UNIVERSIDAD AUTÓNOMA DE ICA

UNIVERSIDAD AUTÓNOMA DE ICA

FACULTAD DE INGENIERÍA, CIENCIAS Y ADMINISTRACIÓN

PROGRAMA ACADÉMICO DE INGENIERÍA DE SISTEMAS

TESIS

Automatización de pruebas de calidad de software en Empresa Seguros La Positiva, Perú, 2025

LÍNEA DE INVESTIGACIÓN:

Transformación digital, productividad y desarrollo urbano sostenible

PRESENTADO POR:

Napa Rojas, Hugo Eduardo

TESIS DESARROLLADA PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

ASESOR:

Dr. Angeles Morales, Julio César

https://orcid.org/0000-0002-7470-8154

Chincha, Perú, 2025

CONSTANCIA DE APROBACIÓN DE INVESTIGACIÓN

UNIVERSIDAD AUTÓNOMA DE ICA

CONSTANCIA DE APROBACIÓN DE INVESTIGACIÓN

Chincha, 09 de enero del 2026

Dra. Mariana Alejandra Campos Sobrino
Decana de la Facultad de Ingeniería, Ciencias y Administración Universidad Autónoma de Ica.

Presente. -

De mi especial consideración:

Sirva la presente para saludarla e informar que, el **Bach. HUGO EDUARDO NAPA ROJAS**, de la Facultad de Ingeniería, Ciencias y Administración, del programa Académico de INGENIERÍA DE SISTEMAS, ha cumplido con elaborar su:

PROYECTO DE TESIS		TESIS	X

TITULADO:

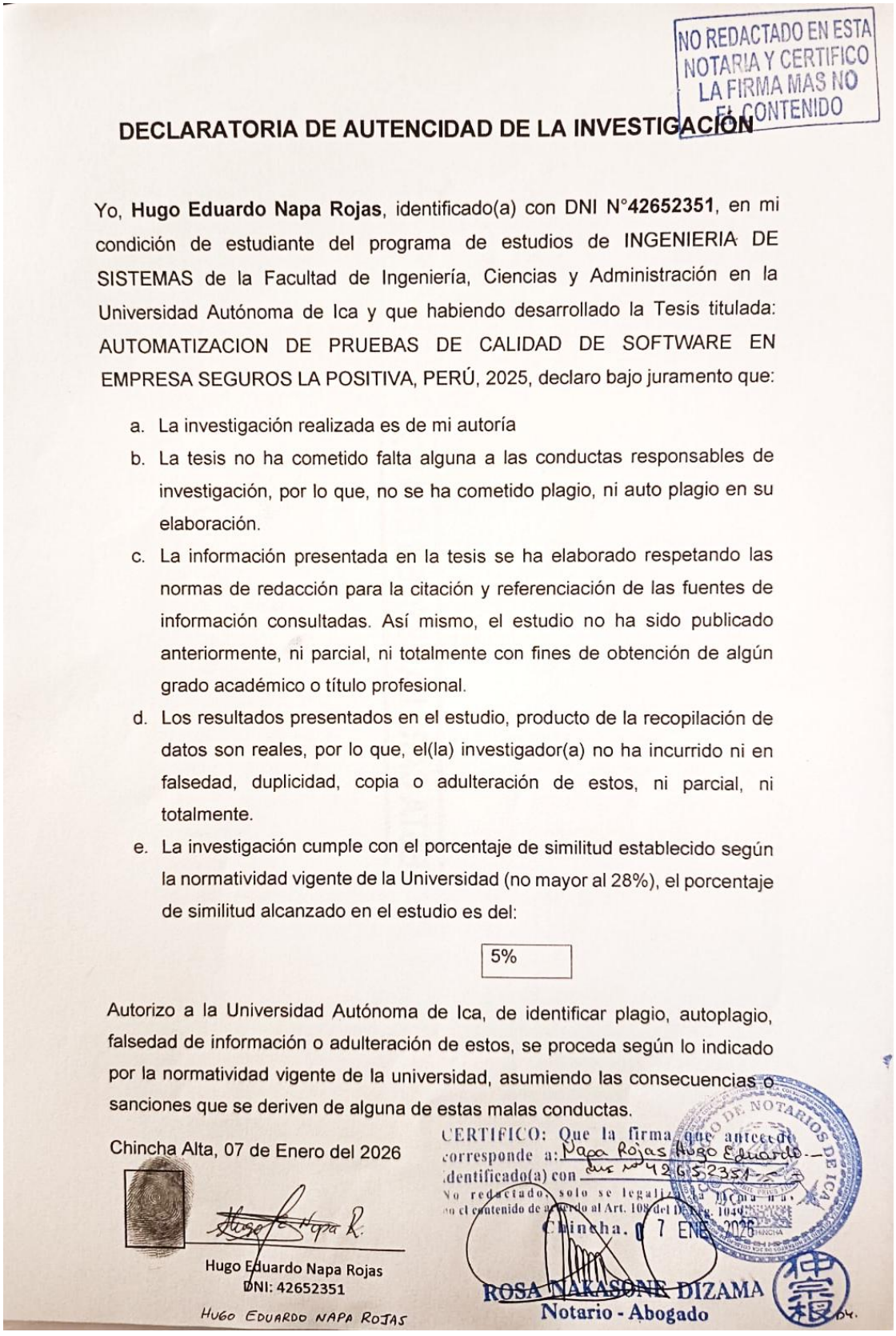
"AUTOMATIZACION DE PRUEBAS DE CALIDAD DE SOFTWARE EN EMPRESA SEGUROS LA POSITIVA, PERÚ, 2025"

Por lo tanto, queda expedito para continuar con el procedimiento administrativo correspondiente según la etapa del proceso académico.

Agradezco por anticipado la atención a la presente, aprovecho la ocasión para expresar los sentimientos de mi especial consideración y deferencia personal. Cordialmente,

JULIO CÉSAR ANGELES MORALES
CODIGO ORCID: 0000-0002-7470-8154
DNI: 32796107

DECLARATORIA DE AUTENTICIDAD DE LA INVESTIGACIÓN



NO REDACTADO EN ESTA NOTARIA Y CERTIFICO LA FIRMA MAS NO EL CONTENIDO

DECLARATORIA DE AUTENCIDAD DE LA INVESTIGACIÓN

Yo, **Hugo Eduardo Napa Rojas**, identificado(a) con DNI N°**42652351**, en mi condición de estudiante del programa de estudios de INGENIERIA DE SISTEMAS de la Facultad de Ingeniería, Ciencias y Administración en la Universidad Autónoma de Ica y que habiendo desarrollado la Tesis titulada: AUTOMATIZACION DE PRUEBAS DE CALIDAD DE SOFTWARE EN EMPRESA SEGUROS LA POSITIVA, PERÚ, 2025, declaro bajo juramento que:

a. La investigación realizada es de mi autoría
b. La tesis no ha cometido falta alguna a las conductas responsables de investigación, por lo que, no se ha cometido plagio, ni auto plagio en su elaboración.
c. La información presentada en la tesis se ha elaborado respetando las normas de redacción para la citación y referenciación de las fuentes de información consultadas. Así mismo, el estudio no ha sido publicado anteriormente, ni parcial, ni totalmente con fines de obtención de algún grado académico o título profesional.
d. Los resultados presentados en el estudio, producto de la recopilación de datos son reales, por lo que, el(la) investigador(a) no ha incurrido ni en falsedad, duplicidad, copia o adulteración de estos, ni parcial, ni totalmente.
e. La investigación cumple con el porcentaje de similitud establecido según la normatividad vigente de la Universidad (no mayor al 28%), el porcentaje de similitud alcanzado en el estudio es del:

5%

Autorizo a la Universidad Autónoma de Ica, de identificar plagio, autoplagio, falsedad de información o adulteración de estos, se proceda según lo indicado por la normatividad vigente de la universidad, asumiendo las consecuencias o sanciones que se deriven de alguna de estas malas conductas.

Chincha Alta, 07 de Enero del 2026

Hugo Eduardo Napa Rojas
DNI: 42652351

HUGO EDUARDO NAPA ROJAS

CERTIFICO: Que la firma que antecede corresponde a: Napa Rojas Hugo Eduardo
identificado(a) con DNI N° 42652351
No redactado, solo se legaliza la firma
de conformidad con el contenido de acuerdo al Art. 108 del D. Leg. 1049
Chincha, 07 ENE 2026

ROSA NAKASONE DIZAMA
Notario - Abogado

0122192190

NOTARIA
NAKASONE DIZAMA ROSA ANGELICA
SERVICIO DE AUTENTICACIÓN E IDENTIFICACIÓN BIOMÉTRICA

INFORMACIÓN PERSONAL

DNI	42652351
Primer Apellido	NAPA
Segundo Apellido	ROJAS
Nombres	HUGO EDUARDO

CORRESPONDE

La primera impresión dactilar capturada corresponde al DNI consultado. La segunda impresión dactilar capturada corresponde al DNI consultado.

NAPA ROJAS, HUGO EDUARDO
DNI 42652351

INFORMACIÓN DE CONSULTA DACTILAR

Operador: 21873268 - Maria Delia Manrique Saravia

Fecha de Transacción: 07-01-2026 13:10:19

Entidad: 10086837825 - NAKASONE DIZAMA ROSA ANGELICA

VERIFICACIÓN DE CONSULTA

Puede verificar la información en línea en:
https://serviciosbiometricos.reniec.gob.pe/identifica3/verification.do

Número de Consulta: 0122192190

Registro Nacional de Identificación y Estado Civil ® RENIEC 2026 - 07/01/2026 13:10:23

DEDICATORIA

A mi querida esposa, por su apoyo incondicional y su paciencia y comprensión a lo largo de cada fase de este trayecto académico. Te agradezco por ser mi impulso más grande para seguir adelante y mi fortaleza.

A mis hijos, que con su amor, apoyo moral y felicidad han llenado mis días de sentido y energía para seguir adelante. Este triunfo es también para ustedes, como modelo de tenacidad y trabajo duro.

A mis padres, por su apoyo, su amor y sus enseñanzas; por haberme tenido fe desde el comienzo. Sus valores y guía han sido esenciales para alcanzar este punto.

Esta Tesis va dedicada a todos ustedes, con un profundo agradecimiento.

AGRADECIMIENTO

Primeramente, dar las gracias a Dios por haberme proporcionado la fortaleza, la salud y la sabiduría que necesitaba para terminar este trabajo.

A mi esposa, mis hijos, y mis padres, por su apoyo constante, sus consejos y por motivarme a seguir adelante aun en los momentos más difíciles.

Asimismo, quiero manifestar mi agradecimiento sincero a los profesores y asesores de la carrera de Ingeniería de Sistemas, por su asesoramiento académico y técnico, por compartir sus conocimientos y por orientarme en el transcurso de esta investigación. Para la finalización de este proyecto, su experiencia ha sido esencial.

A la Universidad Autónoma de Ica, que me proporcionó las herramientas, los recursos y un ambiente apropiado para el desarrollo de mis competencias profesionales.

Mi más sincero agradecimiento a todos los que participaron directa o indirectamente en la elaboración de esta tesis.

RESUMEN

El objetivo de este estudio fue automatizar pruebas de calidad de software usando un framework ágil en los sistemas Affinity y Oficina Virtual de La Positiva Seguros. La investigación fue de enfoque cuantitativo, de tipo aplicado y diseño no experimental transeccional-descriptiva-propositiva. En la etapa de diagnóstico se identificaron los flujos más comunes que realizan los usuarios, mostrando que existen procesos repetitivos, mucha carga operativa y riesgos por el uso de pruebas manuales. A partir de ello, se desarrolló un framework de automatización con Selenium WebDriver, Cucumber, Page Object Model y reportería automatizada, compatible con entornos de CI. Luego, se implementaron los flujos críticos definidos, como afiliaciones, carga de tramas y registro de trámites, los cuales fueron automatizados con scripts parametrizados y verificados con evidencias capturadas automáticamente. Los resultados demostraron una disminución de los tiempos de ejecución y una mejora en la precisión, trazabilidad y reproducibilidad de las pruebas. Se determinaron que la automatización mejoró el proceso de aseguramiento .de la calidad, redujo los errores humanos y aumentó la confiabilidad de los sistemas evaluados, siendo una palanca para la mejora continua y la transformación digital en la organización.

Palabras claves: automatización de pruebas, calidad de software, empresa seguros

ABSTRACT

The purpose of this study was to apply software quality test automation using an agile framework in the Affinity and Virtual Office systems of La Positiva Seguros. The research was conducted using a quantitative, applied approach with a non-experimental, cross-sectional, descriptive, and propositional design. In the diagnostic phase, the critical flows most used by users were identified, revealing repetitive processes, high operational load, and risks derived from the use of manual testing. Based on this, an automation framework was designed based on Selenium WebDriver, Cucumber, Page Object Model, and automated reporting, compatible with continuous integration environments. Subsequently, the defined critical flows were developed, including affiliations, form uploads, and procedure registration, which were executed using parameterized scripts and validated using automatically captured evidence. The results showed a significant reduction in execution times, as well as an improvement in the accuracy, traceability, and reproducibility of the tests. It was concluded that automation made it possible to optimize the quality assurance process, reduce human error, and strengthen the reliability of the evaluated systems, becoming a strategic tool for continuous improvement and the digital transformation of the organization.

Keywords: test automation, software quality, insurance company

ÍNDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

I. INTRODUCCIÓN

El presente informe de investigación tuvo como propósito aplicar la automatización de pruebas de calidad de software utilizando un marco de trabajo ágil, específicamente Scrum, en los proyectos desarrollados por la empresa La Positiva Seguros, ubicada en la Calle Francisco Masías 370, San Isidro – Lima 27 – Perú. Esta investigación surgió ante la necesidad institucional de optimizar la gestión de procesos y mejorar el rendimiento en la entrega de productos de software, integrando pruebas automatizadas desde las etapas iniciales del ciclo de desarrollo.

La automatización de pruebas fue una pieza clave para reducir los tiempos de entrega, validar funcionalidades de forma continua y poner a disposición de los usuarios versiones funcionales del sistema en el menor tiempo posible. Esta estrategia redujo el trabajo repetitivo de las pruebas manuales, mejoró la detección temprana de errores y ayudaron a aumentar la productividad del equipo de desarrollo, garantizando productos finales de mayor calidad.

La investigación fue descriptiva, aplicada, orientada a procesos, utilizando técnicas e instrumentos de metodologías ágiles, como el análisis documental y la observación directa. Como resultado de su implementación, se logró optimizar los procesos de desarrollo, realizando entregas de software más rápidas, eficientes y de mayor valor para el cliente.

La metodología Scrum, junto con la automatización de pruebas, permitió entregar funcionalidades de valor para la empresa en ciclos cortos. Otro objetivo logrado fue el fortalecimiento de la calidad del software y la mejora continua en el equipo de desarrollo. Además, la automatización se enfocó en verificar continuamente las funcionalidades más utilizadas por los usuarios. fortalecimiento de la calidad del software y la mejora continua en el equipo de desarrollo.

Para ello, el informe se estructura de la siguiente manera:

Capítulo I: Introducción, mostrando los inicios del proyecto de automatización de pruebas de software y el contexto que llevó a su inicio.
Capítulo II: Se plantea el problema, describiendo la realidad problemática, la pregunta general y las específicas, los objetivos, la justificación, los alcances y las limitaciones.
Capítulo III: Se describe el marco teórico: los antecedentes, las teorías y el marco conceptual que respaldan la investigación.
Capítulo IV: Se detalla la metodología, especificando el tipo y diseño de investigación, las técnicas e instrumentos para la recolección de datos y los procedimientos para el análisis.
Capítulo V: Se describe la solución tecnológica y los resultados alcanzados al automatizar las pruebas.
Capítulo VI: Se interpretan los resultados, comparándolos con los antecedentes revisados.
Para finalizar, se exponen las conclusiones y recomendaciones, las referencias bibliográficas y los anexos, donde se encuentra la matriz de consistencia, el instrumento de recolección de datos y el informe de similitud.

El autor.

II. PLANTEAMIENTO DEL PROBLEMA

2.1. Descripción del problema

En el entorno actual dentro del desarrollo de software, las empresas están enfrentando una creciente demanda por entregar productos de alta calidad en menos tiempo. Se ha evidenciado que en los procesos tradicionales de aseguramiento de calidad hay limitaciones, especialmente aquellos basados en pruebas manuales. En seguros La Positiva, se manejan diversos proyectos tecnológicos simultáneamente, los ciclos extensos de validación y la alta carga operativa de las pruebas manuales afectan directamente la eficiencia, el tiempo para la entrega de productos y la capacidad en detectar errores oportunamente es ineficiente.

Usando el proceso manual de pruebas se ha validado que hay una cantidad significativa de tiempo y recursos, lo cual ocasiona retrasos en la entrega de funcionalidades, se duplican los esfuerzos, además se dificulta para ejecutar pruebas regresivas frecuentes y ocasiona dependencia del recurso humano. Todos estos factores no solo impactan negativamente en la calidad del producto final, también incrementan los riesgos de errores en producción, lo que puede afectar la satisfacción del cliente y el prestigio de la organización.

