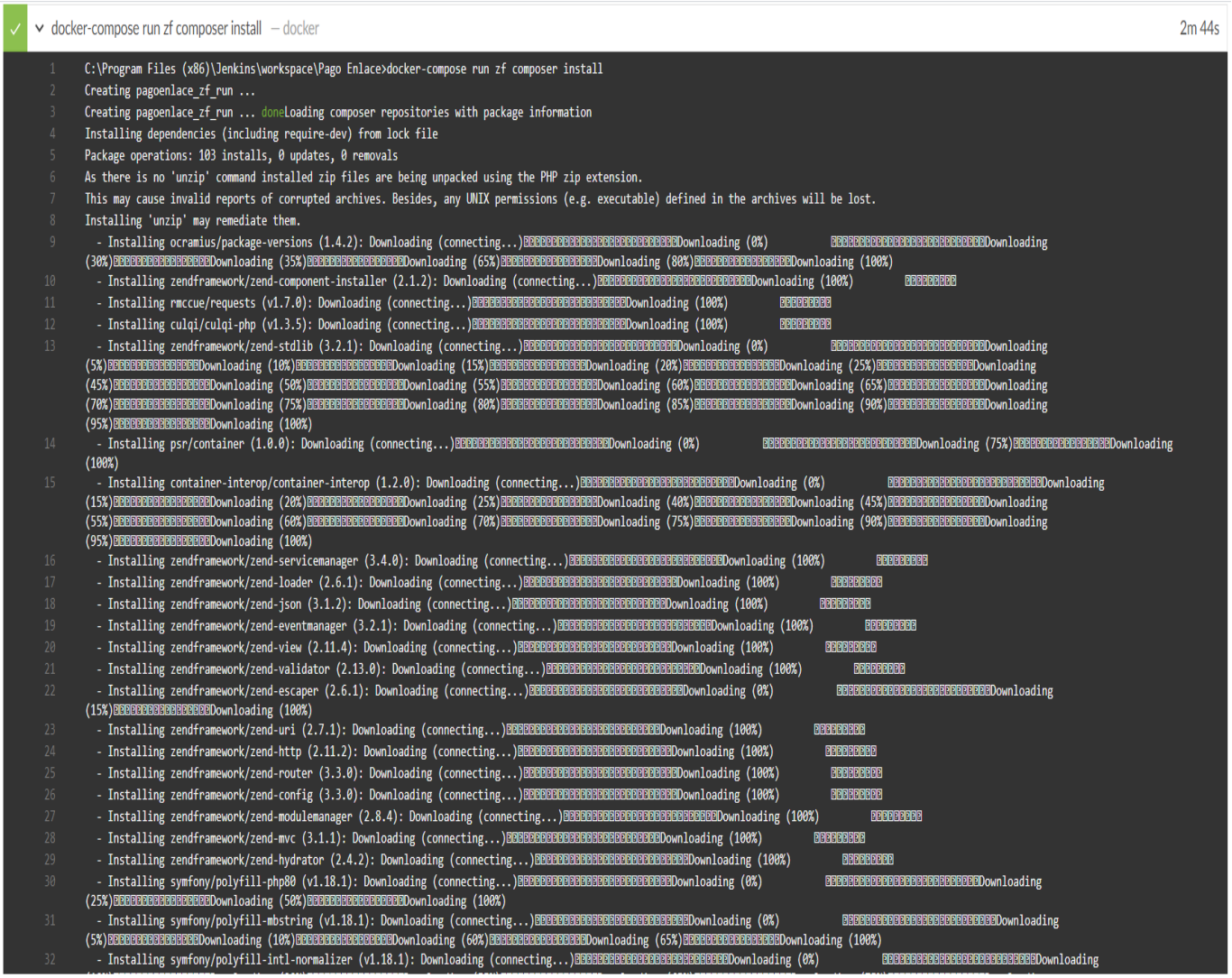

Figura 23: Log de ejecución

Luego de la ejecución de la etapa de construcción de fuentes, el aplicativo cuenta con las librerías necesarias para poder ser probado.

Stage4: ejecutar pruebas de caja blanca

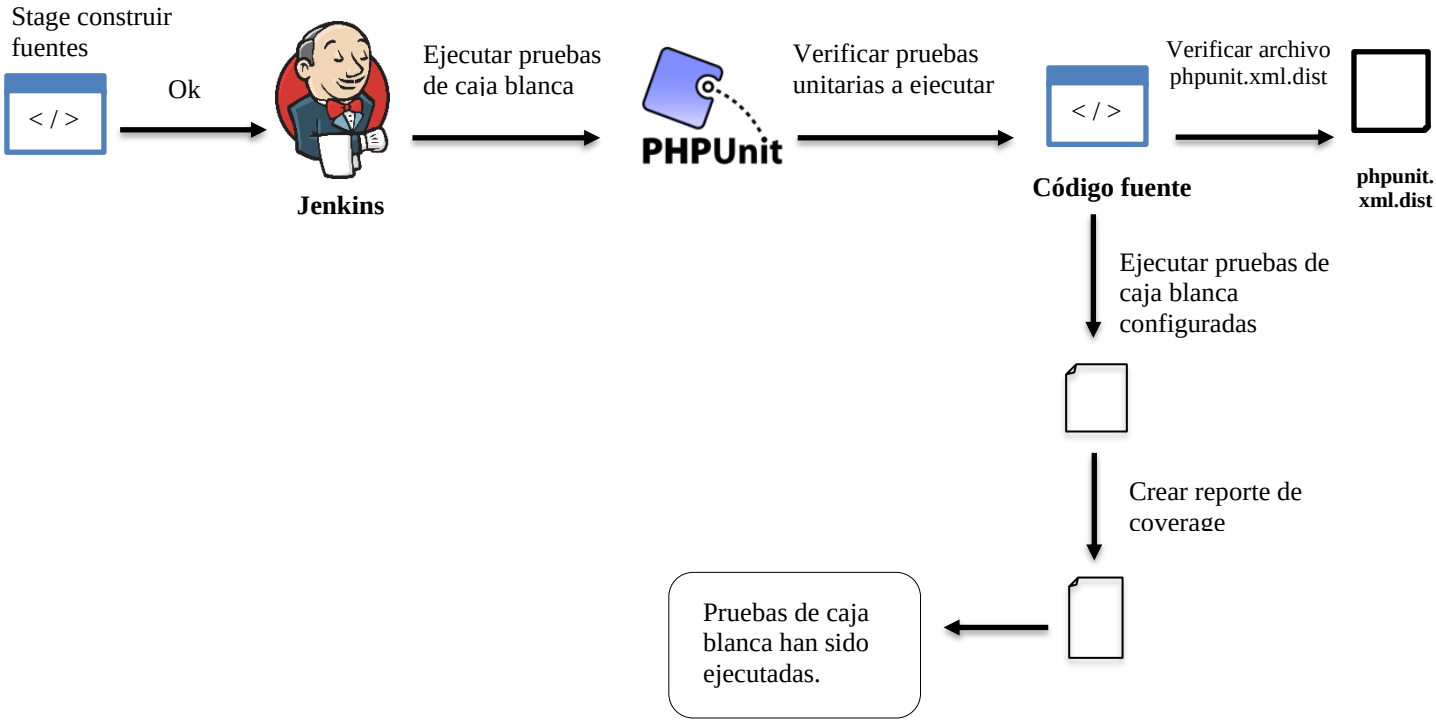

Figura 24: Diagrama de flujo de stage ejecutar pruebas de caja blanca

Script de automatización:

```
stage('Ejecutar pruebas de caja blanca'){

bat label: 'docker', script: 'docker-compose run zf ./vendor/bin/phpunit --configuration=phpunit.xml.dist --coverage-clover data/coverage/coverage.xml'
bat label: 'docker', script: 'docker-compose run zf ./vendor/bin/phpunit --configuration=phpunit.xml.dist --coverage-html data/coverage'

    step([
       $class: 'CloverPublisher',
       cloverReportDir: 'data/coverage',
       cloverReportFileName: 'coverage.xml',
       healthyTarget: [
           methodCoverage: 70,
           conditionalCoverage: 80,
           statementCoverage: 80
       ],
       unhealthyTarget: [
           methodCoverage: 50,
           conditionalCoverage: 50,
           statementCoverage: 50
       ],
       failingTarget: [
           methodCoverage: 0,
           conditionalCoverage: 0,
           statementCoverage: 0
       ]
    ])
}
```

