UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE

"Implementación y Validación de un Framework de Automatización de Pruebas con Katalon Studio, TestOps y TestCloud en una Empresa de Telecomunicaciones"

CAMILO SILVA MOLINA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA

Profesor Guía: JAVIER MALDONADO CARMONA

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: "Implementación y Validación de un Framework de Automatización de pruebas con Katalon Studio, TestOps y TestCloud en una empresa de telecomunicaciomnes"

Nombre del candidato(a): Camilo Hernán Silva Molina

Carrera / Grado: Ingenería en Infotmática.

Campus: Concepcion **;** **Departamento**: Departamento de Electronica e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Javier Alexis Maldonado Carmona, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 14/10/2025 ; **Firma:** 

Estudiante o Candidato(a):

Fecha: 14/10/2025 ; **Firma:** 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Julio - 2025

RESUMEN

La presente memoria de título aborda la implementación y validación de un framework integral en conjunto con la creación e integración de scripts enfocados en automatización para el aseguramiento de calidad automatizado para aplicaciones móviles y web dentro de WOM, una destacada empresa chilena de telecomunicaciones. El trabajo se fundamenta en la experiencia práctica desempeñada en el rol de Analista QA Automatizador, donde se identificaron las limitaciones inherentes al testing manual en un entorno de desarrollo ágil y de alta demanda. La metodología empleada consistió en la automatización de pruebas funcionales utilizando Katalon Studio para la creación de scripts, Katalon TestOps como orquestador de ejecuciones y TestCloud para la ejecución en la nube, permitiendo así el testeo en múltiples dispositivos, múltiples sistemas operativos y múltiples navegadores. Los resultados principales demuestran una mejora significativa en la eficiencia del proceso de pruebas, una ampliación sustancial de la cobertura de testing, una reducción del tiempo de lanzamiento al mercado y un incremento global en la calidad del software entregado. La relevancia de estos hallazgos radica en la demostración de cómo la automatización de pruebas, integrada con soluciones en la nube, se convierte en un imperativo estratégico para el sector de las telecomunicaciones, permitiendo a las empresas mantener su competitividad y asegurar la fiabilidad de sus servicios digitales en un ecosistema tecnológico en constante evolución.

Palabras Clave: Automatización de Pruebas; Calidad de Software; Katalon Studio; Katalon TestOps; Katalon TestCloud; Telecomunicaciones; QA Automatizado; Testing Móvil; Testing Web.

ABSTRACT

Abstract— This capstone project addresses the implementation and validation of a comprehensive framework, together with the development and integration of automation scripts, to enable automated quality assurance for mobile and web applications at WOM, a leading Chilean telecommunications company. The work is grounded in hands-on experience as a QA Automation Analyst, where the inherent limitations of manual testing in an agile, high-demand development environment were identified. The methodology centered on automating functional tests using Katalon Studio to author scripts, Katalon TestOps to orchestrate executions, and TestCloud for cloud-based execution, thereby enabling testing across multiple devices, operating systems, and browsers. The main results show significant improvements in testing efficiency, a substantial expansion of test coverage, reduced time-to-market, and an overall increase in delivered software quality. These findings demonstrate how test automation, integrated with cloud solutions, becomes a strategic imperative for the telecommunications sector, allowing companies to remain competitive and ensure the reliability of their digital services in a constantly evolving technological ecosystem.

Keywords— Test Automation; Software Quality; Katalon Studio; Katalon TestOps; Katalon TestCloud; Telecommunications; Automated QA; Mobile Testing; Web Testing.

GLOSARIO

API: Interfaz de Programación de Aplicaciones — conjunto de reglas para que sistemas se comuniquen.

CI/CD: Integración Continua / Entrega Continua — prácticas para integrar y desplegar software de forma continua.

CSV: Valores Separados por Comas — formato de archivo de datos tabulares.

DDT: Testing Orientado a Datos — ejecución de pruebas con conjuntos de datos externos.

DevOps: Desarrollo y Operaciones — prácticas que integran desarrollo, QA y operaciones.

DOM: Modelo de Objetos del Documento — representación estructurada de una página web.

HTML: Lenguaje de Marcado de Hipertexto — formato estándar para páginas web y reportes.

IaC: Infraestructura como Código — provisión de infraestructura mediante archivos de configuración.

iOS: Sistema operativo móvil de Apple.

JSON: Notación de Objetos de JavaScript — formato ligero de intercambio de datos.

KISS: “Keep It Simple, Stupid” — principio de diseño que prioriza la simplicidad.

KRE: Katalon Runtime Engine — motor de ejecución para correr pruebas fuera del IDE.

MTTD: Mean Time To Detect — Tiempo Medio de Detección de incidentes.

MTTR: Mean Time To Repair — Tiempo Medio de Reparación.

OS: Operating System — Sistema Operativo.

PDF: Formato de Documento Portátil — formato de exportación de reportes.

REST: Representational State Transfer — estilo arquitectónico para APIs.

ROI: Return on Investment — Retorno de la Inversión.

SOAP: Simple Object Access Protocol — protocolo para servicios web.

SUT: System Under Test — Sistema Bajo Prueba.

UI: Interfaz de Usuario.

UTFSM: Universidad Técnica Federico Santa María.

UX: Experiencia de Usuario.

WOM: Word of Mouth Operador de telecomunicaciones WOM (Chile).

XML: Lenguaje de Marcado Extensible — formato de datos estructurados.

ZSmart: Plataforma interna de la empresa WOM

Robot: Librería JAVA

Katalon IDE: Editor de codigo

ÍNDICE

INDICE FIGURAS

INTRODUCCIÓN

En la era digital actual, la calidad del software es un pilar fundamental para el éxito y la competitividad de las empresas, especialmente en el sector de las telecomunicaciones. La industria de las telecomunicaciones, caracterizada por su dinamismo y la constante demanda de nuevos servicios y funcionalidades, requiere sistemas de software robustos, fiables y de alto rendimiento. En este contexto, el aseguramiento de la calidad (QA) se posiciona como una disciplina crítica, encargada de garantizar que las aplicaciones y plataformas cumplan con los más altos estándares antes de llegar al usuario final.

Tradicionalmente, el testing manual ha sido el enfoque predominante para la verificación de software. Sin embargo, las crecientes complejidades de las aplicaciones modernas, los ciclos de desarrollo acelerados y la necesidad de testear en una miríada de dispositivos y entornos han expuesto las limitaciones intrínsecas de este método. La lentitud, la propensión al error humano y la dificultad para escalar las pruebas manuales se han convertido en cuellos de botella significativos, impactando negativamente en la agilidad empresarial y la capacidad de respuesta al mercado.

Frente a estos desafíos, la automatización de pruebas ha emergido como una solución transformadora. Al emplear herramientas de software para ejecutar pruebas de manera repetible y eficiente, la automatización no sólo acelera el ciclo de vida del desarrollo, sino que también mejora la consistencia y la cobertura de las pruebas, permitiendo la detección temprana de defectos y la optimización de recursos. La integración de la automatización con plataformas en la nube amplifica aún más estos beneficios, al proporcionar entornos de ejecución escalables y accesibles para el testeo multidispositivo.

La presente memoria de título documenta la experiencia de implementar un framework de automatización de pruebas en WOM, una empresa chilena de telecomunicaciones, utilizando la suite de herramientas Katalon (Katalon Studio, Katalon TestOps y Katalon TestCloud). El objetivo central es demostrar cómo esta implementación práctica ha abordado las problemáticas del testing manual, mejorando la eficiencia y la calidad del software en un entorno de telecomunicaciones. El trabajo se estructura en cinco capítulos principales, comenzando con la definición detallada del problema, seguido por el marco conceptual, la propuesta de solución, la validación empírica y, finalmente, las conclusiones y futuras mejoras. Este documento busca no solo formalizar una experiencia laboral en un contexto académico, sino también ofrecer un modelo aplicable para otras organizaciones que busquen optimizar sus procesos de aseguramiento de calidad.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

Este capítulo se dedica a la identificación y caracterización del problema que motiva la presente memoria de título . Se examinará el papel del aseguramiento de calidad en el sector de las telecomunicaciones, se detallarán las deficiencias inherentes al testing manual, se analizará la falta de visibilidad y trazabilidad en los procesos de prueba, y se justificará la imperiosa necesidad de adoptar la automatización con integración en la nube.