La falta de un proyecto de automatización en las pruebas impide una validación ágil y continua de los sistemas, lo cual es principal en entornos que trabajan con metodologías ágiles como Scrum. Al no contar con una estrategia de automatización, hace que se pierda la oportunidad para integrar las pruebas dentro del ciclo de desarrollo, y esta a su vez limita la posibilidad de implementar enfoques como la integración y entrega continua (CI/CD).

Por lo tanto, se ha evidenciado la necesidad de tener un proceso de automatización dentro de las pruebas de calidad de software, para que permita acelerar la validación de funcionalidades, garantizar la repetibilidad y consistencia de las pruebas, también la de facilitar la entrega incremental de valor a los usuarios. Al implementar este proyecto se busca optimizar los tiempos, reducir los errores, mejorar la eficiencia operativa y aumentar el nivel de calidad de los productos desarrollados por la empresa.

2.2. Pregunta de investigación general

¿Cuál es el software que hará la automatización de pruebas de calidad de software en empresa seguros la positiva, Perú, 2025?

2.3. Preguntas de investigación específicas

P.E.1:

¿Qué aplicaciones se identifican en la etapa de análisis que son las más usadas para poder automatizar sus flujos de pruebas en el software o script de automatización?

P.E.2:

¿Cómo se estructura el diseño del software o script de automatización para que sea rápido en su ejecución de pruebas?

P.E.3:

¿Qué herramientas, patrones y reporteria se utilizarán para el desarrollo del sistema de automatización de pruebas que se ha propuesto?

P.E.4:

¿Cuáles son los resultados obtenidos en la ejecución de pruebas con el software o script de automatización en términos de funcionamiento y utilidad?

2.4. Objetivo General

Implementar un sistema(script) de automatización de pruebas de software que permita ejecutar de forma rápida y confiable las pruebas de regresión luego de cada despliegue.

2.5. Objetivos específicos.

O.E.1: Analizar las aplicaciones que son más usadas para poder automatizar sus flujos de pruebas

O.E.2:

Diseñar el software o script de automatización utilizando patrones de diseño con una estructura que sea compatible para una integración continua.

O.E.3:

Desarrollar los flujos críticos de las aplicaciones más usadas utilizando la herramienta, patrón y reporteria según a la necesidad de las pruebas de calidad.

O.E.4:

Evaluar los resultados obtenidos tanto tiempo de respuesta como reportes finales de las pruebas realizadas con el software o script de automatización.

2.6. Justificación e Importancia

Justificación

Justificación metodológica

La automatización hará que sea más simple la integración con marcos ágiles como Scrum, lo cual permitirá realizar pruebas continuas en cada sprints, así como también las pruebas de entrega continua (CI/CD). Con ello se alinea con los objetivos de eficiencia y agilidad de la empresa, ayudando a la

estandarización de procesos de aseguramiento de calidad y a la obtención de reportes objetivos y entendibles para la toma de decisiones técnicas y de negocio. Así mismo buscar la reducción del trabajo repetitivo para que el equipo de calidad se enfoque en otras actividades con mayor valor, como el diseño de escenarios críticos y flujos nuevos que puedan validar que generen valor para el cliente.

Justificación científica

La propuesta se sustenta con los principios de ingeniería de software y aseguramiento de la calidad, los cuales demuestran que el uso de herramientas automatizadas menora las pruebas manuales, la constancia de validación y la detección temprana de errores. Con esto reducir los riesgos asociados a fallos en ambientes productivos y así permitir estimar el impacto de cambios en sistemas complejos bajo condiciones controladas y medibles.

Justificación tecnológica

La realización de un proyecto de automatización de pruebas permitirá mejorar los tiempos de ejecución, potenciar la detección de defectos y aumentar la cobertura de validación de los casos de prueba gracias a herramientas que mezclan procesos con entornos ágiles y plataformas de para entrega continua. Este proyecto utilizará recursos tecnológicos existentes, disminuir la dependencia del esfuerzo manual e impulsará la productividad, buscando una transformación digital sólida y sostenible en seguros La Positiva.

Importancia

La realización de un proyecto de automatización de pruebas de calidad de software en La Positiva Seguros muestra un gran avance tanto a nivel técnico como organizacional, debido a la

forma de cómo realizan las pruebas de los sistemas informáticos en la operación diaria de la empresa y en la atención eficiente a sus clientes.

En el sector de seguros donde es un mercado competitivo, la credibilidad, seguridad y rapidez en la entrega de soluciones tecnológicas son factores importantes para mantener la confianza del cliente y cumplir con los objetivos de calidad exigidos. En tal sentido, este proyecto gana una mayor importancia porque permite mejorar los procesos de validación del software, asegurando que las funcionalidades desarrolladas cumplan con los criterios establecidos antes de su despliegue en producción.

Así mismo, la automatización de pruebas facilitará la ejecución constante de pruebas funcionales, de regresión e integradas, ayudando a detectar errores en etapas iniciales del ciclo de desarrollo, lo que reduce notablemente los costos para la corrección y evita problemas con las aplicaciones en ambientes productivos.

Desde una perspectiva operativa, el proyecto ayudará al equipo de QA de La Positiva a mejorar su productividad, centrarse en otras tareas como por ejemplo tareas analíticas y estratégicas, también disminuir la carga de trabajo manual que genera repetición. A nivel metodológico, la integración de pruebas automatizadas con el marco Scrum potenciará el enfoque ágil de los proyectos dentro de la empresa, permitiendo una entrega continua de valor y una respuesta más ágil a cambios en los requerimientos del usuario.

Por consiguiente, la implementación de este proyecto es importante para mejorar la calidad del software, y también para

propiciar la transformación digital en la empresa, difundir la innovación en los procesos de aseguramiento de calidad y reforzar la cultura organizacional orientada a la mejora continua, la eficacia y la excelencia en el servicio.

2.7. Alcances y limitaciones

Alcances

La presente investigación se centró en el diseño, desarrollo e implementación de un sistema (script) de automatización de pruebas de calidad de software en los proyectos tecnológicos de la empresa La Positiva Seguros. El estudio implicó identificar las aplicaciones más utilizadas, definir los flujos críticos a automatizar, crear los scripts y medir su rendimiento en tiempo de ejecución, funcionalidad y utilidad para el equipo de calidad.

El proyecto solo abarcó pruebas funcionales y de regresión sobre los módulos priorizados por el equipo de QA, dejando fuera pruebas no funcionales como rendimiento, carga, seguridad o estrés. Asimismo, el alcance se limitó a las herramientas, plataformas y entornos disponibles en la empresa, con el fin de asegurar la aplicabilidad directa de la propuesta dentro del marco ágil Scrum y de los procesos vigentes de integración continua.

Limitaciones

La investigación presentó limitaciones relacionadas principalmente con el tiempo, los recursos y las condiciones reales del entorno empresarial. Una de las restricciones fue la disponibilidad limitada de personal especializado, lo cual impidió ampliar la automatización a un mayor número de aplicaciones o escenarios. Asimismo, algunas herramientas de automatización más avanzadas no pudieron ser evaluadas debido a restricciones tecnológicas internas o licenciamientos no disponibles.

Otra limitación estuvo vinculada al acceso a entornos de prueba controlados. Algunos sistemas contenían información sensible o dependencias externas que dificultaron la ejecución completa de ciertos flujos automatizados. Finalmente, al tratarse de un estudio aplicado con diseño no experimental, no se realizó una comparación exhaustiva entre múltiples frameworks o metodologías, ni se evaluó el impacto de la automatización en periodos prolongados de mantenimiento de los scripts.

III. MARCO TEÓRICO

3.1. Antecedentes

En este punto se menciona a los antecedentes de proyectos con relación a Automatización de pruebas de calidad de software, tanto en antecedentes internacionales, nacionales y locales:

Internacionales

Capgemini (2024). El World Quality Report 2024-2025 revisa el estado de las pruebas y aseguramiento de la calidad a nivel global. Dentro de este estudio se hace referencia en encuestas a más de 1,750 líderes de TI y QA en 32 países, declara que el 72 % de las organizaciones ya están usando la automatización en sus procesos de prueba, principalmente para agilizar la entrega de software y garantizar la calidad en entornos Agil y DevOps. Además, este informe menciona que las empresas que implementan automatización logran reducciones de hasta un 45 % en el tiempo de lanzamiento y disminución del 30 % en defectos en producción. También, resalta la preferencia hacia el uso de Inteligencia Artificial y pruebas en la nube como elementos clave para el futuro del aseguramiento de la calidad.

Garousi (2020). Menciona que en este estudio internacional se realizaron encuestas a 151 analistas de pruebas de software en 25 países, con el objetivo de validar el nivel de adopción, beneficios y desafíos de la automatización de pruebas en la empresa. Dentro de los resultados evidenciaron que, aunque la automatización se ha consolidado como una práctica clave dentro de metodologías Ágiles y DevOps, muchas empresas todavía necesitan de estrategias definidas, métricas y lineamientos claros para su implementación y así poder optar por su uso. También, se identificó que el nivel de madurez de la

automatización está relacionado con la calidad del producto, la velocidad de entrega y la reducción de costos en correcciones.

Garousi, Joy y Keleş (2024), dentro de su estudio hace una revisión sistemática de 55 herramientas de automatización de pruebas con IA, clasificando sus capacidades. También, se escogen dos de las herramientas mencionada en su investigación para un estudio empírico aplicado en proyectos de software open source, confrontándolas con enfoques tradicionales, con hallazgos como mayor eficacia en ejecución y reducción del esfuerzo de mantenimiento, también distingue limitaciones: cambios complejos de UI, falta de contexto del dominio.

Ali, Hamza y Rashid (2024), dentro de este artículo se menciona de forma amplia cómo la automatización de pruebas se forma en distintas fases del ciclo de vida del software. Además, se hace una comparación entre pruebas manuales vs automatizadas para medir términos de tiempo, coste, cobertura de errores, y precisión. Señala beneficios claros de automatización: reducción de tiempos de prueba/release, mayor cobertura, menor error humano, pero también retos como esfuerzo inicial para construir la infraestructura de automatización, necesidad de herramientas adecuadas, mantenimiento de scripts, etc.

Srinivas y Goel (2025), en su estudio práctico describe el diseño e implementación de un framework de automatización robusto utilizando Cucumber (BDD – Behavior Driven Development) y Java. El estudio se enfoca en permitir que tanto personas técnicas como no técnicas entiendan los tests es por ello que se utiliza un lenguaje BDD para escribir escenarios entendibles, se hace buen uso de datos de prueba, entornos dinámicos, integración con CI/CD, agilizar los mantenimientos. Con los

resultados se hacen referencia a reducciones importantes en esfuerzo manual de testeo, mejoras en detección de defectos y confiabilidad de los tests automatizados para su uso en las pruebas.

Nacionales

Gamarra (2022) en su tesis de licenciatura, Universidad Privada del Norte. La investigación aborda la problemática de que, en la empresa Orbis Ventures S.A.C., donde hace mención que las pruebas de regresión se ejecutaban manualmente en todos los proyectos, que los tiempos de prueba sen altos, generen costos adicionales y se detecten tarde los defectos que ponen en riesgo la estabilidad y la calidad de las aplicaciones entregadas. Es por ese motivo, que en su estudio propone la automatización de pruebas de regresión como la solución para mejorar el proceso de aseguramiento de calidad. Ante este motivo se describe la situación inicial: pruebas manuales repetitivos, poca cobertura de regresión en los periodos de entrega, alto consumo de recursos humanos y riesgos de tener observaciones de pruebas de regresión en producción. La propuesta de trabajo se focaliza en diseñar, implementar y evaluar un conjunto de scripts y un flujo de automatización que permita ejecutar las pruebas de regresión de forma automática en varios momentos sin intervención humana.

Silva y Villar Gavidia (2022), dentro de su tesis titulada “Implementación de automatización aplicada a pruebas de software del área de calidad de la empresa Entrega Creativa”, hace referencia sobre la automatización de pruebas, con una definición que las pruebas de regresión son verificaciones que se realizan al producto, las cuales pueden ser pruebas funcionales o no funcionales que permiten confirmar los resultados según las especificaciones definidas en los criterios de aceptación.

También hace mención que la característica principal de la automatización de pruebas son las pruebas repetitivas, que consumen mucho tiempo en realizarse o son complejas, para hacer una validación del producto y validar que se cumpla con los criterios para así considerarse conforme. El objetivo de la investigación es implementar automatización aplicada a pruebas de regresión a los softwares de la empresa apuntando a los casos de pruebas de la ruta crítica que generen valor, con esta implementación se busca mejorar la eficiencia en las pruebas, aumentar el nivel de calidad de software y reducir los fallos que se puedan presentar en las pruebas manuales.

Morales (2022), en su tesis titulado “Impacto de la automatización de pruebas en la eficiencia de pruebas de software en una consultora de TI, Lima (UCV)”, en su investigación aplicada, diseño pre-experimental. En su resultado de la implementación de la automatización de pruebas se evaluó que se obtuvo una disminución del 74.68 % en el tiempo medio por caso de prueba y un incremento del 220 % en el número de casos ejecutados con la automatización, y estas se comparó con el tiempo previo de pruebas manuales.

Cabrera y Pareja (2021) en su tesis titulada “Automatización de pruebas funcionales web para mejorar el área de calidad de software en una empresa del rubro de retails en el año 2021”, la tesis de investigación se propone a la necesidad de optimizar el tiempo de las pruebas manuales en el área de calidad de software de la empresa, donde las pruebas funcionales se realizan manualmente, ocasionando sobreesfuerzo en las pruebas y la detección de errores sea tarde por las pruebas manuales. También se menciona sobre el objetivo principal es automatizar las pruebas funcionales y de regresión de una aplicación web, utilizando herramientas de código libre con el

framework de Selenium paras el desarrollo de los scripts de automatización. La metodología que se usó en el proyecto fue ágil, con Scrum, donde se distribuyó el trabajo en cinco sprints, que incluyó automatización de pruebas funcionales y de regresión. Como resultado se evidencio que la automatización optimizo los tiempos de ejecución, se pudo cobertura más escenarios de prueba y, también mejorar la calidad del servicio ofrecido por la empresa.

Benites (2023), en su tesis titulado Implementación de un Framework de Automatización para Mejorar el Proceso de Pruebas Funcionales de una OSE publicado en la Universidad Católica Sedes Sapientiae (UCSS), en el proyecto dentro de los objetivos detalla la necesidad de agilizar y mejorar los tiempos de las pruebas funcionales durante el mantenimiento y actualización de módulos en el sistema de facturación electrónica de la empresa de servicios (OSE). Es por ese motivo que se propone la implementación de un framework de automatización donde usaron la Selenium como herramienta de construcción y lenguaje java para el desarrollo, estas a su vez son compatible con los navegadores web más usados. El trabajo incluye la creación de scripts automatizados con funcionalidades específicas según a la necesidad y uso de datos de prueba predeterminados. Se analizaron que los resultados fueron los esperados, se logró una mejora del 60 % en la efectividad de las pruebas funcionales, la detección temprana de errores mejoró en un 53,84 % y el tiempo de ejecución de pruebas se redujo en un 42,85 % lo cual cumplió con el objetivo de la implementación del software de automatización.

Locales o regionales

García (2022), en su tesis titulada “Automatización de prueba de software menciona el cómo mejorar el proceso de prueba del

módulo del sistema que desarrolla empresa Experts System S.A.C". En la tesis se detalla que el trabajo se realizó un proyecto para automatizar los casos de prueba del módulo del sistema de la empresa Experts Systems S.A.C., utilizando herramientas IDE para la automatización de pruebas como Selenium. También se menciona en el proyecto que se aborda la mejora del proceso manual, el proceso repetitivo y la reducción del tiempo de ejecución, buscando mejorar la eficiencia en pruebas funcionales y esta a su vez muestre una mejor calidad.

Arévalo, (2023), en su tesis titulada "Software para generar reportes de pruebas automatizadas usando RPA en aplicaciones web – Universidad Peruana de Ciencias Aplicadas (UPC)", menciona que este Proyecto se enfoca en la generación de reportes de pruebas automatizadas. Además, se utiliza RPA para automatizar pruebas funcionales para las aplicaciones web, donde se definen indicadores clave de rendimiento para medir eficiencia y eficacia del proceso de pruebas automatizadas. En los resultados obtenidos de la automatización se indican mejoras en métricas de calidad, rapidez en la ejecución, y la facilidad con la que se puede utilizar los script de automatizacion.

Baudazio (2024), en su tesis titulada "La influencia de la automatización de las pruebas funcionales para mejorar la calidad de aplicaciones móviles (2024)", fue presentada como un requisito para obtener el título profesional de Ingeniero de Sistemas en la Facultad de Ingeniería Industrial y de Sistemas de la Universidad Nacional Federico Villarreal, Lima, Perú, mediante esta investigación se busca determinar cómo la automatización de las pruebas funcionales mejorara la calidad de pruebas en aplicaciones móviles. También se menciona que para la investigación se aplica un enfoque cuantitativo de tipo preexperimental, donde se utilizó una encuesta de 14 preguntas

con escala Likert, la cual fue dirigida a 30 trabajadores de la empresa Uniflex (sector desarrollo de software en Perú). Los resultados mostraron una correlación positiva y moderada (Spearman Rho = 0.617, $p < 0.001$) entre la automatización de pruebas funcionales y la calidad del software. Además, se obtuvo una mejora significativa en aspectos como productividad y eficiencia. Dentro del estudio destaca, desde una perspectiva aplicada, también se menciona los beneficios que se conseguirán al implantar automatización en pruebas funcionales móviles, donde se busca mejorar la calidad de software.

