UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

VICERRECTORADO

CENTRO DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
FACULTAD DE CIENCIAS Y TECNOLOGÍA

“PROCESO DE AUTOMATIZACIÓN DE PRUEBAS DE INTERFAZ DE USUARIO PARA APLICACIONES WEB DE ARQUITECTURA SPA (SINGLE-PAGE APPLICATIONS)”

TRABAJO EN OPCIÓN A DIPLOMADO EN DEVELOPEMENT

OPERATIONS “DEVOPS" V.1.

AUTOR: NELSON MAITA MIRANDA

SUCRE-BOLIVIA

2024

CESIÓN DE DERECHOS

Al presentar este trabajo como requisito previo para la obtención del Diploma en Developement Operations "DEVOPS" V.1. de la Universidad Mayor, Real y Pontificia de San Francisco Xavier de Chuquisaca, autorizo al Centro de Estudios de Posgrado e Investigación o a la Biblioteca de la Universidad, para que se haga de este trabajo un documento disponible para su lectura según normas de la Universidad

También cedo a la Universidad Mayor, Real y Pontifica de San Francisco Xavier de Chuquisaca, los derechos de publicación de este trabajo o parte de él, manteniendo mis derechos de autor hasta un periodo de 30 meses posterior a su aprobación.

Nelson Maita Miranda

……..

FIRMA:

DEDICATORIA

En primer lugar, a Dios, a mi Padre que desde el cielo me observa, a mi Madre, mis hermanos y hermanas que siempre me motivan a seguir adelante.

A mi familia mi esposa, mis niñas adorada que me aman, por su apoyo constante que son la razón de mi constante esfuerzo.

A mis Docentes del Diplomado y a los Docentes de la Carrera de Ingeniería de Sistemas UMRPSFXCH por compartir su sabiduría y conocimientos.

AGRADECIMIENTOS

A Dios por la sabiduría y guía.

A mis Padres Tomas y Agripina por todo el apoyo brindado.

A mi Esposa Ruth mis hijas Coral, Annel y Abril por todo el amor y apoyo.

A mis hermanos Edwin, Leidy y Veronica que han sido mi ejemplo a seguir.

RESUMEN

La siguiente monografía es aplicada a los procesos de automatización de pruebas de interfaz de usuario (UI) para aplicaciones web de arquitectura SPA (Single Page Application), cuyo objetivo es implementar los fundamentos DevOps para garantizar la calidad del software y proporcionar una experiencia de usuario consistente y libre de errores. Como objetivo específico son los procesos DevOps, incluyendo la integración continua (CI) y despliegue continuo (CD).

La metodología usada para la monografía es la descriptiva por que el proceso de automatización de pruebas de interfaz de usuario para aplicaciones web SPA se centra en describir de manera detallada y objetiva cada etapa del proceso, desde la selección de herramientas hasta la generación de informes y análisis de resultados, sin intervenir en el proceso en sí mismo. Esto proporciona una comprensión clara de cómo se lleva a cabo la implementación de los fundamentos DevOps y cómo contribuye a mejorar la calidad del software y la experiencia del usuario.

Los resultados será el proceso de automatización de pruebas de interfaz de usuario para aplicaciones web SPA es un enfoque sistemático y eficiente con las practicas DevOps para garantizar la calidad del software y mejorar la experiencia del usuario al proporcionar una interfaz funcional.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, el mundo del desarrollo de aplicaciones web ha experimentado un cambio significativo y una de las tendencias más importantes es la adopción de la arquitectura SPA (Single Page Application), Porque las SPA son aplicaciones web más rápidas y fluidas ya que el código solo se carga una vez y se ejecuta en el navegador de los usuarios.

DevOps en el desarrollo de software ha transformado la forma en que se abordan las pruebas de interfaz de usuario (UI) en aplicaciones web de arquitectura SPA (Single Page Application). Antes de la llegada de DevOps, las pruebas manuales eran predominantes y consumían mucho tiempo, lo que afectaba la eficiencia y calidad del desarrollo. Con la automatización de pruebas y la integración continua (CI) y entrega continua (CD), se logran mejoras significativas en la detección temprana de errores.

1. ANTECEDENTES Y JUSTIFICACIÓN

1.1 ANTECEDENTES

Una aplicación de una sola página (SPA) es una aplicación web que usa solo una página web HTML (Fink & Flatow, 2014).

Las SPA utilizan el lenguaje JavaScript para su frontend, complementado con tecnologías HTML y CSS. Para el backend, se puede utilizar cualquier lenguaje porque es independiente del frontend. (Emmit A. , 2015).

Las prácticas de DevOps han ganado mucha atención tanto de la industria del software como de la investigación en ingeniería de software, y ahora se han convertido en una tendencia en ingeniería de software. Aunque no se ha llegado a un consenso en cuanto a una definición del término, podría entenderse como un conjunto de principios, prácticas y un cambio cultural, todo lo cual permite a los desarrolladores y al personal de operaciones trabajar de manera colaborativa (Grupo de Investigación ALARCOS, 2024).

(Devi Nagarajan, 2018), en su tesis titulada “DevOps implementation framework for Agile-based large financial organizations.” Utecht Trecht University. El objetivo es desarrollar un marco de implementación que se adecúe a la implementación del desarrollo ágil. Utiliza el método Cuantitativo incluyendo la revisión de literatura a fin de comprender los principios y organización de impulsores a fin de adoptarlos. El resultado de la Tesis es mostrar diferentes artefactos que responden preguntas que resultan de utilidad para la evaluación de cuán importante es la relación entre la implementación de DevOps y Agile y el impacto directo que esto tiene sobre la obtención de los objetivos de la organización. El aporte de esta tesis radica en la importancia que cumple la transformación digital dentro de las organizaciones utilizando Ágil y DevOps.

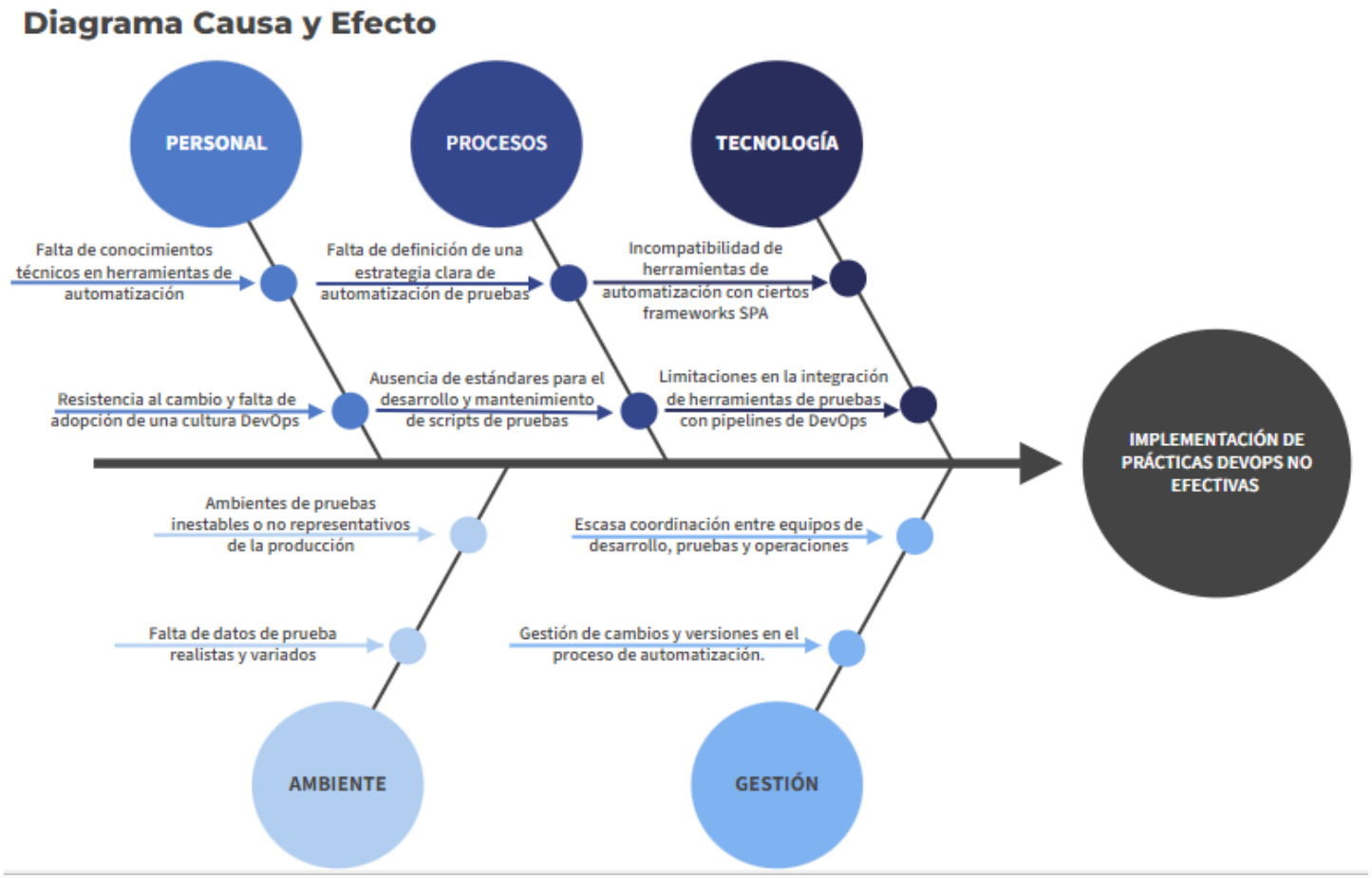
1.2 JUSTIFICACIÓN

Con el crecimiento de las páginas web en todo el mundo, verificación de pruebas en un solo módulo puede ser un gran problema para un equipo de desarrollo, ya que hay que implementar verificaciones asegurando siempre una escalabilidad a futuro. Por lo tanto, aplicar un nuevo enfoque con DevOps se vuelve necesario. La automatización de pruebas en aplicaciones de arquitectura SPA (Single Page Application) ofrece numerosos beneficios y una justificación sólida para su implementación son: Eficiencia, Velocidad, Cobertura Integral, Repetibilidad, Consistencia, Integración Continua y Entrega Continua (CI/CD).

2. SITUACIÓN PROBLÉMICA

La situación problemática en el proceso de automatización de pruebas de interfaz de usuario para aplicaciones web SPA (Single Page Application) con DevOps radica en los desafíos inherentes a la complejidad de las aplicaciones, la diversidad de entornos, la integración con herramientas DevOps, la gestión de datos de prueba y la definición de una estrategia efectiva de automatización. En este caso, aplicaremos el diagrama de Ishikawa para analizar una situación problemática relacionada con la implementación de prácticas DevOps en automatización de pruebas en aplicaciones de arquitectura SPA (Single Page Application).

Ilustración 1 - Situación Problemática (ISHIKAWA)



Fuente: Elaboración Propia

3. FORMULACIÓN DEL PROBLEMA DE INVESTIGACIÓN

¿Cómo mejorar la eficiencia, la calidad y la confiabilidad del proceso de pruebas de interfaz de usuario en aplicaciones web de arquitectura SPA (Single Page Application) mediante la implementación de prácticas de automatización de pruebas integradas con DevOps?

4. OBJETIVO GENERAL

Evidenciar, por medio de una investigación teórica justificada con textos académicos, determinar el proceso de automatización de pruebas de interfaz de usuario para aplicaciones web de arquitectura SPA (Single Page Application), integrado con prácticas de DevOps.