1.1 El rol de QA en empresas de telecomunicaciones

El rol de un Analista QA en una empresa de telecomunicaciones es fundamental para garantizar la fiabilidad y el rendimiento de los servicios digitales que se ofrecen a millones de usuarios. Las responsabilidades de este profesional son multifacéticas y abarcan diversas etapas del ciclo de vida del software. Incluyen el diseño de pautas de prueba basadas en un análisis exhaustivo de la documentación técnica y funcional de los desarrollos, así como la ejecución de pruebas funcionales y no funcionales sobre sistemas nuevos y en mantenimiento, con el objetivo primordial de identificar posibles errores de diseño.[1]

La misión principal de un Analista QA es asegurar la calidad del software mediante la ejecución rigurosa de pruebas funcionales, garantizando que las aplicaciones cumplan con los más altos estándares de calidad. Esto implica una colaboración estrecha con los equipos de desarrollo y análisis, validando datos y reportando resultados de manera efectiva.[2] Las tareas específicas en un entorno de telecomunicaciones son amplias y complejas. Se incluyen la creación y ejecución de pruebas funcionales manuales para nuevas aplicaciones de la compañía, la coordinación y ejecución de pruebas según las necesidades del proyecto, y la validación de la migración de datos y la continuidad del negocio. Además, el rol implica reportar y hacer seguimiento de bugs y errores durante las pruebas, acompañar a los usuarios finales durante las pruebas de aceptación de usuario (UAT), e informar sobre riesgos, plazos y cumplimiento de cronogramas (Gantt).[2]

Más allá de las pruebas funcionales, un Analista QA en telecomunicaciones también realiza pruebas de rendimiento y escalabilidad del software, genera scripts SQL para la validación de datos en bases de datos como SQL Server, y utiliza herramientas como Postman para pruebas de

integración de APIs. La configuración y gestión de entornos de prueba en Test, UAT y Pre-Producción son también responsabilidades clave.[2]

La amplitud y criticidad de las tareas de QA en el sector de las telecomunicaciones, combinadas con la complejidad inherente y la escala de los sistemas, implican que los enfoques de QA tradicionales y de alcance limitado son insuficientes. La naturaleza vital de los servicios de telecomunicaciones, que a menudo constituyen infraestructura crítica, exige niveles excepcionales de fiabilidad, disponibilidad y rendimiento. Cualquier fallo puede acarrear consecuencias financieras y reputacionales significativas. Por lo tanto, la función de QA en este sector no se limita a la mera detección de errores, sino que se erige como un componente estratégico para la mitigación de riesgos y la garantía de la continuidad del servicio. La necesidad de realizar pruebas exhaustivas en diversas funcionalidades y parámetros de rendimiento hace que este rol sea excepcionalmente desafiante, destacando las limitaciones intrínsecas de la ejecución manual para alcanzar los estándares de calidad y la velocidad requeridos en un entorno de telecomunicaciones tan dinámico. Esto sienta las bases para comprender por qué la automatización no es simplemente una mejora de la eficiencia, sino una necesidad estratégica ineludible.

1.2 Problemáticas del testing manual

El testing manual, a pesar de su adaptabilidad inicial y la capacidad de los testers para aplicar creatividad y juicio en escenarios exploratorios [3], presenta una serie de problemáticas significativas que lo convierten en un cuello de botella en los ciclos de desarrollo de software modernos, especialmente en entornos de alta exigencia como el de las telecomunicaciones.

Una de las principales desventajas es el **consumo de tiempo y recursos**.[3] Las pruebas manuales son intrínsecamente lentas y requieren una inversión considerable de mano de obra, lo que se traduce en costos elevados, particularmente en ciclos de prueba extensos o cuando se manejan grandes volúmenes de datos.[3] Esta lentitud ralentiza el proceso de desarrollo y aumenta los gastos del proyecto.[5]

La **propensión al error humano y la inconsistencia** son también desafíos críticos. Los testers manuales, por naturaleza, pueden cometer errores al ejecutar pasos, registrar resultados de forma imprecisa o pasar por alto defectos debido a la monotonía y repetitividad de las tareas.[3] Esta variabilidad humana compromete la fiabilidad de los resultados de las pruebas.[6]

Además, el testing manual exhibe una **escalabilidad y cobertura limitadas**.[3] La ejecución

manual se vuelve inviable en proyectos de gran envergadura o con plazos ajustados, ya que es prácticamente imposible probar todas las combinaciones y escenarios posibles de forma manual, resultando en una cobertura de pruebas incompleta.[3]

Los **desafíos en la regresión y el mantenimiento** son acentuados en proyectos en evolución. Las pruebas de regresión manuales, necesarias para asegurar que los nuevos cambios no afecten funcionalidades existentes, demandan un esfuerzo considerable a medida que el software se modifica, lo que genera una sobrecarga de mantenimiento sustancial.[3] La repetitividad y monotonía de estas pruebas pueden llevar a la fatiga del tester, reduciendo la motivación y aumentando la probabilidad de pasar por alto problemas críticos.[3]

La acumulación de estas problemáticas convierte al testing manual en un obstáculo significativo para la agilidad empresarial. Las ineficiencias y limitaciones inherentes al testing manual entran en conflicto directo con las demandas de velocidad y calidad de los ciclos de desarrollo actuales. El efecto acumulativo de estos problemas es que el testing manual se convierte en un **cuello de botella sustancial en el pipeline de entrega de software**. Esto no solo retrasa el tiempo de comercialización de nuevas características y servicios, sino que también eleva el riesgo de que defectos críticos lleguen a producción, lo que puede derivar en insatisfacción del cliente, interrupciones del servicio y posibles pérdidas financieras para las empresas de telecomunicaciones. El problema trasciende la mera "prueba" para impactar directamente en la **agilidad general del negocio y la experiencia del cliente**.

Para ilustrar de forma concisa las diferencias fundamentales, se presenta la siguiente tabla comparativa:

Aspecto	Testing Manual	Testing Automatizado
Velocidad	Lento [5]	Rápido [5]
Consistencia	Propenso a errores/inconsistente [3]	Consistente/Fiable [6]
Escalabilidad	Limitada [3]	Alta [5]
Cobertura	Limitada [3]	Amplia [6]
Costo Inicial	Menor [3]	Mayor [3]
Costo a Largo Plazo	Mayor [4]	Menor [5]

Detección de Errores	Tardía [6]	Temprana [6]
Reutilización	Baja [5]	Alta [5]
Adaptabilidad a Cambios	Alta (exploratoria) [3]	Media/Alta (con mantenimiento) [6]
Feedback	Lento [6]	Rápido [6]
Requerimientos de Habilidad	Intuición/Juicio [3]	Habilidades de Codificación (para scripts) [11]

Tabla 1.

1.3 Falta de visibilidad y trazabilidad

La falta de visibilidad y trazabilidad en los procesos de aseguramiento de calidad de software representa un desafío crítico para las organizaciones modernas, con repercusiones significativas que van más allá del ámbito técnico. Esta deficiencia se manifiesta en varios frentes, impactando la eficiencia operativa, la gestión financiera y la capacidad de cumplimiento normativo.

En primer lugar, una visibilidad deficiente de TI conduce a un **aumento de los costos y una gestión ineficaz**.[12] Las organizaciones corren el riesgo de gastar en exceso en licencias subutilizadas o innecesarias, lo que sobrecarga los presupuestos de TI y desvía fondos de iniciativas críticas. La gestión fragmentada de los activos de software, a su vez, genera inconsistencias en el despliegue, las políticas de uso y los esfuerzos de cumplimiento, exacerbando la ineficacia y los riesgos.[12]

En segundo lugar, se incrementan los **riesgos de cumplimiento y auditoría**.[12] Los complejos acuerdos de licencia exigen un cumplimiento estricto. Sin una comprensión clara del uso del software, las organizaciones pueden incurrir en incumplimientos, lo que se traduce en sanciones económicas durante las auditorías. La falta de visibilidad completa también implica la

pérdida de oportunidades de optimización.[12] Sin ella, las organizaciones no pueden identificar licencias redundantes, oportunidades de agrupación o mejores condiciones con los

proveedores, lo que impide un ahorro significativo de costos.

Además, la ausencia de visibilidad y trazabilidad obstaculiza la **toma de decisiones basada en datos**.[12] Sin un análisis detallado del uso y una supervisión en tiempo real, las organizaciones carecen de la información necesaria para tomar decisiones informadas sobre la asignación de recursos, la planificación estratégica y la preparación para el futuro.

Desde la perspectiva de la calidad, la trazabilidad es crucial para **demostrar la calidad del software**.[13] Permite proporcionar evidencia de que se han realizado pruebas exhaustivas y que se han cumplido los criterios de aceptación. Sin una trazabilidad adecuada, resulta desafiante auditar las pruebas diseñadas y ejecutadas, comprender los informes de prueba (por ejemplo, requisitos que pasaron, fallaron o están pendientes) y relacionar los aspectos técnicos de la prueba con los objetivos de negocio.[13] La gestión de la trazabilidad consistente de los requisitos se logra idealmente con una herramienta de gestión integral de testing que proporcione el soporte adecuado.[13]

La naturaleza opaca de los procesos de prueba sin una visibilidad y trazabilidad adecuadas transforma la función de QA de un garante de calidad a una **potencial fuente de responsabilidad**. Cuando el estado real de la calidad del software es desconocido, y el impacto de los esfuerzos de prueba no puede medirse o comunicarse con precisión, se impide la gestión proactiva de riesgos y se obstaculizan las iniciativas de mejora continúa, minando la confianza de las partes interesadas. Para una empresa de telecomunicaciones, esto se traduce en una incapacidad para justificar las inversiones en QA, dificultades para cumplir con los requisitos regulatorios y un enfoque reactivo ante los problemas de calidad, lo que impacta directamente en la fiabilidad del servicio y la confianza del cliente.

1.4 Necesidad de automatizar con integración en la nube

La evolución del desarrollo de software y las demandas del mercado han hecho que la automatización de pruebas, especialmente cuando se integra con la infraestructura en la nube, deje de ser una opción para convertirse en una necesidad imperativa. Esta combinación es esencial para superar las limitaciones del testing manual y abordar los desafíos de visibilidad y escalabilidad.