Figura 25: Script de automatización de ejecución de pruebas de caja blanca

Resultado de ejecución

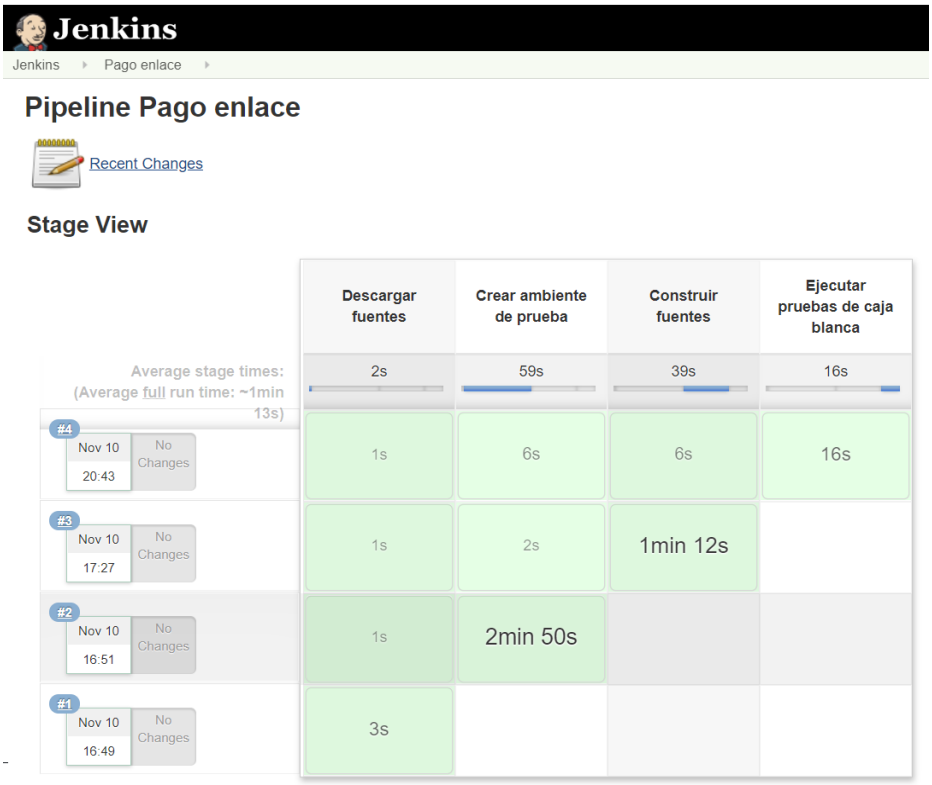

Figura 26: Ejecución de stage de ejecución de pruebas de caja blanca

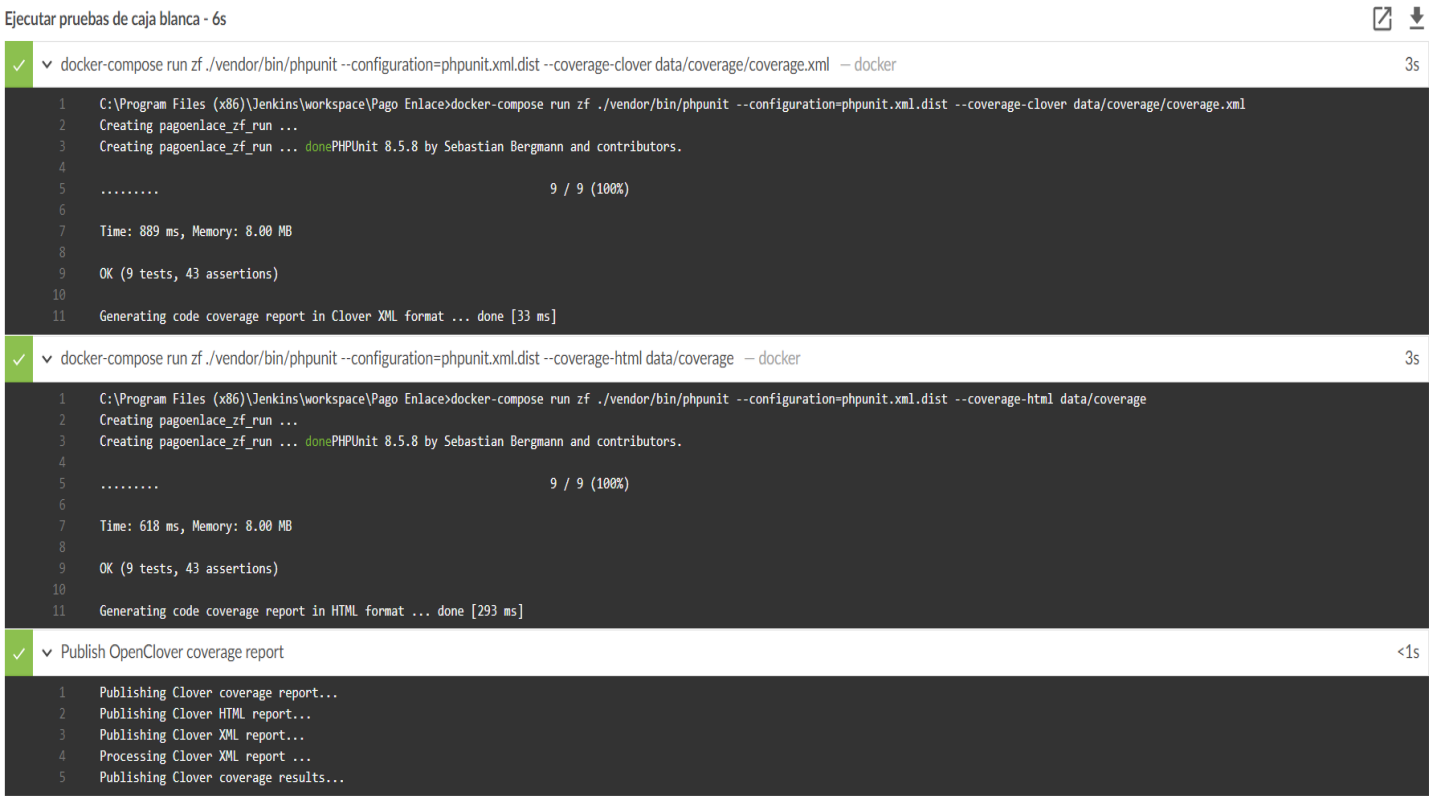

Figura 27: Log de ejecución

Reporte de porcentaje de líneas de código testeadas mediante pruebas unitarias

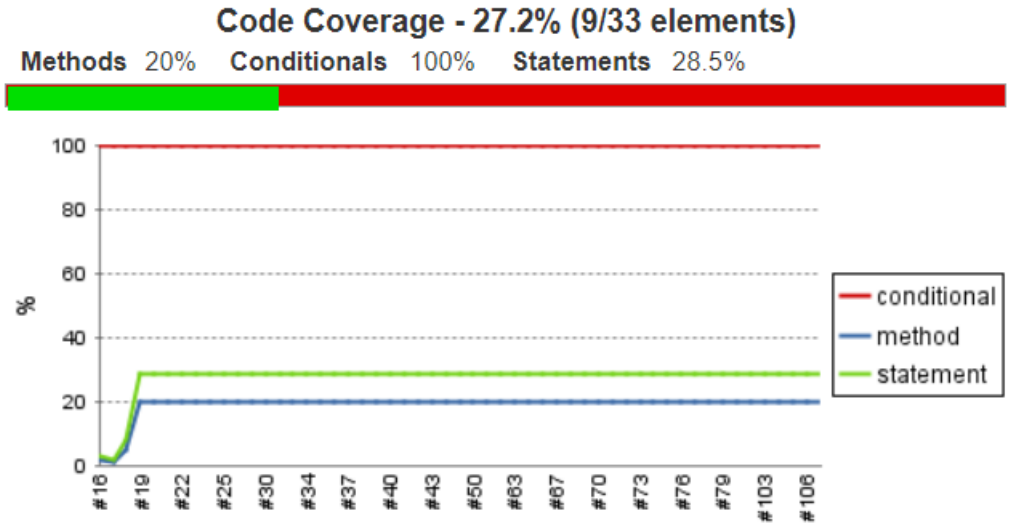

Figura 28: Reporte global de cobertura de pruebas unitarias

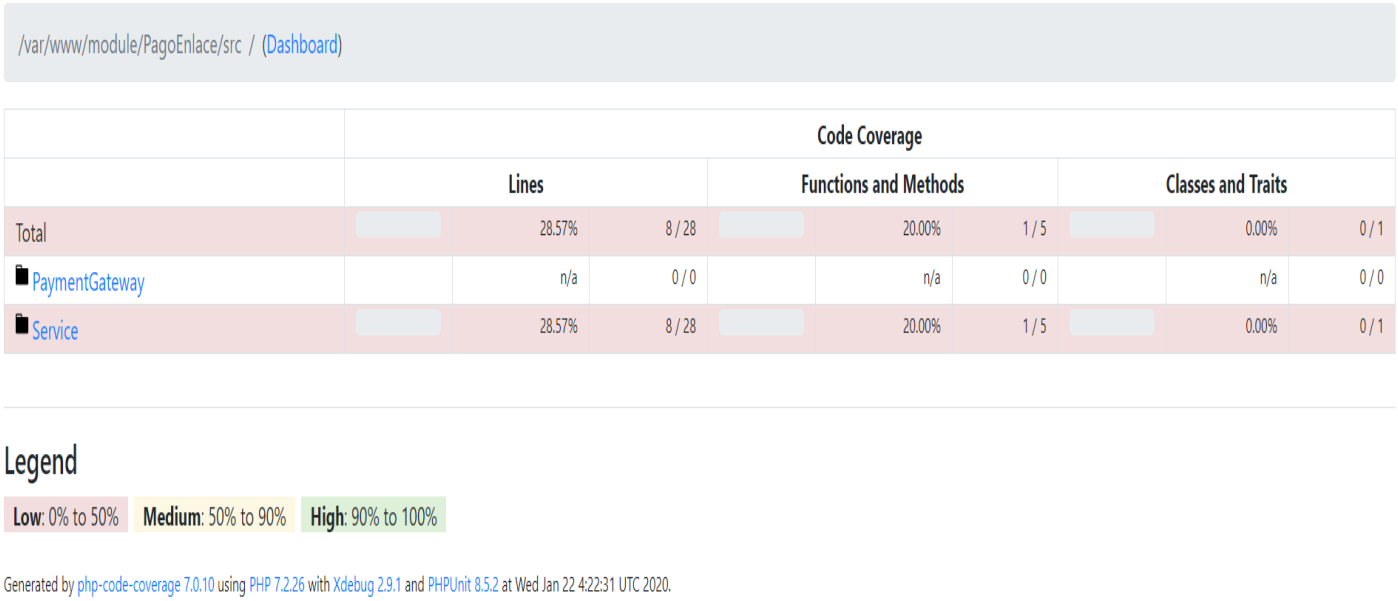

Figura 29: Reporte de módulos testeados

Reporte del porcentaje de líneas de código testeadas por método

/var/www/module/PagoEnlace/src / Service / TransaccionService.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		20.00%	1 / 5	CRAP	28.57%	8 / 28
TransaccionService		0.00%	0 / 1		20.00%	1 / 5	24.86	28.57%	8 / 28
__construct					100.00%	1 / 1	1	100.00%	5 / 5
findById					0.00%	0 / 1	2.06	75.00%	3 / 4
crearCargo					0.00%	0 / 1	6	0.00%	0 / 4
guardar					0.00%	0 / 1	2	0.00%	0 / 2
enviarCorreo					0.00%	0 / 1	2	0.00%	0 / 13

```php
<?php


namespace PagoEnlace\Service;


use Culqi\Culqi;
use Email\Service\EmailService;
use PagoEnlace\Entity\Transacciones;
use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PagoEnlace\PaymentGateway\PaymenGatewayInterface;
use PagoEnlace\Repository\TransaccionRepository;
use Zend\View\Renderer\RendererInterface;

class TransaccionService
{
    /**
     * @var TransaccionRepository
     */
    protected $transaccionRepository;

    /**
     * @var RendererInterface
     */
    protected $renderService;

    /**
     * @var EmailService
     */
    protected $emailService;

    /**
     * @var PaymenGatewayInterface
     */
    protected $paymentGateway;

    /**
     * TransaccionService constructor.
     * @param TransaccionRepository $transaccionRepository
     * @param RendererInterface $renderService
     * @param EmailService $emailService
     */
    public function __construct(TransaccionRepository $transaccionRepository, RendererInterface $renderService, EmailService $emailService, $paymentGateway)
    {
        $this->transaccionRepository = $transaccionRepository;
        $this->renderService = $renderService;
        $this->emailService = $emailService;
        $this->paymentGateway = $paymentGateway;
    }

    /**
     * @param $id
     * @return object|null
     */
    public function findById($id)
    {

        $transaccion = $this->transaccionRepository->findById($id);
        if ($transaccion instanceof Transacciones) {
            return $transaccion;
        }
        throw  new PagoeEnlaceException(self::NOT_FOUND_MESSAGE, self::NOT_FOUND_CODE);
```


Figura 30: Reporte detallado de cobertura de pruebas unitarias

Prueba unitaria implementada: Realizar pago con tarjeta correcta
Archivo testeado: CulqiPaymentGateway.php

TarjetaCorrectaTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;

use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaCorrectaTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaCorrecta.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));

    }

    public function testCargoTarjetaCorrecta() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('charge', $cargo->object);

        $this->assertEquals('venta_exitosa', $cargo->outcome->type);

        $this->assertEquals('Su compra ha sido exitosa.', $cargo->outcome->user_message);

    }

}
```

Figura 31: Tarjeta correcta

Prueba unitaria implementada: Realizar pago con tarjeta perdida
Archivo testeado: CulqiPaymentGateway.php

TarjetaPerdidaTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;

use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaPerdidaTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaPerdida.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));

    }

    public function testTarjetaPerdida() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('error', $cargo->object);

        $this->assertEquals('lost_card', $cargo->decline_code);

        $this->assertEquals('Tarjeta perdida. La tarjeta fue bloqueada y reportada al banco emisor como una tarjeta perdida.'
        ,$cargo->merchant_message);

    }

}
```

Figura 32: Tarjeta perdida

Prueba unitaria implementada: Realizar pago con tarjeta que no cuenta con fondos
Archivo testeado: CulqiPaymentGateway.php

TarjetaFondosInsufcientesTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;

use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaFondosInsuficientesTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaFondosInsufcientes.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));

    }

    public function testTarjetaConFondosInsuficientes() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('error', $cargo->object);

        $this->assertEquals('insufficient_funds', $cargo->decline_code);

        $this->assertEquals('Fondos insuficientes. La tarjeta no tiene fondos suficientes para realizar la compra.'
        ,$cargo->merchant_message);

    }

}
```

Figura 33: Tarjeta no cuenta con fondos

Prueba unitaria implementada: Operación denegada por el banco emisor de tarjeta
Archivo testeado: CulqiPaymentGateway.php

TarjetaOperacionDenegadaContactarEmisorTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;

use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaOperacionDenegadaContactarEmisorTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaAmericaDenegada.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));

    }

    public function testculqiTarjetaAmericaDenegada() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('error', $cargo->object);

        $this->assertEquals('Operación denegada. La operación fue denegada por el banco emisor de la tarjeta por una razón desconocida.', $cargo->merchant_message);

        $this->assertEquals('issuer_decline_operation', $cargo->decline_code);
    }
}
```

Figura 34: Operación denegada

Prueba unitaria implementada: Realizar pago con código de verificación incorrecto
Archivo testeado: CulqiPaymentGateway.php

TarjetaCodigoVerificaciónIncorrectoTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;
use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaCodigoVerificacionIncorrectoTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaCodigoVerificacionIncorrecto.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));
    }

    public function testTarjetaCodigoVerificacionIncorrecto() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('error', $cargo->object);

        $this->assertEquals('CVV incorrecto. El código de seguridad (CVV2, CVC2, CID) de la tarjeta es incorrecto.',
        $cargo->merchant_message);

         $this->assertEquals('incorrect_cvv',$cargo->decline_code);

    }

}
```

Figura 35: Código de verificación incorrecto

Prueba unitaria implementada: Banco emisor de tarjeta no responde
Archivo testeado: CulqiPaymentGateway.php

TarjetaEmisorNoDisponibleTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;

use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaEmisorNoDisponibleTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaEmisorNoDisponible.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));

    }

    public function testTarjetaEmisorNoDisponible() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('error', $cargo->object);

        $this->assertEquals('Emisor no disponible. El banco que emitió la tarjeta no responde. El cliente debe realizar el pago nuevamente.'
        ,$cargo->merchant_message);

        $this->assertEquals('issuer_not_available', $cargo->decline_code);

    }

}
```