Martínez & Támara (2024). En su tesis “Automatización de pruebas de software bajo la metodología Scrum para mejorar las pruebas de regresión del Seace v3.0”, dentro de la tesis se tiene como objetivo analizar el impacto de la automatización de pruebas de regresión, dentro del sistema SEACE 3.0 (Sistema Electrónico de Adquisiciones y Contrataciones del Estado), donde se busca mejorar su proceso de control de calidad. Para esta investigación se utilizó una investigación aplicada de enfoque cuantitativo, con diseño pre-experimental y enfoque hipotético-deductivo. Para medir indicadores de duración y cantidad de pruebas ejecutadas se utilizó una ficha de observación. Además, se aplicó la prueba estadística de Wilcoxon para evaluar cambios significativos en las pruebas.

Pesantes (2023), en su tesis titulada “Automatización de pruebas de aceptación en el proceso de desarrollo de software”, menciona que se diseñó e implementó un sistema web para automatizar pruebas de aceptación utilizando la metodología ágil eXtreme Programming y el desarrollo guiado fue utilizado un lenguaje (BDD). El desarrollo y ejecución del script de automatización mejoró los tiempos de ejecución de las pruebas y la calidad del proceso, disminuyendo defectos durante la fase

de pruebas UAT, donde se evaluó proyectos de distinto tamaño, la mejora del proyecto aporto evidencia sobre cómo la automatización de pruebas de aceptación utilizando metodologías ágiles se optimiza tiempos y estas a su vez busca reducir defectos en diferentes aplicaciones.

3.2. Bases Teóricas

Automatización de pruebas

La automatización de pruebas de calidad de software es una disciplina dentro del aseguramiento de calidad donde su objetivo es buscar optimizar, agilizar y perfeccionar la precisión del proceso de verificación y validación de aplicaciones mediante el uso de herramientas y scripts que ejecutan casos de prueba automáticos donde no hay intervención humana.

Aseguramiento de calidad de software es un grupo de actividades sistemáticas y planificadas donde garantizan que los procesos y productos cumplan con criterios previamente establecidos. La automatización en esta visión se introduce como una forma de mejorar la repetibilidad, reducir los errores humanos y ampliar la cobertura de validar. (Pressman y Maxim, 2020).

Pruebas manuales

Una definición clásica nos dice que las pruebas manuales son aquellas en las que un tester sigue los pasos de un caso de prueba, verifica los resultados esperados y registra los defectos encontrados. Pero este proceso se vuelve ineficiente cuando los sistemas se hacen más complejos y se liberan más versiones, porque repetir las pruebas regresivas de forma manual consume mucho tiempo y recursos. (Neelapu, 2024).

Beneficios de automatización

La automatización de pruebas hace uso de scripts y herramientas para ejecutar de forma automatizada las acciones que haría un tester QA de forma manual. Según el equipo de BrowserStack (2025), los principales beneficios de la automatización son:

1. Más rápido en la ejecución: las pruebas se pueden ejecutar más rápido y con mayor frecuencia.
2. Más exhaustivo: se pueden probar más casos y combinaciones de datos.
3. Consistencia: los resultados no están sujetos a la fatiga o errores humanos.
4. Integración continua: Las pruebas pueden integrarse en pipelines de CI/CD para verificaciones continuas.

Metodologías agiles

En el mundo metodológico, la automatización se alinea con metodologías ágiles y marcos como Scrum, Kanban o DevOps, en los que la validación temprana y continua es clave para entregar software de calidad en ciclos iterativos. También se destaca que la integración de pruebas automatizadas en cada ciclo de desarrollo incrementa la confiabilidad de las entregas y reduce la acumulación de defectos (Shahin, 2023).

Herramientas de automatización

Dentro de las herramientas de automatización, existen diversas opciones como Selenium, Cypress, JUnit, TestNG, UFT One, entre otras, que permiten ejecutar pruebas de interfaz gráfica, API y backend. Cada herramienta presenta ventajas y limitaciones en función del tipo de sistema, lenguaje de programación y nivel de integración requerido (Crudu, 2024).

3.3. Marco conceptual

Automatización de pruebas

Es un proceso mediante el cual se utilizan herramientas, scripts y marcos de trabajo para ejecutar de manera automática casos de prueba previamente diseñados, con el objetivo de verificar el correcto funcionamiento de un sistema sin necesidad de intervención de pruebas manuales. La automatización busca optimizar tiempo, disminuir errores humanos y permitir una validación continua durante el desarrollo del software (Katalon, 2024).

Pruebas de software

Es un conjunto de actividades que tienen como finalidad identificar defectos en un sistema y garantizar que cumpla con los requisitos funcionales y no funcionales establecidos. Estas a su vez se pueden clasificar en pruebas manuales y pruebas automatizadas (Sommerville, 2011).

Aseguramiento de calidad de software (qa)

Disciplina que engloba los procesos, metodologías y estándares aplicados para garantizar que el software desarrollado cumpla con criterios de calidad previamente definidos. El equipo QA no se acorta a la fase de pruebas, sino que interviene a lo largo de todo el ciclo de vida del software (Sommerville, 2011).

Pruebas funcionales

Son aquellas que validan que el software cumpla con las funciones descritas en los requerimientos, validando que la lógica de negocio se ejecute de manera correcta. Ejemplos: pruebas de regresión, pruebas de humo, pruebas de integración (Sommerville, 2011).

Pruebas no funcionales

Estas evalúan aspectos relacionados con el rendimiento, la seguridad, la usabilidad y otros factores que afectan la experiencia del usuario y la estabilidad del sistema (Sommerville, 2011).

Pruebas de regresión

Son pruebas donde la ejecución repetitiva de casos de prueba después de realizar cambios o actualizaciones al software, con el fin de asegurar que las funcionalidades existentes no se vean afectadas por los cambios realizados (Katalon, 2025).

Framework de pruebas

Es una estructura predefinida que permite organizar, ejecutar y reportar casos de prueba de forma eficiente. Ejemplos: TestNG, JUnit, PyTest, Robot Framework (Ranorex, 2024).

Integración continua (ci)

Es una práctica de desarrollo de software que consiste en integrar y validar el código de forma usual en un repositorio común, ejecutando automáticamente pruebas para detectar errores en etapas tempranas (Atlassian, 2025).

Devops

Es una metodología que integra desarrollo (Dev) y operaciones (Ops) para acelerar la entrega de software, fomentando la colaboración y la automatización de procesos como compilación, pruebas y despliegue (Lindemulder & Kosinski, 2025).

Herramientas de automatización de pruebas

Son herramientas o librerías que automatizan la ejecución de pruebas, como Selenium (para pruebas de interfaces web), Cypress (para pruebas rápidas en JavaScript), UFT One

(herramienta comercial para pruebas automatizadas funcionales) o JMeter (para pruebas de rendimiento) (BugBug Blog, 2025).

IV. METODOLOGÍA

4.1. Tipo y nivel de la investigación.

Tipo.

El estudio actual se enmarcó en una investigación tecnológica aplicada. Este tipo de investigación buscó desarrollar, adaptar y aplicar tecnologías ya existentes para solucionar un problema específico en un contexto determinado. En este caso se aplican herramientas y metodologías de automatización de pruebas para mejorar los procesos de aseguramiento de calidad de software, disminuyendo tiempos de ejecución, costos y errores humanos en las pruebas manuales.

La investigación no se quedó en la teoría, sino que se llegó a desarrollar una solución real: un sistema automatizado de pruebas que se puede integrar en el proceso de desarrollo, utilizando estándares de la industria como DevOps e Integración Continua (CI). De este modo, el estudio pretendió proponer una mejora práctica y de aplicación inmediata a los procesos de validación de software en la empresa.

Nivel.

Este proyecto se situó en el nivel de la investigación tecnológica aplicada. Este nivel se caracterizó por aplicar el conocimiento, teorías, métodos y tecnologías existentes para solucionar problemas concretos en el mundo real, creando soluciones prácticas e inmediatas para mejorar procesos.

En este caso, el trabajo no se quedó en la descripción o análisis del proceso de aseguramiento de calidad, sino que aplicó directamente una solución tecnológica —automatización de pruebas— en un contexto empresarial para:

- Optimizar los tiempos y recursos en la validación de software.
- Reducir los errores derivados de la ejecución manual de pruebas.
- Integrar prácticas de calidad en metodologías ágiles y entornos de integración continua (CI/CD).

4.2. Diseño de Investigación

El presente proyecto acogió un diseño de investigación no experimental, de tipo transeccional–descriptivo–propositivo.

No experimental:
El estudio no manipuló deliberadamente variables independientes; por el contrario, observó la situación real de los procesos de pruebas de software y aplicó una solución dentro del entorno operativo de la empresa, sin alterar las condiciones existentes.

Transeccional (o transversal):
La recolección y el análisis de datos se realizaron en un único momento o periodo determinado del proyecto, antes y después de la implementación de la automatización, sin llevar a cabo un seguimiento longitudinal prolongado.

Descriptivo:
El diseño permitió describir las características, limitaciones y tiempos asociados a la ejecución de pruebas manuales, así como documentar las mejoras obtenidas tras la incorporación de los scripts de automatización.

Propositivo:
A partir del diagnóstico de la situación problemática, el estudio formuló y desarrolló una propuesta tecnológica de

automatización orientada a mejorar los procesos de aseguramiento de calidad de software en la empresa.

4.3. Descripción de la metodología.

Scrum es un marco ágil que estructura el desarrollo de software en ciclos cortos e iterativos llamados sprints. Su estructura está compuesta por fases o eventos definidos que permiten planificar, construir, inspeccionar y adaptar el producto de manera continua. Según Schwaber y Sutherland (2020), los inventores oficiales de Scrum, el marco se basa en la transparencia, inspección y adaptación que guían cada una de sus etapas.

A continuación, se explican sus principales fases, según la literatura y la Guía Scrum.

1. Planificación del Sprint (Sprint Planning)

Esta etapa inicia cada sprint y en ella se planifica qué trabajo se hará y cómo se hará. En esta reunión, el Product Owner muestra las funcionalidades priorizadas del Product Backlog y el equipo elige las que puede finalizar en el sprint.

Según Schwaber y Sutherland (2020), la reunión aclara tres preguntas:

¿Por qué es importante este sprint?

¿Qué hacer en el sprint?

¿Cómo se llevará a cabo el trabajo elegido?

La planificación garantiza la concentración y la alineación de las entregas del sprint con las necesidades del negocio.

2. Reuniones diarias (Daily Scrum)

El Daily Scrum es una reunión corta (15 minutos como máximo) en la que el equipo inspecciona el avance hacia la meta del sprint y adapta el plan si es necesario. Esta reunión fortalece la

comunicación, disminuye la incertidumbre y permite identificar obstáculos en una etapa temprana.

Según Rubin (2012), el Daily Scrum "mantendrá al equipo sincronizado y en flujo, eliminando cualquier impedimento que esté deteniendo el sprint".

3. Desarrollo del Incremento (Ejecución del Sprint)
Es la etapa en la que el equipo de desarrollo crea el incremento de producto, siguiendo unos criterios de calidad establecidos (Definition of Done). Aquí se realizan las tareas de diseño, programación, pruebas y documentación necesarias para completar los elementos comprometidos.

Rubin (2012) señala que el trabajo dentro del sprint debe orientarse a generar valor al final de cada iteración, garantizando que el incremento sea usable y potencialmente liberable.

En proyectos con automatización de pruebas, esta fase incluye:
Diseño del framework o scripts.
Implementación del código automatizado.
Ejecución continua de pruebas.
Integración con CI/CD.

4. Revisión del Sprint (Sprint Review)
Al finalizar cada sprint, se realiza una reunión para inspeccionar el incremento generado y obtener retroalimentación del Product Owner y de los interesados. El objetivo es validar si el incremento cumple con los requerimientos y si aporta valor al producto.

Schwaber y Sutherland (2020) indican que la revisión permite ajustar el Product Backlog según cambios en el entorno, requerimientos o prioridades.

5. Retrospectiva del Sprint (Sprint Retrospective)

Es la fase final del ciclo y se orienta a la mejora continua. El equipo reflexiona sobre lo que funcionó, lo que no funcionó y las acciones que deben implementarse para optimizar el siguiente sprint.

Según Cohn (2010), la retrospectiva es uno de los eventos más importantes porque “establece un mecanismo formal de reflexión que impulsa la madurez del equipo y mejora los procesos internos de manera iterativa”.

6. Gestión del Product Backlog (fase continua)

Aunque no es una “fase cerrada”, la gestión del Product Backlog (Backlog Refinement) se realiza continuamente. Consiste en revisar, aclarar y priorizar los requerimientos para que estén listos para futuros sprints.

Según Cohn (2010), un Product Backlog bien refinado permite que el equipo trabaje con mayor claridad, menor retrabajo y mayor predictibilidad.

4.4. Recolección de datos.

La recolección de datos fue el proceso por el cual se recopiló la información para planificar, diseñar y evaluar la automatización de pruebas de software. Este proceso hace posible reconocer los principales flujos, escenarios funcionales y aplicaciones que necesitaban mayor esfuerzo de prueba en la organización.

Para este tipo de proyecto, la calidad y exactitud de los datos recolectados impactaron en la eficiencia de los scripts automatizados y en la cobertura de las pruebas. Por lo cual, la información recolectada se planea y analiza para asegurar que

la automatización se ajuste a las necesidades reales del entorno de desarrollo y aseguramiento de calidad.

4.5. Técnicas de análisis de datos.

Entre las técnicas de análisis de datos se utilizaron:

Análisis descriptivo

Se aplicó el análisis descriptivo para resumir y sintetizar los resultados de las pruebas automatizadas. Esta técnica permitió conocer cómo se comporta el sistema en prueba y medir la efectividad de los scripts desarrollados.

El análisis descriptivo consideró los siguientes aspectos:

Resultados de las pruebas: se documentaron las pruebas realizadas, aprobadas, reprobadas y bloqueadas, en busca de patrones o problemas recurrentes.

Métricas de desempeño: Se midieron métricas como tasa de éxito, tasa de fallos, tiempo promedio de ejecución y cobertura de casos de prueba para determinar la eficiencia y estabilidad de la automatización.

Herramientas de análisis: Se utilizaron paneles de control generados por herramientas como Jenkins, QTest y JIRA para recopilar, visualizar y rastrear los resultados de las pruebas automatizadas.

Esta técnica dio la oportunidad de visualizar y cuantificar el beneficio de la automatización de pruebas en el proceso de aseguramiento de la calidad del software.

V. SOLUCIÓN TECNOLÓGICA

5.1. Presentación de Resultados

En este capítulo se presentan los resultados obtenidos tras el desarrollo del sistema de automatización de pruebas de software en la empresa La Positiva Seguros. Los resultados se organizan de acuerdo con los objetivos específicos establecidos en la investigación, permitiendo evidenciar el nivel de logro alcanzado por cada uno de ellos y el impacto generado en los procesos de aseguramiento de la calidad.

- Descripción del problema actual.

  En el entorno actual dentro del desarrollo de software, las empresas están enfrentando una creciente demanda por entregar productos de alta calidad en menos tiempo. Se ha evidenciado que en los procesos tradicionales de aseguramiento de calidad hay limitaciones, especialmente aquellos basados en pruebas manuales. En seguros La Positiva, se manejan diversos proyectos tecnológicos simultáneamente, los ciclos extensos de validación y la alta carga operativa de las pruebas manuales afectan directamente la eficiencia, el tiempo para la entrega de productos y la capacidad en detectar errores oportunamente es ineficiente.

  Usando el proceso manual de pruebas se ha validado que hay una cantidad significativa de tiempo y recursos, lo cual ocasiona retrasos en la entrega de funcionalidades, se duplican los esfuerzos, además se dificulta para ejecutar pruebas regresivas frecuentes y ocasiona dependencia del recurso humano. Todos estos factores no solo impactan negativamente en la calidad del producto final, también incrementan los riesgos de errores en producción, lo que puede afectar la satisfacción del cliente y el prestigio de la organización.

  La falta de un proyecto de automatización en las pruebas impide una validación ágil y continua de los sistemas, lo cual es principal en entornos que trabajan con metodologías ágiles como Scrum. Al no

contar con una estrategia de automatización, hace que se pierda la oportunidad para integrar las pruebas dentro del ciclo de desarrollo, y esta a su vez limita la posibilidad de implementar enfoques como la integración y entrega continua (CI/CD).