5. OBJETIVOS ESPECÍFICOS

- Observar las características y requerimientos específicos de las aplicaciones web de arquitectura SPA (Single Page Application) para las pruebas automatizadas.
- Revisar herramientas y *frameworks* de automatización de pruebas adecuados para aplicaciones web SPA (Single Page Application).
- Comparar las pruebas automatizadas en los procesos de integración continua y entrega continua (CI/CD) de DevOps.
- Documentar el proceso de automatización de pruebas SPA con Devops.

6. DISEÑO METODOLÓGICO

6.1 TIPO DE INVESTIGACIÓN

El tipo de investigación usado en esta monografía es descriptivo, se refiere a que el objeto de investigación solo llegara a la descripción. (Maldonado Pinto, 2018, pág. 24).

El tipo de investigación descriptivo se centra en describir características, comportamientos, situaciones o fenómenos tal como son, sin modificarlos ni alterarlos.

6.2 MÉTODOS

Los métodos de investigación para "Proceso de Automatización de Pruebas de Interfaz de Usuario para Aplicaciones Web de Arquitectura SPA (Single Page Application) con DevOps", seria mixta (cualitativa y cuantitativa) sería la más adecuada para esta monografía.

El método Cuantitativo, está orientado al esfuerzo del investigador a descifrar y responder las interrogantes ¿Qué?, ¿Cómo? y ¿Cuándo? Ocurren los fenómenos que son objeto de estudio, en el caso del método Cualitativo existe mayor interés por identificar ¿Por qué? Y ¿Cómo? Acontecen dichos fenómenos. (Pimienta Prieto & De la Orden Hoz, 2017, pág. 61)

6.2.1 MÉTODOS TEÓRICOS

Investigación Documental: Para realizar una investigación documental con el objetivo de determinar un proceso de automatización de pruebas de interfaz de usuario para aplicaciones web de arquitectura SPA integrado con prácticas de DevOps

Consiste en el análisis de documentos, libros, artículos científicos, informes técnicos, leyes, reglamentos y cualquier otro tipo de material escrito relacionado con el tema de estudio. Este método teórico es fundamental para revisar el estado del arte, identificar antecedentes, teorías y conceptos relevantes, así como para sustentar teóricamente la investigación.

6.2.2 METODO ANÁLISIS – SÍNTESIS

Análisis: Este método fue el que más utilizado a lo largo del desarrollo de la monografía, puesto que tiene gran utilidad para la búsqueda y el procesamiento de la información (clasificación, descomposición, división), fue aplicado en las siguientes etapas: en la delimitación del tema, redacción de la situación problemática, la formulación del problema, redacción de los objetivos, así como en el marco teórico.

Síntesis: Este método fue utilizado en esta monografía en la integración de la información analizada para crear una visión completa y coherente del proceso de automatización de pruebas en aplicaciones SPA con DevOps, en la Construcción de argumentos y así Generar las conclusiones: Llegar a conclusiones basadas en la síntesis de la información analizada, destacando las mejoras potenciales en la eficiencia, calidad y confiabilidad del desarrollo de software mediante la automatización de pruebas en aplicaciones SPA con DevOps.

6.2.3 MÉTODOS EMPIRICOS

"Proceso de Automatización de Pruebas de Interfaz de Usuario para Aplicaciones Web de Arquitectura SPA (Single Page Application) con DevOps", en el método empírico utilizaremos el Análisis de datos, los datos recopilados lo analizaremos utilizando los métodos de análisis cualitativos o cuantitativos.

6.3 TÉCNICAS

Las técnicas utilizadas en el contexto del Proceso de Automatización de Pruebas de Interfaz de Usuario para Aplicaciones Web de Arquitectura SPA con DevOps según nuestros objetivos específicos de la investigación presentan las siguientes técnicas que pueden ser relevantes:

Revisión de Documentación y Literatura: Realizar una revisión exhaustiva de la literatura, artículos, libros y documentos relacionados con la automatización de pruebas en aplicaciones SPA y la integración con DevOps. Esto proporcionará una base sólida de conocimientos teóricos y prácticos.

Análisis de Datos: Utilizar técnicas de análisis de datos cualitativos y cuantitativos para analizar los resultados de las pruebas automatizadas, métricas de calidad, tiempos de ejecución, cobertura de pruebas y otros aspectos relevantes.

6.4 PROCEDIMIENTOS E INSTRUMENTOS DE INVESTIGACIÓN

6.4.1.1 INSTRUMENTOS DE INVESTIGACIÓN

Análisis Estadístico: Utilización de herramientas y técnicas de análisis estadístico para procesar y analizar los datos recopilados a través de cuestionarios, entrevistas y registros de pruebas, con el objetivo de identificar patrones, tendencias y relaciones significativas.

Software de Automatización de Pruebas: Empleo de herramientas de automatización de pruebas, para analizar casos de prueba en aplicaciones web SPA, y evaluar su compatibilidad con los procesos de DevOps.

Tabla 1 - Metodología

TIPO DE MONOGRAFÍA		TIPO DE INVESTIGACIÓN		
ANÁLISIS DE EXPERIENCIAS		DESCRIPTIVA		
OBJETIVOS	**MÉTODOS**	**TÉCNICAS**	**INSTRUMENTO**	**RESULTADOS**
Observar las características y requerimientos específicos de las aplicaciones web de arquitectura SPA (Single Page Application) para las pruebas automatizadas.	Cualitativo	Análisis Documental	Revisión Documental	Crear una matriz que relacione cada requisito con los escenarios de prueba.
Revisar herramientas y *frameworks* de automatización de pruebas adecuados para aplicaciones web SPA (Single Page Application).	Cuantitativo	Revisión Bibliográfica	Herramientas de Investigación	Recomendaciones sobre las herramientas y *frameworks* más apropiados para aplicaciones web SPA, con las necesidades específicas del proyecto y las mejores prácticas de automatización de pruebas.
Comparar las pruebas automatizadas en los procesos de integración continua y entrega continua (CI/CD) de DevOps.	Mixto, cualitativo y cuantitativo	Análisis de Datos Cuantitativos	Herramientas de CI/CD	Obtener métricas sobre el rendimiento de las pruebas automatizadas en los procesos de CI/CD, con tiempos de ejecución, cobertura de pruebas, índice de fallos.
Documentar el proceso de automatización de pruebas.	Cuantitativo	Análisis de Documentos	Revisión de Documentos	Generar una documentación completa y detallada que describa el proceso de automatización de pruebas, incluyendo procedimientos, herramientas utilizadas, casos de uso, informes de resultados.

Fuente: Elaboración Propia

CAPÍTULO I

MARCO TEÓRICO Y CONTEXTUAL

1.1 MARCO CONCEPTUAL

El marco conceptual en esta monografía se refiere a la estructura teórica y conceptual que sustenta el estudio sobre la automatización de pruebas de interfaz de usuario en aplicaciones web SPA con DevOps.

1.1.1 DEFINICIÓN DE CONCEPTOS CLAVE

Términos usados en esta monografía como "automatización de pruebas", "interfaz de usuario", "aplicaciones web SPA", "prácticas de DevOps", "eficiencia", "calidad" y "confiabilidad del desarrollo de software". Estas definiciones ayudan a establecer un lenguaje común y aclarar el alcance del estudio.

1.1.1.1 AUTOMATIZACIÓN DE PRUEBAS

La automatización de pruebas de software se ha movido más allá de un lujo para convertirse en una necesidad (Fewster & Graham, 2012). Las pruebas manuales no son suficientes para aplicaciones complejas en constante cambio.

La automatización de pruebas es una recomendación, aunque no una obligación, útil para crear un entorno de desarrollo satisfactorio (Fewster & Graham, 2012). Los casos de prueba aseguran que la aplicación funciona correctamente, incluso después de cambios en su código interno, gracias a las pruebas automatizadas. Además, este tipo de pruebas obligan a los programadores a crear pruebas en un formato estandarizado y muy sólido que pueda ser procesado por un framework de pruebas (Fewster & Graham, 2012).

1.1.1.2 APLICACIÓN WEB

Una aplicación web es un software que realiza tareas a través de internet, para que una aplicación web funcione correctamente se necesita un servidor web, un servidor de

aplicaciones y un servidor de base de datos. El servidor web procesa las solicitudes del cliente y el servidor de aplicaciones completa las tareas solicitadas. Para acceder a una aplicación web se utiliza un navegador web como Google Chrome, Mozilla Firefox entre otros (Rouse, 2023). Una aplicación web puede tener uno o varios clientes, cada uno con una interfaz de usuario diferente. Las aplicaciones web pueden ser accedidas desde diferentes dispositivos móviles o de escritorio, cada cliente puede tener una interfaz de usuario ligeramente diferente, pero cada cliente interactúa con la misma aplicación al conectarse al servidor. El cliente interactúa con el servidor realizando peticiones de Creación, Lectura, Actualización y Eliminación de datos (CRUD por sus siglas en inglés) (Fox, 2020).

Ilustración 2 - Esquema básico de una Aplicación Web.

Fuente: https://unprogramador.com/crear-interfaz-retrofit2/

1.1.1.3 APLICACIÓN WEB SPA (SINGLE-PAGE APPLICATION)

Una Single Page Application es una aplicación basada en la web que utiliza solo una página HTML como la columna vertebral de todas las sub páginas de la aplicación, no necesita actualizarse durante su utilización, una vez cargado el fichero HTML se descarga todo el código fuente necesario para la funcionalidad de la aplicación (Solovei, Olshevska, & Bortsova, 2017). Para mejorar la interacción del usuario con la aplicación se implementa JavaScript y CSS. A diferencia de los sitios web tradicionales que se cargan en el servidor, las aplicaciones SPA entrega esa responsabilidad al navegador (Borzemski, Grzech, Świątek, & Wilimowska, 2017).

Las SPA realizan la petición del marcado (HTML) y los datos (JSON) de forma independiente, una vez cargado el fichero HTML y los datos, el navegador es responsable de renderizar y visualizar la página web. Una SPA interactúa con el servidor web únicamente con la transferencia de datos, lo que reduce la sobrecarga en la red, permitiendo a la aplicación ser mucho más rápida en comparación a las aplicaciones web tradicionales (Solovei, Olshevska, & Bortsova, 2017).

Ilustración 3 - Interacción una aplicación SPA

Fuente: https://www.omatech.com/blog/2022/08/31/que-es-una-web-spa/

Una SPA puede estar compuesta por una gran cantidad de scripts y estilos, pero no se descargan todos los archivos con la petición inicial, los recursos se solicitan al servidor dependiendo a la necesidad de la sub página, posterior se almacena en la caché del navegador para no volver a solicitarlos (Solovei, Olshevska, & Bortsova, 2017).

Ventajas

- Existen varios *frameworks* que permiten el desarrollo, despliegue y depuración de forma sencilla.
- Al cargar todo el contenido en una sola página, el contenido se puede mostrar de una manera simple, elegante y veloz.
- Una vez que la aplicación realiza la solicitud inicial y muestra el contenido en el navegador, solo enviará y recibirá datos, reduciendo así el tiempo de respuesta a las acciones del usuario.