La automatización en la nube es fundamental para materializar la promesa y el valor del cloud computing.[14] La flexibilidad y el acceso a recursos bajo demanda que ofrece la nube se ven mermados si la provisión, configuración y desmantelamiento de entornos de prueba siguen

siendo tareas manuales. En flujos de trabajo modernos, con múltiples despliegues diarios, es laborioso y difícil enviar consistentemente código de alta calidad sin automatización. La automatización corrige este problema al permitir el aprovisionamiento, la configuración y la optimización automáticos de los recursos en la nube.[14]

Esta integración también ayuda a abordar puntos débiles comunes de las plataformas en la nube, como las facturas desorbitadas y los entornos complejos. Al proporcionar mayor control y visibilidad, facilita la eficiencia de los recursos y alivia la carga de tareas repetitivas de los equipos de TI, permitiéndoles enfocarse en actividades de mayor valor.[14]

La automatización es indispensable para las metodologías DevOps y los pipelines de Integración Continua/Entrega Continua (CI/CD).[6] En estos entornos dinámicos, donde el código nuevo se envía varias veces al día, la eficiencia se perdería si los entornos de TI tuvieran que aprovisionarse y configurarse manualmente para cada despliegue. La automatización permite la Infraestructura como Código (IaC) y la configuración automática de una infraestructura versionada y documentada. Las herramientas de automatización pueden monitorear, detectar problemas y realizar cambios en tiempo real, aumentando la estabilidad y escalabilidad de la infraestructura de TI.[14]

Los beneficios de la automatización de pruebas son numerosos y directos:

- **Eficiencia y ahorro de tiempo:** Las pruebas automatizadas son significativamente más rápidas que las manuales, reduciendo los tiempos de ejecución y liberando a los testers humanos para tareas más estratégicas.[5]
- **Consistencia y fiabilidad:** La automatización elimina el error humano, asegurando resultados de prueba consistentes y fiables en todas las iteraciones, lo que genera confianza en la calidad del software.[6]
- **Mejora del alcance y escalabilidad:** Las pruebas automatizadas pueden ejecutarse simultáneamente en múltiples plataformas (navegadores, sistemas operativos, dispositivos), mejorando drásticamente la cobertura y manejando miles de scripts de prueba con facilidad.[6]
- **Costo-eficiencia a largo plazo:** Aunque la configuración inicial puede implicar costos, la automatización reduce la necesidad de recursos humanos, los tiempos de prueba y el costo de corregir errores detectados tempranamente, lo que se traduce en ahorros significativos a largo plazo.[5]
- **Detección temprana de errores y adaptabilidad:** La automatización facilita la detección temprana de errores y es ideal para pruebas de regresión, asegurando que las nuevas modificaciones no afecten las funcionalidades existentes, lo cual es crucial en entornos de desarrollo dinámicos.[6]

La integración de la automatización con la nube no es simplemente una mejora técnica, sino un

imperativo estratégico que impulsa la transformación digital en las empresas de telecomunicaciones. Al automatizar las pruebas en la nube, organizaciones como WOM pueden lograr una agilidad sin precedentes en el despliegue de nuevos servicios, garantizando su calidad en una vasta gama de dispositivos y plataformas, y manteniendo una ventaja competitiva en un mercado altamente dinámico. Esto transforma el rol de QA de un "guardián" tradicional a un "acelerador de valor" dentro de todo el ciclo de vida de entrega de software.

1.5 Objetivo General

Implementar y validar un framework integral de aseguramiento de calidad automatizado para aplicaciones móviles y web dentro de WOM, utilizando Katalon Studio, Katalon TestOps y Katalon TestCloud, con el fin de mejorar significativamente la eficiencia de las pruebas, la cobertura y la calidad general del software.

1.6 Objetivos Específicos

- **1.6.1** Analizar los procesos de testing manual actuales en WOM y cuantificar sus limitaciones en términos de tiempo, costo, error humano y escalabilidad.
- **1.6.2** Establecer un marco conceptual robusto para la automatización de pruebas, detallando los principios, beneficios y las funcionalidades específicas de Katalon Studio, TestOps y TestCloud.
- **1.6.3** Proponer una arquitectura modular y escalable para el framework de automatización, incluyendo las mejores prácticas para la implementación de scripts, la gestión de objetos y el manejo de datos.
- **1.6.4** Implementar e integrar scripts de prueba automatizados para escenarios funcionales clave de las aplicaciones móviles y web de WOM, aprovechando Katalon TestCloud para la ejecución multi-entorno y Katalon TestOps para la orquestación.
- **1.6.5** Definir y recolectar métricas de calidad relevantes para validar cuantitativamente la efectividad de la solución automatizada, comparando los resultados con las líneas base pre-automatización cuando sea posible.
- **1.6.6** Analizar los resultados obtenidos, identificar las contribuciones clave del framework automatizado y proponer futuras mejoras, incluyendo la potencial integración de técnicas de Inteligencia Artificial/Machine Learning.

CAPÍTULO 2: MARCO CONCEPTUAL

Este capítulo establece la base teórica y técnica sobre la cual se construye la presente memoria de titulo. Se definirán los conceptos fundamentales de la automatización de pruebas, se describirán en detalle las herramientas de la suite Katalon (Katalon Studio, Katalon TestOps y Katalon TestCloud), y se abordará la importancia de los reportes y métricas de calidad en el contexto del software testing.

2.1 Automatización de Pruebas

La automatización de pruebas se define como el uso de herramientas y técnicas de software para automatizar los procesos de testing, con el objetivo de determinar y mejorar la calidad de un producto de software.[15] Este enfoque se basa en principios fundamentales que garantizan la robustez de la suite de pruebas, incluyendo la independencia de las pruebas, la idempotencia (la capacidad de una operación de producir el mismo resultado si se ejecuta varias veces) y la claridad.[16]

Los beneficios de la automatización de pruebas son extensos y bien documentados. Contribuye a una mayor eficiencia, reduciendo significativamente los costos y el tiempo asociados con el proceso de pruebas.[5] Además, mejora la cobertura de las pruebas y su fiabilidad, ya que los scripts automatizados ejecutan las validaciones de manera consistente y sin errores humanos.[8] Los ciclos de retroalimentación se aceleran, permitiendo a los desarrolladores identificar y corregir errores mucho más rápido.[8] La escalabilidad también se ve mejorada, ya que un framework de automatización bien diseñado puede ampliarse fácilmente para adaptarse a conjuntos de pruebas más grandes a medida que la aplicación crece en complejidad.[5] La detección temprana de defectos es otro beneficio crucial, ya que los problemas se identifican en etapas iniciales del ciclo de vida del desarrollo.[6] Finalmente, la automatización brinda un soporte invaluable para escenarios complejos, permitiendo la ejecución de un mismo script de prueba con distintas entradas de datos.[6]

Existen diversos tipos de pruebas que pueden ser automatizadas, abarcando desde la validación de funcionalidades básicas hasta la evaluación de aspectos no funcionales. Entre ellos se incluyen las pruebas funcionales, pruebas no funcionales (como las de rendimiento y carga),

pruebas unitarias, pruebas de integración, pruebas de API, pruebas de interfaz de usuario (UI), pruebas de regresión, *smoke tests* y *sanity tests*.[9]

Para que un proceso de automatización sea exitoso, es fundamental considerar varios factores críticos. Estos incluyen el Sistema Bajo Prueba (SUT), los tipos y el número de pruebas a automatizar, la selección de la herramienta de prueba adecuada, aspectos humanos y organizacionales, y factores transversales.[11] Es importante señalar que, en el testing automatizado, el ingeniero de pruebas o el especialista en aseguramiento de calidad de software debe poseer habilidades de codificación, ya que los casos de prueba se escriben en forma de código fuente.[11]

El valor de la automatización de pruebas trasciende la mera velocidad de ejecución; transforma el aseguramiento de calidad en una actividad continua e integrada dentro del ciclo de vida del desarrollo. Este cambio permite una **retroalimentación de calidad constante**, lo que significa que los defectos se identifican y corrigen mucho antes en el proceso, reduciendo el costo asociado a los errores y acelerando la entrega general del software. Para una empresa de telecomunicaciones, esto se traduce en un despliegue más rápido y fiable de nuevos servicios y características, lo que impacta directamente en la satisfacción del cliente y la ventaja competitiva. El objetivo es incorporar la calidad *desde el inicio*, en lugar de solo probarla *al final*.

2.2 Katalon Studio

Katalon Studio es una herramienta de automatización de pruebas multiplataforma, lanzada en 2015, que se ha consolidado como una plataforma todo en uno diseñada para resolver los desafíos de la automatización de pruebas.[19] Su filosofía central radica en tender un puente entre el testing basado en scripts, dirigido a usuarios técnicos, y el testing sin código, accesible para principiantes, ofreciendo soporte para aplicaciones web, API, móviles y de escritorio en un único entorno.[19]

Entre sus características y funcionalidades clave se destacan:

- **Creación de Pruebas:** Katalon Studio permite la creación de pruebas de diversas maneras. Ofrece una funcionalidad de grabación y reproducción (*record-and-playback*) que facilita la generación rápida de pruebas simulando interacciones de usuario en navegadores o aplicaciones.[19] Para usuarios más avanzados, proporciona un modo de script que permite la codificación directa en Groovy o Java, con acceso completo a las funciones de Selenium WebDriver y una función de "Content Assist" para autocompletado de código.[19]
- **Gestión del Repositorio de Objetos:** La herramienta centraliza la gestión de elementos de

la interfaz de usuario (botones, campos, etc.) en un repositorio de objetos. Esto facilita el mantenimiento y la reutilización de los elementos, reduciendo la redundancia en los scripts de prueba. Katalon identifica de forma unívoca los elementos, aunque requiere interacción para seleccionar el "Locator" adecuado.[19]