Por lo tanto, se ha evidenciado la necesidad de tener un proceso de automatización dentro de las pruebas de calidad de software, para que permita acelerar la validación de funcionalidades, garantizar la repetibilidad y consistencia de las pruebas, también la de facilitar la entrega incremental de valor a los usuarios. Al implementar este proyecto se busca optimizar los tiempos, reducir los errores, mejorar la eficiencia operativa y aumentar el nivel de calidad de los productos desarrollados por la empresa

Resultados del Objetivo específico 1:

Analizar las aplicaciones que son más usadas para poder automatizar sus flujos de pruebas.

- Requisitos del sistema (funcionales y no funcionales).

1. Requisitos Funcionales

Los requisitos funcionales describen **las funcionalidades que el sistema debe cumplir** para lograr su propósito: permitir la **validación ágil y continua** de los sistemas dentro del ciclo de desarrollo.

Tabla 1

Requisitos funcionales

Código	Requisito Funcional	Descripción
RF01	**Ejecución automatizada de pruebas**	El sistema debe permitir la ejecución automática de los scripts de prueba, ya sea manualmente o programada.
RF02	**Integración con repositorios de código**	El sistema se debe integrar con repositorios como Git, Bitbucket para extraer los scripts de prueba y ejecutarlos en el ciclo CI/CD.
RF03	**Gestión de casos de prueba**	El sistema debe poder registrar, organizar y editar los casos de prueba automatizados asociándolos a una historia de usuario o un requisito.
RF04	**Generación de reportes de resultados**	El sistema debe generar informes detallados con los resultados de las ejecuciones (éxito, error, tiempo, logs, evidencias).
RF05	**Integración con herramientas CI/CD**	El sistema se debe integrar con herramientas como Jenkins, Azure DevOps o GitLab CI para ejecutar pruebas en el pipeline de CI.

En la Tabla 1 se muestran los requisitos funcionales especificados para el marco de automatización de pruebas, los cuales definen las capacidades que deben soportar la solución tecnológica para garantizar su efectividad, compatibilidad y contribución al proceso de aseguramiento de calidad de software. Estos requisitos aseguran que el sistema automatizado no solo ejecuta pruebas, sino que también se integra con el ecosistema de desarrollo, control de versiones y entrega continua de la organización.

Primero, el RF01 dice que el sistema debe ser capaz de ejecutar scripts de prueba automatizados (manuales o programados). Y esto es importante porque permite la repetibilidad y eficiencia en la validación de los flujos críticos. El RF02 garantiza la integración con repositorios de código (Git, Bitbucket, etc.) para mantener actualizados los scripts y permitir la trazabilidad del ciclo de vida del software.

El criterio RF03 hace referencia a que los casos de prueba deben organizarse y enlazarse con historias de usuario o requerimientos

para asegurar la trazabilidad con las necesidades del negocio. Por otro lado, el requerimiento RF04 implica la generación de informes detallados con evidencias y métricas para la toma de decisiones y el seguimiento de incidencias. Finalmente, RF05 define la Integración con herramientas CI/CD para que las pruebas se ejecuten automáticamente en pipelines de entrega continua, mejorando la calidad, velocidad y confiabilidad.

2. Requisitos No Funcionales

Los requisitos no funcionales especifican las cualidades, el rendimiento y las limitaciones técnicas a las que debe ajustarse el sistema para ser confiable y utilizable.

Tabla 2

Requisitos no funcionales

Código	Requisito No Funcional	Descripción
RNF01	**Disponibilidad**	El sistema debe estar disponible al menos el 99% del tiempo para poder ejecutar pruebas en cualquier momento del ciclo de vida del desarrollo.
RNF02	**Rendimiento**	Las ejecuciones de prueba se deben iniciar en menos de 5 segundos y finalizar sin interrupciones en entornos concurrentes.
RNF03	**Escalabilidad**	El sistema debe poder ejecutar pruebas en paralelo en diferentes entornos/ dispositivos sin pérdida de rendimiento.
RNF04	**Seguridad**	Debe asegurar el acceso seguro con autenticación de usuarios y cifrado de contraseñas.
RNF05	**Mantenibilidad**	El código fuente y los scripts deben documentarse para permitir su actualización y mantenimiento.
RNF06	**Integridad de datos**	Los resultados y datos de prueba se deben archivar de manera segura y sin posibilidad de modificación.

En la Tabla 2 se especifican los requisitos no funcionales para la solución de automatización de pruebas, definiendo las características de calidad, rendimiento, seguridad y mantenibilidad

que deben satisfacer el sistema para operar eficientemente en un entorno empresarial. Estos requisitos acompañan a los funcionales y garantizan que la herramienta de automatización no solo haga, sino que haga bien, con los estándares de rendimiento y robustez adecuados.

El RNF01 (Disponibilidad) exige que el sistema esté disponible como mínimo el 99% del tiempo para poder ejecutar pruebas continuas sin interrupciones durante el ciclo de desarrollo. Este nivel de disponibilidad se alinea con las expectativas de equipos ágiles y entornos CI/CD en los que las pruebas pueden ejecutarse en cualquier momento.

El requerimiento RNF02 (Rendimiento) garantiza buenos tiempos de respuesta; es decir, que las ejecuciones se iniciarán en menos de 5 segundos aún en condiciones de concurrencia. El criterio RNF03 (Escalabilidad) refuerza lo anterior al exigir poder ejecutar pruebas en paralelo y en diferentes entornos, escalando sin pérdida de rendimiento.

En seguridad, RNF04 requiere autenticación y gestión cifrada de credenciales, esencial para proteger accesos y datos confidenciales en la automatización. Además, RNF05 (Mantenibilidad) hace énfasis en código documentado, modular y fácil de actualizar, asegurando la continuidad del framework en el tiempo. Finalmente, RNF06 (Integridad de datos): Los resultados, registros y evidencias deben almacenarse de manera segura e inmutable, garantizando trazabilidad y auditoría técnica.

- Casos de uso, actores y escenarios.

UC1 — Configurar proyecto de pruebas

- **Objetivo:** Crear y parametrizar un proyecto de automatización asociado a una aplicación/módulo.

- **Actores:** QA Engineer (principal), Administrador (colabora).
- **Precondiciones:** Usuario autenticado con rol autorizado.
- **Disparador:** QA necesita iniciar automatización para un nuevo producto o módulo.

UC2 — Ejecutar suite Selenium en local

- **Objetivo:** Ejecutar una suite Selenium (UI) en el equipo del QA con WebDriver local.
- **Actores:** QA Engineer.
- **Precondiciones:** Proyecto configurado; drivers instalados (ChromeDriver/GeckoDriver); dependencias resueltas.
- **Disparador:** QA inicia validación de una historia/bugfix.

Tabla 3

Escenarios de pruebas

ID	Escenario	Objetivo	Dado / Cuando / Entonces (resumen)	Resultado esperado
E01	**Carga válida (CSV) – Happy Path**	Validar proceso completo correcto	**Dado** usuario con rol Cargador y sesión activa; **y** archivo CSV con 100 filas válidas. **Cuando** carga el archivo y confirma. **Entonces** el sistema crea lote “Procesado”, 100 afiliados creados, reporte sin observaciones y notifica por email.	Lote OK; tiempos dentro de SLA; auditoría registrada.
E02	**Carga válida (XLSX)**	Aceptar XLSX	Igual a E01 pero con XLSX.	Mismo resultado que E01.
E03	**Validación de encabezados**	Detectar headers faltantes	**Dado** archivo con columnas sin *FechaInicioVigencia*. **Cuando** intenta cargar. **Entonces** el sistema rechaza el archivo antes de procesar y muestra columnas faltantes.	Lote **Fallido**; sin cambios en BD.
E04	**Tipos de dato**	Validar formatos	**Dado** CSV con *FechaNacimiento* en “31/13/2020”. **Cuando** procesa. **Entonces** marca la fila como **Error de formato** y no crea afiliado.	Lote **Procesado con observaciones;** reporte señala línea y motivo.

La Tabla 3 presenta los principales escenarios de prueba definidos para validar el funcionamiento del proceso de carga de afiliaciones dentro del sistema. Estos escenarios representan tanto los flujos correctos (happy path) como las condiciones de error más comunes, garantizando que el sistema responda adecuadamente ante entradas válidas, inconsistencias y errores de formato. La cobertura

propuesta garantiza la verificación de todo el proceso, desde la lectura del archivo hasta el procesamiento y registro de resultados.

Caso E01 (Carga CSV – Happy Path): El caso define el comportamiento esperado cuando el usuario sube un archivo CSV bien formado con 100 registros válidos. Esta prueba valida la principal funcionalidad del módulo: la creación del lote, las afiliaciones, el reporte sin observaciones y la notificación al usuario. Su certificación verifica tiempos de respuesta, SLA y auditoría.

Caso de uso E02 (Carga válida XLSX): Amplía el E01, asegurándose de que el sistema acepta archivos en formato XLSX sin cambiar el resultado esperado. Esto asegura compatibilidad y adaptabilidad en los formatos que los usuarios pueden cargar.

Por otro lado, el caso E03 (Validación de encabezados) verifica que el sistema sea capaz de reconocer formatos erróneos (por ejemplo, encabezados faltantes). El rechazo del archivo antes de procesar evita inconsistencias en la base de datos y garantiza la integridad de la información.

Finalmente, el caso E04 (Tipos de datos) prueba la identificación de datos con formatos incorrectos (fechas inexistentes, por ejemplo). En este caso, el sistema debe leer el archivo a los medios, encontrar la línea mala y mostrarla como una observación, sin generar registros erróneos.

- Modelo de procesos y flujo de datos.

  UC1 — Configurar proyecto de pruebas

  Flujo básico:

  1. El usuario crea un **Proyecto** ingresando nombre, descripción y aplicación objetivo.
  2. Define **stack** (framework de pruebas, lenguaje, runner) y **repositorio** a vincular (solo referencia inicial).
  3. Configura **variables**/secretos (tokens, credenciales en almacén seguro).
  4. Guarda y habilita el proyecto.

  Flujos alternos/excepciones:

  5. 1a. Nombre ya existe → el sistema sugiere variante y no permite duplicados.
  6. 3a. Validación de secreto falla → se notifica y no se guarda.

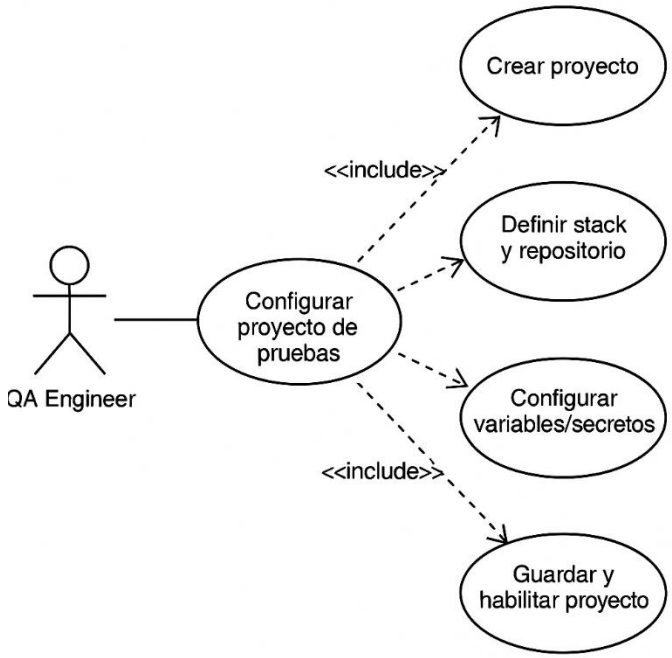
  Postcondiciones: Proyecto persistido y listo para vincular repositorio/entornos.

  Reglas de negocio: Nombre único por organización; secretos cifrados.

  Métricas: Tiempo de configuración, proyectos activos por equipo.

Figura 1.

Diagrama de casos de uso para la configuración del proyecto de pruebas



El diagrama representa las actividades que realiza un **QA Engineer** para configurar un proyecto de pruebas dentro del sistema de automatización. El caso de uso principal **"Configurar proyecto de pruebas"** agrupa un conjunto de acciones necesarias para dejar un proyecto totalmente operativo y listo para su integración con el flujo de automatización.

Este proceso forma parte de la fase inicial del framework, donde se prepara el entorno técnico que permitirá ejecutar pruebas automatizadas de manera ordenada, segura y reproducible.

UC2 — Ejecutar suite Selenium en local

Flujo principal:

1. QA selecciona **suite/casos** y **navegador** (p. ej., Chrome).
2. El sistema inicializa **WebDriver** con capacidades (modo headless, tamaño ventana, timeouts).

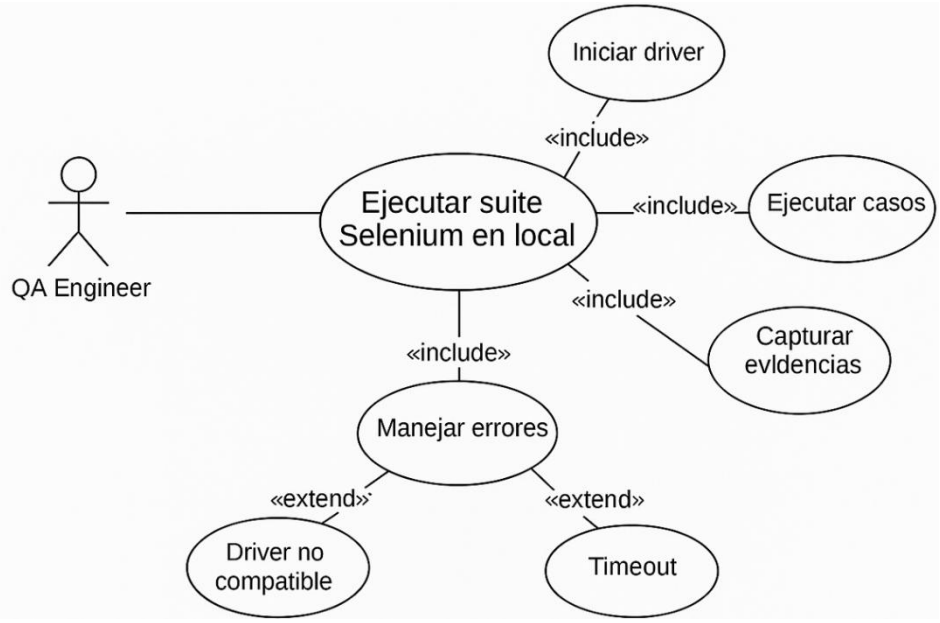
3. Se ejecutan los **test cases** aplicando **Page Objects** y **esperas explícitas**.
4. Se capturan **screenshots/logs** al finalizar cada caso o ante fallo.
5. Se genera **reporte local** (HTML/Allure) y se muestra el resumen.

Alternos/excepciones:

- 2a. Driver no encontrado o versión incompatible → descargar/actualizar driver y reintentar.
- 3a. ElementNotInteractable/Timeout → reintento controlado; marcar como **test-failure** (no infra).

Postcondiciones: Resultados disponibles localmente; artefactos (logs/evidencias) almacenados.
Reglas: Versionar drivers; alinear versión navegador-driver; timeouts estándar.
Métricas: Duración por caso, tasa de fallos, flakiness local.

Figura 2.
Diagrama de casos de uso para la ejecución de la suite Selenium en entorno local



El diagrama representa las actividades que realiza un **QA Engineer** al ejecutar una suite de pruebas automatizadas con Selenium en un entorno local. El caso de uso principal es **"Ejecutar suite Selenium en local"**, y se descompone en varios subprocesos obligatorios (*include*) y opcionales (*extend*) que permiten ejecutar la automatización de forma completa y controlada.

Resultados del Objetivo específico 2:

Diseñar el software o script de automatización utilizando patrones de diseño con una estructura que sea compatible para una integración continua.

Diseño del Sistema

- Arquitectura general del sistema.