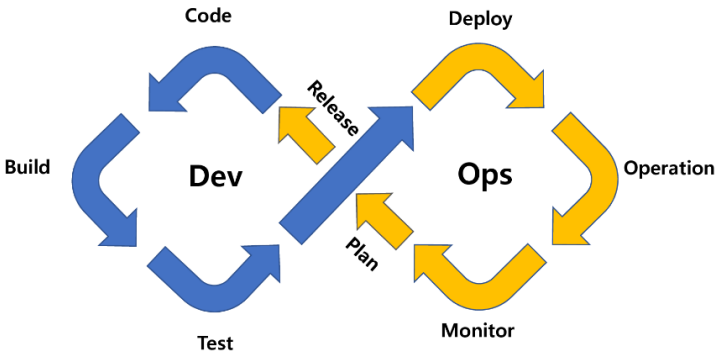
- El navegador es el encargado de ejecutar la lógica, lo que permite que la interacción del usuario sea más rápida.
- Las aplicaciones SPA aceleran el proceso de carga al minimizar el código JavaScript, eliminar las reglas CSS no utilizadas y fusionar recursos.

1.1.1.4 PRACTICAS DEVOPS

DevOps (Development plus Operations) ha tomado recientemente el centro del escenario en el SDLC (Software Development Life Cycle). DevOps ofrece marcos de procesos mejorados con herramientas de código abierto para integrar todas las fases del ciclo de vida de la aplicación y garantizar que funcionen como una unidad cohesionada (Srincharan , 218).

Nacido de la necesidad de mejorar la agilidad de prestación del servicio de TI el movimiento DevOps enfatiza la comunicación, colaboración y la integración entre desarrolladores de software y operaciones de TI. En lugar de ver a estos dos grupos como silos que se van pasando las cosas pero que no trabajan realmente juntos, DevOps reconoce la interdependencia del desarrollo del software y las operaciones de TI y ayuda a una organización a producir software y servicios de TI de forma más rápida, con frecuentes interacciones (Hutterman, September 2012).

Ilustración 4 - Ciclo DEVOPS



Fuente: https://www.brainz.co.kr/company-story/view/page/3/id/158#u

En un ambiente de DevOps, la funcionalidad cruzada, las responsabilidades compartidas y la confianza son un valor en alza. DevOps esencialmente extiende el continuo desarrollo de objetivos del movimiento Ágil a una continua integración y lanzamiento. Para hacer más fáciles los lanzamientos, DevOps fomenta la automatización del cambio, configuración y procesos de lanzamiento (Hutterman, September 2012) (Soni, 2016).

Los beneficios de este enfoque son muchos, incluyendo:

- Mejora en la frecuencia de instalación, lo cual puede llevar a sacarlo al mercado en un periodo de tiempo más breve.
- Tasa de error más baja.
- Un tiempo de espera más corto.
- Facilita la comunicación entre los desarrolladores, los analistas de calidad y operaciones, mostrándose como un punto de intersección entre los 3 grupos mencionados.

Ilustración 5 - Relación DEVOPS y demás actores

Fuente: https://www.xeridia.com/blog/sabes-realmente-que-es-devops

1.1.1.5 INTEGRACIÓN CONTINUA

El proceso de IC (integración continua) tiene como objetivo principal comprobar que cada actualización del código fuente no genere problemas en una aplicación que se está desarrollando. La integración continua fue utilizada por IBM para el desarrollo del OS/360 en los años 60 (IBM Redbooks;, 2015).

La integración continua no es una herramienta sino más bien una práctica salida del eXtreme Programming (XP) (IBM Redbooks;, 2015).

La práctica de la integración continua lleva varios años ayudando a gestionar mejores proyectos en multitud de organismos debido a que proporciona una gran cantidad de ventajas como por ejemplo la reducción de tiempo y cantidad de errores en la integración de código, la disponibilidad de la última versión del código a los miembros del equipo, visibilidad de los cambios en código, etc. (IBM Redbooks;, 2015). Sin embargo, dada la dificultad de uso de las herramientas actuales, aún no es una práctica conocida y utilizada intensivamente en los desarrollos de software (IBM Redbooks;, 2015).

1.1.1.6 ENTREGA CONTINUA

Se logra la entrega continua, integrando continuamente el software realizado por el equipo de desarrollo, la creación automáticamente de versiones instalables y la ejecución de pruebas automatizadas en esos ejecutables para la detección automática de errores. Además, se deben poder mover los ejecutables a entornos de producción. Para establecer cuál debe ser el flujo de tareas automatizadas (construcción de los instalables, pruebas automáticas, puntos de chequeos, instalación en diferentes entornos…) se define lo que se denomina un "Deployment Pipeline" (Swartout, 2014).

Para lograr la implementación exitosa de la entrega continua es necesario:

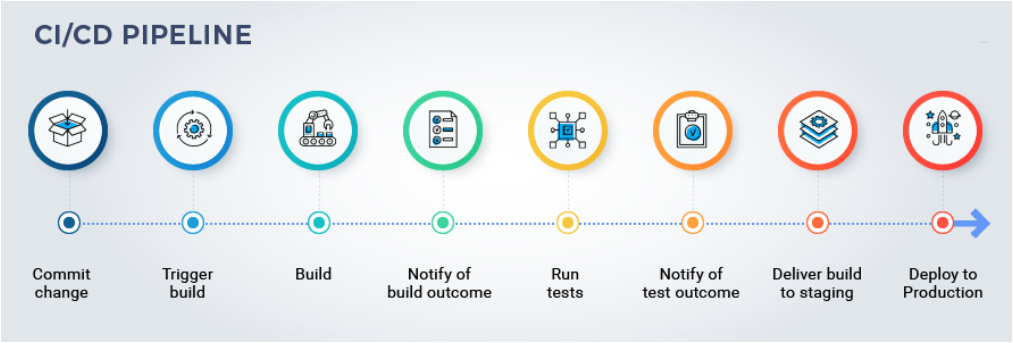
- Una estrecha relación de trabajo colaborativo entre todos los involucrados en la entrega.
- Extensa automatización de todas las partes posibles del proceso de entrega, definido completa y correctamente a través de un Deployment Pipeline.

Se debe tener en cuenta que la entrega continua no es únicamente la instalación en producción, sino la puesta en producción pasando por procesos de validación y calidad automatizados. El objetivo es por tanto que la versión que se está instalando en producción se haga de manera rápida (sin necesidad de una planificación especial), se haga de manera

eficaz (sin afectar a equipos con otras tareas planificadas) y con todas las garantías posibles (todas las comprobaciones que se puedan llevar a cabo de manera automática y asegure que no se están introduciendo nuevos errores) (Berta, 211).

Algunas grandes empresas como Amazon, Google o Twitter (entre otros) llevan años aplicando esta metodología con resultados exitosos (Swartout, 2014).

Ilustración 6 - Deployment Pipeline



Fuente: https://biplus.com.vn/ci-cd-pipeline-tools/

1.1.1.7 PRUEBAS

Existen diferentes aproximaciones o metodologías que orientan el diseño de las pruebas. Cada metodología puede usarse con los diferentes tipos de pruebas e incluso combinarse para adaptarse a las necesidades de los proyectos (Black, 2014).

- **Caja Negra:** Se evalúan los componentes como una caja negra. Solo las entradas y salidas (Black, 2014).
- **Caja Blanca:** Se evalúan los estados internos y llamados internos a otros componentes (Black, 2014).
- **Caja Gris:** Es una mezcla entre el método de caja blanca y caja negra. Básicamente se evalúa la especificación de alto nivel del sistema y los componentes y llamados internos (Black, 2014).

- **Adhoc:** Es una metodología de pruebas donde se validan los componentes sin documentación o especificación. El Ingeniero de calidad trata de encontrar bugs sin ningún plan basándose únicamente en su intuición (Black, 2014).

1.1.1.8 EFICIENCIA

Esta característica permite evaluar la relación entre el nivel de funcionamiento del software y la cantidad de recursos usados. Los aspectos a evaluar son:

- *Comportamiento con respecto al Tiempo*: Atributos del software relativos a los tiempos de respuesta y de procesamiento de los datos.
- *Comportamiento con respecto a Recursos:* Atributos del software relativos a la cantidad de recursos usados y la duración de su uso en la realización de sus funciones.

1.1.1.9 CONFIABILIDAD

La confiabilidad agrupa un conjunto de atributos que se refieren a la capacidad del software de mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo establecido (Abud Figueroa, 2012) Las sub características que el estándar sugiere son:

- *Nivel de Madurez:* Permite medir la frecuencia de falla por errores en el software.

- *Tolerancia a fallas:* Se refiere a la habilidad de mantener un nivel específico de funcionamiento en caso de fallas del software o de cometer infracciones de su interfaz específica.

- *Recuperación:* Se refiere a la capacidad de restablecer el nivel de operación y recobrar los datos que hayan sido afectados directamente por una falla, así como al tiempo y el esfuerzo necesarios para lograrlo.

1.1.1.10 CALIDAD DE SOFTWARE

Hablar de calidad del software implica la necesidad de contar con parámetros que permitan establecer los niveles mínimos que un producto de este tipo debe alcanzar para que se considere de calidad. El problema es que la mayoría de las características que definen al software no se pueden cuantificar fácilmente; generalmente, se establecen de forma cualitativa, lo que dificulta su medición, ya que se requiere establecer métricas que permitan evaluar cuantitativamente cada característica dependiendo del tipo de software que se pretende calificar (Abud Figueroa, 2012).

1.2 MARCO TEÓRICO

1.2.1 PRUEBA DE INTERFAZ DE USUARIO (UI)

La interfaz de usuario, o UI, es la plataforma que se utiliza para interactuar con un determinado software. La interfaz de usuario es el lugar en el que se pueden introducir instrucciones, introducir datos o ver información desde una pantalla o monitor (Zap Chernyak).

Las pruebas de la interfaz de usuario (UI), a veces conocidas como pruebas de la interfaz gráfica de usuario (GUI) según el contexto, son una serie de acciones utilizadas para medir el rendimiento y la funcionalidad general de los elementos visuales de una aplicación. Busca verificar y validar varias funciones de la interfaz de usuario y se asegura de que no haya resultados inesperados, defectos o errores.

Algunas de las metodologías de aproximación a las pruebas funcionales y no funcionales más comunes son las siguientes:

- ***Pruebas de regresión:*** Las pruebas de regresión son un tipo de prueba de interfaz de usuario que examina cualquier cambio en la codificación de la aplicación o el sitio web. Garantiza que toda la funcionalidad de la aplicación es la prevista después de realizar cambios en partes del código (Pressman & Maxim, 2014).

- ***Pruebas funcionales:*** Las pruebas funcionales buscan validar la aplicación para asegurarse de que cumple todos los requisitos funcionales. Este tipo de pruebas de interfaz de usuario suele centrarse en las pruebas de caja negra, que no examinan el código fuente. Suele comprobar cosas como la interfaz de usuario, cualquier API asociada, la comunicación entre cliente y servidor o la seguridad (Kaner, Falk, & Nguyen, 1999).

- ***Pruebas de aceptación:*** Las pruebas de aceptación, a veces conocidas como pruebas de aceptación del usuario (UAT), son una forma de prueba de la interfaz de usuario que realiza el usuario final de la aplicación para verificar el sistema antes de pasarlo a producción (Spillner, Linz, & Schaefer, 2014). Este tipo de prueba de la interfaz de usuario se encuentra con mayor frecuencia en las fases finales de las pruebas, una vez que se han verificado las demás áreas (Spillner, Linz, & Schaefer, 2014).