- **Organización de Pruebas:** Permite organizar las pruebas en casos de prueba, *test suites* y *test suite collections*, lo que facilita una ejecución estructurada y una gestión eficiente de los activos de prueba.[19]
- **Palabras Clave Integradas y Personalizadas:** Katalon Studio se lanza con conjuntos predefinidos de palabras clave o acciones de uso común, lo que permite a los usuarios implementar la mayoría de los casos de prueba rápidamente. Para necesidades más avanzadas, ofrece la posibilidad de crear palabras clave personalizadas, que son extensiones de las integradas y pueden compartirse entre usuarios, promoviendo la reutilización de código.[22]
- **Testing Orientado a Datos (DDT):** Soporta la consulta de datos desde fuentes externas como archivos CSV, Excel y bases de datos relacionales. Esto permite ejecutar el mismo script de prueba con diferentes entradas, lo que es especialmente útil para probar funciones como el inicio de sesión con múltiples cuentas.[22]
- **Pruebas de API y Servicios Web:** Ofrece compatibilidad completa con la automatización de pruebas de APIs REST y SOAP. Los usuarios pueden diseñar y ejecutar pruebas de *endpoints*, validar respuestas JSON o XML, gestionar tokens de autenticación y encadenar peticiones para simular escenarios de negocio reales.[20]
- **Pruebas de Aplicaciones Móviles:** Proporciona soporte integral para la automatización de pruebas en aplicaciones móviles nativas e híbridas, tanto en dispositivos Android como iOS. Permite grabar interacciones directamente desde un dispositivo o emulador y ejecutar pruebas en múltiples dispositivos en paralelo, lo que es clave para garantizar una experiencia de usuario consistente.[20]
- **Testing Visual:** Incluye funcionalidades para realizar testing visual automatizado, detectando cambios en la interfaz de usuario mediante comparaciones basadas en píxeles o en el diseño (*layout*), con umbrales de tolerancia ajustables. Esto acelera la verificación de la interfaz y la detección de cualquier diferencia.[26]
- **Integraciones:** Katalon Studio se integra de forma fluida con diversas tecnologías y herramientas externas, como Jira, Azure DevOps, Jenkins, GitLab, Git, Slack y TestRail, facilitando la gestión de incidencias, el control de versiones y la integración continua.[19]

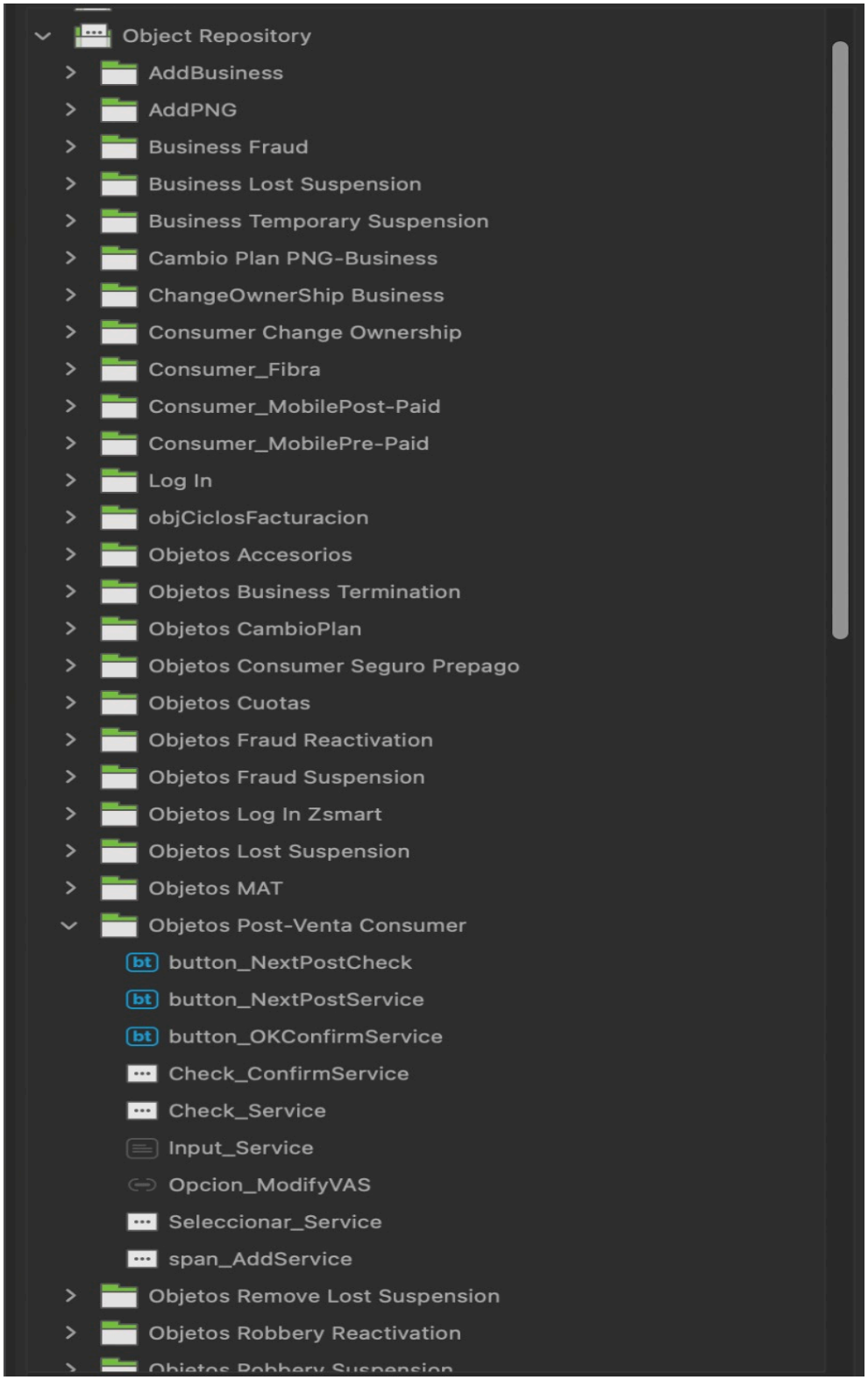

Figura 1:Gestión Repositorio Objetos.

Fuente: Elaboración Propia

La capacidad de Katalon Studio para ser una plataforma "todo en uno" y su enfoque híbrido que atiende tanto a usuarios técnicos como no técnicos [19] es un elemento clave. Esta característica permite a organizaciones como WOM incorporar y empoderar a una fuerza laboral de QA diversa, reduciendo el

cuello de botella del factor humano en la adopción de la automatización. Esto significa que los esfuerzos iniciales de automatización pueden ser rápidamente escalados por usuarios con menos experiencia técnica, mientras que los ingenieros experimentados pueden construir suites de prueba complejas y robustas, fomentando un ecosistema de QA más colaborativo y productivo.

2.3 Katalon TestOps y TestCloud

Katalon TestOps y Katalon TestCloud son componentes complementarios de la plataforma Katalon que extienden las capacidades de Katalon Studio, proporcionando orquestación, gestión y ejecución escalable de pruebas en la nube.

Katalon TestOps: El Centro de Orquestación

Katalon TestOps es una plataforma de orquestación de pruebas abierta y completa, diseñada para optimizar los pipelines de DevOps. Actúa como un centro de comando centralizado, conectando todos los datos de prueba y a los miembros del equipo en un ciclo de retroalimentación estrecho.[26]

Sus funcionalidades clave incluyen:

- **Planificación y Organización de Pruebas:** Permite planificar, organizar y programar ejecuciones de pruebas, crear *test suites* y agrupar pruebas para objetivos específicos. Ofrece la posibilidad de agendar nuevas corridas de pruebas, especificando fechas, horas y las *test suites* a ejecutar.[26]
- **Orquestación de Ejecuciones:** Soporta la ejecución remota en diversos entornos, incluyendo agentes locales, Docker, Kubernetes y CircleCI. Proporciona planificadores de pruebas inteligentes y mecanismos de planificación para optimizar las ejecuciones.[27] También ofrece soluciones

serverless, balanceo de carga automático y ejecución paralela para un ciclo de entrega más eficiente.[29]

- **Colaboración y Eficiencia:** Mejora la colaboración entre los equipos de desarrollo y QA, maximiza el retorno de la inversión (ROI) y la utilización de recursos, y reduce los pasos manuales en la ejecución de pruebas de software.[29]
- **Gestión DevOps Orientada a Datos:** Brinda información valiosa e inmediata a partir de los resultados de las pruebas, redefiniendo cómo se utilizan los análisis para garantizar la calidad del producto. Permite a los equipos gestionar lanzamientos sin seguimientos continuos y repetitivos, y descubrir y abordar rápidamente los problemas encontrados en cualquier punto del pipeline.[29]

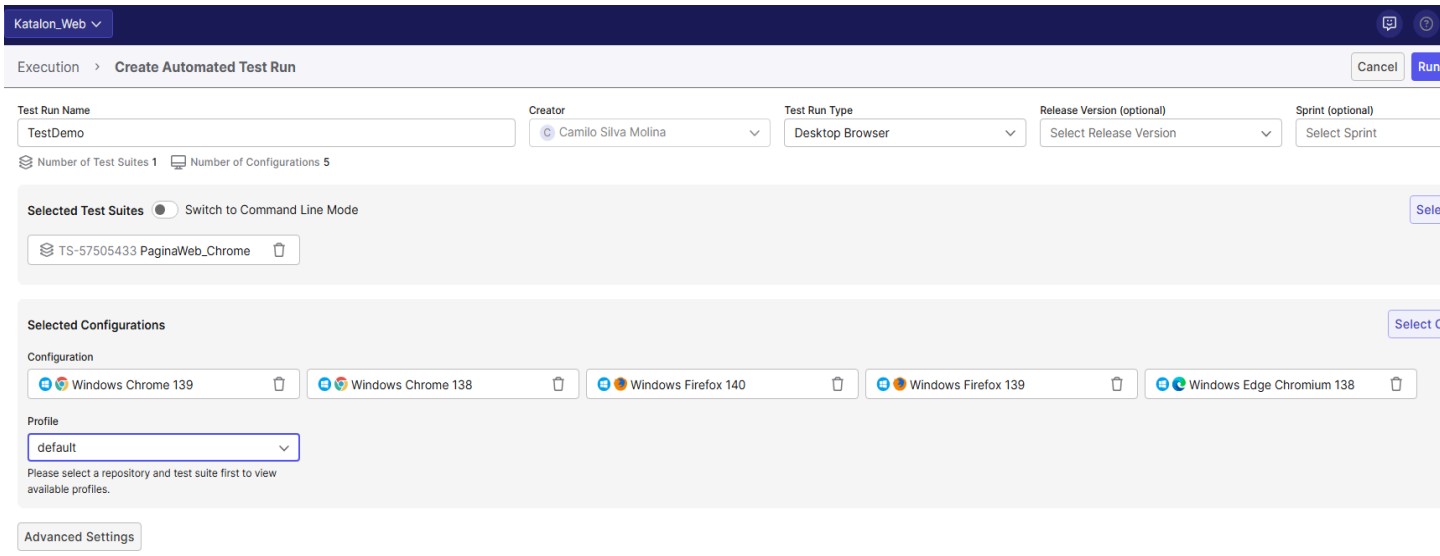

Figura 2:Ejecución múltiples navegadores TestOps/TestCloud.