Figura 36: Emisor de tarjeta no responde

Prueba unitaria implementada: Error de procesamiento al procesar pago.
Archivo testeado: CulqiPaymentGateway.php

TarjetaErrordeProcesamientoTest.php 1 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\PaymentGateway;
use PagoEnlace\PaymentGateway\CulqiPaymentGateway;
use PHPUnit\Framework\TestCase;

class TarjetaErrordeProcesamientoTest extends TestCase {

    protected $culqiPaymentGatewayMock;

    protected function setUp(): void {

        $this->culqiPaymentGatewayMock = $this
                ->getMockBuilder(CulqiPaymentGateway::class)
                ->disableOriginalConstructor()
                ->getMock();

        $obj = json_decode(file_get_contents(__DIR__ . '/../Data/culqiTarjetaErrordeProcesamiento.json'));

        $this->culqiPaymentGatewayMock->expects($this->once())
                ->method('crearCargo')
                ->will($this->returnValue($obj));

    }

    public function testTarjetaErrordeProcesamiento() {

        $monto = '100';
        $currency = 'PEN';
        $email = 'demo@yopmail.com';
        $token = 'tkn_123';
        $nombres = 'Luis';
        $apellidos = 'Garcia Torres';
        $cargo = $this->culqiPaymentGatewayMock->crearCargo($monto, $currency, $email, $token, $nombres, $apellidos);

        $this->assertEquals('error', $cargo->object);

        $this->assertEquals('Error de procesamiento. Ocurrió un error mientras procesabamos la compra. Contáctate con culqi.com/soporte para que te demos una solución.',
        $cargo->merchant_message);

        $this->assertEquals('processing_error', $cargo->decline_code);

    }

}
```

Figura 37: Error de procesamiento

Prueba unitaria implementada: Prueba unitaria a métodos de la clase TransacciónService.php

TransaccionServiceTest.php 4 KB

Editar | Web IDE | Reemplazar | Eliminar

```php
<?php

namespace PagoEnlaceTest\Service;

use PagoEnlace\Entity\Transacciones;
use PagoEnlace\PaymentGateway\PaymenGatewayInterface;
use PagoEnlace\Repository\TransaccionRepository;
use PagoEnlace\Service\TransaccionService;
use Email\Service\EmailService;
use Zend\View\Renderer\RendererInterface;
use PHPUnit\Framework\TestCase;

class TransaccionServiceTest extends TestCase
{

    protected $transaccionRepositoryMock;
   //  protected $comercioEntityMock;

    /**
     * @var TransaccionService
     */

    protected $transaccionService;
    protected $emailServiceMock;
    protected $renderServiceMock;
    protected $transaccionServiceMock;
    protected $paymenGatewayMock;
    protected $transaccionEntityMock;
    protected $transaccionEntity;
    protected $API_KEY;
    protected $PUBLIC_API_KEY;

    protected function setUp(): void
    {
        /*Mock a la entidad transaccion*/

        $this->transaccionEntityMock = $this
            ->getMockBuilder(Transacciones::class)
            ->disableOriginalConstructor()
            ->getMock();

        $this->transaccionEntityMock->expects($this->once())
            ->method('getId')
            ->will($this->returnValue(101));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getDocumento')
            ->will($this->returnValue('70121059'));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getTransaccionCodigo')
            ->will($this->returnValue('123'));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getMonto')
            ->will($this->returnValue(300));
```

```php
        $this->transaccionEntityMock->expects($this->once())
            ->method('getCurrency')
            ->will($this->returnValue('pen'));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getCodigoReferencia')
            ->will($this->returnValue('123'));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getNombres')
            ->will($this->returnValue('Carlos'));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getApellidos')
            ->will($this->returnValue('Sing Ramos'));

        $this->transaccionEntityMock->expects($this->once())
            ->method('getCorreoElectronico')
            ->will($this->returnValue('demo@yopmail.com'));


    /*Fin de mock a entidad transaccion*/


        $this->transaccionRepositoryMock = $this
            ->getMockBuilder(TransaccionRepository::class)
            ->disableOriginalConstructor()
            ->getMock();

        $this->transaccionRepositoryMock->expects($this->once())
            ->method('findById')
            ->will($this->returnValue($this->transaccionEntityMock));

//        $this->transaccionRepositoryMock->expects($this->once())
//            ->method('save')
//            ->will($this->returnValue($this->transaccionEntityMock));

        $this->emailServiceMock = $this
            ->getMockBuilder(EmailService::class)
            ->disableOriginalConstructor()
            ->getMock();

        $this->renderServiceMock = $this
            ->getMockBuilder(RendererInterface::class)
            ->disableOriginalConstructor()
            ->getMock();


        $this->transaccionService = new TransaccionService(
            $this->transaccionRepositoryMock,
            $this->renderServiceMock,
            $this->emailServiceMock,
            $this->paymenGatewayMock
        );

    }

    public function testFindById()
    {

        $transaccion = $this->transaccionService->findById(101);

        $this->assertEquals(101, $transaccion->getId());
        $this->assertEquals('70121059', $transaccion->getDocumento());
        $this->assertEquals('123', $transaccion->getTransaccionCodigo());
        $this->assertEquals('300', $transaccion->getMonto());
        $this->assertEquals('pen', $transaccion->getCurrency());
        $this->assertEquals('123', $transaccion->getCodigoReferencia());
        $this->assertEquals('Carlos', $transaccion->getNombres());
        $this->assertEquals('demo@yopmail.com', $transaccion->getCorreoElectronico());
    }

}
```

Figura 38: TransaccionServiceTest

Stage5: restaurar base de datos

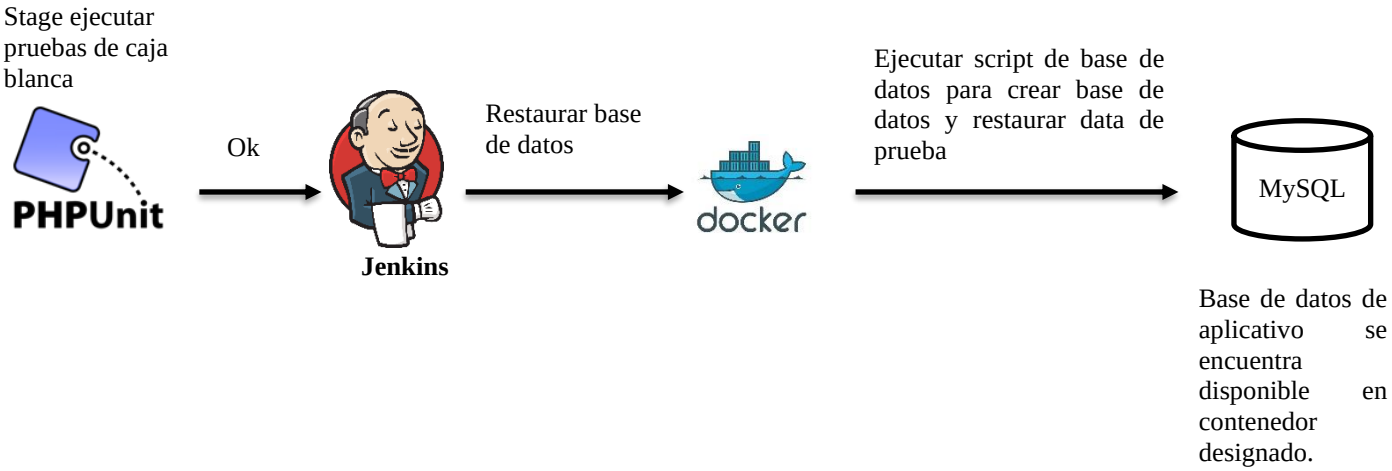

Figura 39: Diagrama de flujo de stage restaurar base de datos

Script de automatización:

```
stage('Restaurar base de datos'){

    bat label: 'docker', script: "docker-compose run zf cat data/bd/dump.sql | docker exec -i pagoenlace_db_1 mysql -uroot -ppass777 pago_app"