- ***Pruebas unitarias:*** Las pruebas unitarias buscan inspeccionar los componentes individuales de una aplicación para validar que funciona como se pretende. Suele realizarse durante la fase de codificación, por lo que suele recaer en los desarrolladores la realización de este tipo de pruebas de interfaz de usuario. Las pruebas unitarias consisten en separar un trozo de código para asegurarse de que funciona como se espera. Esta pieza individual de código puede ser un módulo específico, una función, un objeto o cualquier otra parte individual de la aplicación (Thomas & Hunt , 2014).

- ***Pruebas de rendimiento:*** Las pruebas de rendimiento tratan de evaluar la optimización de la aplicación, examinando aspectos como la velocidad, la estabilidad, la capacidad de respuesta y la escalabilidad de la aplicación cuando se utiliza. Este tipo de pruebas de interfaz de usuario tiene como objetivo encontrar cualquier área de preocupación en la aplicación o cuellos de botella en el flujo de datos. Las tres áreas principales que analiza son la velocidad, la escalabilidad y la estabilidad de la aplicación (Menascé, Almeida, Dowdy, & Dowdy, 2004).

- ***Prueba de la interfaz gráfica de usuario:*** Las herramientas de prueba de la interfaz gráfica de usuario (GUI) buscan inspeccionar la interfaz gráfica de usuario de una aplicación para asegurarse de que toda la funcionalidad funciona como se espera. Esto incluye el examen de los activos gráficos y los controles de la aplicación, como los botones, las barras de herramientas y los iconos. La interfaz gráfica de usuario es con lo que el usuario final interactúa y ve cuando utiliza una aplicación (Feudjio & Schieferdecker, 2009).

1.2.2 AUTOMATIZACIÓN DE PRUEBAS DE INTERFAZ DE USUARIO

La automatización de estas pruebas implica utilizar herramientas y marcos de trabajo para ejecutar casos de prueba automáticamente (The Continuous Delivery Foundation, 2021):

Beneficios:

- Repetibilidad: Las pruebas se pueden ejecutar de manera consistente.
- Eficiencia: Ahorra tiempo y recursos en comparación con las pruebas manuales.
- Detección temprana de defectos: Identifica problemas antes del despliegue.

Enfoques:

- Pruebas manuales: Realizadas por evaluadores humanos.
- Prueba de grabación y reproducción: Graba acciones y las reproduce automáticamente.
- Pruebas basadas en modelos: Utiliza modelos para generar casos de prueba.
- Escenarios de prueba de interfaz de usuario: Simula interacciones del usuario.

1.2.3 INTEGRACIÓN CON DEVOPS

La automatización de pruebas de interfaz de usuario se integra en el flujo de trabajo de DevOps. El objetivo es asegurar la calidad en todas las etapas del ciclo de vida del software, desde el desarrollo hasta el despliegue. Ejecución temprana y continua de pruebas dentro de la canalización de CI/CD2 (Hering, 2018).

1.2.4 SELECCIÓN DE HERRAMIENTAS

Elegir la herramienta de automatización adecuada es crucial.

- *Repositorio de objetos:* Facilita la reutilización de elementos de la interfaz (Garg, 2019).
- *Herramientas sin código:* Simplifican la creación de scripts (Garg, 2019).
- *Estándares de revisión de código organizacional:* Garantizan la calidad del código (Garg, 2019).

1.3 ESTADO DEL ARTE

1.3.1 DIFERENCIA ENTRE APLICACIÓN WEB TRADICIONAL Y UNA APLICACIÓN SPA

- **Navegación y flujo de la aplicación:**

  Aplicación web tradicional: En una aplicación web tradicional, cada acción del usuario (como hacer clic en un enlace) provoca una solicitud HTTP al servidor y una nueva página web se carga completamente en el navegador. Esto implica que las pruebas deben cubrir la navegación entre páginas y validar que cada página se carga correctamente. (Japikse, Grossnicklaus, & Dewey, 2017)

  Aplicación SPA: En una aplicación SPA, la interfaz de usuario se carga solo una vez y la navegación entre las diferentes secciones de la aplicación se realiza de manera dinámica mediante JavaScript y manipulación del DOM. Esto requiere pruebas para asegurar que el enrutamiento entre componentes y la actualización de la interfaz de usuario sean suaves y precisos. (Brown, 2014)

- **Interacción con el servidor:**

  Aplicación web tradicional: En una aplicación tradicional, las interacciones con el servidor son frecuentes ya que cada solicitud del usuario puede requerir una respuesta del servidor y una carga de página completa. Por lo tanto, las pruebas

deben cubrir la comunicación con el servidor, incluida la validación de respuestas y tiempos de carga. (Holdener III, 2008)

Aplicación SPA: En una SPA, las interacciones con el servidor suelen ser menos frecuentes debido al uso de tecnologías como AJAX para cargar datos de forma asíncrona. Las pruebas deben garantizar que estas interacciones asíncronas funcionen correctamente y que los datos se actualicen correctamente en la interfaz de usuario. (Ludin & Garza, 2017)

- **Estado de la aplicación:**

Aplicación web tradicional: En una aplicación tradicional, el estado de la aplicación se mantiene principalmente en el servidor, lo que significa que las pruebas deben verificar la consistencia de los datos y el estado de la sesión del usuario en el servidor. (Casciaro, 2014)

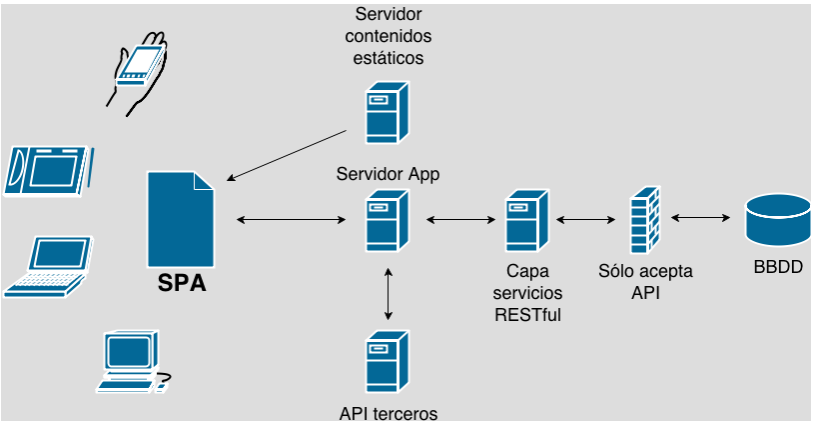
Aplicación SPA: En una SPA, gran parte del estado de la aplicación se maneja en el cliente a través de estados locales y la gestión del estado de la aplicación se vuelve más compleja. Las pruebas deben abordar la gestión adecuada del estado local, como el manejo de datos en caché, la gestión de sesiones y la consistencia de los datos locales con el servidor. (Herron, 2016)

- **Componentización y reactividad:**

Aplicación web tradicional: Las aplicaciones web tradicionales tienden a tener una estructura más monolítica, con páginas completas que se cargan y recargan. Las pruebas deben abordar la funcionalidad de cada página y su interacción con otros componentes. (Macrae, 2021)

Aplicación SPA: Las SPA se caracterizan por su enfoque en la componentización y la reactividad, lo que implica que las pruebas deben centrarse en probar componentes individuales, su comunicación entre ellos y su reactividad a los cambios de estado. (Stefanov & Freeman, 2016)

Ilustración 7 – Estructura SPA



Fuente: https://jonmircha.com/spa

1.3.2 DISEÑAR CASOS DE PRUEBA AUTOMATIZADOS PARA APLICACIONES SPA

Interacción Asincrónica: Las SPA suelen tener interacciones asincrónicas con el servidor, lo que significa que las respuestas pueden llegar en momentos no predecibles. Por lo tanto, los casos de prueba deben tener en cuenta estas interacciones y asegurarse de manejar correctamente las respuestas asincrónicas. (Mikowski & Powell, 2013)

Manejo de Estados: Las SPA pueden tener múltiples estados dentro de una sola página, como cambios de vista o actualizaciones de datos en tiempo real. Los casos de prueba deben verificar el correcto manejo de estos estados y garantizar que la interfaz de usuario refleje correctamente los cambios. (Monteiro, 2014)

Navegación y Enrutamiento: Las SPA suelen utilizar enrutadores para gestionar la navegación dentro de la aplicación. Los casos de prueba deben probar la navegación entre diferentes vistas y asegurar que el enrutamiento funcione correctamente. (Filipova, 2017)

Carga Dinámica de Contenido: Las SPA cargan contenido dinámicamente a medida que el usuario interactúa con la aplicación. Los casos de prueba deben

verificar la carga y visualización correcta de contenido dinámico, como listas o detalles de elementos. (Horton & Vice, 2016)

Gestión de Eventos: Las SPA suelen tener muchos eventos de usuario, como clics, desplazamientos, cambios de estado, etc. Los casos de prueba deben cubrir la gestión adecuada de estos eventos y garantizar que la aplicación responda correctamente a las acciones del usuario. (Wieruch, 2017)

Compatibilidad con Navegadores: Debido a la complejidad de las SPA, es importante probar la aplicación en diferentes navegadores y dispositivos para garantizar una experiencia consistente. Los casos de prueba deben abarcar la compatibilidad con diferentes navegadores y dispositivos. (Abril, 2019)

Pruebas de Rendimiento: Además de las pruebas funcionales, es crucial realizar pruebas de rendimiento para evaluar la velocidad de carga, la capacidad de respuesta y la escalabilidad de la aplicación SPA bajo diferentes cargas de usuarios. (Proskurin, 2018)

Seguridad: Las SPA también deben ser probadas en términos de seguridad, incluyendo la protección contra ataques comunes como XSS (Cross-Site Scripting) y CSRF (Cross-Site Request Forgery), así como la seguridad en la gestión de datos y sesiones. (Uluca, 2018)

1.3.3 HERRAMIENTAS Y TECNOLOGIA

- **Framework de Pruebas:**

Selenium: Selenium es un framework popular para pruebas de interfaz de usuario. Se utiliza junto con Selenium WebDriver para automatizar las acciones del usuario en un navegador web y realizar pruebas funcionales en una aplicación web SPA (Zhan, 2015).

Cypress: Cypress es un framework moderno de pruebas de extremo a extremo que se enfoca en pruebas de interfaz de usuario. Proporciona una API fácil de usar y está diseñado específicamente para aplicaciones web modernas, incluidas las SPA (Rippon, 2021).

Puppeteer: Puppeteer es una herramienta desarrollada por Google que permite controlar el navegador Chrome o Chromium. Se utiliza para automatizar tareas en el navegador, como pruebas de rendimiento y pruebas de interfaz de usuario en aplicaciones SPA (Booth, 2019).

- **Herramientas de Pruebas de Carga y Rendimiento:**

***JMeter*:** Apache JMeter es una herramienta de pruebas de carga y rendimiento que se puede utilizar para simular la carga de usuarios en una aplicación SPA y medir su rendimiento bajo diferentes condiciones (Gomes Rodrigues, 2019).

Gatling: Gatling es otra herramienta de pruebas de carga y rendimiento que permite simular la carga de usuarios concurrentes en una aplicación web y analizar su rendimiento y capacidad de respuesta (John C. & Tragura, 2015).