Fuente: Elaboración Propia

Katalon TestCloud: Ejecución Basada en la Nube

Katalon TestCloud es una solución que facilita la ejecución de pruebas en la nube, permitiendo el testeo multidispositivo, múltiples sistemas operativo y múltiples navegadores sin la necesidad de mantener una infraestructura local extensa.[32]

Sus características principales abarcan:

- **Configuración de Entornos de Ejecución:** Permite configurar fácilmente entornos de prueba para navegadores de escritorio (seleccionando el sistema operativo, navegador y versión), navegadores móviles (seleccionando el sistema operativo, versión y dispositivo, con Chrome para Android y Safari para iOS por defecto) y aplicaciones móviles nativas (seleccionando el sistema operativo, versión y dispositivo).[33]
- **Integración con Studio y Runtime Engine:** Se integra sin problemas con Katalon Studio para la configuración y ejecución de pruebas, y con Katalon Runtime Engine (KRE) para su uso en pipelines de CI/CD.[33]
- **Monitoreo en Vivo:** Permite visualizar el progreso de la ejecución y los metadatos en tiempo real.[32]
- **Consideraciones de Rendimiento:** Es importante reconocer que la ejecución en TestCloud puede ser más lenta que la ejecución local debido a la latencia de internet y al protocolo Remote WebDriver, especialmente para pruebas con muchos pasos individuales.[36]

La combinación de Katalon Studio (para la creación de pruebas), TestOps (para la orquestación y el análisis) y TestCloud (para los entornos de ejecución escalables) conforma un **ecosistema integrado y potente**. Esta sinergia es crucial para alcanzar la **madurez en la automatización de pruebas a escala empresarial**. Permite a una empresa como WOM ir más allá de los scripts de automatización aislados para adoptar una estrategia de testing holística, gestionada y escalable, capaz de seguir el ritmo de la entrega rápida de software, garantizar la calidad en una vasta base de usuarios y proporcionar información procesable para la mejora continua, a pesar de las posibles consideraciones de rendimiento para ejecuciones específicas en la nube.

2.4 Reportes y Métricas de Calidad

Los reportes y métricas de calidad son herramientas indispensables en el aseguramiento de calidad de software. Proporcionan una visión objetiva y cuantificable del estado del producto y del proceso de testing, lo que es fundamental para la toma de decisiones informadas y la mejora continua.

La importancia de las métricas de pruebas radica en varios aspectos clave:

- **Evaluación Imparcial:** Ofrecen una evaluación objetiva de las herramientas y sistemas de pruebas, permitiendo a desarrolladores y QA determinar si cumplen con los estándares requeridos o si necesitan modificaciones para aumentar la eficacia.[37]
- **Mejor Toma de Decisiones:** Guían las elecciones estratégicas basándose en datos concretos, en lugar de la intuición, lo que es crucial para priorizar tareas y ajustar estrategias.[37]
- **Visibilidad Mejorada:** Aumentan la comprensión del progreso de las pruebas e identifican las áreas más propensas a errores, lo que facilita la conexión entre las acciones de los usuarios y los fallos de la aplicación.[37]
- **Detección Temprana de Cuellos de Botella:** Ayudan a identificar ineficiencias en el proceso en una fase temprana del ciclo de pruebas.[37]
- **Mejor Mitigación de Riesgos:** Permiten identificar patrones comunes y definir áreas de alto riesgo del producto de software, posibilitando una asignación inteligente de recursos y un énfasis en los aspectos de alto impacto.[37]
- **Mayor Calidad del Producto y Satisfacción del Cliente:** Contribuyen a mantener la aplicación estable, lanzar actualizaciones regularmente y, en última instancia, ofrecer una experiencia de usuario sin problemas.[37]

Las métricas de pruebas suelen clasificarse en cuatro categorías principales [37]:

- **Métricas de Proceso:** Relacionadas con la mejora de la eficacia del proceso de pruebas, como la tasa de defectos abiertos vs. cerrados o el *lead time* (tiempo desde la solicitud de una funcionalidad hasta su entrega).[37]
- **Métricas del Producto:** Miden la calidad del producto, incluyendo métricas de pruebas de rendimiento, el número total de errores graves o la densidad de defectos.[37]
- **Métricas del Proyecto:** Evalúan el rendimiento del equipo de QA en su conjunto, como la tasa de finalización de casos de prueba, la cobertura de automatización de pruebas y la tasa de resolución de defectos.[37]
- **Métricas de Personas:** Evalúan las habilidades y la productividad de los testers individuales y del equipo, como los problemas por reportero o la tasa de descubrimiento de defectos.[37]

Entre las métricas de pruebas de software más importantes se encuentran:

- **Cobertura de las Pruebas:** Permite estimar la exhaustividad del proceso de pruebas.[37]
- **Cobertura de Automatización de Pruebas:** Porcentaje de casos de prueba automatizados.[37]
- **Densidad de Defectos:** Número de defectos encontrados por módulo o componente, útil para identificar áreas que acumulan errores.[37]
- **Tiempo Medio de Detección (MTTD):** Mide el tiempo transcurrido entre el inicio de un incidente y el momento en que se detecta, con el objetivo de mantener valores bajos.[39]
- **Tiempo Medio de Reparación (MTTR):** Se calcula dividiendo el tiempo total de mantenimiento no planificado invertido en un activo por el número total de fallas experimentadas.[39]
- **Fiabilidad de las Pruebas:** Se refiere a la cantidad de casos de prueba que no brindan comentarios útiles debido a su falta de fiabilidad.[39]
- **Porcentaje de Casos de Prueba Superados:** La proporción de pruebas superadas respecto al total de pruebas ejecutadas.[37]

Katalon TestOps ofrece capacidades robustas de reporte y análisis:

- **Dashboards Completos:** Proporciona dashboards visualmente atractivos con historial de ejecución de pruebas, tendencias, análisis de fallos, detección de pruebas "flaky" (intermitentes), métricas de cobertura de pruebas y paneles de rendimiento del equipo.[19]
- **Opciones de Exportación:** Permite exportar informes en varios formatos, como HTML, PDF, Excel y CSV.[19]
- **Notificaciones Automatizadas:** Configuración para el envío automático de informes de pruebas por correo electrónico al finalizar las ejecuciones, asegurando una comunicación oportuna a los stakeholders relevantes.[41]
- **Integración con Katalon Studio:** Una vez habilitada la integración, todas las ejecuciones de pruebas desde Katalon Studio se cargan automáticamente a TestOps, contribuyendo a la plataforma centralizada de análisis.[41]

El uso estratégico de métricas de calidad y herramientas de reporte robustas como Katalon TestOps transforma la función de QA de una función técnica reactiva a un **socio estratégico proactivo y basado en datos**. Esto permite a las empresas de telecomunicaciones no solo comprender el estado actual de la calidad del software, sino también identificar tendencias, predecir problemas futuros y demostrar el retorno de inversión tangible de sus inversiones en QA a los stakeholders del negocio. Las métricas se convierten en el "lenguaje" a través del cual QA comunica su valor, fomentando una cultura de mejora continua y una toma de decisiones informada en toda la organización.

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

Este capítulo detalla la propuesta de solución para la automatización de pruebas en WOM, centrándose en la arquitectura del framework, la metodología de implementación de scripts desarrollados y la integración de estos con entornos en la nube y la generación de reportes automáticos.