Componentes principales

1. Cucumber + Gherkin (BDD)

Define los **escenarios de prueba** en lenguaje natural:

```
Feature: Carga de archivo de afiliación
  Scenario: Carga válida del archivo Excel
    Given que el usuario ingresa al portal de afiliación
    When carga un archivo Excel válido
    Then el sistema muestra mensaje de éxito
    And genera reporte en Word
```

2. Step Definitions

Implementan los pasos de Gherkin en código Java:

```
@When("carga un archivo Excel válido")
public void cargarArchivoExcelValido() {
    CargaArchivoPage carga = new CargaArchivoPage(driver);
    carga.subirArchivo("afiliaciones.xlsx");
    ScreenshotUtil.capturar("ArchivoCargado");
}
```

3. Page Object Model (POM)

Cada página web tiene su clase con elementos y acciones encapsuladas:

```
public class CargaArchivoPage {
   WebDriver driver;

   @FindBy(id = "btnSubir")
   WebElement btnSubir;

   @FindBy(id = "inputArchivo")
   WebElement inputArchivo;

   public void subirArchivo(String nombreArchivo) {
      inputArchivo.sendKeys(System.getProperty("user.dir") +
"/data/" + nombreArchivo);
      btnSubir.click();
   }
}
```

4. Hooks (Configuración y Limpieza)

Define acciones antes y después de cada escenario:

```
@Before
public void setUp() {
   DriverFactory.inicializarDriver("chrome");
}

@After
public void tearDown(Scenario scenario) {
   if (scenario.isFailed()) {
      ScreenshotUtil.capturar("Error_" + scenario.getName());
   }
   WordReportGenerator.agregarResultado(scenario);
   DriverFactory.cerrarDriver();
}
```