}
```

Figura 40: Script de automatización de restauración de base de datos

Resultado de la ejecución

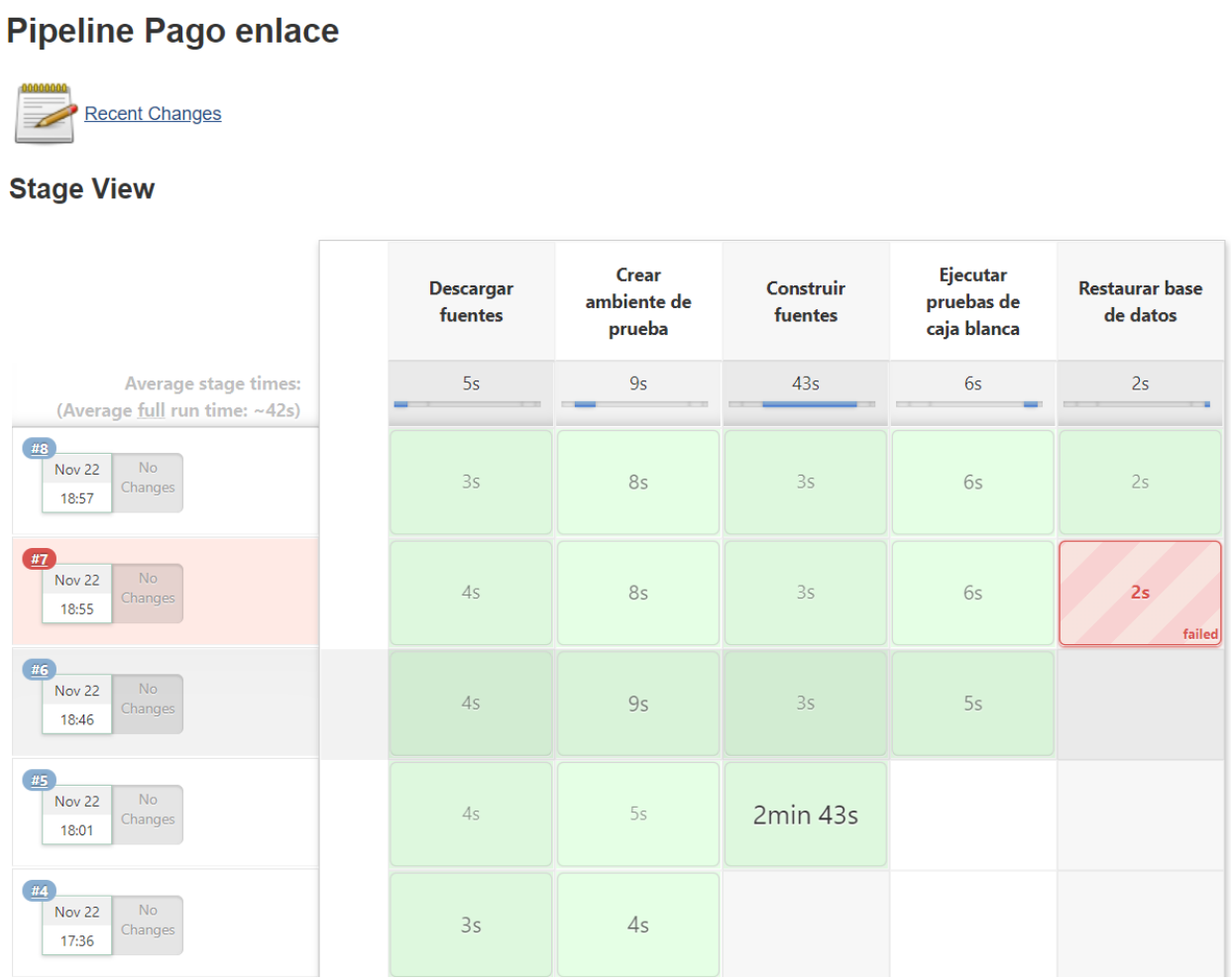

Figura 41: Ejecución de stage de restauración de base de datos

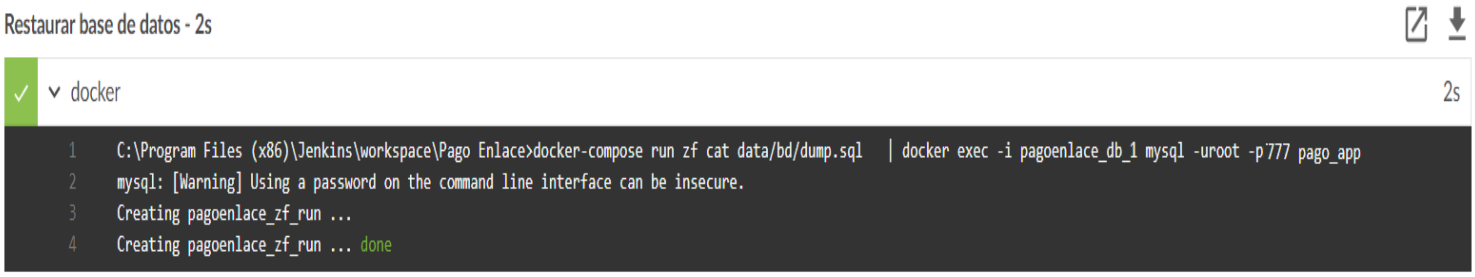

Figura 42: Log de ejecución

Luego de ejecutado el stage se tendrá acceso a la base de datos del aplicativo de pago enlace

Figura 43: Base de datos de aplicativo

Se tiene acceso al aplicativo y se pueden realizar las pruebas de caja negra a través de la URL: http://localhost:8090/indominus/pen

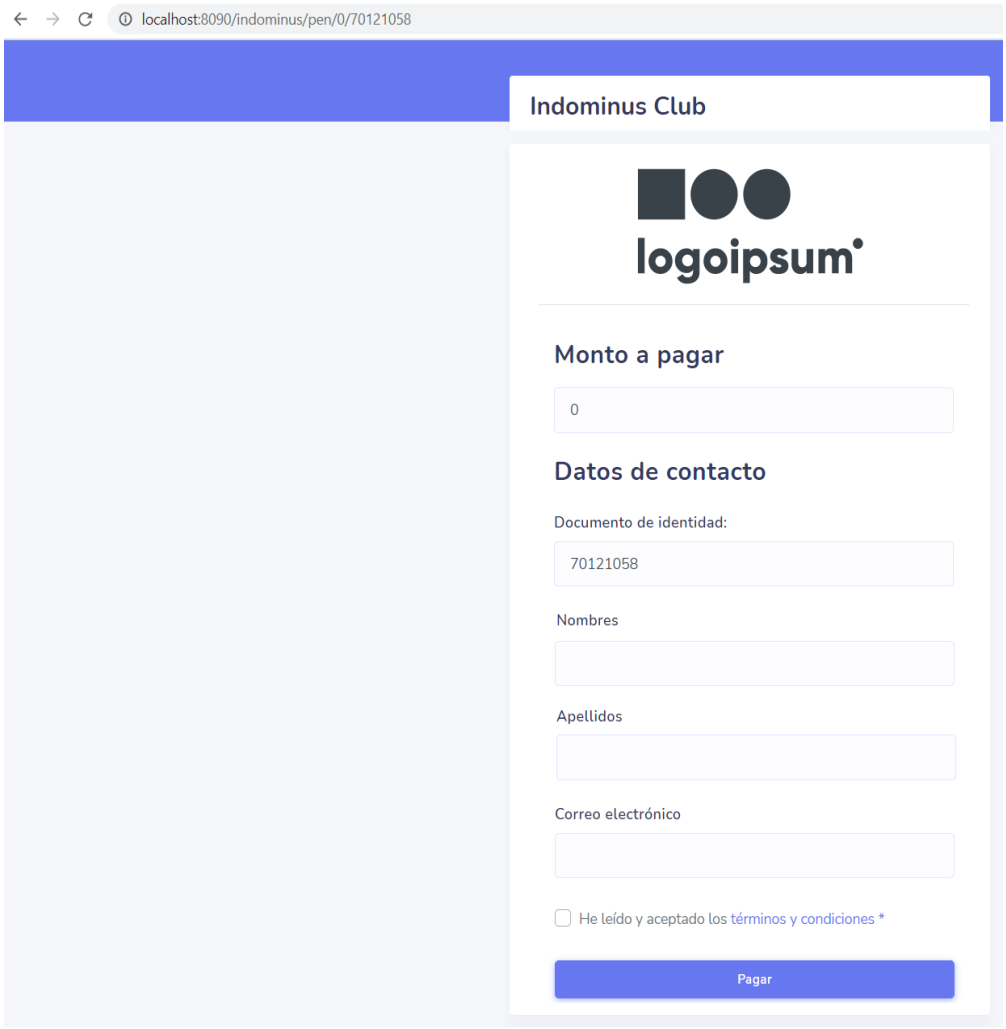

Figura 44: Aplicativo Pago Enlace

Resultados de Sprint2

Implementación de Fase DevOps de entrega continúa, en esta fase ejecutarán de manera automatizada pruebas de caja negra. Al finalizar esta fase se contará con una versión final del aplicativo (release) lista para ser desplegado en producción.

Figura 45: Actividades de fase de entrega continúa DevOps

Stage6: Ejecutar pruebas de caja negra

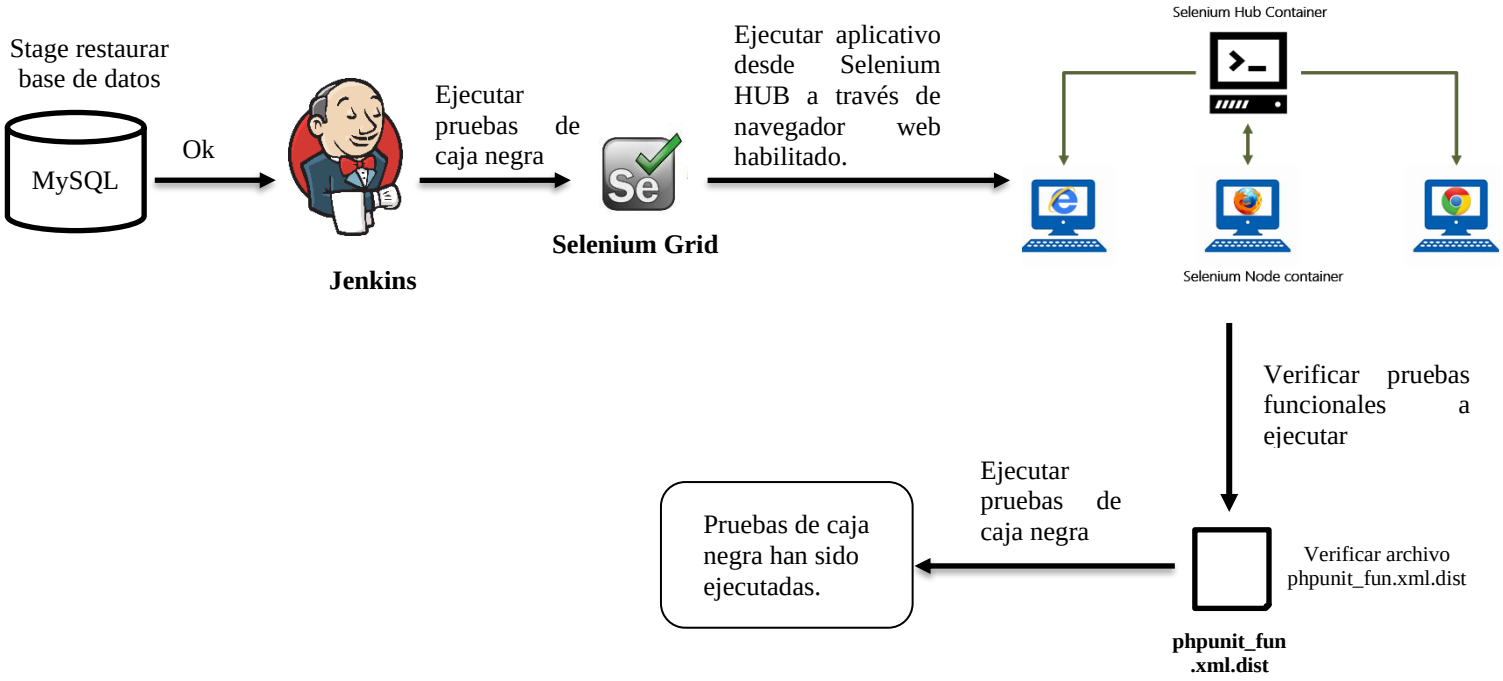

Figura 46: Diagrama de flujo de stage ejecutar pruebas de caja negra

Script de automatización:

```
stage('Ejecutar pruebas de caja negra'){
    bat label: 'docker', script: 'docker-compose  run zf ./vendor/bin/phpunit --configuration=phpunit_fun.xml.dist'
}
```

Figura 47: Script de automatización de ejecución de pruebas de caja negra

Resultado de la ejecución

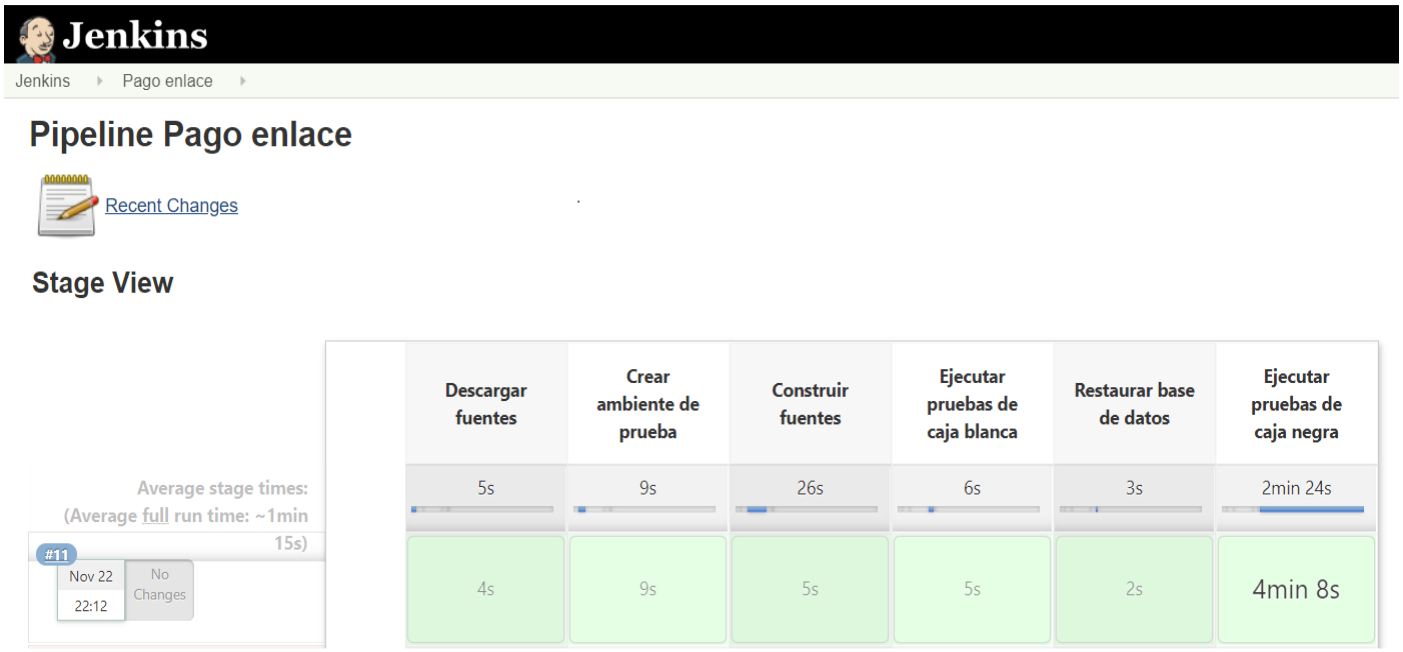

Figura 48: Ejecución de stage de ejecución de pruebas de caja negra

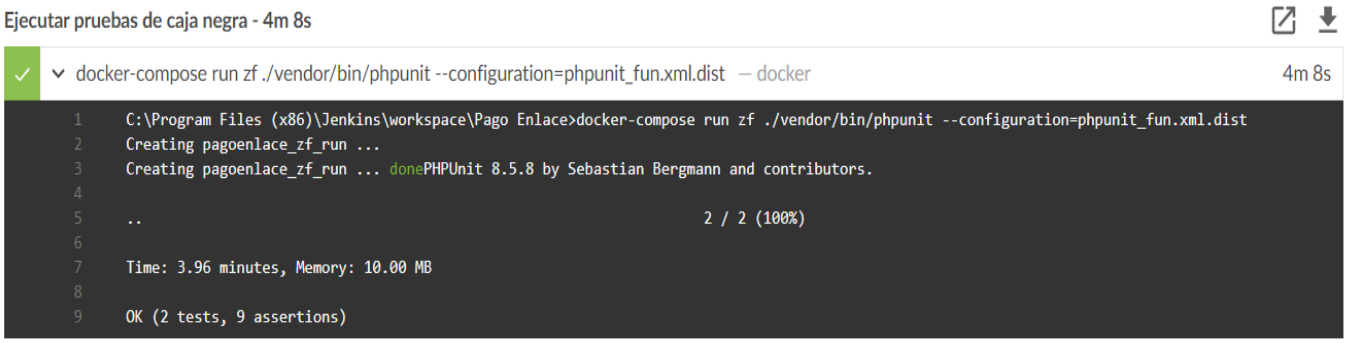

Figura 49: Log de ejecución

Pruebas de caja negra implementadas

CheckoutPageTest.php