- **Librerías de Pruebas Unitarias y de Integración:**

Jest: Jest es una librería de pruebas unitarias desarrollada por Facebook. Se utiliza principalmente para pruebas de componentes individuales en aplicaciones React, incluidas las SPA (Chinnathambi, 2018).

JUnit y TestNG: Estas son librerías de pruebas unitarias para Java que también se pueden utilizar para pruebas de integración en aplicaciones SPA que utilizan tecnologías como Spring Boot y Angular (Menon, 2013).

- **Herramientas de Virtualización y Contenedores:**

Docker: Docker se utiliza para crear y gestionar contenedores que pueden contener aplicaciones web y sus dependencias. Se utiliza en entornos de pruebas para crear

entornos aislados y reproducibles para las pruebas automatizadas de aplicaciones SPA (Tankersley, 2019).

Kubernetes: Kubernetes es una herramienta de orquestación de contenedores que se utiliza para gestionar y escalar contenedores en un clúster. Se puede utilizar para desplegar y gestionar entornos de pruebas automatizadas para aplicaciones SPA (Poulton, 2020).

- **Herramientas de Integración Continua (CI) y Entrega Continua (CD):**

  Jenkins: Jenkins es una herramienta de CI/CD que se utiliza para automatizar el proceso de construcción, pruebas y despliegue de aplicaciones. Se puede integrar con herramientas de pruebas automatizadas para ejecutar pruebas automáticamente en cada ciclo de integración (Smart, 2011).

  Travis CI: Travis CI es otra herramienta de CI/CD que se utiliza para construir y probar aplicaciones automáticamente en un entorno controlado antes de desplegarlas en producción (Salunke, 2021).

1.3.4 DEVOPS ENTREGA E INTEGRACION CONTINUA

La integración continua (CI) y la entrega continua (CD) son prácticas clave en DevOps que juegan un papel fundamental en la automatización de pruebas para aplicaciones SPA (Single Page Application). Se detalla cómo estas prácticas se integran y su impacto en las pruebas automatizadas:

- **Integración Continua (CI):**

  Automatización de Builds: Con CI, se automatizan los builds de la aplicación cada vez que se realiza un cambio en el código fuente. Esto garantiza que las nuevas funcionalidades se integren correctamente con el código existente y se detecten rápidamente posibles conflictos o errores (Duvall, Matyas, & Glover, 2007).

Ejecución Automatizada de Pruebas: La CI permite ejecutar automáticamente una suite de pruebas cada vez que se realiza un build. Esto incluye pruebas unitarias, pruebas de integración y pruebas de extremo a extremo para garantizar que las nuevas funcionalidades no introduzcan regresiones en la aplicación (Kawalerowicz & Berntson, 2011).

Retroalimentación Inmediata: Al ejecutar las pruebas automáticamente como parte del proceso de CI, se obtiene una retroalimentación inmediata sobre la calidad del código. Esto ayuda a identificar y corregir errores de forma temprana, reduciendo el tiempo y los costos asociados con la resolución de problemas más adelante en el ciclo de desarrollo (Smith, 2017).

- **Entrega Continua (CD):**

  Automatización de Despliegues: Con CD, se automatizan los despliegues de la aplicación en entornos de desarrollo, pruebas y producción. Esto incluye la configuración de servidores, la instalación de dependencias y la ejecución de scripts de despliegue (Humble & Farley, 2011).

  Pruebas de Aceptación Automatizadas: La CD también permite ejecutar pruebas de aceptación automatizadas en entornos similares al de producción. Esto asegura que la aplicación funcione correctamente en un entorno real antes de ser desplegada a los usuarios finales (Leszko, 2018).

  Retroalimentación Rápida y Continua: La CD proporciona una retroalimentación continua sobre el estado de la aplicación en diferentes entornos. Esto permite identificar problemas de integración, rendimiento o funcionalidad de manera rápida y continua, lo que mejora la calidad y confiabilidad de la aplicación final (Forsgren, Humble, & Kim , 2018).

1.3.5 DOCUMENTACIÓN Y GESTIÓN DE LOS RESULTADOS

En un entorno DevOps, la documentación y gestión de los resultados de las pruebas automatizadas son aspectos cruciales para garantizar la eficiencia y calidad del proceso. Aquí algunas prácticas comunes para documentar y gestionar estos resultados:

Generación de Informes Automatizados: Utilizar herramientas de automatización de pruebas que generen informes detallados automáticamente al finalizar las pruebas. Estos informes deben incluir resultados de pruebas, cobertura de código, métricas de rendimiento, errores encontrados y más (Kim, Debois, Willis , & Humble, 2016).

Integración con Herramientas de Gestión de Proyectos: Integrar los resultados de las pruebas automatizadas con herramientas de gestión de proyectos como Jira, Trello o Asana. Esto permite vincular los errores encontrados en las pruebas con tareas específicas y realizar un seguimiento más eficiente de su resolución (Kim, Debois, Willis , & Humble, 2016).

Almacenamiento en Repositorios de Código: Almacenar los resultados de las pruebas junto con el código fuente en repositorios de control de versiones como Git. Esto facilita la colaboración entre equipos y proporciona un historial de cambios que puede ser útil para futuras referencias (Kim, Debois, Willis , & Humble, 2016).

Utilización de Dashboards de Pruebas*:* Crear dashboards o paneles de control que muestren de manera visual el estado de las pruebas automatizadas. Estos dashboards pueden incluir gráficos de tendencias, métricas de calidad, errores críticos y otras estadísticas relevantes (Davis & Daniels, 2016).

Notificaciones Automáticas: Configurar notificaciones automáticas para alertar al equipo sobre los resultados de las pruebas. Esto puede incluir correos electrónicos, mensajes en canales de comunicación interna o integraciones con herramientas de mensajería como Slack (Davis & Daniels, 2016).

Revisión y Análisis Continuo: Establecer procesos para revisar y analizar regularmente los resultados de las pruebas automatizadas. Esto ayuda a identificar patrones, tendencias y áreas de mejora en el proceso de desarrollo y pruebas (Davis & Daniels, 2016).

Registro de Incidentes: Registrar de manera detallada los incidentes encontrados durante las pruebas automatizadas, incluyendo información como pasos para reproducir el problema, capturas de pantalla y prioridad de resolución (Kim, Debois, Willis , & Humble, 2016).

1.4 MARCO CONTEXTUAL

El método contextual se centra en comprender el entorno y las condiciones específicas en las que se llevarán a cabo las pruebas automatizadas. Esto implica considerar factores como el tipo de aplicación web (SPA), los requerimientos de DevOps, las herramientas de automatización disponibles, las capacidades del equipo de desarrollo, entre otros aspectos.

Algunas actividades que se pueden llevar a cabo en el método contextual incluyen:

- Análisis del entorno de desarrollo y operación de las aplicaciones SPA.
- Evaluación de los procesos de desarrollo y entrega continua (CI/CD) de DevOps.
- Identificación de requisitos y características específicas de las pruebas de interfaz de usuario en el contexto de aplicaciones SPA y DevOps.
- Selección de herramientas y tecnologías adecuadas para la automatización de pruebas en este entorno.
- Definición de estrategias y prácticas de pruebas que se alineen con los objetivos y procesos de DevOps.
- Establecimiento de métricas y criterios de evaluación para medir la eficiencia y efectividad de las pruebas automatizadas en el contexto de DevOps.

CAPÍTULO II

DIAGNOSTICO

2.1 INTRODUCCIÓN

Para realizar un diagnóstico del "Proceso de Automatización de Pruebas de Interfaz de Usuario para Aplicaciones Web de Arquitectura SPA (Single-Page Application) con DevOps" el instrumento de recolección de datos que aplicaremos en esta monografía es la revisión de documentos para recopilar información.

2.1.1 PROCESAMIENTO Y ANALISIS DE DATOS

2.1.1.1 FORMULACION DEL PROBLEMA

Para cada objetivo específico se plantearon las siguientes preguntas de investigación, así:

Objetivo Especifico 1

Observar las características y requerimientos específicos de las aplicaciones web de arquitectura SPA (Single Page Application) para las pruebas automatizadas.

Preguntas de Investigación Objetivo Especifico 1

- ¿Qué diferencias existen en los requerimientos de pruebas automatizadas entre una aplicación web tradicional y una aplicación SPA?
- ¿Qué requerimientos específicos deben considerarse al diseñar casos de prueba automatizados para aplicaciones SPA?

Acciones para dar respuesta a las preguntas de investigación Especifico 1

Revisión bibliográfica y documentación del estado del arte.

Objetivo Especifico 2

Revisar herramientas y *frameworks* de automatización de pruebas adecuados para aplicaciones web SPA (Single Page Application).

Preguntas de Investigación Objetivo Especifico 2

- ¿Cuáles son las principales herramientas y tecnologías utilizadas para realizar pruebas automatizadas en aplicaciones SPA?

Acciones para dar respuesta a las preguntas de investigación Especifico 2

Revisión bibliográfica y documentación del estado del arte.

Objetivo Especifico 3

Comparar las pruebas automatizadas en los procesos de integración continua y entrega continua (CI/CD) de DevOps.

Preguntas de Investigación Objetivo Especifico 3

¿Qué papel juega la integración continua (CI) y la entrega continua (CD) en la automatización de pruebas para aplicaciones SPA con prácticas Devops?

Acciones para dar respuesta a las preguntas de investigación Especifico 3

Revisión bibliográfica y documentación del estado del arte.

Objetivo Especifico 4

Documentar el proceso de automatización de pruebas SPA con DevOps.

Preguntas de Investigación Objetivo Especifico 4

¿Cómo se documentan y gestionan los resultados de las pruebas automatizadas en un entorno DevOps?

Acciones para dar respuesta a las preguntas de investigación Especifico 3

Revisión bibliográfica y documentación del estado del arte. Examina las prácticas de documentación de resultados de pruebas, gestión de incidencias, generación de informes y seguimiento de métricas clave en un contexto de DevOps.

2.1.1.2 REVISIÓN BIBLIOGRÁFICA

Se recopila información existente sobre la automatización de pruebas de interfaz de usuario en aplicaciones web SPA con DevOps, utilizando diversas fuentes como sitios web, revistas, artículos científicos, libros y trabajos académicos. Esta investigación ofrece una visión actualizada del estado del tema o problema seleccionado.