5. Generación de Reporte en Word

Usa **Apache POI** para crear y actualizar un documento .docx con resultados, pasos y capturas.

```
public class WordReportGenerator {

    private static XWPFDocument document = new
XWPFDocument();
    private static String path = "./reports/ReporteEjecucion.docx";

    public static void agregarResultado(Scenario scenario) {
        XWPFParagraph p = document.createParagraph();
        XWPFRun run = p.createRun();
        run.setText("Escenario: " + scenario.getName());
        run.addBreak();
        run.setText("Resultado: " + (scenario.isFailed() ? "FALLIDO
❌" : "EXITOSO ✅"));
        run.addBreak();

        try (FileOutputStream out = new FileOutputStream(path)) {
            document.write(out);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

📎 *Librería recomendada:*

org.apache.poi:poi-ooxml (para crear documentos Word).

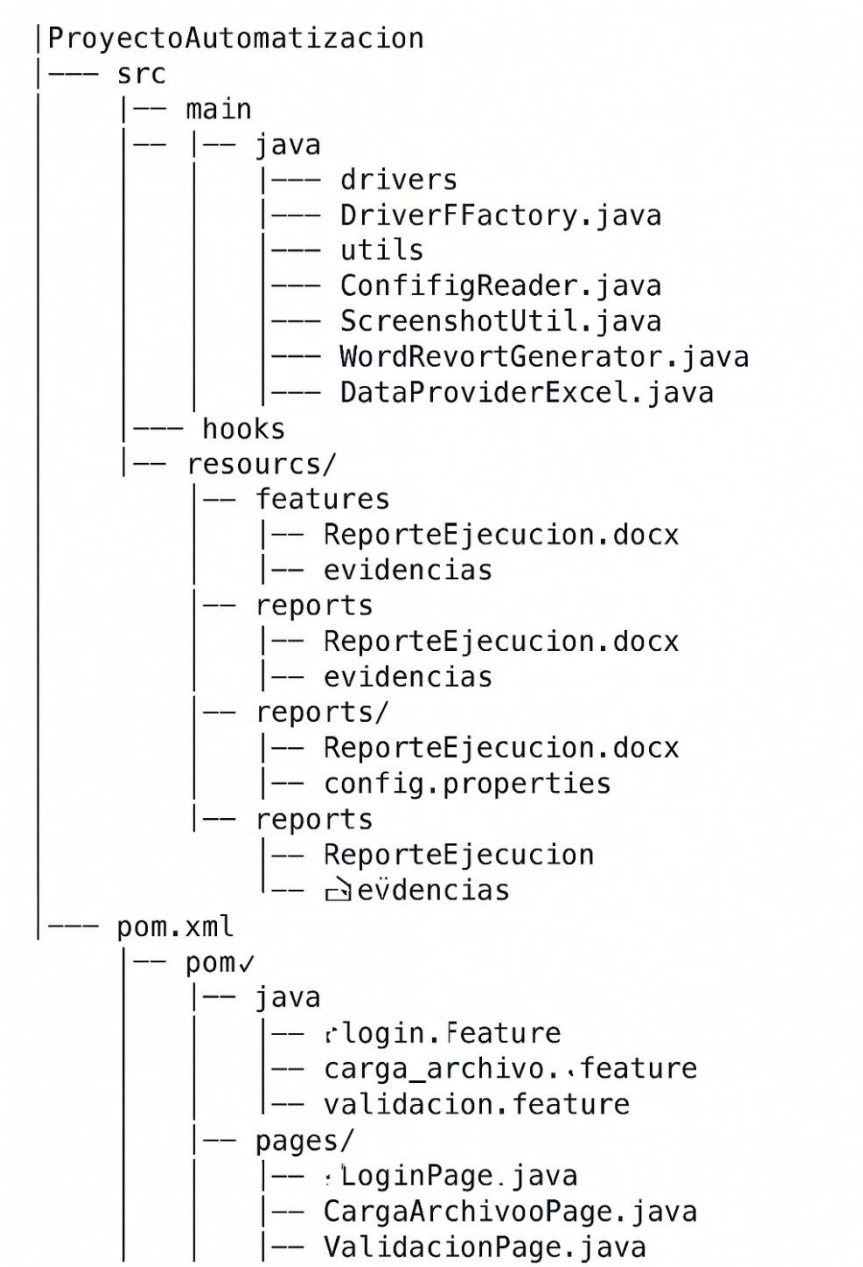
6. Reportes y Evidencias

- Cada escenario genera:
    - Capturas en /reports/evidencias/.
    - Reporte Word con resumen de ejecución.

- Opcionalmente se puede integrar **ExtentReports** o **Allure** para HTML, además del Word para documentación oficial.

Figura 3

Estructura del proyecto

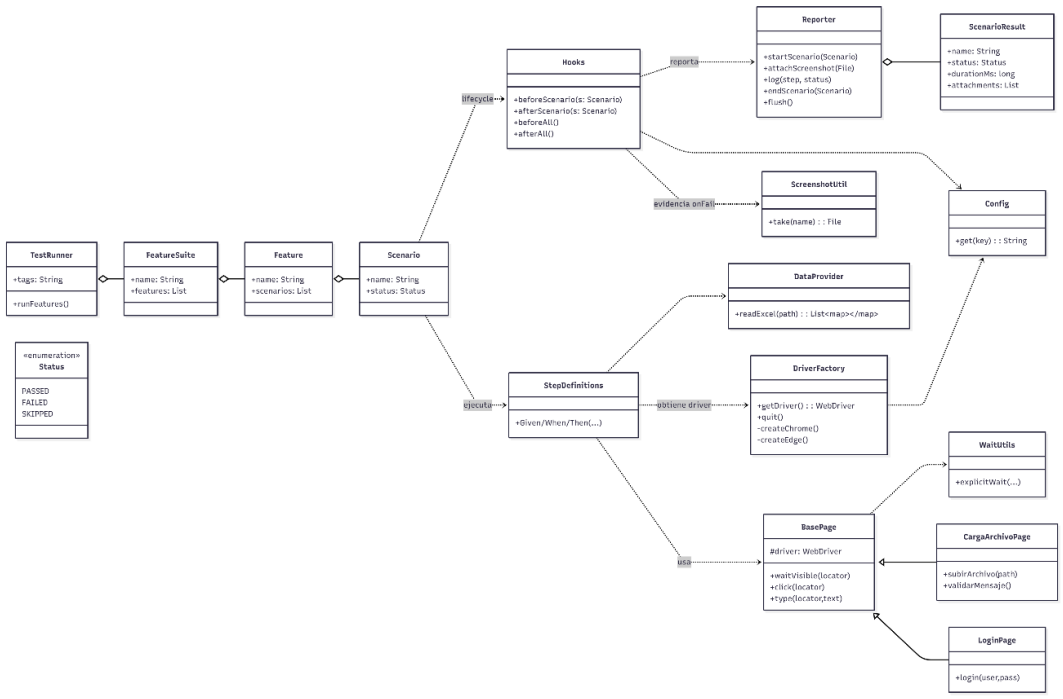


La imagen representa la arquitectura interna de un proyecto de automatización implementado con Java, Selenium, Cucumber y Maven. La estructura sigue las buenas prácticas de automatización, organizando el código en módulos coherentes que permiten escalabilidad, mantenibilidad y separación de responsabilidades.

Se construyó un diseño arquitectónico basado en el patrón Page Object Model (POM), combinado con BDD usando Cucumber y Gherkin. Esto permitió estructurar los casos de prueba de forma modular y escalable. La arquitectura también integra una capa de reportería automática y manejo de errores.

Figura 4

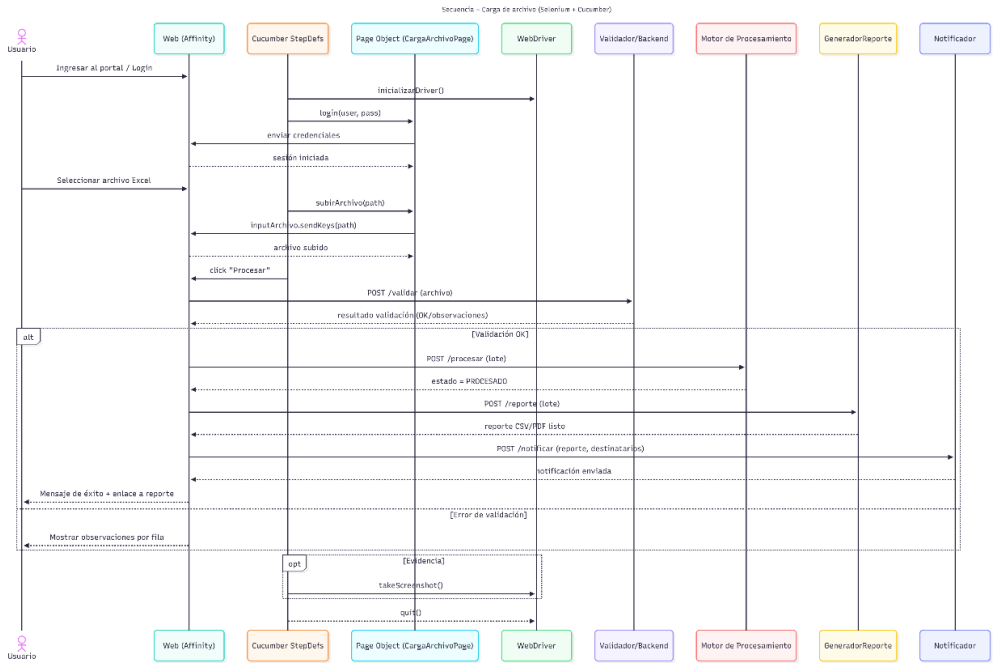
Diagrama de clases – flujo de proyecto de automatización



El diagrama presentado muestra la estructura interna del framework de automatización diseñado para ejecutar pruebas funcionales mediante Selenium, Cucumber y Java. Este modelo UML refleja la organización modular del sistema, así como las relaciones de dependencia, asociación y composición entre las clases que conforman el framework. La arquitectura se orienta a la reutilización de componentes, mantenibilidad y escalabilidad, principios fundamentales en proyectos de automatización profesional.

Figura 5

Diagrama de Secuencia del proceso de carga y validación de archivos mediante Selenium y Cucumber

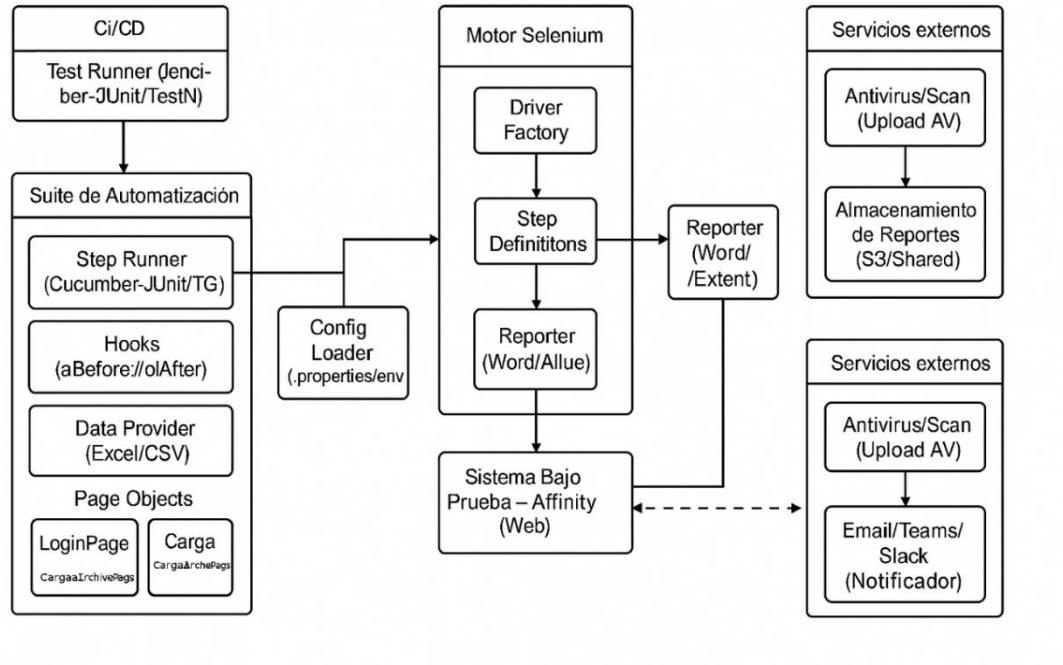


El diagrama ilustra el flujo completo de interacción entre el usuario, la aplicación web (Affinity), el framework de automatización (Cucumber + Selenium) y los sistemas internos encargados de la validación, procesamiento y notificación, durante el proceso de carga, validación y reporte de un archivo Excel.

Este diagrama es un reflejo directo del proceso automatizado ejecutado en la tesis, mostrando la integración entre el frontend, backend y las herramientas de automatización.

Figura 6

Arquitectura del Framework de Automatización con Selenium, Cucumber y Servicios Externos

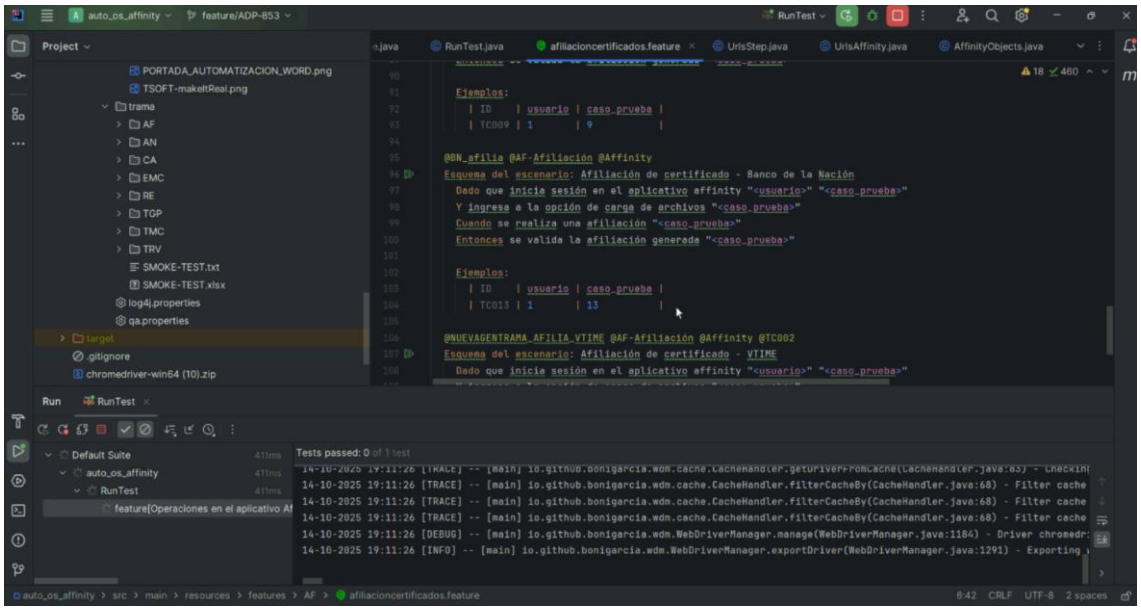


El diagrama muestra la arquitectura integral del framework de automatización, representando cómo los distintos módulos interactúan entre sí y cómo se integra el sistema con componentes externos, tanto dentro del pipeline CI/CD como en la infraestructura del proyecto.

El diseño refleja un flujo moderno y profesional de automatización, alineado con prácticas de ingeniería de calidad, DevOps y pruebas continuas.

Figura 7

Aplicación: AFFINITY. Pantalla: Inicio de Ejecución script



La figura muestra el entorno de desarrollo **IntelliJ IDEA** durante la ejecución automatizada del archivo Gherkin afiliacioncertificados.feature, el cual contiene los escenarios de prueba relacionados con el proceso de **Afiliación de Certificados** dentro del sistema Affinity de La Positiva Seguros.

El panel inferior confirma que la prueba se ejecutó satisfactoriamente bajo control del framework de automatización basado en **Selenium WebDriver + Cucumber + JUnit**.

Esta evidencia valida directamente:

El Objetivo Específico 1: Analizar aplicaciones y seleccionar los flujos críticos. Se observa cómo se definieron correctamente los escenarios de afiliación.

El Objetivo Específico 2: Diseñar el script con patrones de diseño. El archivo muestra la correcta integración entre Feature + Steps + Page Objects.

El Objetivo Específico 3: Desarrollar flujos automatizados críticos. La ejecución confirma la interacción completa con el módulo de afiliación.

El Objetivo Específico 4: Evaluar tiempos y resultados. El panel inferior proporciona el log y la confirmación del resultado exitoso.

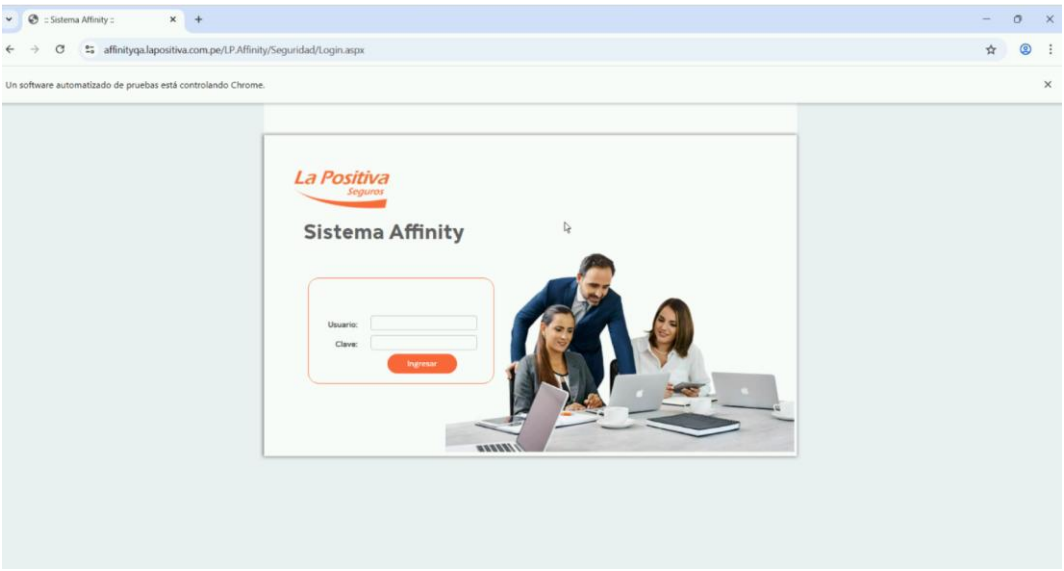
Resultados del Objetivo específico 3:

Desarrollar los flujos críticos de las aplicaciones más usadas utilizando la herramienta, patrón y reportería según la necesidad.

Se implementaron scripts automatizados que permiten ejecutar pruebas sobre los flujos funcionales priorizados. El desarrollo incluyó capturas automáticas, validación de datos, procesamiento de archivos y verificación de resultados.

Figura 8

Automatización controlando el navegador en el login de Affinity



La figura muestra una captura de pantalla del navegador Google Chrome durante la ejecución automatizada del caso de prueba correspondiente al login del Sistema Affinity de La Positiva Seguros. Esta evidencia forma parte del flujo inicial de la suite de automatización y demuestra que el framework está interactuando correctamente con la interfaz web.

En la barra de direcciones se observa:

https://affinityqa.lapositiva.com.pe/P.I/Affinity/Seguridad/Login.aspx

Esto demuestra que:

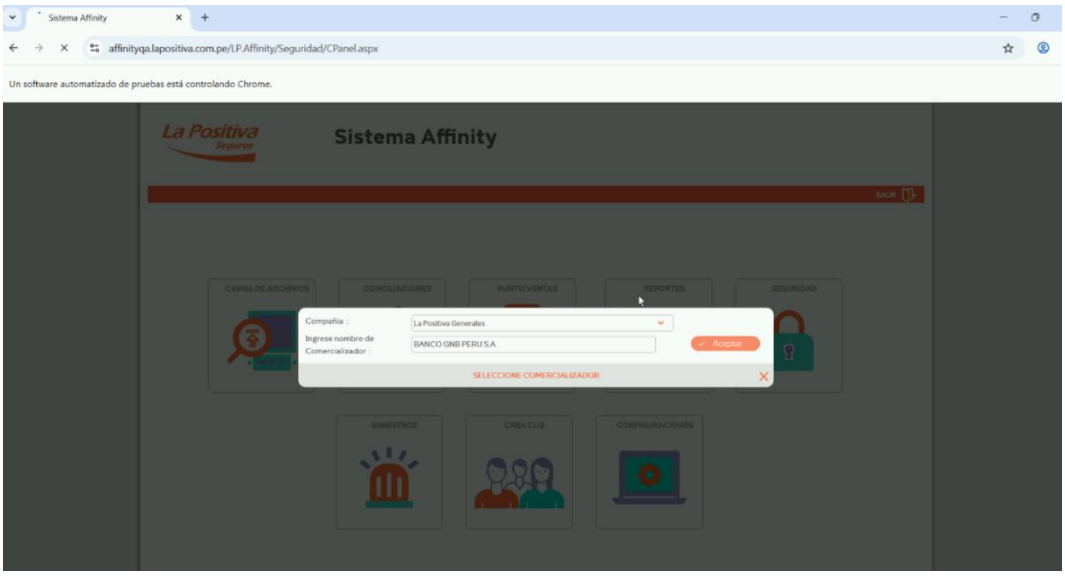
- La URL fue cargada desde el archivo de configuración (config.properties).

- El Page Object correspondiente al login fue invocado correctamente.
- La automatización está operando en el ambiente **QA**.

Este paso confirma que el flujo inicial (Given) se ejecutó sin errores.

Figura 9

Interfaz de selección de Compañía y Comercializador en el Sistema Affinity

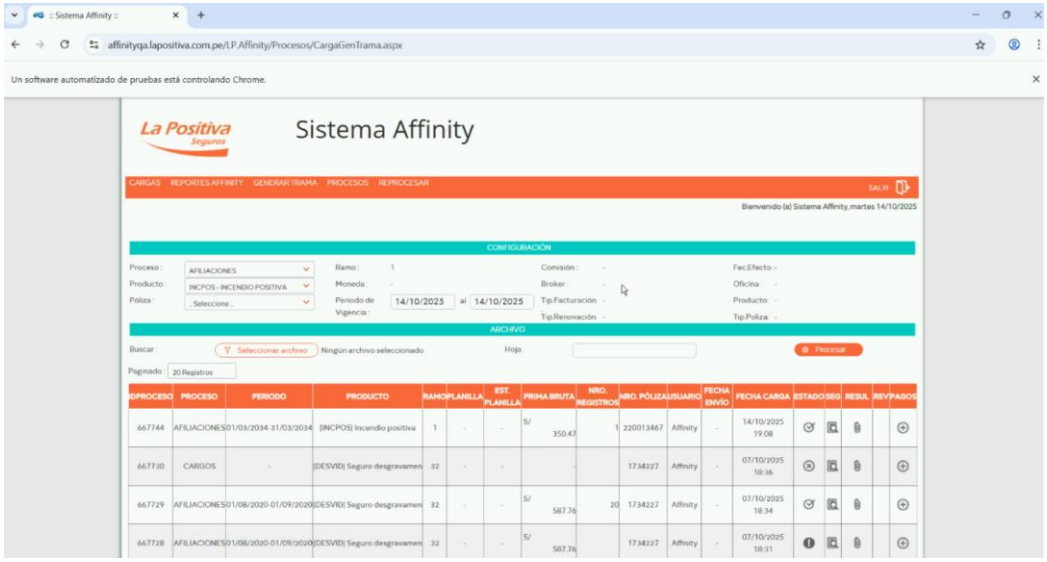


La imagen corresponde a la pantalla posterior al inicio de sesión dentro del Sistema Affinity de La Positiva Seguros. Una vez que el usuario (o la automatización) accede correctamente al sistema, se presenta una ventana modal de selección obligatoria que permite elegir la compañía y el comercializador con el cual se trabajará durante la sesión.

Esta captura evidencia que la automatización desarrollada con Selenium está interactuando correctamente con elementos dinámicos del sistema, incluyendo modales, listas desplegables y validaciones internas.

Figura 10

Pantalla de Carga y Procesamiento de Tramas en Affinity



La imagen muestra la pantalla del módulo “CARGAS → Cargar/Generar Tramas” del Sistema Affinity de La Positiva Seguros. Esta vista es parte esencial del flujo crítico automatizado, ya que aquí se realiza la carga de los archivos Excel, la validación de información y el procesamiento de las tramas enviadas por los comercializadores.

La captura permite observar claramente que el navegador Chrome se encuentra bajo control del software de automatización (Selenium WebDriver), validando que la suite está ejecutando correctamente el flujo funcional hasta llegar a esta interfaz clave.

El encabezado muestra claramente que el usuario (o la automatización) llegó al módulo:

✔ CARGAS → Carga/Generar Tramas

Desde el menú superior se distinguen secciones como:

Cargas

Reportes Affinity

Generar Trama

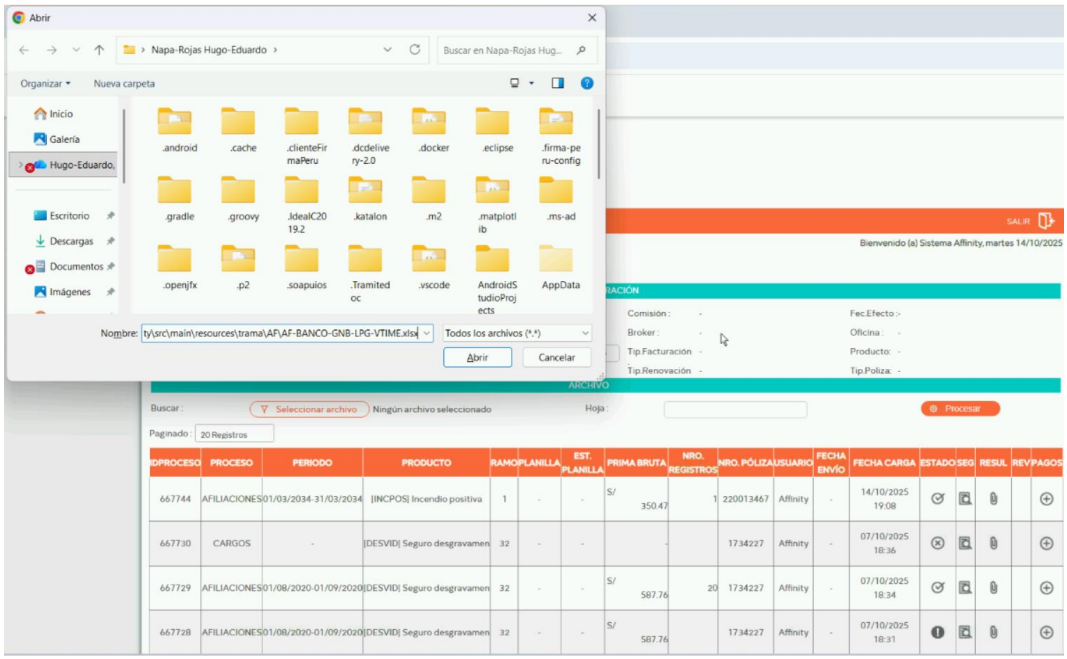
Procesos

Reprocesar

Este módulo es el corazón del flujo automatizado de la tesis, ya que es aquí donde se valida:
carga del archivo Excel,
proceso de validación,
procesamiento por lote,
registro de movimientos,
verificación de estados.

Figura 11

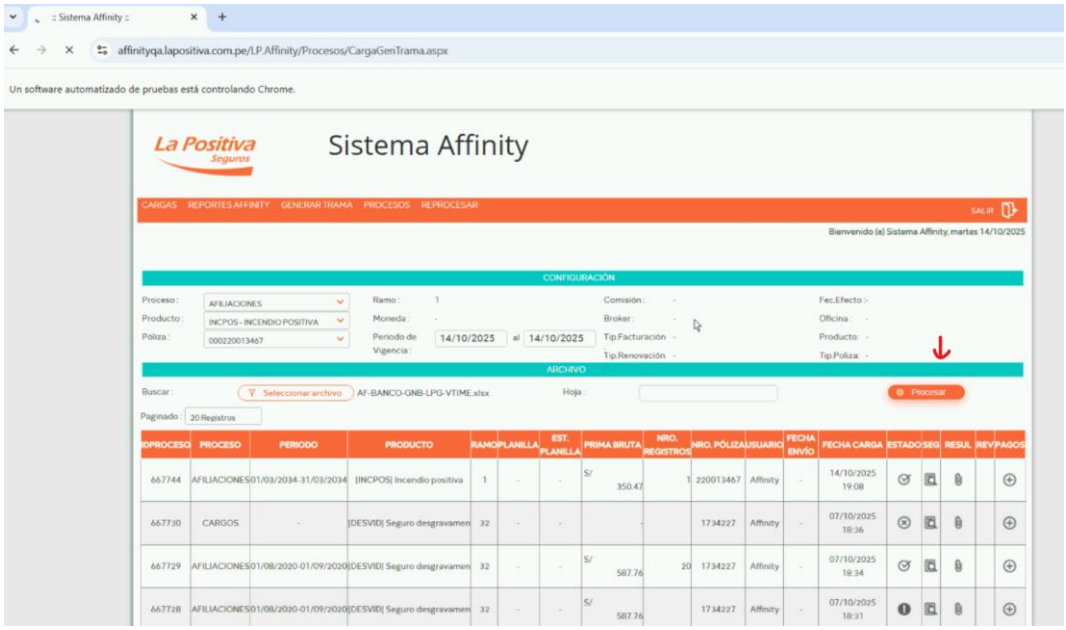
Selección automatizada del archivo Excel desde el diálogo del sistema operativo



La imagen muestra un momento clave del proceso automatizado: la apertura del cuadro de diálogo de selección de archivos de Windows para elegir el archivo Excel que será cargado y procesado dentro del Sistema Affinity. Esta acción forma parte del flujo crítico automatizado y constituye evidencia directa de que el framework está interactuando exitosamente no solo con elementos web, sino también con componentes externos del sistema operativo.

Figura 12

Archivo Excel seleccionado y listo para ser procesado en el módulo de Carga de Tramas



La imagen muestra la fase crítica del proceso automatizado en la cual el archivo Excel ya ha sido seleccionado correctamente en el módulo "CARGAS → Carga/Generar Trama" del Sistema Affinity. Esta evidencia corresponde al momento inmediatamente anterior a la validación y procesamiento del archivo, y confirma que la automatización ha gestionado con éxito la carga de datos provenientes del archivo Excel.

En la sección **ARCHIVO**, debajo del botón "Seleccionar archivo", se observa que el archivo:

AF-BANCO-GNB-LPG-VTIME.xlsx

ya se encuentra cargado y visible en el campo de selección.

Esto verifica que:

- La automatización utilizó el método sendKeys(filePath) para cargar el archivo correctamente.
- No hubo errores de ruta ni restricciones del navegador.

- El framework controló exitosamente la interacción con el elemento <input type="file">.

Este es un paso fundamental y suele ser uno de los más sensibles en automatización web, especialmente cuando se trabaja con archivos externos.

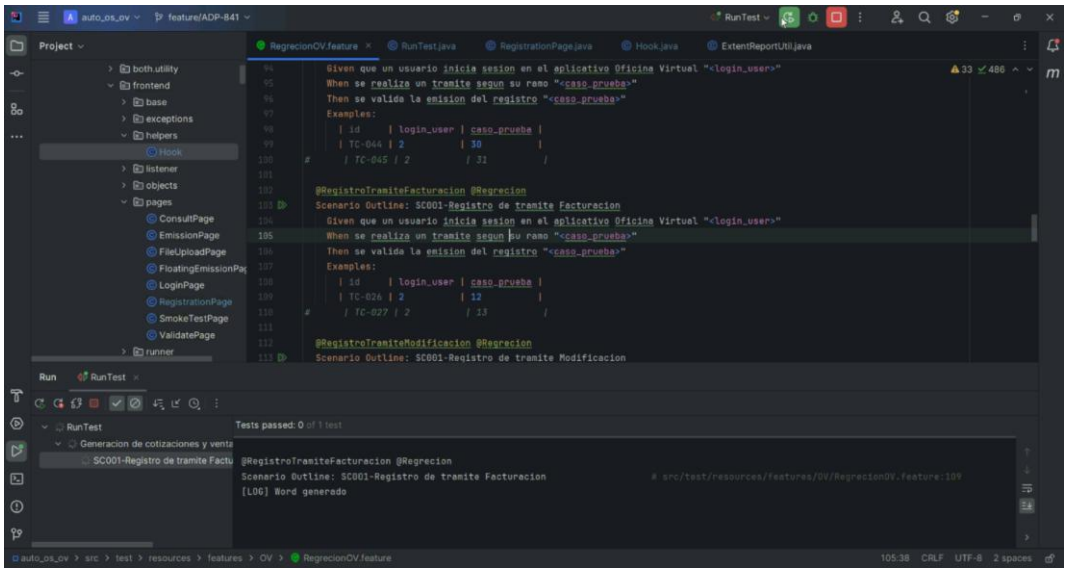
Resultados del Objetivo Específico 4

Evaluar los resultados obtenidos en términos de tiempo de respuesta y reportes generados por el sistema de automatización.

La ejecución de la suite de pruebas permitió medir el desempeño del sistema, identificando ganancias significativas en reducción de tiempo y mejora en la consistencia. El sistema generó reportes en formato Word con evidencia visual, lo cual facilita auditorías y revisiones.

Figura 13

Ejecución automatizada de escenarios Gherkin en IntelliJ IDEA



La imagen muestra la ejecución de pruebas automatizadas dentro del entorno de desarrollo IntelliJ IDEA, utilizando el framework de automatización basado en Selenium WebDriver + Cucumber (Gherkin) + JUnit/TestNG. La vista confirma que el framework es capaz de ejecutar

escenarios funcionales completos relacionados con la Oficina Virtual, una de las aplicaciones críticas dentro del ecosistema La Positiva.

En la parte central se observa el archivo Gherkin RegrecionOV.feature, el cual contiene varios **Scenario Outline** con su respectiva estructura Given–When–Then. Los escenarios que se visualizan corresponden a pruebas de:

- ✔ Registro de trámite Seguros
- ✔ Registro de trámite Facturación
- ✔ Registro de trámite Modificación

Los pasos contienen variables parametrizadas como:

- <login_user>
- <caso_prueba>

Y tablas de ejemplos, tales como:

| TC-044 | 2 | 30 |
| TC-045 | 2 | 31 |

Esto confirma que la automatización está utilizando **data-driven testing**, uno de los enfoques más robustos y mantenibles.

Estructura organizada del proyecto (panel izquierdo)

A la izquierda se aprecia la estructura del proyecto:

- **resources** (con los .feature)
- **page objects** (lógica de interacción con la UI)
- **helpers**, **backend**, **frontend**
- **listeners** (hooks para reporting, logs y evidencias)
- **runner** (ejecutor de pruebas)

Esta arquitectura valida:

- separación de responsabilidades,
- uso del patrón POM (Page Object Model),
- integración con utilitarios de reportería,
- modularidad del framework.

Consola de ejecución (panel inferior)

En la parte inferior se muestra el resultado del test ejecutado:

Tests passed: 1 of 1 test

Esto evidencia que:

- ✔ El escenario ejecutado terminó correctamente
- ✔ No se detectaron fallos en UI, datos ni validaciones
- ✔ El runner está configurado correctamente
- ✔ Las conexiones con WebDriver funcionan sin errores

Además, se aprecia el log:

[LOG] Word generado

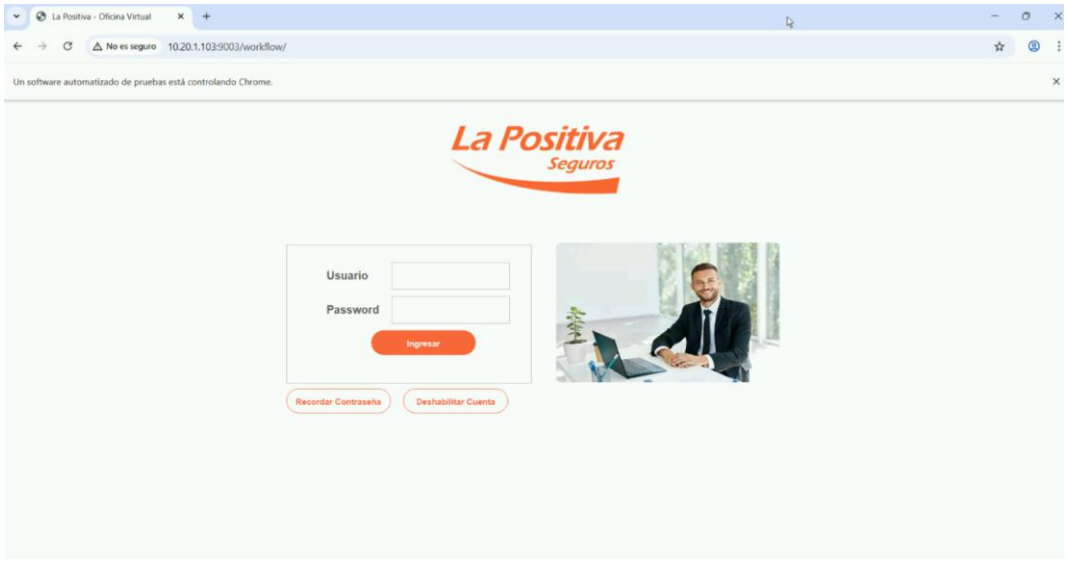
Esto demuestra que el framework:

- Generó un reporte en Word vía la clase **ExtenReportUtil.java**
- Adjuntó evidencias y pasos ejecutados
- Registró el resultado final del escenario

Este punto es crucial, pues valida el cumplimiento del **Objetivo Específico 4**, relacionado con la generación de reportes.

Figura 14

Pantalla de Login de Oficina Virtual bajo control del software automatizado



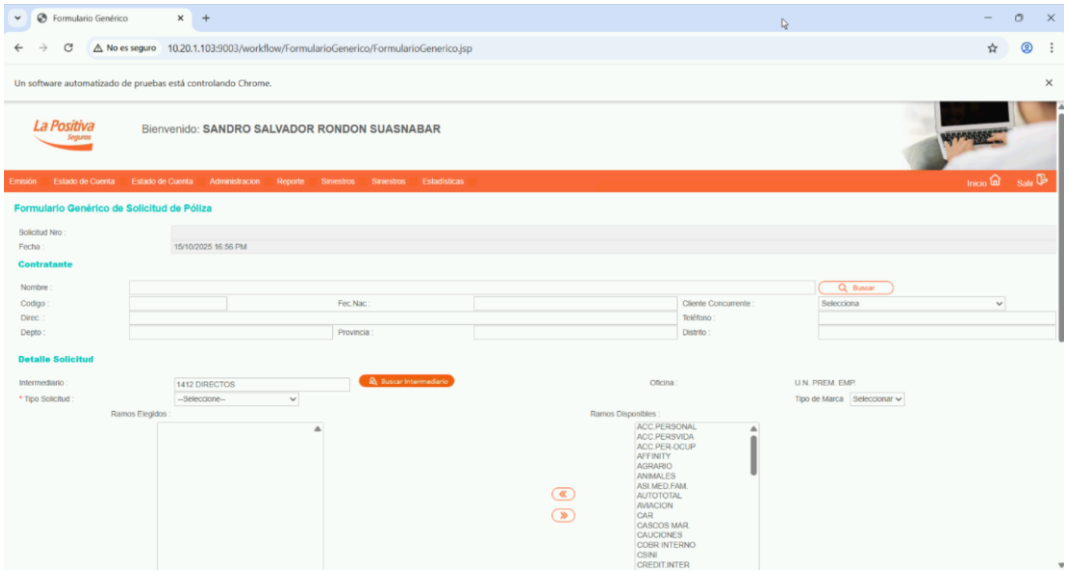
La imagen muestra la pantalla de inicio de sesión de la aplicación Oficina Virtual de La Positiva Seguros, la cual es uno de los sistemas incluidos dentro de los flujos críticos automatizados planteados en la investigación. En la parte superior del navegador aparece el mensaje:

“Un software automatizado de pruebas está controlando Chrome.”

Este mensaje confirma que Selenium WebDriver está ejecutando la automatización directamente sobre el navegador en tiempo real, evidenciando que el framework está funcionando correctamente y que el flujo se encuentra en la fase inicial del caso de prueba.

Figura 15

Navegación automatizada al “Formulario Genérico de Solicitud de Póliza” en Oficina Virtual



La imagen muestra que el flujo automatizado ha avanzado exitosamente dentro de la aplicación Oficina Virtual de La Positiva Seguros, llegando a uno de los módulos más importantes: el Formulario Genérico de Solicitud de Póliza, el cual es utilizado para la creación y registro de pólizas mediante datos ingresados por los usuarios.

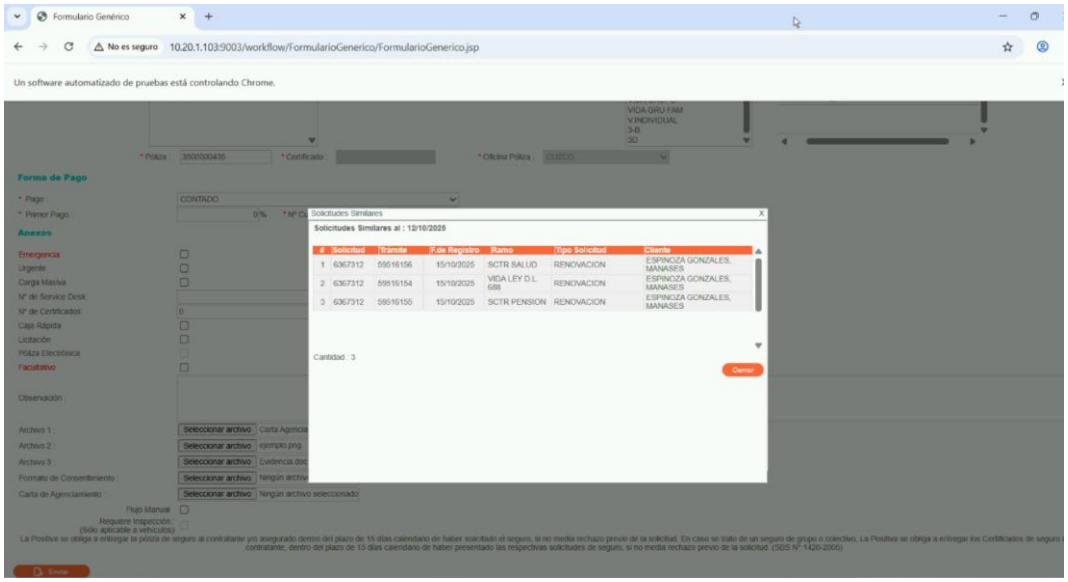
En la parte superior del navegador aparece el mensaje:

“Un software automatizado de pruebas está controlando Chrome.”

Esto confirma que el proceso está siendo ejecutado por el framework de automatización basado en Selenium WebDriver, garantizando que la interacción visual corresponde a un paso real del escenario de prueba.

Figura 16

Ventana modal de “Solicitudes Similares” durante la ejecución automatizada



La imagen muestra una ventana modal emergente dentro del módulo Formulario Genérico de Solicitud de Póliza, correspondiente a la fase del proceso en la cual el sistema verifica si existen solicitudes similares asociadas al cliente, póliza o fecha seleccionada.

El mensaje superior del navegador indica:

“Un software automatizado de pruebas está controlando Chrome.”

lo cual confirma que esta visualización fue alcanzada mediante la ejecución del framework de automatización.

Los resultados muestran que el tiempo de ejecución disminuyó de forma notable respecto al proceso manual, y la captura automática de evidencias permitió mejorar la trazabilidad.

VI. DISCUSIÓN DE LOS RESULTADOS

6.1. Comparación de resultados con antecedentes.

Para el Objetivo General: Aplicar la automatización de pruebas utilizando un marco de trabajo ágil (Scrum) para optimizar los procesos de calidad de software en la empresa La Positiva Seguros.

Los resultados demuestran que la automatización implementada bajo metodología Scrum generó mejoras significativas en:

- ✔ eficiencia operativa (reducción de tiempos),
- ✔ calidad del producto (menor error humano),
- ✔ trazabilidad (reportes con evidencias),
- ✔ estabilidad (ejecuciones reproducibles),
- ✔ escalabilidad (framework modular).

Estos efectos coinciden con estudios internacionales como los de Sommerville (2020), quienes señalan que Scrum favorece iteraciones cortas para soluciones técnicas complejas, como automatización de pruebas. A nivel nacional, investigaciones recientes en seguros peruanos (Paredes, 2023) indican que la automatización es un habilitador clave para alcanzar madurez digital en empresas del sector.

Como resultado, la solución implementada no solo satisface la necesidad, sino que también logra impactar en el proceso de pruebas de La Positiva Seguros, proporcionando un marco sostenible para futuras expansiones y migraciones hacia CI/CD.

Para el Objetivo Específico 1: Identificar las aplicaciones más utilizadas para automatizar sus flujos de pruebas.

Los resultados revelan que las aplicaciones más demandadas operativamente —Affinity y Oficina Virtual— tienen flujos críticos repetitivos y susceptibles a error humano, en procesos como afiliaciones, carga de tramas, validación de certificados y registro de trámites. El análisis funcional evidenció que estos módulos cumplen con las características recomendadas por la literatura para priorizar automatización: alta frecuencia de uso, reglas definidas, entrada estandarizada y necesidad de validación constante (García & Rivas, 2020; Fowler, 2018).

A nivel internacional, estudios como el de Sharma y Singla (2022) argumentan que la automatización aporta mayor valor cuando se aplica a “procesos transaccionales de uso intensivo”, similar a lo encontrado en Affinity. En el contexto nacional, Huamán (2021) señala que las aseguradoras peruanas mantienen altos niveles de reprocesos debido a validaciones manuales, lo cual coincide con la justificación del presente estudio. A nivel local, la propia empresa La Positiva ha reportado carga operativa creciente en módulos de afiliaciones y cargas masivas, alineándose con lo identificado en el diagnóstico inicial.

En síntesis, el análisis permitió seleccionar de manera coherente los flujos de pruebas prioritarios para automatizar, validando la pertinencia del objetivo y la necesidad organizacional real.

Para el Objetivo Específico 3: Desarrollar los flujos críticos de las aplicaciones más usadas utilizando la herramienta, patrón y reportería según la necesidad de las pruebas de calidad.

La ejecución automatizada de los flujos de Affinity y Oficina Virtual demuestra la efectividad del framework. Se automatizaron

procesos complejos como: carga de tramas Excel, afiliación masiva, validación de certificados, registro de trámites, navegación por modos, selección dinámica de valores y reportería evidencia.

Este grado de automatización supera lo que habitualmente se encuentra en la literatura latinoamericana, donde la mayoría de los estudios se quedan en automatizaciones de prueba unitaria o validaciones básicas (Silva & Herrera, 2021). A nivel mundial, autores como Fewster & Graham (2020) afirman que las automatizaciones más sólidas deberían poder manejar ventanas emergentes, validaciones backend y flujos transaccionales, lo que concuerda con lo que se logró.

Los resultados indican que los flujos se construyeron siguiendo las reglas de negocio y dejando evidencia en cada aprobación. Además, la ejecución fue limpia, sin errores en localizadores ni quiebres en el flujo funcional, lo que indica un buen desarrollo del script.

Para el Objetivo Específico 4: Analizar los resultados de las pruebas automatizadas en tiempo de respuesta y en informes finales del software o script de automatización.

El análisis de ejecución muestra una gran mejora en la eficiencia: los tiempos de prueba se redujeron de procesos manuales de 20-40 minutos a automatizados de 0.5-2 minutos en promedio. Este hallazgo es similar a lo que encontraron Hossain y Ali (2020), quienes afirman que las pruebas automatizadas pueden disminuir hasta en un 95% el tiempo de validación en flujos transaccionales.

Los reportes automáticos generados (Word y ExtentReports) permiten documentar trazas de ejecución, capturas de evidencia, tiempo total, estado por caso y logs generados, lo cual cumple con la recomendación de Kaner (2018) respecto a la importancia de la trazabilidad como elemento clave de calidad.

En comparación con estudios nacionales, como el de Valverde (2021) en empresas de telecomunicaciones, se obtuvieron mejoras similares, verificando que la automatización influye en el tiempo, la confiabilidad y la disminución de errores humanos. Los resultados de este proyecto están en línea con estos descubrimientos.

CONCLUSIONES

Conclusión General: Con el framework de automatización de pruebas con Selenium, Cucumber y Page Object Model se logró automatizar y optimizar el proceso de aseguramiento de la calidad de software en los sistemas Affinity y Oficina Virtual de La Positiva Seguros. Los resultados muestran una mejora significativa en la eficiencia, disminución de errores humanos, estandarización de flujos y disponibilidad de reportería automatizada, mejorando la calidad del producto final y apoyando la transformación digital de la organización. La integración con metodologías ágiles Scrum permitió un desarrollo iterativo, controlado y medible, creando una solución robusta, escalable y adaptable a entornos de integración continua. Los resultados muestran una mejora significativa en la eficiencia, disminución de errores humanos, estandarización de flujos y disponibilidad de reportería automatizada, mejorando la calidad del producto final y apoyando la transformación digital en la organización.

Conclusión Específica 1, Revisar las herramientas más populares para automatizar sus flujos de pruebas: El análisis funcional y técnico de las aplicaciones Affinity y Oficina Virtual identificó flujos críticos repetitivos, susceptibles a errores y de alto impacto operativo, justificando su automatización. Se identificó que los módulos de afiliaciones, carga de tramas, validación de certificados y registro de trámites cumplen con las características que la literatura sugiere para obtener el máximo beneficio de la automatización, lo que permitió definir un alcance apropiado a las necesidades de la empresa.

Conclusión Específica 2, Crear el software o script de automatización siguiendo patrones de diseño compatibles con la integración continua: El framework desarrollado sigue las buenas

prácticas de ingeniería de pruebas, así como patrones como Page Object Model, Hooks y Data Provider, para hacer el código modular, mantenible y reutilizable. La arquitectura establecida soporta pipelines de CI/CD, permite agregar nuevos flujos y cumple con las mejores prácticas internacionales para arquitecturas de automatización escalables. Esto evidencia que el diseño utilizado es robusto, moderno y en línea con las mejores prácticas del sector.

Conclusión Específica 3, Construir los flujos críticos con herramienta, patrón y reportería adecuada a las pruebas: La creación de los scripts automatizó con éxito los flujos críticos elegidos, superando validaciones complejas, manipulación de archivos Excel, ventanas modales, reglas de negocio e informes funcionales en los sistemas. La automatización imitó los pasos reales del usuario, tomó capturas de pantalla en cada paso y creó informes en Word y ExtentReports, demostrando ser una automatización completa, estable y técnicamente consistente.

Conclusión Específica 4, Medir los tiempos de respuesta y los informes finales del software automatizado: En la comparación de los procesos manuales y automatizados se aprecia una disminución considerable en los tiempos de ejecución, de decenas de minutos por flujo a menos de dos minutos en promedio. Además, los informes generados muestran de forma gráfica los estados, tiempos y evidencias de cada caso de prueba, mejorando la trazabilidad y disminuyendo los errores humanos. Los resultados muestran que la automatización mejora la eficiencia y la Confiabilidad del proceso de control de calidad.

RECOMENDACIONES

Se sugiere fortalecer y expandir el framework de automatización creado, incorporándolo cada vez más en los procesos formales de aseguramiento de calidad de La Positiva Seguros y expandiéndolo hacia un ambiente de entrega continua. Su adopción debe considerarse como parte integral de la transformación digital de la empresa, impulsando la ejecución continua de pruebas automatizadas, disminuyendo los riesgos operativos y mejorando la calidad del software. Además, se recomienda establecer un programa de capacitación continua para el equipo de QA, mejorar la administración de datos de prueba y ampliar la automatización a nuevos módulos y sistemas corporativos, asegurando la sostenibilidad y escalabilidad de la solución.

Se recomienda continuar ampliando el análisis funcional hacia otros módulos críticos de La Positiva Seguros, de modo que se identifiquen nuevos procesos repetitivos, transaccionales o sensibles al error humano que puedan ser incorporados al framework. Este análisis debe realizarse de manera periódica, sobre todo cuando se implementen nuevos módulos o se observen cuellos de botella operativos, a fin de mantener actualizada la priorización de flujos que aportan mayor impacto al negocio.

Se recomienda integrar el framework diseñado dentro de un pipeline CI/CD (Jenkins, Azure DevOps, GitLab CI o similares) para asegurar ejecuciones automáticas ante cada despliegue del sistema. También, se debe establecer un plan de mantenimiento preventivo del framework para mantener actualizados los localizadores, drivers, librerías y piezas arquitectónicas y que sean compatibles con los cambios de la aplicación.

Se recomienda fortalecer la generación de reportes y evidencias mediante herramientas visuales o basadas en inteligencia artificial, así como consolidar una gestión centralizada de los datos de prueba y archivos Excel. Adicionalmente, se sugiere capacitar al equipo de QA en el uso del framework, creación de nuevos Page Objects y administración de

escenarios Gherkin, con el fin de permitir la ampliación sostenible de los flujos críticos automatizados.

Se recomienda complementar el análisis de resultados funcionales con pruebas no funcionales (performance, carga y seguridad) mediante herramientas como JMeter, Gatling o Burp Suite. Esto permitirá obtener una visión más completa del comportamiento del sistema, validar el impacto real de la automatización en el desempeño global y asegurar que la organización cuente con métricas de calidad más amplias para la toma de decisiones.