```
<?php

namespace PagoEnlaceTest\Funcional\Pages;

use Facebook\WebDriver\Chrome\ChromeOptions;
use Facebook\WebDriver\Remote\DesiredCapabilities;
use Facebook\WebDriver\Remote\RemoteWebDriver;
use Facebook\WebDriver\Remote\WebDriverBrowserType;
use Facebook\WebDriver\Remote\WebDriverCapabilityType;
use Facebook\WebDriver\WebDriverBy;
use Facebook\WebDriver\WebDriverPlatform;
use PHPUnit\Framework\TestCase;


class CheckoutPageTest extends TestCase
{

    protected $driver;

    protected function setUp(): void
    {

        $host = 'http://selenium-hub:4444/wd/hub'; // this is the default

        $capabilities = DesiredCapabilities::chrome();
        // $capabilities = DesiredCapabilities::firefox();
                                $this->driver = RemoteWebDriver::create($host, $capabilities);

        parent::setUp(); // TODO: Change the autogenerated stub
    }

    public function testFlujoNormalVisa()
    {

        $inputFileType = 'Xlsx';
        $inputFileName = __DIR__ . '/TarjetasPrueba.xlsx';
        $sheetname = 'Tarjetas';
        $reader = \PhpOffice\PhpSpreadsheet\IOFactory::createReader($inputFileType);
        $reader->setLoadSheetsOnly($sheetname);
        $spreadsheet = $reader->load($inputFileName);
        $activeSheet = $spreadsheet->getActiveSheet();
        $totalFilas = $activeSheet->getHighestRow();

        foreach (range(2, $totalFilas) as $index) {

            $monto = $activeSheet->getCell('A' . $index)->getValue();
            $documento = $activeSheet->getCell('B' . $index)->getValue();
            $nombres = $activeSheet->getCell('C' . $index)->getValue();
            $apellidos = $activeSheet->getCell('D' . $index)->getValue();
            $email = $activeSheet->getCell('E' . $index)->getValue();
            $cardNumber = $activeSheet->getCell('F' . $index)->getValue();
            $cardExp = $activeSheet->getCell('G' . $index)->getValue();
            $cardCVV = $activeSheet->getCell('H' . $index)->getValue();
            $cardEmail = $activeSheet->getCell('I' . $index)->getValue();
            $message = $activeSheet->getCell('J' . $index)->getValue();
            $pruebaDescripcion = $activeSheet->getCell('K' . $index)->getValue();
```

```
        $this->driver->manage()->window()->maximize();
        $this->driver->get("http://zf/indominus/pen/0/70121058");
        $this->driver->findElement(WebDriverBy::id('monto'))->clear();
        $this->driver->findElement(WebDriverBy::id('monto'))->sendKeys($monto);
        $this->driver->findElement(WebDriverBy::id('documento'))->clear();
        $this->driver->findElement(WebDriverBy::id('documento'))->sendKeys($documento);
        $this->driver->findElement(WebDriverBy::id('nombres'))->sendKeys($nombres);
        $this->driver->findElement(WebDriverBy::id('apellidos'))->sendKeys($apellidos);
        $this->driver->findElement(WebDriverBy::id('email'))->sendKeys($email);
        $this->driver->findElement(WebDriverBy::id('btnPagar'))->click();
        sleep(10);

        $my_frame = $this->driver->findElement(WebDriverBy::className('culqi_checkout'));
        $this->driver->switchTo()->frame($my_frame);
        $this->driver->findElement(WebDriverBy::name('cardNumber'))->sendKeys($cardNumber);
        $this->driver->findElement(WebDriverBy::name('cardExp'))->sendKeys($cardExp);
        $this->driver->findElement(WebDriverBy::name('cardCVV'))->sendKeys($cardCVV);
        $this->driver->findElement(WebDriverBy::name('cardEmail'))->sendKeys($cardEmail);
        $this->driver->findElement(WebDriverBy::className('btnAction'))->click();
        sleep(15);
        $this->driver->switchTo()->defaultContent();
        $mensaje = $this->driver->findElement(WebDriverBy::id('message'))->getText();
        sleep(1);
        $this->assertEquals($message, $mensaje, $pruebaDescripcion);
      }
    }

    public function tearDown(): void
    {

      $this->driver->quit();

    }
}
```

Figura 50: Pruebas de caja negra implementadas con selenium

Configuración de casos de prueba a ejecutar, en este archivo se colocan los datos de las tarjetas de prueba

Archivo: TarjetasPrueba.xlsx

monto	documento	nombres	apellidos	email	cardNumber	cardExp	cardCVV	cardEmail	message	pruebaDescripcion
100	70121058	Carlos	Sing Ramo	carlossing@gmail.com	4111 1111 1111 1111	09/25	123	zerus@yopmail.com	Su compra ha sido exitosa.	Tarjeta Correcta
400	70121058	Carlos	Sing Ramo	carlossing@gmail.com	4000 0300 0000 0009	08/25	836	zerus@yopmail.com	La operación ha sido denegada por la entidad emisora de tu tarjeta. Contáctate con el banco para conocer el motivo de la denegación	Tarjeta perdida
500	70121058	Carlos	Sing Ramo	carlossing@gmail.com	4000 0400 0000 0008	03/25	295	zerus@yopmail.com	Su tarjeta no tiene fondos suficientes. Para realizar la compra, verifica tus fondos disponibles con el banco emisor o inténta nuevamente con otra tarjeta.	fondos insuficientes
500	70121058	Carlos	Sing Ramo	carlossing@gmail.com	5400 0000 0000 0005	01/25	492	zerus@yopmail.com	La operación ha sido denegada por la entidad emisora de tu tarjeta. Contáctate con el banco para conocer el motivo de la denegación o intenta nuevamente con otra tarjeta.	Contactar emisor
500	70121058	Carlos	Sing Ramo	carlossing@gmail.com	5400 0200 0000 0003	07/25	203	zerus@yopmail.com	El código de seguridad (CVV) es incorrecto. Verifica tu tarjeta e ingresa los dígitos correctamente. Si es denegada nuevamente, contáctate con el banco emisor de tu tarjeta.	CVV incorrecto
500	70121058	Carlos	Sing Ramo	carlossing@gmail.com	3700 010000 00000	04/25	2511	zerus@yopmail.com	El banco emisor de tu tarjeta no responde y la operación no pudo ser completada. Vuelve a realizar el pago en unos minutos o intenta nuevamente con otra tarjeta.	Emisor no disponible
500	70121058	Carlos	Sing Ramo	carlossing@gmail.com	3700 020000 00008	05/25	1810	zerus@yopmail.com	La operación ha sido denegada por la entidad emisora de tu tarjeta. Contáctate con el banco para conocer el motivo de la denegación o intenta nuevamente con otra tarjeta.	Operación denegada
500	70121058	Carlos	Sing Ramo	carlossing@gmail.com	3600 010000 0007	12/25	820	zerus@yopmail.com	Ocurrió un error mientras procesabamos tu compra. Contáctate con culqi.com/soporte para que te demos una solución.	Error de procesamiento

Figura 51: Archivo de configuración de pruebas a ejecutar

Stage7: Automatizar versionado de release

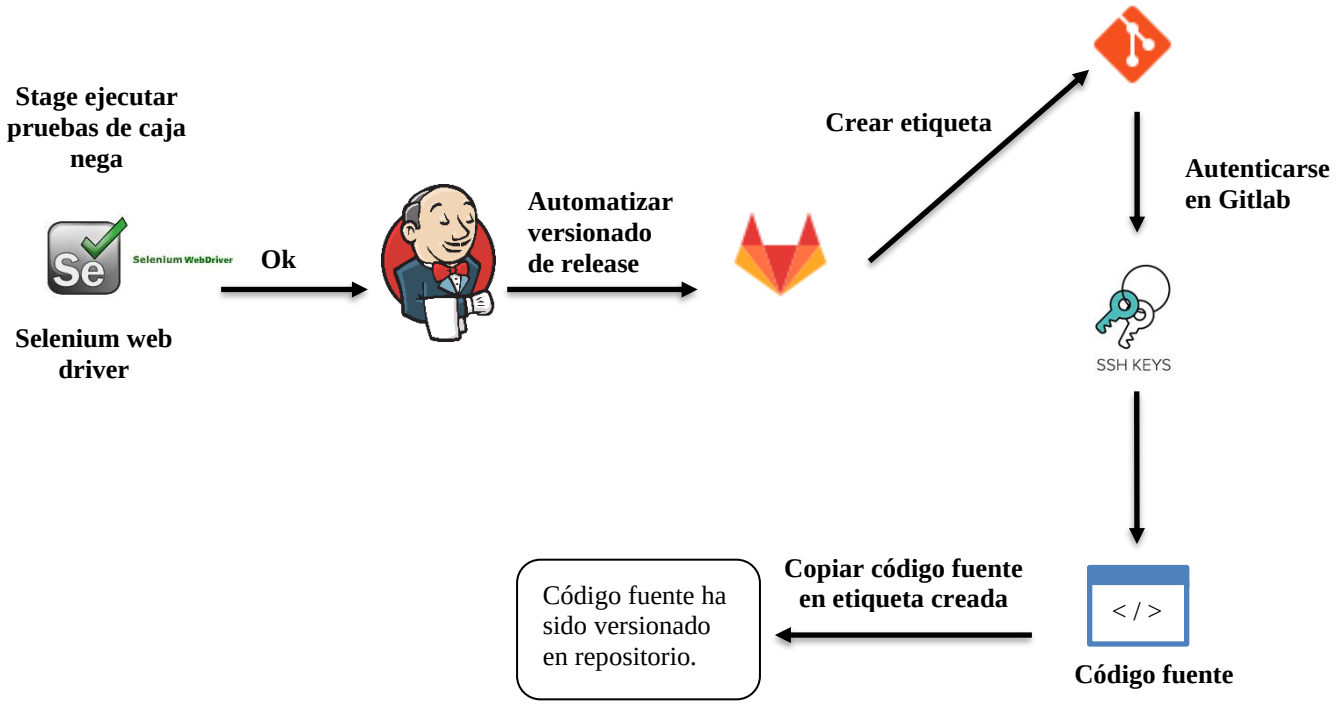

Figura 52: Diagrama de flujo de stage automatizar versionado de release

Script de automatización:

```
stage('Automatizar versionado de release'){

    withCredentials([usernamePassword(credentialsId: 'gitlabUser', passwordVariable: 'GIT_PASS', usernameVariable: 'GIT_USER')]) {

        bat label: 'usuario_gitlab', script: "git config user.name 'miqueridatechi'"
        bat label: 'email_gitlab', script: "git config user.email 'miqueridatechi@gmail.com'"
        echo "User:${USERPROFILE}"
        echo "V1.0.${currentBuild.number}"
        bat label: 'create tag', script: "git tag -a v1.0.${currentBuild.number} -m \"software paso pruebas de caja blanca y caja negra\""
        bat label: 'push tag', script: "git push https://${GIT_USER}:${GIT_PASS}@gitlab.com/pago-enlace/pagoapp.git v1.0.${currentBuild.number}"