Tabla 2 – Revisión Bibliográfica

Título	Autor	Enfoque	Editorial / Institución
Calidad en la Industria del Software. La Norma ISO-9126	Abud Figueroa, M. A.	Normas de calidad en software	nacionmulticultural.unam.mx
Modern Web Development with React	Abril, A. (2019)	Desarrollo web moderno con React	Packt Publishing
Incorporación de la integración Continua en el Desarrollo de Software	Berta, O. (2011)	Integración continua en el desarrollo de software	Universidad de Piura
Advanced Software Testing	Black, R. (2014)	Pruebas de software avanzadas	Rocky Nook
Puppeteer Succinctly	Booth, J. (2019)	Uso de Puppeteer para pruebas de interfaz	Syncfusion Inc.
Information Systems Architecture and Technology	Borzemski, L., Grzech, A., Świątek, J., & Wilimowska, Z. (2017)	Arquitectura y tecnología de sistemas de información	37th International Conference on Information Systems Architecture and Technology – ISAT 2016
Web Development with Node and Express	Brown, E. (2014)	Desarrollo web con Node.js y Express	O'Reilly Media
Node.js Design Patterns	Casciaro, M. (2014)	Patrones de diseño en Node.js	Packt Publishing
Learning React: A Hands-On Guide to Building Web Applications Using React and Redux	Chinnathambi, K. (2018)	Aprendizaje de React para desarrollo web	Addison-Wesley Professional
Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale	Davis, J., & Daniels, R. (2016)	Prácticas y cultura DevOps	O'Reilly Media

Fuente: *Elaboración propia*

2.1.1.3 MODELO PROPUESTO PARA ADOPCIÓN DE PRACTICAS DEVOPS PARA PRUEBAS DE INTERFACE DE USUARIO EN APLICACIONES SPA

La implementación de las prácticas DevOps para las pruebas de interface de usuario en aplicaciones SPA, se pretende contribuir con conocimiento capaz de orientar en los aspectos claves para abordar un proceso de implementación de buenas prácticas DevOps. Los elementos aquí expuestos son resultado de:

Tabla 3 – Adopción de Practicas DevOps

Elemento	Descripción
Origen	Revisión bibliográfica durante la monografía, experiencias documentadas y publicadas por profesionales, experiencia en modelos de gestión y estándares en TI.
Alcance	Prácticas técnicas de DevOps orientadas específicamente a pruebas de interfaz de usuario en aplicaciones SPA.
Inclusión de prácticas	Incluye prácticas específicas relacionadas con la automatización de pruebas de interfaz de usuario en aplicaciones SPA.
Exclusión de prácticas	No aborda prácticas relacionadas con arquitectura, gestión de procesos y productos de software, monitorización y cultura DevOps.
Enfoque	Orientado a proporcionar conocimientos clave para abordar un proceso de implementación de buenas prácticas DevOps en pruebas de interfaz de usuario.

Fuente: *Elaboración propia*

2.1.1.4 SELECCIONAR UNA APLICACIÓN PARA IMPLEMENTAR LAS PRÁCTICAS DEVOPS.

Para implementar prácticas DevOps, es crucial un cambio cultural y comenzar con una aplicación que permita experimentar y aprender, mostrar resultados rápidos y tener visibilidad. Las aplicaciones SPA son ideales debido a su flexibilidad y potencial para implementar prácticas como gestión de la configuración, integración continua, pruebas y despliegue continuo.

Tabla 4 – Criterio de Selección de Aplicación SPA

Criterio	Ventajas de seleccionar una aplicación SPA para implementar prácticas DevOps
Posibilidad de experimentación y aprendizaje	Facilidad para probar nuevas prácticas y experimentar con diferentes enfoques en un entorno menos rígido y establecido.
Capacidad de mostrar logros en el corto plazo	Mayor probabilidad de obtener resultados tangibles en un tiempo más corto debido a la naturaleza ágil y modular de las SPA.
Visibilidad para generar confianza y extrapolación	Mejor visibilidad de los resultados y procesos de implementación de DevOps, lo que aumenta la confianza y facilita la replicación en otros proyectos.
Potencial para implementar prácticas de DevOps	La simplicidad de las SPA permite una implementación más rápida y efectiva de prácticas como la gestión de la configuración, integración continua, pruebas continuas y despliegue continuo.

Fuente: *Elaboración propia*

2.1.2 TABULACIÓN Y CODIFICACIÓN DE DATOS

Tabla de tabulación y codificación de datos para la diferencia entre una aplicación web tradicional y una aplicación SPA:

Tabla 5 - Diferencia entre Aplicación Web Tradicional y SPA

Categoría	Aplicación Web Tradicional	Aplicación Web SPA
Arquitectura	Multi-página con múltiples cargas	Única página, carga dinámica de contenido
Interacción con el servidor	Recarga de página completa en cada solicitud	Intercambio de datos por medio de API o AJAX
Experiencia del usuario	Menor rapidez y fluidez, más interrupciones	Mayor rapidez, sensación de aplicación nativa
Desarrollo y mantenimiento	Mayor tiempo de desarrollo y mantenimiento	Menor tiempo de desarrollo y mantenimiento
Recursos requeridos	Mayor carga de recursos al cargar páginas completas	Menor carga de recursos debido a la carga dinámica
SEO	Menos optimizado para motores de búsqueda	Mejor optimizado para motores de búsqueda

Fuente: *Elaboración propia*

Tabla diseño de casos de prueba automatizados para aplicaciones SPA:

Tabla 6 - Pruebas automatizados SPA

Categoría	Descripción	Codificación
Interacción Asincrónica	Pruebas para garantizar que las respuestas asincrónicas del servidor se manejen correctamente.	IA-01, IA-02, ...
Manejo de Estados	Verificación del manejo adecuado de múltiples estados dentro de la SPA.	ME-01, ME-02, ...
Navegación y Enrutamiento	Pruebas para comprobar la navegación entre vistas y el correcto funcionamiento del enrutamiento.	NE-01, NE-02, ...
Carga Dinámica de Contenido	Verificación de la carga y visualización correcta del contenido dinámico.	CDC-01, CDC-02, ...
Gestión de Eventos	Pruebas para evaluar la gestión de eventos de usuario y la respuesta de la aplicación.	GE-01, GE-02, ...
Compatibilidad con Navegadores	Pruebas para asegurar la compatibilidad de la SPA en diferentes navegadores y dispositivos.	CN-01, CN-02, ...
Pruebas de Rendimiento	Evaluación de la velocidad de carga, capacidad de respuesta y escalabilidad de la SPA.	PR-01, PR-02, ...
Seguridad	Pruebas para garantizar la seguridad de la SPA contra ataques como XSS y CSRF.	SE-01, SE-02, ...

Fuente: *Elaboración propia*

Tabla de tabulación y codificación de datos para las herramientas y tecnologías utilizadas en el proceso de automatización de pruebas de interfaz de usuario para aplicaciones SPA:

Tabla 7 - Herramientas pruebas IU SPA

Categoría	Herramienta	Descripción
Framework de Pruebas	Selenium	Framework popular para pruebas de interfaz de usuario, utilizado con Selenium WebDriver para automatizar acciones del usuario en un navegador web.
	Cypress	Framework moderno de pruebas de extremo a extremo, enfocado en pruebas de interfaz de usuario para aplicaciones web modernas, incluidas las SPA.
	Puppeteer	Herramienta de control de navegador desarrollada por Google, utilizada para automatizar tareas en el navegador, como pruebas de rendimiento en aplicaciones SPA.
Herramientas de Pruebas de Carga / Rendimiento	JMeter	Herramienta de pruebas de carga y rendimiento para simular la carga de usuarios y medir el rendimiento de una aplicación SPA bajo diferentes condiciones.

	Gatling	Herramienta de pruebas de carga y rendimiento para simular la carga de usuarios concurrentes y analizar el rendimiento y capacidad de respuesta en aplicaciones web SPA.
Librerías de Pruebas Unitarias / Integración	Jest	Librería de pruebas unitarias para componentes individuales en aplicaciones React, incluidas las SPA.
	JUnit y TestNG	Librerías de pruebas unitarias e integración para Java, utilizadas en aplicaciones SPA que emplean tecnologías como Spring Boot y Angular.
Herramientas de Virtualización / Contenedores	Docker	Herramienta para crear y gestionar contenedores que contienen aplicaciones web y sus dependencias, utilizada para pruebas aisladas y reproducibles de aplicaciones SPA.
	Kubernetes	Herramienta de orquestación de contenedores para gestionar y escalar contenedores en un clúster, utilizada para desplegar entornos de pruebas automatizadas en aplicaciones SPA.
Herramientas de Integración Continua (CI) / Entrega Continua (CD)	Jenkins	Herramienta de CI/CD para automatizar procesos de construcción, pruebas y despliegue de aplicaciones, integrada con pruebas automatizadas en cada ciclo de integración.
	Travis CI	Herramienta de CI/CD para construir y probar aplicaciones automáticamente en un entorno controlado antes de desplegarlas en producción.

Fuente: *Elaboración propia*

Tabla de tabulación y codificación de datos para la sección DevOps entrega e integración continua

Tabla 8 - Devops entrega e integración

Categoría	Aspecto	Descripción
Integración Continua (CI)	Automatización de Builds	Automatización de los builds de la aplicación cada vez que se realiza un cambio en el código fuente.
	Ejecución Automatizada de Pruebas	Ejecución automática de una suite de pruebas cada vez que se realiza un build.
	Retroalimentación Inmediata	Obtención de retroalimentación inmediata sobre la calidad del código.
Entrega Continua (CD)	Automatización de Despliegues	Automatización de los despliegues en diferentes entornos.
	Pruebas de Aceptación Automatizadas	Ejecución automática de pruebas de aceptación en entornos similares a producción.
	Retroalimentación Rápida y Continua	Obtención de retroalimentación continua sobre el estado de la aplicación en diferentes entornos.
Documentación y Gestión de Datos	Generación de Informes Automatizados	Utilización de herramientas para generar informes detallados automáticamente al finalizar las pruebas.
	Integración con Herramientas de Proyectos	Integración de los resultados de las pruebas con herramientas de gestión de proyectos.

	Almacenamiento en Repositorios de Código	Almacenamiento de resultados de pruebas junto con el código fuente en repositorios de control de versiones.
	Utilización de Dashboards de Pruebas	Creación de dashboards visuales que muestren el estado de las pruebas automatizadas.
	Notificaciones Automáticas	Configuración de notificaciones automáticas para alertar al equipo sobre resultados de pruebas.
	Revisión y Análisis Continuo	Establecimiento de procesos para revisar y analizar regularmente resultados de pruebas automatizadas.
	Registro de Incidentes	Registro detallado de incidentes encontrados durante las pruebas automatizadas.

Fuente: *Elaboración propia*

2.1.3 ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Objetivo Especifico 1

> Observar las características y requerimientos específicos de las aplicaciones web de arquitectura SPA (Single Page Application) para las pruebas automatizadas.

La revisión bibliográfica y las características de SPA proporciona una visión detallada de las diferencias y los requerimientos específicos de pruebas automatizadas entre una aplicación web tradicional y una aplicación SPA (Single Page Application). Esto reflejan la complejidad y las especificidades que deben considerarse al diseñar casos de prueba automatizados para aplicaciones SPA en comparación con aplicaciones web tradicionales. La componentización, la interacción asíncrona con el servidor y la gestión del estado local son aspectos clave que requieren atención en las pruebas automatizadas de aplicaciones SPA.

Objetivo Especifico 2

> Revisar herramientas y *frameworks* de automatización de pruebas adecuados para aplicaciones web SPA (Single Page Application).

Frameworks de Pruebas: Se escogen Selenium, Cypress y Puppeteer sobre otras herramientas debido a su capacidad para abordar diferentes aspectos y requerimientos de

las pruebas en aplicaciones SPA, desde la compatibilidad y flexibilidad de Selenium, la eficiencia y facilidad de uso de Cypress, hasta el control preciso y capacidades de rendimiento de Puppeteer. Estas herramientas ofrecen un conjunto completo de funcionalidades que se adaptan bien a los principios y prácticas de automatización de pruebas en entornos DevOps.