3.1 Arquitectura del Framework

El diseño de la arquitectura del framework de automatización es un pilar fundamental para garantizar su escalabilidad, mantenibilidad y robustez a largo plazo. Se ha adherido a principios de diseño clave, como el principio "Keep It Simple, Stupid" (KISS) y un enfoque modular.[43] Esto implica descomponer las pruebas complejas en módulos más pequeños y reutilizables, con componentes débilmente acoplados, lo que facilita la comprensión, el mantenimiento y la reutilización de fragmentos de código.[16]

La **gestión del repositorio de objetos** es un aspecto crítico. Katalon Studio centraliza los elementos de la interfaz de usuario (UI) en un repositorio de objetos, lo que simplifica su mantenimiento cuando la aplicación cambia.[19] Se ha implementado una estrategia de nombres significativos para los objetos, incluyendo la ubicación del elemento en la página, su tipo y un nombre descriptivo, lo que mejora la legibilidad y la organización.[25] Además, se han considerado técnicas para manejar estructuras DOM dinámicas, ajustando los métodos de selección predeterminados en la utilidad de grabación para asegurar una identificación robusta de los objetos.[25]

La **estrategia de gestión de datos de prueba** es otro componente vital. Se ha priorizado el uso de fuentes de datos externas como archivos CSV, Excel o bases de datos relacionales para alimentar las pruebas.[22] Una práctica recomendada crucial es manejar la configuración de los datos de prueba y las precondiciones a través de interacciones con la API o la base de datos, en lugar de hacerlo directamente a través de la interfaz de usuario. Este enfoque mejora la velocidad y la estabilidad de las pruebas, ya que las pruebas de UI tienden a ser más frágiles y lentas para la configuración de datos.[43] Se ha evitado el uso de Excel para la gestión de datos de prueba debido a sus limitaciones en el control de versiones, la colaboración, el rendimiento y la integridad de los datos.[43]

La creación y organización de **componentes reutilizables**, como las palabras clave personalizadas, es esencial. Estas encapsulan funcionalidades de prueba comunes y reutilizables, clasificándolas (por ejemplo, funciones genéricas aplicables a todos los proyectos o funciones específicas de la aplicación).[22] Esto reduce la redundancia en los scripts de prueba y simplifica los esfuerzos de mantenimiento.[8]

La **integración con un sistema de control de versiones** (como Git) es fundamental para gestionar los scripts de prueba, permitir la colaboración entre los miembros del equipo y mantener un historial de cambios.[19] Esto facilita las reversiones y el desarrollo paralelo en las pruebas.[8]

El framework incorpora principios de un **Framework de Automatización Híbrido**, estructurado en torno a la independencia, idempotencia y claridad de las pruebas.[16] Su arquitectura presenta un diseño multicapa que separa los componentes de la UI del navegador de los scripts de prueba, e integra metodologías híbridas como el Scripting Lineal, el Testing Basado en Módulos, el Testing Orientado a Palabras Clave y el Testing Orientado a Datos.[16]

Las decisiones arquitectónicas para el framework de automatización no son meras preferencias técnicas, sino **determinantes críticos del éxito y la sostenibilidad a largo plazo** de la iniciativa de automatización. Al adherirse a principios como la modularidad, una gestión robusta de objetos y un manejo estratégico de datos, el framework asegura que la solución de automatización permanezca adaptable, escalable y fácil de mantener, incluso a medida que las aplicaciones subyacentes evolucionan. Este enfoque proactivo en el diseño aborda directamente el desafío del "mantenimiento de escenarios de prueba complejos" [20], salvaguardando la inversión inicial y garantizando que la automatización siga siendo un activo, no una carga, para WOM.

3.2 Implementación de Scripts

La implementación de scripts de prueba en Katalon Studio aprovecha la flexibilidad de la herramienta, que permite la creación de casos de prueba tanto a través de una interfaz gráfica como mediante codificación directa.

El **proceso de creación de casos de prueba** se puede iniciar en la **vista Manual**, donde los pasos de prueba se definen de manera intuitiva a través de una interfaz amigable. Estos pasos se traducen automáticamente a un script en Groovy.[23] Para escenarios más complejos o para usuarios con habilidades de programación, la

vista Script permite la codificación directa en Groovy. Aquí, se utilizan las palabras clave WebUI integradas de Katalon (como WebUI.openBrowser, WebUI.navigateToUrl, WebUI.click, WebUI.verifyElementPresent, WebUI.closeBrowser) y la función "Content Assist" que proporciona sugerencias de código contextuales, agilizando el proceso de escritura.[23]

La **interacción con los elementos de la interfaz de usuario** se realiza refiriéndose a los objetos almacenados en el Repositorio de Objetos. Esto se puede lograr escribiendo la sintaxis findTestObject('{Object ID}') o simplemente arrastrando y soltando el objeto desde el repositorio al editor de casos de prueba.[23]

Para construir flujos de prueba complejos y dinámicos, se implementan diversas **estructuras de control de flujo** utilizando sentencias de programación:

- **Lógica Condicional:** Se emplean sentencias if-else y switch para gestionar diferentes condiciones y adaptar las pruebas a cambios esperados en el comportamiento de la aplicación sin necesidad de duplicar scripts.[44]
- **Sentencias de Bucle:** Los bucles for y while se utilizan para automatizar acciones repetitivas y para iterar eficientemente sobre conjuntos de datos, como probar una funcionalidad con múltiples entradas.[44]
- **Sentencias de Ramificación:** Las sentencias break y continue se aplican para reducir acciones innecesarias y mejorar la eficiencia de los scripts de prueba, controlando el flujo de ejecución dentro de los bucles.[44]

La **gestión de errores y la robustez** de los scripts se logran mediante el uso de sentencias de manejo de excepciones. Las sentencias try-catch previenen fallos inesperados al capturar y manejar errores, mientras que la sentencia finally asegura que las acciones de limpieza esenciales se realicen independientemente de si ocurrieron errores. La sentencia throw permite activar excepciones personalizadas para un control de errores más preciso, haciendo que los

scripts sean más resilientes.[44]

Finalmente, se aprovecha la capacidad de **palabras clave personalizadas** para encapsular funcionalidades reutilizables. Esto asegura la consistencia y reduce la duplicación de código en los scripts, contribuyendo a un framework más mantenible.[22]

La existencia de ambos modos de implementación de scripts en Katalon Studio es crucial para maximizar la eficiencia y la robustez de la solución de automatización. Si bien la grabación y reproducción, junto con el modo manual, aceleran la creación inicial de pruebas, la capacidad de trabajar directamente en la vista de script con Groovy/Java e implementar construcciones de programación avanzadas (condicionales, bucles, manejo de errores) garantiza que las pruebas puedan manejar escenarios complejos y sean resilientes a problemas inesperados. Esta flexibilidad permite a WOM optimizar el desarrollo de pruebas en función de la complejidad del escenario y las habilidades del equipo, lo que conduce a **activos de automatización más fiables y mantenibles** a lo largo del tiempo.

3.3 Desarrollo de scripts de automatización y orquestación

En esta memoria de titulo se abordó tanto la integración de nuevas tecnologías como el desarrollo de scripts de automatización. Inicialmente se puso principal énfasis en la automatización web de la plataforma interna de “WOM” llamada “ZSmart”, la cual funciona como fuente principal de datos. Para ello se desarrollaron numerosos scripts automatizados empleando Katalon Studio como IDE (entorno de desarrollo integrado). Estos scripts están escritos en Groovy/Java y utilizan las capacidades de Selenium/WebDriver, junto con otras bibliotecas, para interactuar con la aplicación web de ZSmart. El desarrollo incluyó la creación de funciones y rutinas personalizadas orientadas a reproducir las interacciones de un usuario en la interfaz, cubriendo los pasos críticos de cada flujo de negocio.

Se automatizan varios flujos críticos del sistema ZSmart, entre ellos:

- **Creación de usuarios:** Script que ingresa los datos de un nuevo usuario y verifica que la creación sea exitosa, asegurando que se asignen correctamente los roles y permisos correspondientes.

- **Creación de cuentas de facturación:** Automatización del proceso de registro de una nueva cuenta de facturación en el sistema, validando que los datos financieros queden guardados correctamente en la plataforma.

- **Flujos de venta:** Simulación de transacciones de venta, desde la selección de productos/servicios hasta la generación de la orden, garantizando que las reglas

comerciales se apliquen correctamente en cada transacción.

- **Flujos de postventa:** Scripts para procesos posteriores a la venta (como cambios de plan, cancelaciones o soporte al cliente), comprobando que el sistema responda adecuadamente en cada caso y que las operaciones de postventa se registren correctamente.

Estos scripts de prueba no solo interactúan con la interfaz web, sino que también comprueban que la lógica de negocio y la integridad de los datos no se vean comprometidas tras cada acción automatizada. En cada caso se implementaron mecanismos de verificación (aserciones) para validar que el resultado obtenido en la plataforma coincide con el esperado; por ejemplo, corroborando que un usuario efectivamente quede creado con los atributos correctos, o que una venta genere las actualizaciones correspondientes en el sistema.

La estructura de las pruebas automatizadas se diseñó de forma modular, agrupando casos de prueba individuales por funcionalidad dentro de *suites* de prueba más amplias para lograr flujos completos de negocio. Esto permitió reutilizar y encadenar distintos casos de prueba según se necesitara. Por ejemplo, se definió un caso de prueba para la “Creación de Cliente Nuevo” y otro para la “Creación de Cuenta Móvil Postpago”; al ejecutarlos juntos en una suite de pruebas, forman un flujo integral que cubre desde el alta de un nuevo cliente hasta la activación de una línea postpago. Gracias a este enfoque modular, cada caso de prueba también puede ejecutarse por sí solo si se requiere (por ejemplo, solamente la creación de un cliente sin incluir la creación de su cuenta de facturación).

Además, se incorporó el uso de archivos de datos externos (como archivos CSV o Excel) para la ejecución de pruebas con múltiples iteraciones, siguiendo un enfoque *data-driven*. Esto permitió alimentar los scripts con diversos conjuntos de datos y crear múltiples registros en una sola ejecución. Por ejemplo, un archivo CSV con una lista de información de clientes nuevos puede ser utilizado por la suite de pruebas para iterar automáticamente y crear cada cliente definido en el archivo, generando tantos registros como filas tenga el CSV. Esta estrategia incrementó la cobertura y eficiencia de las pruebas, ya que facilitó la ejecución masiva de escenarios variando únicamente los datos de entrada, sin requerir cambios en el código de los scripts.