        }
    }
```

Figura 53: Script de automatización de versionado de release

Resultado de la ejecución

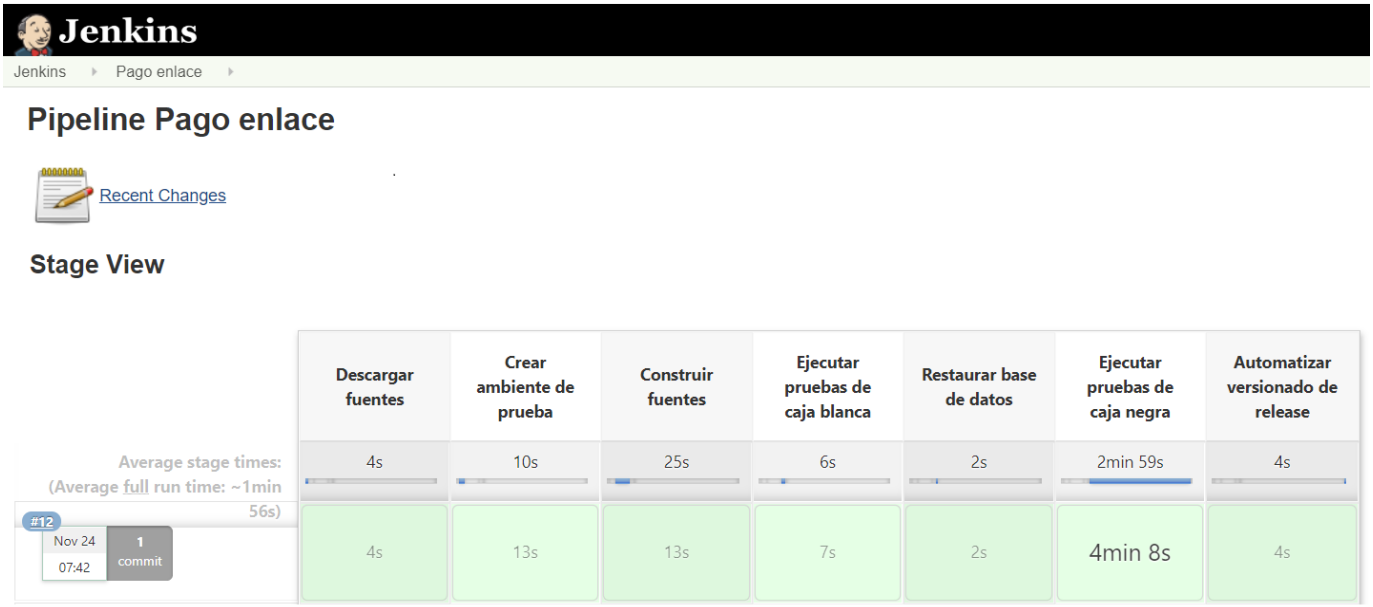

Figura 54: Ejecución de stage de automatización de versionado de release

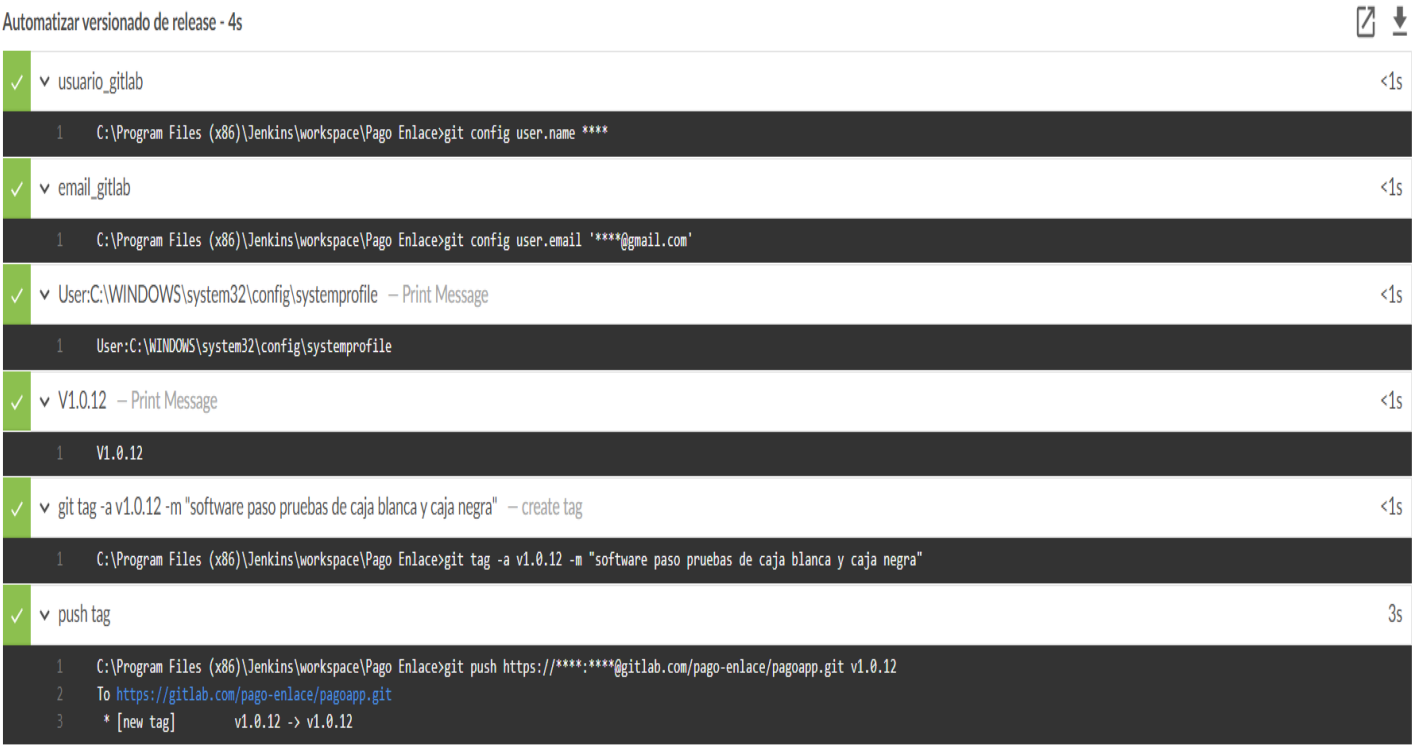

Figura 55: Log de ejecución

Verificar creación de etiqueta en repositorio de Gitlab. Este stage permite versionar con una etiqueta y subir al repositorio de Gitlab una versión del aplicativo que ha pasado de manera exitosa las pruebas de caja blanca y negra. De esta manera se tendrá acceso a una versión del aplicativo que ha pasado las pruebas y que se encuentra listo para ser desplegado en producción.

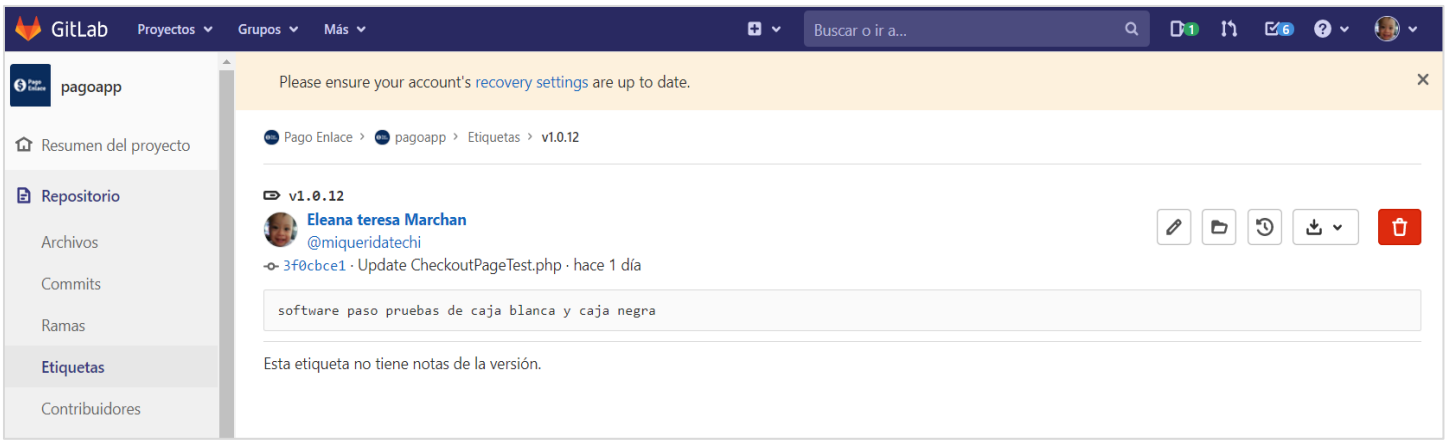

Figura 56: etiquetas creadas en Gitlab

Stage8: Eliminar ambiente de prueba

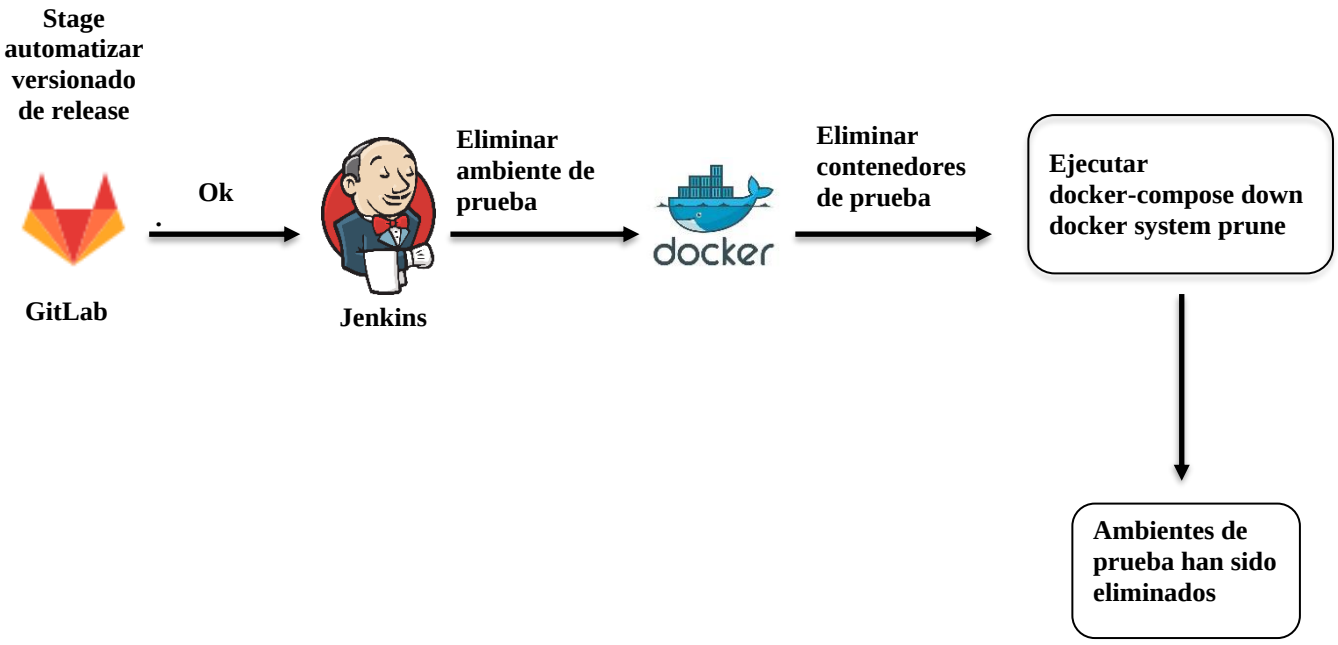

Figura 57: Diagrama de flujo de stage eliminar ambientes de prueba

Script de automatización:

```
stage('Eliminar ambiente de prueba'){
    bat label: 'docker', script: 'docker-compose down'
    bat label: 'docker', script: 'docker system prune --force'
}
```

Figura 58: Script de automatización de eliminación de ambiente de prueba

Resultado de la ejecución

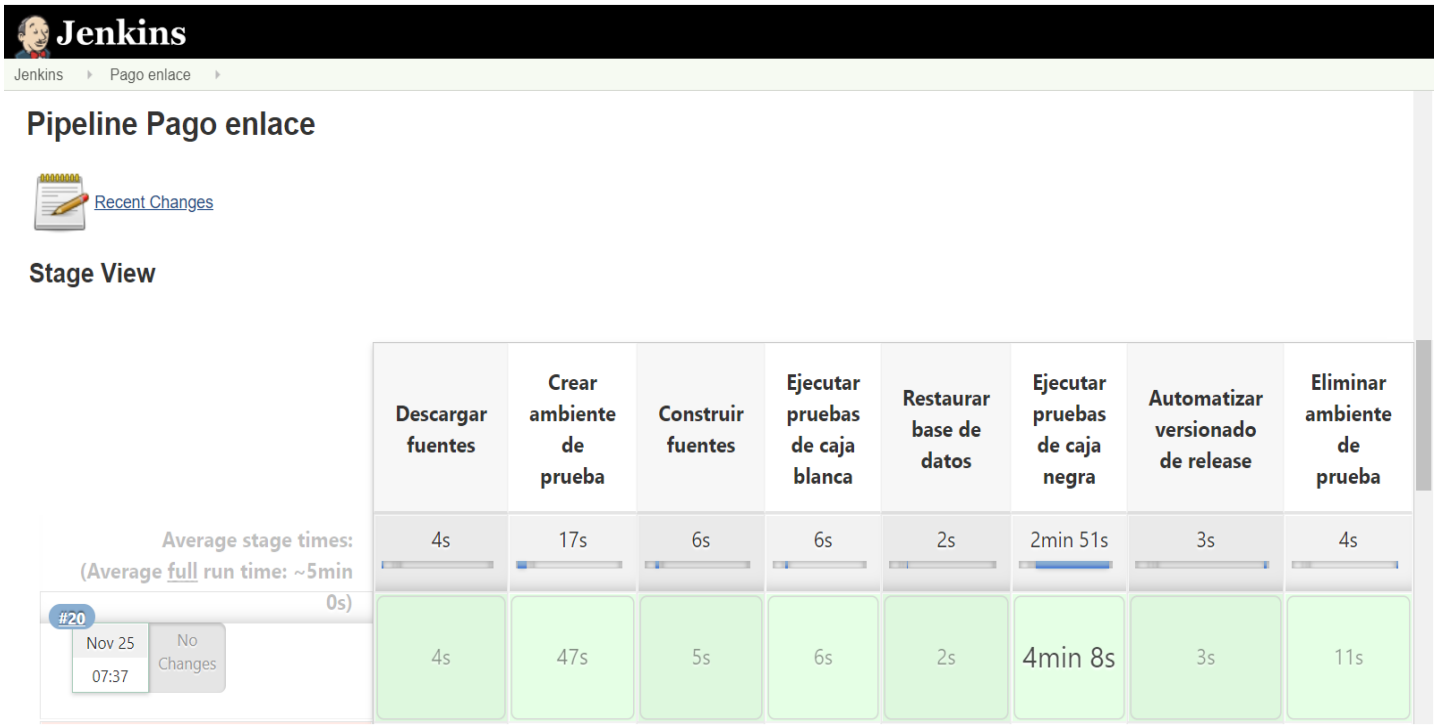

Figura 59: Ejecución de stage de automatización para eliminación de ambiente de prueba

4. Análisis y discusión

De los resultados obtenidos, los encuestados respondieron que la automatización del despliegue y ejecución de pruebas bajo el enfoque DevOps mejora de diversas formas su trabajo, entre las que destacan la detección temprana de errores, otros encuestados destacaron la adopción de buenas prácticas en el desarrollo de software y otros destacaron que el software es sometido a distintas pruebas tomando en cuenta diversos escenarios, lo que permite reducir la deuda técnica de los aplicativos.

En otras palabras, existen múltiples razones para la implementación de un flujo de trabajo automatizado para el control de calidad de aplicaciones web bajo el enfoque DevOps, asi Farias coincide con los resultados de nuestro proyecto en que la implementación de DevOps mejora la colaboración entre los equipos de desarrollo y operaciones e introduce un cambio en la cultura de trabajo de las empresas. Esto debido a que la calidad es asumida desde etapas tempranas por todo el equipo de desarrollo y no solo por el equipo de QA.

Por otro lado, al igual que Tasato, en el presente trabajo se eligió que funcionalidad probar y los escenarios de pruebas a automatizar con el flujo DevOps, mediante el uso de la metodología SCRUM se definieron las historias de usuario a desarrollar para la implementación del flujo automatizado de pruebas, esto permitió generar las fases de integración y entrega continúa, permitiendo la ejecución de pruebas de caja blanca y negra de manera continua y la detección temprana de errores.

Coincidimos con Cespedes y Belalcázar en que es necesario el compromiso de la empresa para poder implementar marcos de trabajo como DevOps, ya que DevOps no se trata solo de herramientas de automatización sino de un cambio en la cultura de trabajo de la empresa eliminando la división entre los equipos de trabajo promoviendo el uso de metodologías agiles de desarrollo, la automatización, el monitoreo y medición del uso de los aplicativos puestos en producción y por último la transferencia de conocimiento, a través de documentación general o código bien documentado.

5. Conclusiones

- Se estableció que es necesario contar con un repositorio central y actualizado de código fuente y aplicar buenas prácticas en el desarrollo del software para tener un código testeable que pueda ser automatizado.
- En la construcción del diseño e implementación de un flujo de trabajo automatizado para el control de calidad de las aplicaciones web bajo el enfoque DevOps, es necesaria la participación de los equipos de operaciones, desarrollo y QA y de herramientas como Git, Jenkins, PhpUnit, Docker y Selenium.
- Se logró establecer que un flujo automatizado de pruebas de software bajo el enfoque DevOps, permite que el software pueda ser sometido a más pruebas en distintos escenarios y en menos tiempo, lo que a su vez mejorará la productividad del equipo de desarrollo.

Recomendaciones

- Se recomienda automatizar pruebas de caja blanca y caja negra, dándole prioridad a las pruebas de caja blanca.
- Se recomienda hacer despliegues continuos en un repositorio central y mantenerlo actualizado para que sirva de base para el flujo DevOps.
- Se recomienda contar con un ambiente dedicado para la implementación de los flujos de integración y entrega continúa DevOps.

AGRADECIMIENTOS

A mis padres, amigos y docentes que, con su apoyo incondicional y conocimientos, forjaron en mí el deseo de superación constante y así culminar mi carrera profesional.

8. Referencias Bibliográficas

Alonso Álvarez. (2020). La empresa Ágil. Madrid: Anaya Multimedia.

Belalcázar. (2017). "*Arquitectura de un data center con herramientas devops"*. Tesis de Doctorado. Universidad Nacional de la Plata, La Plata, Argentina. Recuperado de: http://sedici.unlp.edu.ar/bitstream/handle/10915/63838/Documento_completo__.pdf-PDFA2u.pdf?sequence=1

Céspedes. (2017). *"Propuesta de implementación del proceso de soporte de aplicaciones de seguridad de la información para que sea brindado por InfoSec de Intel Costa Rica".* Proyecto para optar el grado de Maestría Profesional con énfasis en Sistemas de Información. Instituto Tecnológico de Costa Rica Escuela de Computación Programa de Maestría. Cartago, Costa Rica. Recuperado de: https://repositoriotec.tec.ac.cr/handle/2238/9390

Farias. (2017). **"***Definición de un ambiente de construcción de aplicaciones empresariales a través de Devops, Microservicios y Contenedores"*. Tesis de Título. Universidad Técnica Particular de Loja, Loja, Ecuador. Recuperado de: http://dspace.utpl.edu.ec/bitstream/20.500.11962/21205/1/Far%C3%ADas%20Alejandro%20Ivonne%20Karina%20tesis.pdf

Ian Sommerville. (2011). Ingeniería de software. México: Pearson Educación

Joel Francia. (2017). *¿Qué es Scrum?* Recuperado de: https://www.scrum.org/resources/blog/que-es-scrum

Juan Quijano. (2018). *El ciclo de DevOps, una guía para iniciarse en las fases que lo componen.* DevOps. Recuperado de: https://www.genbetadev.com/programacion-en-la-nube/el-ciclo-de-devops-una-guia-para-iniciarse-en-las-fases-que-lo-componen

Mike Cohn. (2009). The Forgotten Layer of the Test Automation Pyramid. Recuperado de: https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid

Mike Wacker. (2015). Just Say No to More End-to-End Tests. Recuperado de: https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html

Pallavi Poojary. (2016). *What is DevOps – Facebook's UseCase | DevOps Tools*. DEVOPS TUTORIAL SERIES. Recuperado de: https://www.edureka.co/blog/what-is-devops/

Santos. (2016). *"Management del aseguramiento de la calidad en desarrollos de software de telecomunicaciones".* Proyecto para optar el grado de maestria en Gestión de Servicios Tecnologicos y Telecomunicaciones. Universidad San Andrés. Buenos Aires, Argentina. Recuperado de: http://repositorio.udesa.edu.ar/jspui/bitstream/10908/11881/1/%5BP%5D%5BW%5D%20T.M.%20Ges.%20Gonz%C3%A1lez%2C%20Santos%20Ram%C3%B3n.pdf

Sam Guckenheimer. (2018). *Learn DevOps / What is Monitoring?* DevOps Resource Center. Recuperado de: https://docs.microsoft.com/en-us/azure/devops/what-is-monitoring

Tasato. (2013). "*Desarrollo de una infraestructura de software para realizar pruebas automatizadas de sistemas de información desarrollados en lenguaje cobol en el contexto bancario"*. Tesis de Título. Pontificia Universidad Católica del Perú, Lima, Perú. Recuperado de: http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/5424/TASAT

O_KENJY_SOFTWARE_SISTEMAS_INFORMACION_LENGUAJE_COBOL_BANCARIO.pdf?sequence=1&isAllowed=y

Tony Stafford. (2017). How to Implement DevOps: The CAMS Approach. Recuperado de: https://shadow-soft.com/how-to-implement-devops/