Tabla 9 – Frameworks de Pruebas

Categoría	Selenium	Cypress	Puppeteer
Descripción	Framework para pruebas de interfaz de usuario, se utiliza con Selenium WebDriver.	Framework moderno de pruebas de extremo a extremo enfocado en aplicaciones web modernas.	Herramienta de Google para controlar el navegador Chrome o Chromium.
Uso Principal	Automatización de acciones del usuario en un navegador web y pruebas funcionales en SPA.	Pruebas de extremo a extremo en SPA.	Pruebas de rendimiento y de interfaz de usuario en SPA.
Facilidad de Uso	Ampliamente utilizado con una curva de aprendizaje moderada.	API fácil de usar y diseñado específicamente para aplicaciones web modernas.	Facilidad de uso con control directo sobre el navegador.
Compatibilidad	Amplia compatibilidad con navegadores web.	Compatibilidad con navegadores modernos.	Controla el navegador Chrome o Chromium.
Ventajas	Soporte para múltiples lenguajes de programación y amplia comunidad.	Enfoque en pruebas end-to-end y simplicidad en la escritura de pruebas.	Control directo sobre el navegador y pruebas de rendimiento detalladas.
Desventajas	Configuración inicial más compleja para entornos complejos.	Menos flexibilidad en algunos aspectos comparado con Selenium.	Menos soporte de la comunidad en comparación con Selenium.
Ejemplo de Uso	Automatización de formularios y acciones de usuario en SPA.	Pruebas de navegación y validación de elementos en SPA.	Automatización de tareas de navegación y pruebas de carga en SPA.

Fuente: *Elaboración propia*

Herramientas de Pruebas de Carga y Rendimiento: JMeter y Gatling son escogidas sobre otras herramientas similares debido a su amplia adopción, versatilidad, facilidad de uso, funcionalidades avanzadas y capacidad para simular escenarios de carga realistas en el contexto de aplicaciones SPA. Estas herramientas son fundamentales para evaluar el rendimiento, la capacidad de respuesta y la estabilidad de las aplicaciones bajo

condiciones de carga simuladas, lo que es crucial en el desarrollo y despliegue de aplicaciones en entornos DevOps.

Tabla 10 - Herramientas de Pruebas de Carga y Rendimiento

Categoría	JMeter	Gatling
Descripción	Herramienta de pruebas de carga y rendimiento para simular la carga de usuarios en SPA.	Herramienta de pruebas de carga y rendimiento para simular la carga de usuarios concurrentes en SPA.
Uso Principal	Simular la carga de usuarios y medir el rendimiento en SPA bajo diferentes condiciones.	Simular la carga de usuarios concurrentes y analizar el rendimiento y capacidad de respuesta en SPA.
Facilidad de Uso	Interfaz gráfica intuitiva pero con opciones avanzadas.	Configuración basada en código con opciones avanzadas para pruebas complejas.
Compatibilidad	Amplia compatibilidad con sistemas y protocolos web.	Amplia compatibilidad con protocolos web y facilidad de integración con sistemas existentes.
Ventajas	Soporte para escenarios complejos de pruebas de carga.	Escalabilidad y rendimiento en la simulación de cargas pesadas.
Desventajas	Mayor curva de aprendizaje y configuración inicial.	Mayor esfuerzo de configuración para escenarios muy complejos.
Ejemplo de Uso	Simulación de cientos de usuarios concurrentes en una SPA y medición del rendimiento.	Análisis del rendimiento bajo diferentes condiciones de carga en una SPA.

Fuente: *Elaboración propia*

Librerías de Pruebas Unitarias/Integración: Jest se elige por su afinidad con aplicaciones basadas en React, su facilidad de uso y su amplio conjunto de funcionalidades, mientras que JUnit y TestNG son preferidos por su compatibilidad con Java, su historial de uso y su integración con herramientas de CI/CD, lo que los hace a ambas herramientas sólidas y confiables para la automatización de pruebas en aplicaciones SPA con DevOps.

Tabla 11 - Librerías de Pruebas Unitarias/Integración

Categoría	Jest	JUnit y TestNG
Descripción	Librería de pruebas unitarias desarrollada por Facebook para aplicaciones React, incluidas las SPA.	Librerías de pruebas unitarias para Java, utilizadas también en pruebas de integración en SPA.
Uso Principal	Pruebas unitarias de componentes individuales en SPA.	Pruebas unitarias e integración en aplicaciones SPA con tecnologías como Spring Boot y Angular.
Facilidad de Uso	Fácil de usar con integración directa en proyectos de React.	Ampliamente utilizadas con una curva de aprendizaje moderada.
Compatibilidad	Enfoque en aplicaciones React pero adaptable a otras tecnologías.	Amplia compatibilidad con aplicaciones Java y *frameworks* como Spring Boot y Angular.
Ventajas	Integración directa con proyectos de React y simplicidad en la escritura de pruebas.	Amplio soporte para pruebas unitarias e integración en aplicaciones Java.
Desventajas	Limitaciones en el soporte fuera del ecosistema React.	Curva de aprendizaje moderada para principiantes.
Ejemplo de Uso	Pruebas unitarias de componentes React en una SPA.	Pruebas de integración en una SPA utilizando Spring Boot y TestNG.

Fuente: *Elaboración propia*

Herramientas de Virtualización/Contenedores: Docker y Kubernetes son elegidos sobre otras herramientas de virtualización y contenedores debido a su amplia adopción, características avanzadas de orquestación, eficiencia en el uso de recursos, automatización de operaciones y capacidad para proporcionar entornos de prueba escalables, fiables y consistentes en el contexto de DevOps y pruebas automatizadas de aplicaciones SPA.

Tabla 12 - Herramientas de Virtualización/Contenedores

Categoría	Docker	Kubernetes
Descripción	Herramienta para crear y gestionar contenedores de aplicaciones web y sus dependencias.	Herramienta de orquestación de contenedores para gestionar y escalar contenedores en un clúster.
Uso Principal	Creación de entornos aislados y reproducibles para pruebas automatizadas en SPA.	Gestión de entornos de pruebas automatizadas para aplicaciones SPA en un clúster escalable.

Facilidad de Uso	Fácil de usar para la creación y gestión de contenedores en entornos de pruebas.	Requiere conocimientos avanzados para la configuración y gestión de clústeres de contenedores.
Compatibilidad	Amplia compatibilidad con sistemas operativos y plataformas.	Integración con diferentes proveedores de servicios en la nube y sistemas operativos.
Ventajas	Creación rápida de entornos aislados y reproducibles para pruebas.	Escalabilidad y gestión eficiente de contenedores en un clúster.
Desventajas	Curva de aprendizaje inicial para usuarios nuevos en contenedores.	Configuración y gestión avanzada requerida para entornos complejos.
Ejemplo de Uso	Creación de entornos de pruebas aislados para ejecutar pruebas automatizadas en SPA.	Gestión de entornos de pruebas automatizadas en un clúster utilizando Kubernetes.

Fuente: *Elaboración propia*

Objetivo Especifico 3

Comparar las pruebas automatizadas en los procesos de integración continua y entrega continua (CI/CD) de DevOps.

Comparando las pruebas automatizadas en los procesos de integración continua (CI) y entrega continua (CD) de DevOps para aplicaciones SPA (Single Page Applications) con prácticas DevOps implica examinar varios aspectos clave. Aquí se presenta un enfoque estructurado para responder a la pregunta al objetivo específico:

Automatización de Pruebas en CI y CD:

En la integración continua (CI), la automatización de pruebas se centra en ejecutar pruebas unitarias, pruebas de integración y pruebas funcionales básicas para validar la estabilidad del código cada vez que se realiza una integración de cambios. Esto asegura la detección temprana de errores.

La entrega continua (CD), la automatización de pruebas se expande para incluir pruebas más exhaustivas, como pruebas de aceptación del usuario, pruebas de carga y rendimiento,

y pruebas de seguridad. Estas pruebas garantizan que la aplicación sea desplegada de manera confiable y con un alto nivel de calidad.

Cobertura de Pruebas:

En CI, la cobertura de pruebas se enfoca en verificar la funcionalidad básica y la integración de los componentes del software. Esto se refleja en una cobertura de código más limitada pero fundamental para garantizar la estabilidad de las integraciones.

En CD, la cobertura de pruebas se amplía para abarcar una variedad de escenarios, incluyendo casos de uso del usuario final, rendimiento bajo carga y vulnerabilidades de seguridad. Esto conduce a una cobertura de código más completa y robusta.

Tiempo y Eficiencia:

En términos de tiempo de ejecución, las pruebas automatizadas en CI suelen ser más rápidas y centradas en la validación rápida de cambios de código. Esto permite una retroalimentación inmediata al equipo de desarrollo.

En CD, si bien las pruebas pueden llevar más tiempo debido a la amplitud de los escenarios cubiertos, su automatización permite mantener un proceso de entrega rápido y confiable. La eficiencia radica en la detección temprana de problemas potenciales que podrían impactar la experiencia del usuario final.

Feedback y Mejoras Continuas:

En CI como en CD, el feedback generado por las pruebas automatizadas es fundamental. En CI, se detectan rápidamente errores de integración y se resuelven antes de avanzar en el proceso de desarrollo. En CD, el feedback se extiende a aspectos como rendimiento, seguridad y experiencia del usuario, lo que impulsa mejoras continuas en la aplicación.

Roles Complementarios de CI y CD:

La integración continua (CI) se enfoca en la validación temprana del código y la integración fluida de cambios, mientras que la entrega continua (CD) amplía este enfoque para garantizar una aplicación confiable, segura y eficiente en su despliegue y funcionamiento.

Al evaluar estos aspectos, se evidencia que la combinación de la integración continua y la entrega continua desempeña un papel crítico en la automatización de pruebas para aplicaciones SPA bajo prácticas DevOps, garantizando la calidad, confiabilidad y eficiencia del proceso de desarrollo y entrega.

Objetivo Especifico 4

Documentar el proceso de automatización de pruebas SPA con Devops.

La documentación del proceso de automatización de pruebas en aplicaciones SPA con DevOps es crucial para garantizar la comprensión, transparencia y mejora continua de este proceso. La documentación detallada proporciona un marco de referencia claro para todo el equipo de desarrollo y ayuda a mantener la calidad y confiabilidad de la aplicación a lo largo del tiempo.

2.2 CONCLUSIONES Y RECOMENDACIONES

2.2.1 CONCLUSIONES

Según el Objetivo General, en esta monografía se evidencio, por medio de una investigación teórica justificada con textos académicos de manera positiva, se determinó el proceso de automatización de pruebas de interfaz de usuario para aplicaciones web de arquitectura SPA(Single Page Application) es esencial para abordar la complejidad y los cambios frecuentes inherentes a este tipo de aplicaciones, la integración de estas pruebas en un enfoque DevOps amplía estos beneficios garantiza una entrega continua de alta calidad y fomenta una colaboración efectiva entre los equipos de desarrollo y operaciones.

Según el Objetivo Específico 1, sobre las características y requerimientos específicos de las aplicaciones web de arquitectura SPA (Single Page Application) para las pruebas automatizadas se cumplió nuestro objetivo con la revisión bibliográfica y el análisis de las características de las SPA nos proporcionó una visión detallada de las diferencias y los requerimientos específicos de las pruebas automatizadas en comparación con las aplicaciones web tradicionales. Las SPA presentan una serie de desafíos y particularidades que deben ser considerados al diseñar casos de prueba automatizados. La componentización, la interacción asíncrona con el servidor y la gestión del estado local son aspectos clave que requieren una atención especial en las pruebas automatizadas de aplicaciones SPA.