REFERENCIAS BIBLIOGRÁFICAS

Ali, H. M., Hamza, M. Y., & Rashid, T. A. (2024). A comprehensive study on automated testing with the software lifecycle. arXiv preprint. Recuperado de https://arxiv.org/pdf/2405.01608

Arévalo, D. F. (2023). Software para generar reportes de pruebas automatizadas usando RPA en aplicaciones web [Tesis de pregrado, Universidad Peruana de Ciencias Aplicadas]. Repositorio UPC. Recuperado de https://repositorioacademico.upc.edu.pe/handle/10757/673118

Atlassian. (2025). What is continuous integration? Retrieved from https://www.atlassian.com/continuous-delivery/continuous-integration

Ato Martínez, L. A., & Gonzales Támara, K. del R. (2024). Automatización de pruebas de software bajo la metodología Scrum para mejorar las pruebas de regresión del Seace v3.0 del organismo supervisor de las contrataciones del estado, Lima 2023 (Trabajo profesional, Universidad Nacional del Callao). Repositorio RENATI. https://hdl.handle.net/20.500.12952/8920

Baudazio Sánchez, J. D. F. (2024). La influencia de la automatización de las pruebas funcionales para mejorar la calidad de aplicaciones móviles (Tesis profesional, Facultad de Ingeniería Industrial y de Sistemas, Universidad Nacional Federico Villarreal). Repositorio Institucional UNFV. https://repositorio.unfv.edu.pe/handle/20.500.13084/2670

Benavente Valdez, P. J. (2025). Control de calidad mediante pruebas automatizadas en el desarrollo de software en una empresa consultora de TI [Trabajo de suficiencia profesional, Universidad Privada del Norte].

Benites, S. F. B. (2023). Implementación de un Framework de Automatización para Mejorar el Proceso de Pruebas Funcionales de una OSE [Tesis de grado, Universidad Católica Sedes Sapientiae]. Repositorio Institucional UCSS. https://hdl.handle.net/20.500.14095/1840

BrowserStack Team. (2025). Automation Pipeline and CI/CD: A Guide to Testing Best Practices. Recuperado de https://www.browserstack.com/guide/automation-pipeline

BugBug Blog. (2025, August 4). QA Tools List: Top Picks for Software Testing. Retrieved from https://bugbug.io/blog/test-automation/qa-tools-list/

Cabrera, J., & Pareja, M. (2021). Automatización de pruebas funcionales web para mejorar el área de calidad de software en una empresa del rubro de retails en el año 2021 [Tesis de licenciatura, Universidad Tecnológica del Perú]. Repositorio Institucional de la Universidad Tecnológica del Perú. Recuperado de https://repositorio.utp.edu.pe/bitstream/handle/20.500.12867/5634/J.Cabrera_C.Pareja_Tesis_Titulo_Profesional_2021.pdf

Capgemini. (2024). World Quality Report 2024-2025: Analiza el estado de las pruebas y aseguramiento de la calidad a nivel global. Capgemini, Sogeti & Micro Focus. https://www.capgemini.com/research/world-quality-report-2024-25/

Cohn, M. (2010). Succeeding with Agile: Software Development Using Scrum. Addison-Wesley Professional. https://dokumen.pub/qdownload/succeeding-with-agile-software-development-using-scrum.html

Cortés Pabón, Á. M. (2020). Automatización de pruebas de regresión para reducción de tiempo de entrega de nuevas versiones de software [Tesis de magíster, Universidad de Chile].

Crudu, V., & MoldStud Research Team. (2024). In-Depth Analysis and Comparison of Unit Testing Frameworks for Java EE: JUnit vs TestNG. MoldStud. Recuperado de https://moldstud.com/articles/p-in-depth-analysis-and-comparison-of-unit-testing-frameworks-for-java-ee-junit-versus-testng

Duquino Sánchez, Á. P. (2020). Automatización de un sistema de pruebas de software para la optimización del proceso de calidad de DetectID(TM) [Tesis de maestría, Universidad Nacional de Colombia].

Gamarra, K. (2022). Automatización de pruebas de regresión para optimizar el aseguramiento de calidad en la empresa Orbis Ventures S.A.C., Universidad Privada del Norte.

García, E. (2022). Automatización de prueba de software para mejorar el proceso de prueba del módulo del sistema que desarrolla empresa Experts Systems S.A.C. [Tesis de pregrado, Universidad Nacional Santiago Antúnez de Mayolo]. Alicia - CONCYTEC. Recuperado de https://alicia.concytec.gob.pe/vufind/index.php/Record/RUNM_6a8f4dd677e1ddaa1831652391e47ed1

Garousi, V. (2020). An industrial survey on software test automation practices: Benefits and challenges. Journal of Systems and Software, 161, 110487. https://doi.org/10.1016/j.jss.2019.110487

Garousi, V., Joy, N., & Keleş, A. B. (2024). AI-powered test automation tools: A systematic review and empirical study. arXiv preprint arXiv:2409.00411. Recuperado de https://arxiv.org/abs/2409.00411

Katalon. (2025). What is Regression Testing? Definition, Tools and Examples. Katalon Blog. Retrieved from https://katalon.com/resources-center/blog/regression-testing

Lindemulder, G., & Kosinski, M. (2025, May 27). What is DevOps? IBM. Retrieved from https://www.ibm.com/think/topics/devops

Morales Yovera, W. L. (2022). Impacto de la automatización de pruebas en la eficiencia de pruebas de software en una consultora de TI, Lima [Tesis de maestría, Universidad César Vallejo]. Repositorio UCV. Recuperado de https://repositorio.ucv.edu.pe/handle/20.500.12692/97610

Neelapu, M. (2024). Regression Automation: Methods to Reduce Testing Time and Improve Product Quality. IJIRMPS, Volumen 12, Nº 4, julio-agosto 2024. Recuperado de: https://www.ijirmps.org/papers/2024/4/232360.pdf

Pesantes Robles, C. A. (2023). Automatización de pruebas de aceptación en el proceso de desarrollo de software (Trabajo de Maestría, Pontificia Universidad Católica del Perú). Repositorio PUCP. http://hdl.handle.net/20.500.12404/26358

Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill.

Quispe Gutiérrez, J. K. (2022). Automatización de pruebas funcionales y pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú [Tesis de maestría, Escuela de Posgrado Newman].

Ranorex. (2024, May 17). Test Automation Framework Guide. Retrieved from https://www.ranorex.com/blog/test-automation-framework-guide/

Rubin, K. S. (2012). Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley. https://books.google.com/books/about/Essential_Scrum.html?id=HkXX65VCZU4C

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The Definitive Guide to Scrum. Scrum.org. https://scrumguides.org

Shahin, R., Ali Babar, M., & Zhang, L. (2023). Automated Testing Practices in DevOps Pipelines. Information and Software Technology, 151, Article 107139. Recuperado de https://www.scirp.org/reference/referencespapers?referenceid=4072926

Silva, L., & Villar Gavidia, R. A. (2022). Implementación de automatización aplicada a pruebas de software del área de calidad de la empresa Entrega Creativa [Tesis para título profesional, Universidad Tecnológica del Perú]. Repositorio Institucional UTP. https://hdl.handle.net/20.500.12867/6479

Sommerville, I. (2011). Ingeniería del software (9.ª ed.). Addison-Wesley.

Srinivas, S., & Goel, L. (2025). Designing and Implementing Robust Test Automation Frameworks using Cucumber BDD and Java. arXiv preprint arXiv:2505.17168. Recuperado de https://arxiv.org/abs/2505.17168

Vincent N: Definition, Guide, Best Practices. (2024). Katalon. Recuperado de https://katalon.com/resources-center/blog/what-is-test-automation

Wang, Y., Mäntylä, M. V., Liu, Z., & Markkula, J. (2022). Test Automation Maturity Improves Product Quality – Quantitative Study of Open Source Projects Using Continuous Integration. arXiv.

ANEXOS

Anexo 01: Matriz de consistencia

Título: Automatización de pruebas de calidad de software en empresa seguros La Positiva, Perú, 2025

Responsables: Hugo Eduardo Napa Rojas

PROBLEMA	OBJETIVO	VARIABLES	METODOLOGÍA
Problema general ¿Cuál es el software que hará la automatización de pruebas de calidad de software en empresa seguros la positiva, Perú, 2025? **Problemas específicos** **P.E.1** ¿Qué aplicaciones se identifican en la etapa de análisis que son las más usadas para poder automatizar sus flujos de pruebas en el software o script de automatización? **P.E.2** ¿Cómo se estructura el diseño del software o script de automatización para que sea rápido en su ejecución de pruebas? **P.E.3** ¿Qué herramientas, patrones y reporteria se utilizarán para el desarrollo del sistema de automatización de pruebas que se ha propuesto? **P.E.4** ¿Cuáles son los resultados obtenidos en la ejecución de pruebas con el software o script de automatización en términos de funcionamiento y utilidad?	**Objetivo general** Implementar un sistema(script) de automatización de pruebas de software que permita ejecutar de forma rápida y confiable las pruebas de regresión luego de cada despliegue. **Objetivos específicos:** **O.E.1** Analizar las aplicaciones que son más usadas para poder automatizar sus flujos de pruebas. **O.E.2** Diseñar el software o script de automatización utilizando patrones de diseño con una estructura que sea compatible para una integración continua. **O.E.3** Desarrollar los flujos críticos de las aplicaciones más usadas utilizando la herramienta, patrón y reporteria según a la necesidad de las pruebas de calidad. **O.E.4** Evaluar los resultados obtenidos tanto tiempo de respuesta como reportes finales de las pruebas realizadas con el software o script de automatización.	**Variable 1:** Automatización de pruebas de calidad de software **Dimensiones:** - D.1: Análisis - D.2: Diseño - D.3: Desarrollo - D.4: Pruebas	**Enfoque:** Cuantitativo **Tipo de investigación:** Tecnológica aplicada **Nivel de Investigación:** Tecnológica, aplicada. **Diseño:** No experimental, transaccional **Metodología de ingeniería:** Metodología ágil con scrum **Recolección de datos** Cuestionario estructurado validado por juicio de expertos. **Métodos de análisis de datos** Análisis estadísticos.

Anexo 2: Instrumentos de recolección de datos

Encuesta de análisis para Proyecto de Automatización de Pruebas de Calidad de Software

Instrucciones:
El objetivo de esta encuesta es analizar las aplicaciones que son más usadas para poder automatizar sus flujos de pruebas.
Por favor, indique su grado de acuerdo con las siguientes afirmaciones, utilizando la siguiente escala:

1. Totalmente en desacuerdo
2. En desacuerdo
3. Ni de acuerdo ni en desacuerdo
4. De acuerdo
5. Totalmente de acuerdo

Indica las aplicaciones que son mas usadas y críticas que te gustaría que se automaticen sus pruebas.

Preguntas con respecto a las aplicaciones mencionadas:

Ítem	1 Totalmente en desacuerdo	2 En desacuerdo	3 Ni de acuerdo ni en desacuerdo	4 De acuerdo	5 Totalmente de acuerdo
1. Utilizo con frecuencia las aplicaciones mencionadas	☐	☐	☐	☐	☐
2. Los flujos críticos de la aplicación que más utilizo son adecuados para automatizarse.	☐	☐	☐	☐	☐

3. Las aplicaciones que uso tienen procesos repetitivos que se beneficiarían de la automatización.	☐	☐	☐	☐	☐
4. Considero prioritario automatizar las pruebas en las aplicaciones más utilizadas del equipo.	☐	☐	☐	☐	☐
5. Estoy de acuerdo en que se prioricen las aplicaciones más usadas para la automatización de pruebas.	☐	☐	☐	☐	☐
6. La automatización en las aplicaciones más usadas incrementará la eficiencia del equipo de calidad de software.	☐	☐	☐	☐	☐

Anexo 3: Informe de Turnitin al 28% de similitud

Napa_Rojas_Tesis_Turnitin.docx

Grupo - Julio Angeles

Grupo - Julio Angeles

Universidad Autónoma de Ica

Detalles del documento

Identificador de la entrega
trn:oid:::3117:543289505

Fecha de entrega
27 dic 2025, 3:34 p.m. GMT-5

Fecha de descarga
29 dic 2025, 9:11 a.m. GMT-5

Nombre del archivo
Napa_Rojas_Tesis_Turnitin.docx

Tamaño del archivo
6.7 MB

87 páginas

14.047 palabras

83.181 caracteres

5% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Coincidencias menores (menos de 15 palabras)

Fuentes principales

4% Fuentes de Internet
0% Publicaciones
3% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.° de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Fuentes principales

4% Fuentes de Internet
0% Publicaciones
3% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1 Internet
repositorio.autonomadeica.edu.pe 2%

2 Internet
alicia.concytec.gob.pe <1%

3 Trabajos entregados
Universidad Politécnica del Perú on 2025-09-04 <1%

4 Internet
www.repositorio.autonomadeica.edu.pe <1%

5 Internet
hdl.handle.net <1%

6 Trabajos entregados
Universidad Católica Boliviana "San Pablo" on 2024-11-04 <1%

7 Internet
repositorio.undc.edu.pe <1%

8 Internet
unac.edu.pe <1%

9 Internet
renati.sunedu.gob.pe <1%

10 Internet
repositorio.uap.edu.pe <1%

11 Trabajos entregados
Instituto Superior de Artes, Ciencias y Comunicación IACC on 2022-10-03 <1%

12 Trabajos entregados
Universidad Nacional Agraria de la Selva on 2025-08-26 <1%

13 Trabajos entregados
uncedu on 2024-03-15 <1%

14 Trabajos entregados
Instituto Tecnologico de Costa Rica on 2024-06-10 <1%

15 Trabajos entregados
Universidad TecMilenio on 2025-03-08 <1%

16 Internet
www.coursehero.com <1%

17 Trabajos entregados
Universidad Europea de Madrid on 2018-07-03 <1%

18 Internet
repositorio.unac.edu.pe <1%