Anexos

Cuestionario: Coordinador de proyectos

El presente cuestionario tiene por finalidad recopilar información sobre los beneficios obtenidos tras aplicar DevOps en la empresa SingLabz S.A.C

1. ¿La automatización del despliegue y ejecución de pruebas, bajo el enfoque DevOps mejoró el tiempo de entrega del software?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []
   Totalmente desacuerdo []

2. ¿La automatización promueve la adopción de buenas prácticas en el desarrollo de software?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []
   Totalmente desacuerdo []

3. ¿DevOps mejoro el trabajo colaborativo entre el equipo de desarrollo y operaciones?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []
   Totalmente desacuerdo []

Cuestionario: Desarrollador

El presente cuestionario tiene por finalidad recopilar información sobre los beneficios obtenidos tras aplicar DevOps

4. ¿Fue más fácil la detección de errores utilizando un proceso automatizado de pruebas de caja blanca bajo el enfoque DevOps?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []
   Totalmente desacuerdo []

5. ¿DevOps promovió la creación de una estrategia de versionado dentro del equipo de desarrollo?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []
   Totalmente desacuerdo []

6. ¿Es más fácil acceder a versiones anteriores estables de un aplicativo?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []
   Totalmente desacuerdo []

7. ¿DevOps cambio el enfoque que decía que la calidad de software depende solo del equipo de QA?
   Totalmente de acuerdo []
   De acuerdo []
   Neutral []
   En desacuerdo []

Totalmente desacuerdo []

8. ¿Con DevOps la atención a un cambio solicitado por el cliente es más rápido?

   Totalmente de acuerdo []

   De acuerdo []

   Neutral []

   En desacuerdo []

   Totalmente desacuerdo []

9. ¿A través de DevOps el software es sometido a más pruebas?

   Totalmente de acuerdo []

   De acuerdo []

   Neutral []

   En desacuerdo []

   Totalmente desacuerdo []

10. ¿Se siente satisfecho con el control de calidad implementado bajo el enfoque de DevOps?

    Totalmente de acuerdo []

    De acuerdo []

    Neutral []

    En desacuerdo []

    Totalmente desacuerdo []

1.- La automatización del despliegue y ejecución de pruebas, bajo el enfoque DevOps mejoró el tiempo de entrega del software

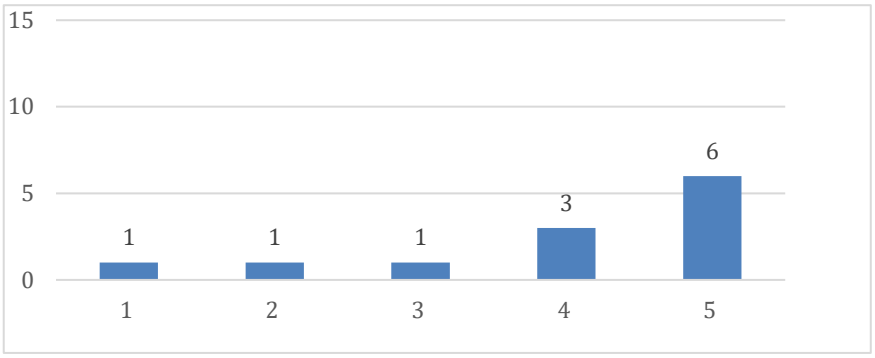

Figura N° 60: Pregunta 1

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 50% de los encuestados opinan que la automatización del despliegue y ejecución de pruebas con DevOps mejoró el tiempo de entrega del software.

2.- La automatización promueve la adopción de buenas prácticas en el desarrollo de software

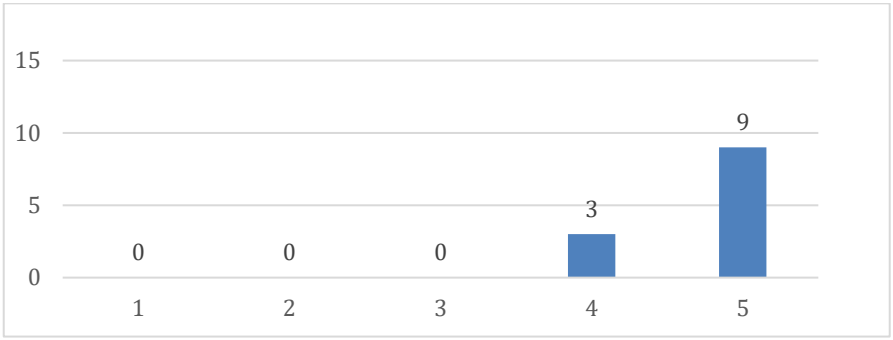

Figura N° 61: Pregunta 2

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 75% de los encuestados opinan que la automatización promueve la adopción de buenas prácticas en el desarrollo de software

3.- DevOps mejoro el trabajo colaborativo entre el equipo de desarrollo y operaciones

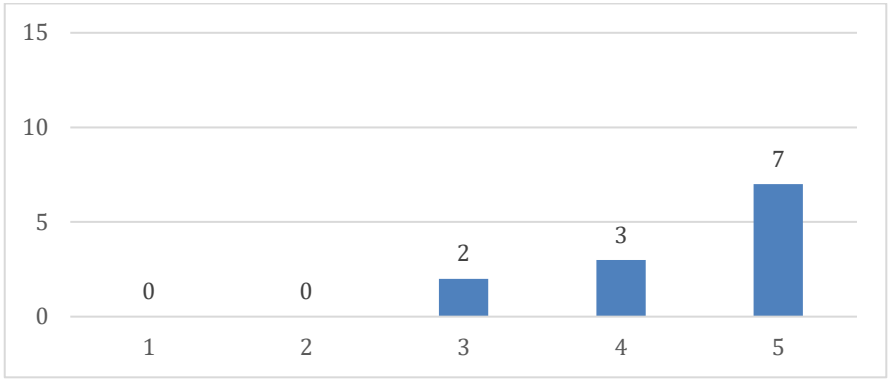

Figura N° 62: Pregunta 3

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 58.33% de los encuestados opinan que la automatización mejoró el trabajo colaborativo entre el equipo de desarrollo y operaciones

4.- Fue más fácil la detección de errores utilizando un proceso automatizado de pruebas de software bajo el enfoque DevOps

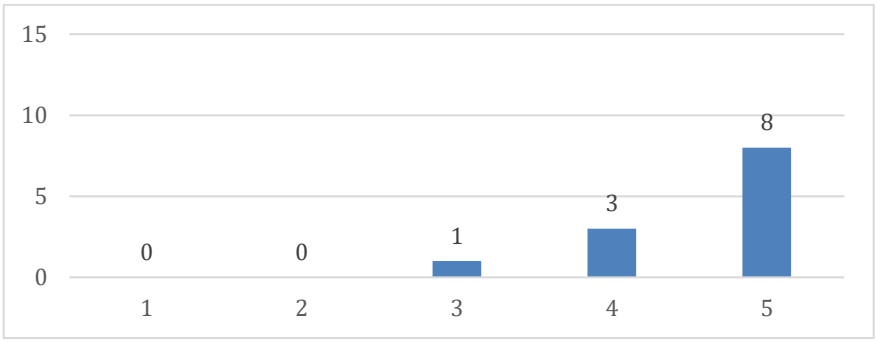

Figura N° 63: Pregunta 4

Interpretación: El gráfico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 66.66% de los encuestados opinan que la automatización hace más fácil la detección de errores utilizando un proceso automatizado de pruebas de software bajo el enfoque DevOps.

5.- DevOps promovió la creación de una estrategia de versionado dentro del equipo de desarrollo

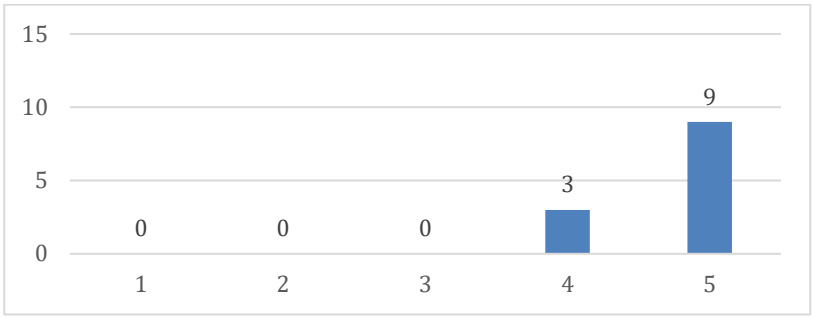

Figura N° 64: Pregunta 5

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 75% de los encuestados opinan que DevOps promovió la creación de una estrategia de versionado dentro del equipo de desarrollo.

6.- Es más fácil acceder a versiones anteriores estables de un aplicativo

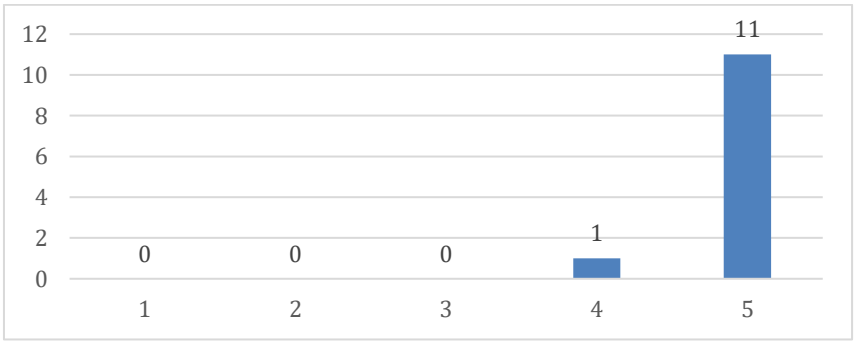

Figura N° 65: Pregunta 6

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 91.66% de los encuestados opinan que es más fácil acceder a versiones anteriores estables de un aplicativo

7.- DevOps cambio el enfoque que decía que la calidad de software depende solo del equipo de QA

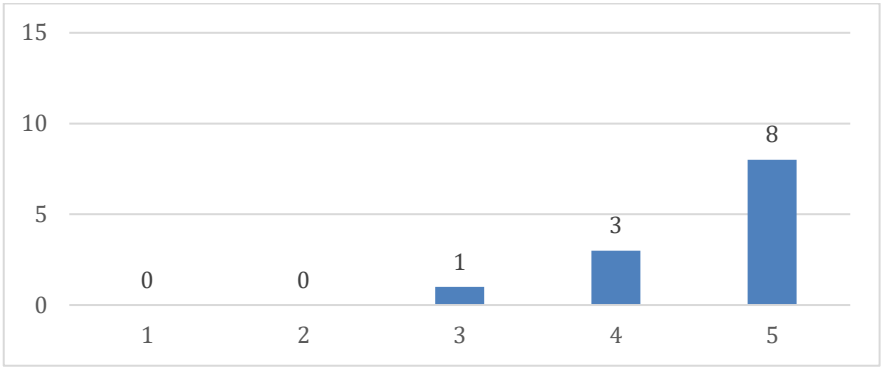

Figura N° 66: Pregunta 7

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 66.66% de los encuestados opinan que cambio el enfoque que decía que la calidad de software depende solo del equipo de QA.

8.- Con DevOps la atención a un cambio solicitado por el cliente es más rápido.

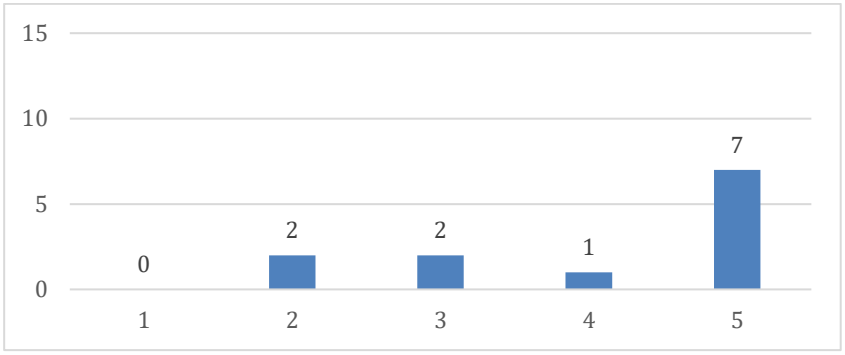

Figura N° 67: Pregunta 8

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 58.33% de los encuestados opinan que con DevOps la atención a un cambio solicitado por el cliente es más rápido.

9.- A través de DevOps el software es sometido a más pruebas

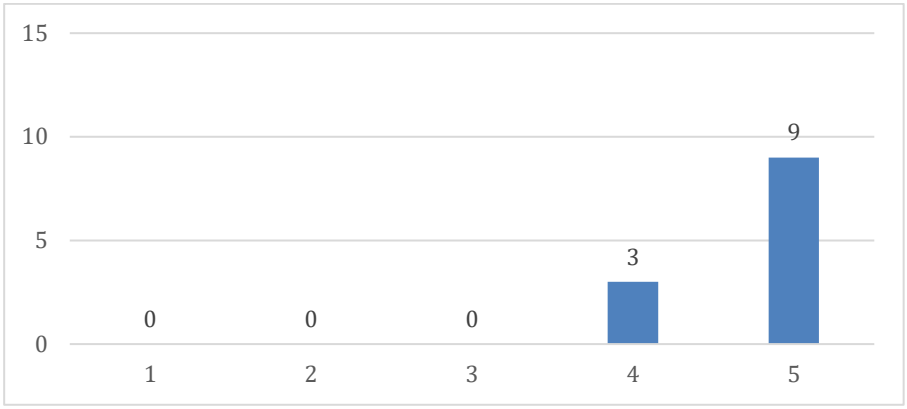

Figura N° 68: Pregunta 9

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 75.00% de los encuestados opinan que a través DevOps el software es sometido a más pruebas.

10.- Se siente satisfecho con el control de calidad implementado bajo el enfoque de DevOps

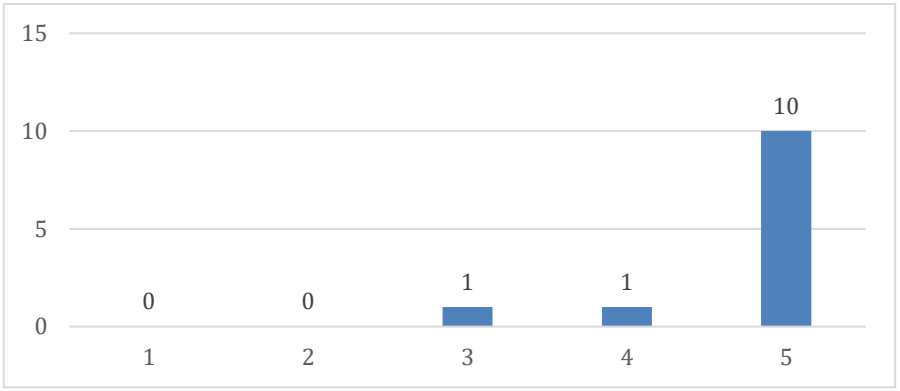

Figura N° 69: Pregunta 10

Interpretación: El grafico nos muestra la opinión de los trabajadores sobre la usabilidad, indicando que el 83.33% de los encuestados opinan que se siente satisfecho con el control de calidad implementado bajo el enfoque de DevOps.

Plan de costos

1. **Recursos disponibles**

1.1.- Personal investigador:

- Marchan Marquina Eleana Teresa

1.2.- Materiales y equipos:

1.2.1 Bienes

- **Recursos informáticos**
  - Laptop - Intel Core I7, 12GB RAM, 1 TB HDD, 2.40 GHz
  - Sistema Operativo Windows 8.1.
  - Microsoft Office profesional plus 2013.
  - Docker Destokp para Windows 64 bits
- **Útiles de escritorio**
  - Lapiceros.
  - Papel Bond 01 millar.
  - Resaltadores.
  - Lápices.
  - Clips.
  - Borradores.
  - Grapas.
  - Folder

1.2.2 Servicios

- Telefonía e Internet.

1.3.- Locales:

- Casa del personal investigador - (Marchan Marquina Eleana Teresa)
- Empresa SingLabz Solutions S.A.C
- Universidad San Pedro– Sede Chimbote

2. Presupuesto

2.1 Materiales y Equipos

2.1.1 Bienes

- **Útiles de Escritorio**

Tabla 1: Útiles de Escritorio

DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO S/.	TOTAL, S/.
Papel Bond A4	2 millares	12.00	24.00
Sub Total:			**24.00**

Fuente: Elaboración propia.

2.1.2 Servicios

Tabla 2: Servicios

DESCRIPCIÓN	TOTAL, S/.
Anillado y empastado	250.00
Impresión y fotocopias	150.00
Movilidad	1000.00
Sub Total:	**1400.00**

Fuente: Elaboración propia.

2.2 Presupuesto Total

Tabla 3: Tabla Presupuesto Total

DESCRIPCIÓN	TOTAL, S/.
Tabla 1	24.00
Tabla 2	1400.00
Total:	**1424.00**

Fuente: Elaboración propia.