Como parte del desarrollo de los scripts, se incluyeron soluciones técnicas para automatizar interacciones complejas con la aplicación. Por ejemplo, a continuación se muestra un fragmento

de código utilizado para automatizar el inicio de sesión en ZSmart haciendo uso de la clase Robot de Java. Esta técnica permitió simular pulsaciones de teclas en situaciones donde la automatización tradicional de elementos web resulta difícil, como en ventanas emergentes de autenticación o controles no estándar:

```
import java.awt.Robot;

import java.awt.event.KeyEvent;


Robot robot = new Robot();

robot.delay(2000); // Espera para que aparezca la ventana
emergente


// Función para escribir texto y mostrar en consola lo que se
está escribiendo

void escribirTexto(Robot robot, String texto) {

    String resultado = ""; // Almacenar lo que se escribe

    for (char c : texto.toCharArray()) {

        int keyCode = KeyEvent.getExtendedKeyCodeForChar((int)
c);


        if (Character.isUpperCase(c)) { // Si es mayúscula, usar
Shift

            robot.keyPress(KeyEvent.VK_SHIFT);

            robot.keyPress(keyCode);

            robot.keyRelease(keyCode);

            robot.keyRelease(KeyEvent.VK_SHIFT);

        } else if (Character.isLowerCase(c) ||
```

```
Character.isDigit(c)) { // Minúsculas y números
            robot.keyPress(keyCode);
            robot.keyRelease(keyCode);
        } else if (c == '-') { // Si es el guión "-"
            robot.keyPress(KeyEvent.VK_SHIFT);
            robot.keyPress(KeyEvent.VK_MINUS);
            robot.keyRelease(KeyEvent.VK_MINUS);
            robot.keyRelease(KeyEvent.VK_SHIFT);
        } else {
            println("⚠️ No se pudo escribir el carácter: " + c);
        }

        resultado += c; // Agregar el carácter a la salida
        robot.delay(200); // Pequeña pausa entre teclas
    }

    println("✅ Se escribió: " + resultado); // Mostrar lo que
se escribió en consola
}

// 1️⃣ Escribir usuario "miwom"
escribirTexto(robot, "miwom");
```

```
robot.delay(500); // Pequeña espera antes de cambiar de campo

robot.keyPress(KeyEvent.VK_TAB); // Cambiar al campo de
contraseña

robot.keyRelease(KeyEvent.VK_TAB);


robot.delay(500); // Pequeña espera antes de escribir la
contraseña


// 2️⃣ Escribir contraseña "M1W0M-dev"

escribirTexto(robot, "M1W0M-dev");


robot.delay(500); // Pequeña espera antes de confirmar

robot.keyPress(KeyEvent.VK_ENTER); // Presionar Enter para
iniciar sesión

robot.keyRelease(KeyEvent.VK_ENTER);


println("✅ Se presionó ENTER para enviar el formulario.");
```

En el código anterior, la función `escribirTexto` recibe un objeto `Robot` y una cadena de texto, simulando la pulsación en secuencia de teclas para escribir ese texto en la interfaz. Se agrega una pequeña demora (`delay`) entre cada pulsación y se imprime en la consola cada carácter escrito. La función contempla casos especiales como letras mayúsculas (manteniendo presionada la tecla Shift), minúsculas, dígitos e incluso el carácter especial "-" (guión). Tras definir esta función, el script la utiliza para escribir automáticamente el nombre de usuario “miwom” y la contraseña “M1W0M-dev”, navegando entre los campos con la tecla Tab y finalmente simulando la pulsación de Enter para enviar el formulario de inicio de sesión. Este fragmento demuestra cómo se desarrolló código personalizado para sortear obstáculos en la automatización web (por ejemplo, interactuar con ventanas emergentes de autenticación) y

asegurar que el flujo de login pudiera ejecutarse sin intervención manual.

```groovy
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
import static com.kms.katalon.core.testdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import static com.kms.katalon.core.testobject.ObjectRepository.findWindowsObject
import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
import com.kms.katalon.core.cucumber.keyword.CucumberBuiltinKeywords as CucumberKW
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.core.testdata.TestData as TestData
import com.kms.katalon.core.testng.keyword.TestNGBuiltinKeywords as TestNGKW
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import com.kms.katalon.core.windows.keyword.WindowsBuiltinKeywords as Windows
import internal.GlobalVariable as GlobalVariable
import org.openqa.selenium.Keys as Keys

WebUI.openBrowser('')

WebUI.navigateToUrl('URL')

WebUI.maximizeWindow()

WebUI.waitForPageLoad(5)

WebUI.setText(findTestObject('Object Repository/Objetos Log In Zsmart/Page_User Index/input_espaol_inputUserName'), GlobalVariable.Username)

WebUI.setText(findTestObject('Object Repository/Objetos Log In Zsmart/Page_User Index/input_UserName_inputPasswd'), GlobalVariable.Pass)

WebUI.click(findTestObject('Object Repository/Objetos Log In Zsmart/Page_User Index/button_Sign in'))

WebUI.delay(GlobalVariable.Time)

WebUI.click(findTestObject('Object Repository/Objetos Log In Zsmart/Page_User Index/td_Televentas'))

WebUI.click(findTestObject('Object Repository/Objetos Log In Zsmart/Page_User Index/button_OK'))

WebUI.delay(2)

WebUI.delay(GlobalVariable.Time)
```

Figura 3:Script LogIn Zsmart.

Fuente: Elaboración Propia

En paralelo al desarrollo de los scripts, se llevó a cabo la integración de varias herramientas para lograr la orquestación y gestión eficiente de las pruebas automatizadas. En este proyecto se integraron los siguientes componentes tecnológicos:

- **Katalon Studio:** IDE donde se desarrollaron y estructuraron los scripts de automatización. Desde Katalon Studio se organizaron los casos de prueba y *suites* de prueba, aprovechando sus bibliotecas integradas para interacciones web, generación de reportes, etc.

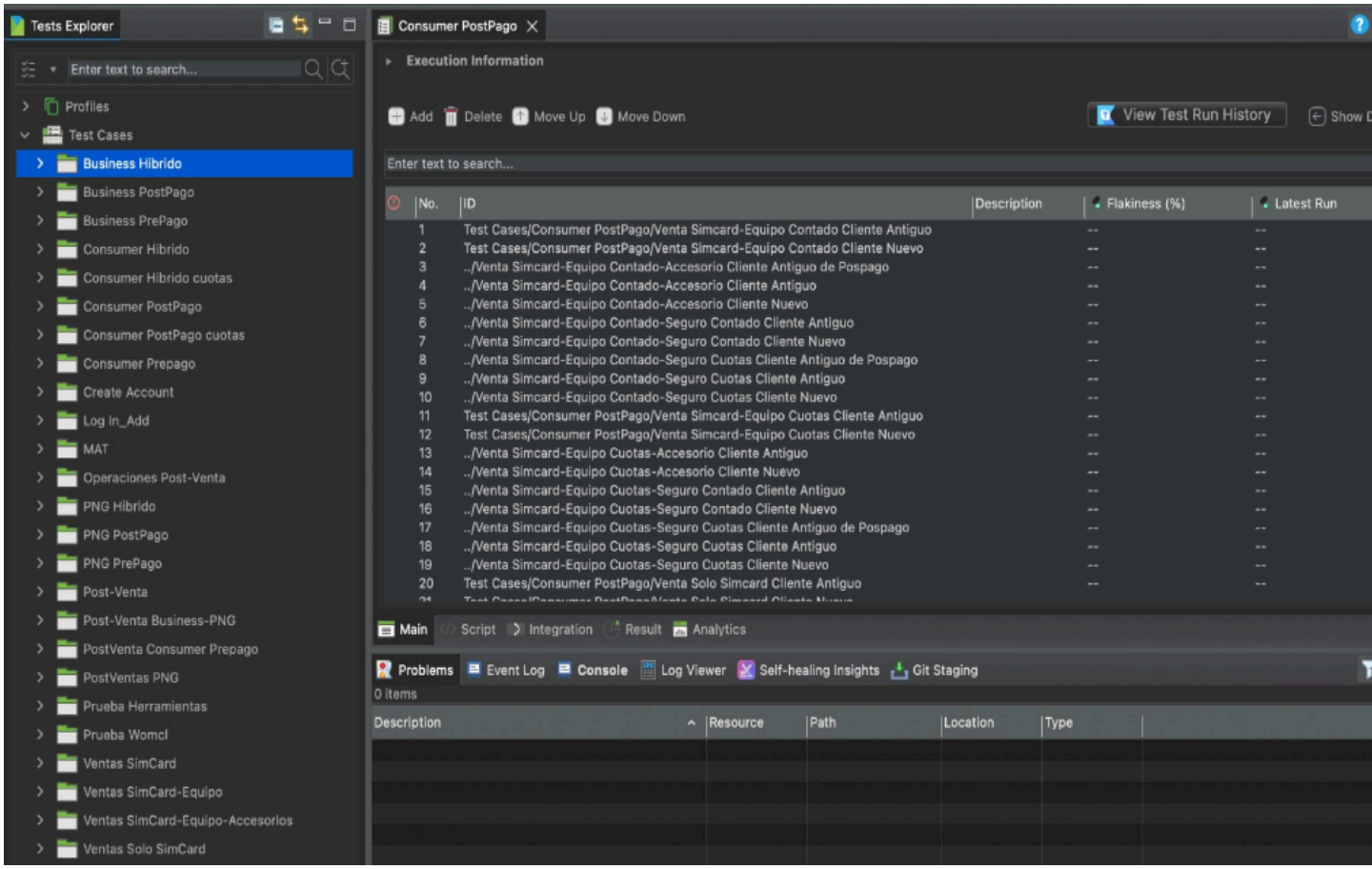

Figura 4:Estructura Testcase y Testsuite.