Según el Objetivo Específico 2, sobre la revisión de herramientas y frameworks de automatización de pruebas adecuados para aplicaciones web SPA (Single Page Application), se cumplió este objetivo basado el criterios específicos y adaptabilidad a entornos DevOps. Los frameworks Selenium, Cypress y Puppeteer se destacan para pruebas de interfaz de usuario, mientras que JMeter y Gatling se eligen para pruebas de carga y rendimiento. Las librerías Jest, JUnit y TestNG son preferidas para pruebas unitarias/integración, y Docker y Kubernetes se seleccionan como herramientas de virtualización y contenedores. Estas elecciones se basan en capacidades específicas y compatibilidad con los principios de automatización de pruebas en el contexto de DevOps y aplicaciones SPA.

Según el Objetivo Específico 3, se cumplió el objetivo con la comparación de pruebas automatizadas en CI y CD de DevOps para aplicaciones SPA revela diferencias complementarias. En CI, las pruebas se enfocan en estabilidad y detección temprana de errores, mientras que en CD se amplían para incluir pruebas exhaustivas como de carga y seguridad. La cobertura en CI es básica pero crucial, mientras que en CD es más completa. En términos de tiempo, CI es rápido y ágil, mientras que CD es más amplio pero confiable. El feedback de las pruebas impulsa mejoras continuas. En resumen, CI valida el código tempranamente, CD garantiza una aplicación confiable, y la combinación de ambos asegura calidad y eficiencia.

Según el Objetivo Específico 4, la documentación del proceso de automatización de pruebas en aplicaciones SPA con DevOps es esencial para asegurar la comprensión, transparencia y mejora constante del proceso. Proporciona un marco claro para el equipo de desarrollo y contribuye a mantener la calidad y confiabilidad de la aplicación a largo plazo.

2.2.2 RECOMENDACIONES

Se recomienda desde una perspectiva académica la Integración de Inteligencia Artificial en pruebas de interfaz de usuario para explorar cómo estas técnicas pueden mejorar la detección de errores y la eficiencia en las pruebas de aplicaciones SPA, incluyendo el desarrollo de modelos predictivos y la automatización de casos de prueba.

Se recomienda también la Seguridad en Pruebas Automatizadas para profundizar en el desarrollo de herramientas y técnicas para integrar pruebas de seguridad automatizadas de manera efectiva en DevOps, centrándose en la detección de vulnerabilidades específicas de SPA y la evaluación de riesgos en tiempo real.

Desde una perspectiva práctica o de implementación en empresas se recomienda el Desarrollo de Dashboards de Pruebas Automatizadas para implementar dashboards interactivos que muestren métricas clave de pruebas automatizadas en tiempo real, como cobertura de pruebas, resultados de pruebas, tendencias de errores y calidad del código.

Con estas recomendaciones se buscan extender el estudio hacia áreas más avanzadas y ofrecer enfoques prácticos y concretos para la implementación de prácticas DevOps en pruebas de interfaz de usuario para aplicaciones SPA.

REFERENCIAS BIBLIOGRÁFICAS

Abud Figueroa, M. A. (2012). *Calidad en la Industria del Software. La Norma ISO-9126*. Obtenido de nacionmulticultural.unam.mx: https://www.nacionmulticultural.unam.mx/empresasindigenas/docs/2094.pdf

Abril, A. (2019). *Modern Web Development with React.* Packt Publishing.

Berta, O. (211). *Incorporación de la integración Continua en el Desarrollo de Software.* Facultad de ingeniería; Universidad de Piura.

Black, R. (2014). *Advanced Software Testing.* Rocky Nook.

Booth, J. (2019). *Puppeteer Succinctly.* Syncfusion Inc.

Borzemski, L., Grzech, A., Świątek, J., & Wilimowska, Z. (2017). *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016 – Part I.* ISAT.

Brown, E. (2014). *Web Development with Node and Express: Leveraging the JavaScript Stack.* O'Reilly Media.

Casciaro, M. (2014). *Node.js Design Patterns.* Packt Publishing.

Chinnathambi, K. (2018). *Learning React: A Hands-On Guide to Building Web Applications Using React and Redux.* Addison-Wesley Professional.

Davis, J., & Daniels, R. (2016). *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale.* O'Reilly Media.

Devi Nagarajan, A. (2018). *DevOps implementation framework for Agile-based large financial organizations.* Utecht Techt University.

Duvall, P., Matyas, S., & Glover, A. (2007). *Continuous Integration: Improving Software Quality and Reducing Risk.* Addison-Wesley Professional.

Emmit A. , S. J. (2015). *SPA Design and Architecture: Understanding single-page web applications.* North Carolina: Manning; 1st edición.

Feudjio, & Schieferdecker, I. (2009). *Test automation design patterns for reactive software systems.* Feudjio.

Fewster, M., & Graham, D. (2012). *Experience of Test Automation: Case Studies of Software Test Automation.* Addison-Wesley Professional.

Filipova, O. (2017). *Vue.js 2 and Bootstrap 4 Web Development.* Packt Publishing.

Fink , G., & Flatow, I. (2014). *Pro Single Page Application Development Using Backbone.js and ASP.NET.* Apress, Berkeley, CA.

Forsgren, N., Humble, J., & Kim , G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations.* Portland, Oregon: IT Revolution.

Fox, N. (Enero de 2020). *Secure Web Application Development*. Obtenido de edtechbooks.org: https://edtechbooks.org/studentguide/secure_web_development

Garg, M. (2019). *Test Automation using Selenium WebDriver with Java.* Packt Publishing.

Gomes Rodrigues, A. (2019). *Master Apache JMeter: From load testing to DevOps.* Packt Publishing.

Grupo de Investigación ALARCOS, U. d.-L. (01 de 01 de 2024). *¿Vale la pena adoptar prácticas de DevOps en la ingeniería de software global? Posibles desafíos y beneficio.* Obtenido de ScienceDirect Revistas y Libros: https://www.sciencedirect.com/science/article/pii/S092054892300048X

Hering, M. (2018). *DevOps for the Modern Enterprise: Winning Practices to Transform Legacy IT Organizations.* O'Reilly Media.

Herron, D. (2016). *Web Development with Node.js, 2nd Edition.* Packt Publishing.

Holdener III, A. (2008). *Ajax: The Definitive Guide: Interactive Applications for the Web.* O'Reilly Media.

Horton , A., & Vice, R. (2016). *Mastering React.* Packt Publishing.

Humble, J., & Farley, D. (2011). *CONTINUOUS DELIVERY Reliable Software Releases Through Build, Test, and Deployment Automation.* Boston: Pearson Education Inc.

Hutterman, M. (September 2012). *DevOps for Developers.* Apress.

IBM Redbooks;. (2015). *Using Liberty for DevOps, Continuous Delivery, and Deployment.* IBM Redbooks.

Japikse, P., Grossnicklaus, K., & Dewey, B. (2017). *Building Web Applications with Visual Studio 2017: Using .NET Core and Modern JavaScript Frameworks.* Apress.

John C. , S., & Tragura. (2015). *Gatling Cookbook: Over 60 recipes to get you up and running with Gatling as a stress-testing tool for web applications.* Packt Publishing.

Kaner, C., Falk, J., & Nguyen, H. Q. (1999). *Testing Computer Software. Wiley.* Wiley.

Kawalerowicz, M., & Berntson, C. (2011). *Continuous Integration in .NET.* Manning Publications.

Kim, G., Debois, P., Willis , J., & Humble, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations.* IT Revolution Press.

Leszko, R. (2018). *Continuous Delivery with Docker and Jenkins: Delivering software at scale.* Packt Publishing.

Ludin, S., & Garza, J. (2017). *Learning HTTP/2: A Practical Guide for Beginners.* O'Reilly Media.

Macrae, C. (2021). *Vue.js: Up and Running: Building Accessible and Performant Web Apps.* O'Reilly Media.

Maldonado Pinto, J. E. (2018). *METODOLOGÍA DE LA INVESTIGACIÓN SOCIAL.* Bogota: Ediciones de la U.

Menascé, D., Almeida, V., Dowdy, L., & Dowdy, L. (2004). *Performance by Design: Computer Capacity Planning By Example.* Prentice Hall.

Menon, V. (2013). *TestNG Beginner's Guide.* Packt Publishing.

Mikowski, M., & Powell, J. (2013). *Single Page Web Applications: JavaScript end-to-end First.* Manning.

Monteiro, F. (2014). *Learning Single-page Web Application Development.* Packt Publishing.

Pimienta Prieto, J. H., & De la Orden Hoz, A. (2017). *Metodología de la Investigación.* México: Pearson Educación de México S.A.

Poulton, N. (2020). *The Kubernetes Book: Updated December 2020.* Independently published.

Pressman, R., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach 8th Edición.* McGraw Hill.

Proskurin, O. (2018). *React Performance Optimization: Techniques and strategies to improve React app performance.* Packt Publishing.

Rippon, C. (2021). *Learning Cypress: A Beginner's Guide to End-to-End Testing with JavaScript.* Packt Publishing.

Rouse, M. (Enero de 2023). *Aplicación web (aplicación web)*. Obtenido de WhatIs.com, TechTarget: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

Salunke, S. (2021). *Continuous Integration, Delivery, and Deployment: Reliable and faster software releases with automated CI/CD using Travis CI.* Packt Publishing.

Smart, J. F. (2011). *Jenkins: The Definitive Guide.* O'Reilly Media.

Smith, A. k. (2017). *Implementing Continuous Integration and Continuous Delivery: Building Reliable and Secure Software from Source to Deployment.* Packt Publishing.

Solovei, V., Olshevska, O., & Bortsova, Y. (2017). The Difference Between Developing Single Page Application And Traditional Web Application Based On Mechatronics Robot Laboratory Onaft Application. *Odessa National Academy of Food Technologies, Odessa, Ukraine*, 1-5.

Soni, M. (2016). *DevOps for Web Development.* Packt Publishing.

Spillner, A., Linz, T., & Schaefer, H. (2014). *Software Testing Foundations (4th ed.).* Rocky Nook.

Srincharan , V. (218). *DevOps: Continuous Delivery, Integration, and Deployment with DevOps.* Packt Publishing.

Stefanov, S., & Freeman, A. (2016). *React: Up & Running: Building Web Applications.* O'Reilly Media.

Swartout, P. (2014). *Continuous Delivery and DevOps.* Packt Publishing.

Tankersley, C. (2019). *Docker for Developers: Develop and run your application with Docker containers using DevOps tools for continuous delivery.* Packt Publishing.

The Continuous Delivery Foundation. (2021). *Continuous Testing: The CD Foundation's Guide to Accelerating and Improving Software Quality.* O'Reilly Media.

Thomas, D., & Hunt , A. (2014). *The Pragmatic Programmer: Your Journey to Mastery (2nd ed.).* Addison-Wesley Professional.

Uluca, D. (2018). *Securing Angular Applications: Protect your web applications against threats by learning Angular concepts.* Packt Publishing.

Wieruch, R. (2017). *The Road to learn React: Your journey to master plain yet pragmatic React.js.* Independently Published.

Zap Chernyak, A. (s.f.). *que-es-la-prueba-de-software-de-interfaz-de-usuario-profundizacion-en-los-tipos-procesos-herramientas-y-aplicacion.* Obtenido de ZAPTEST: Https://www.zaptest.com/es/que-es-la-prueba-de-software-de-interfaz-de-usuario-profundizacion-en-los-tipos-procesos-herramientas-y-aplicacion?pdf=44013

Zhan, Z. (2015). *Selenium WebDriver Recipes in C#: Second Edition.* Apress.