Fuente: Elaboración Propia

- **Katalon TestOps:** Plataforma de orquestación y análisis de pruebas de Katalon. Se utilizó como orquestador de ejecuciones, permitiendo programar la ejecución de las suites de pruebas automatizadas, recopilar los resultados en un solo lugar y obtener análisis/reportes centralizados sobre cada ejecución (tiempos, tasas de éxito, historiales, entre otros).

- **GitLab:** Repositorio de código donde se versionaron los scripts de prueba. El proyecto de automatización se mantuvo bajo control de versiones en GitLab, facilitando la colaboración entre los miembros del equipo y permitiendo integrar las pruebas en la tubería de CI/CD. Cada actualización de los scripts podía desencadenar ejecuciones automáticas de pruebas, manteniendo así un ciclo de retroalimentación continuo durante el desarrollo.

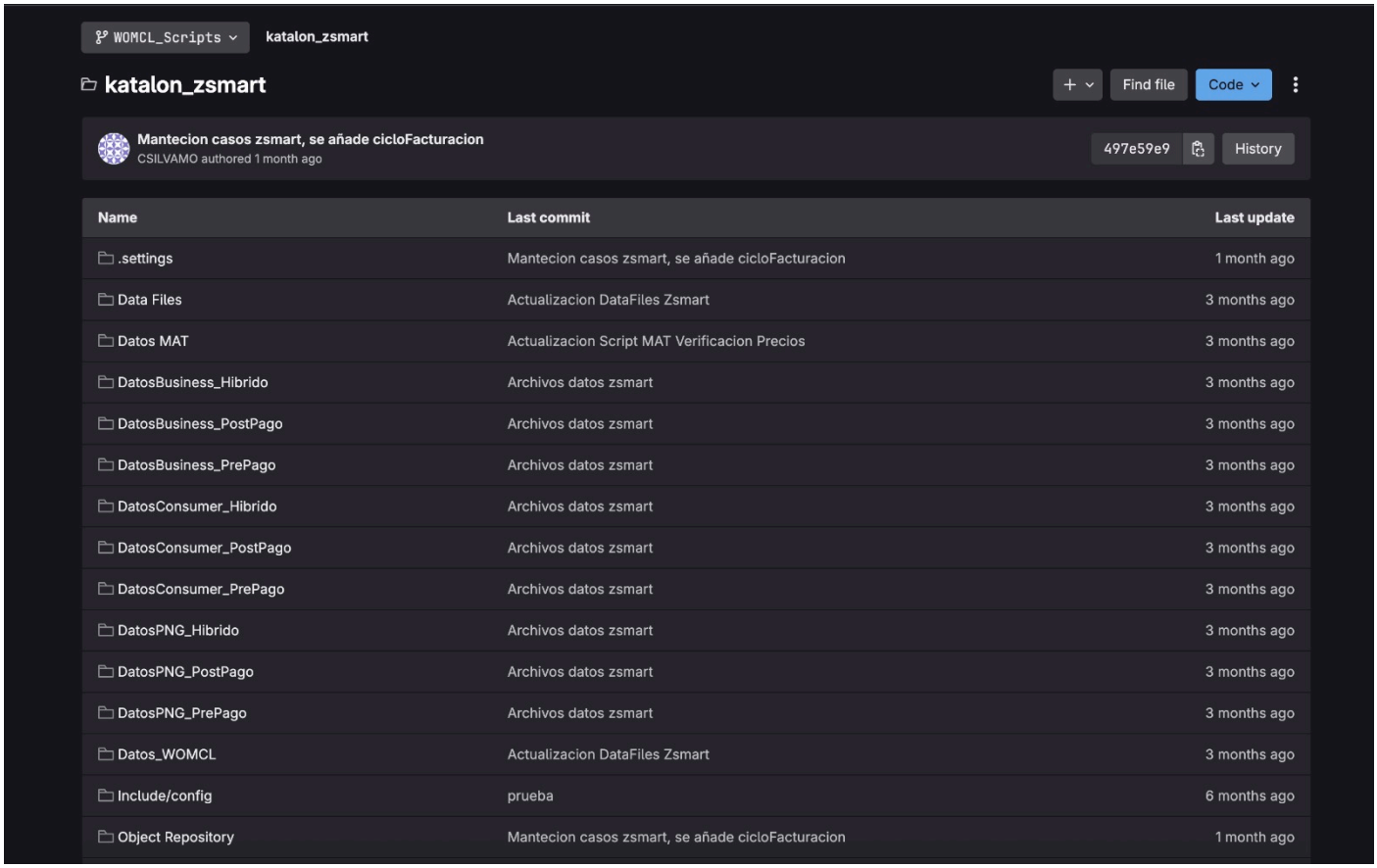

Figura 5:Integración GitLab.

Fuente: Elaboración Propia

- **Katalon TestCloud:** Plataforma de ejecución de pruebas en la nube proporcionada por Katalon. Se integró para realizar pruebas distribuidas en distintos sistemas operativos y navegadores sin necesidad de infraestructura local. Esto aseguró que los scripts fueran probados en entornos heterogéneos (por ejemplo, distintas versiones de Chrome y Firefox, en Windows, Linux, etc.), aumentando la robustez de la automatización al cubrir múltiples escenarios representativos de usuarios finales.

Gracias a esta integración de herramientas, se logró establecer un flujo de trabajo automatizado completo para las pruebas. Por ejemplo, los scripts desarrollados en Katalon Studio (y almacenados en GitLab) podían ser ejecutados mediante Katalon TestOps de forma programada o al desencadenarse ciertos eventos, como un *commit* en el repositorio. TestOps orquestaba la ejecución enviando las tareas de prueba a Katalon TestCloud u otros agentes disponibles, donde las pruebas corrían en paralelo sobre diferentes navegadores y sistemas operativos. Al finalizar cada ciclo, los resultados se centralizaban nuevamente en TestOps, donde el equipo podía revisar los reportes, métricas e históricos detallados de cada ejecución.

En resumen, se implementa tanto el desarrollo de scripts de automatización como su orquestación integrada, dando como resultado una solución de pruebas automatizadas robusta y escalable. Esto permitió cubrir los flujos críticos de la plataforma ZSmart de manera confiable y eficiente, reduciendo errores humanos y acelerando los ciclos de prueba en el proceso de aseguramiento de la calidad.

3.3 Integración en la Nube

La integración de la solución de automatización con la infraestructura en la nube, específicamente a través de Katalon TestCloud y Katalon TestOps, es fundamental para lograr escalabilidad, cobertura y eficiencia en las ejecuciones de pruebas.

El primer paso es **habilitar la integración con TestCloud** dentro de Katalon Studio. Esto se realiza

configurando los ajustes del proyecto para conectar el entorno local con la plataforma en la nube, asegurando que el proyecto esté asociado a una organización con una suscripción activa a TestCloud.[34]

Una vez habilitada la integración, se procede a **configurar los entornos de ejecución** en TestCloud. Esta funcionalidad permite seleccionar y preparar una amplia gama de entornos para satisfacer diversas necesidades de testing:

- **Navegadores de Escritorio:** Se especifica el sistema operativo (OS), el navegador y la versión deseada (por ejemplo, Windows 10 con Chrome 120).[33]
- **Navegadores Móviles:** Se selecciona el sistema operativo móvil, su versión y el dispositivo específico (por ejemplo, Android 13 en un Samsung S22). Por defecto, Chrome se utiliza para Android y Safari para iOS.[33]
- **Aplicaciones Nativas Móviles:** Se define el sistema operativo, su versión y el dispositivo para el testing de aplicaciones nativas.[33]

Para las aplicaciones alojadas en **entornos privados o locales**, Katalon TestCloud ofrece la funcionalidad de **TestCloud Tunnel**. Esta característica establece una conexión segura entre el entorno local y la nube, permitiendo que las pruebas en la nube accedan a las aplicaciones internas.[33]

La **orquestación de las ejecuciones con TestOps** es un componente crucial de la integración en la nube. Primero, el proyecto Katalon se almacena en un repositorio Git (como GitHub, GitLab o Azure DevOps), y se configura un "Script Repository" en TestOps con los accesos correspondientes para gestionar los scripts de prueba.[27] Luego, se configuran los agentes de Katalon en las máquinas locales (o en entornos Docker/Kubernetes) para que se conecten con TestOps y permitan la ejecución remota de las pruebas.[27] Finalmente, TestOps se utiliza para

programar las ejecuciones de pruebas, definiendo fechas, horas y especificando qué *test suites* deben ejecutarse de forma automática o manual.[26]

La integración en los **pipelines de CI/CD** es un beneficio significativo de esta solución. Katalon TestCloud y Katalon Runtime Engine (KRE) se integran sin problemas con plataformas de CI/CD como Jenkins, GitLab CI, Azure DevOps y GitHub Actions.[35] La ejecución de pruebas se puede disparar mediante argumentos de línea de comandos con KRE, especificando el tipo de navegador, el ID del entorno y los IDs de dispositivo/OS/aplicación móvil para TestCloud.[35] Las ventajas de esta integración en CI/CD incluyen una configuración rápida y sencilla (sin instalación manual de Studio), la capacidad de trabajar en máquinas sin pantalla (

headless execution) y la asignación de pantallas virtuales aisladas para aumentar la estabilidad

de la ejecución.[35]

La integración de Katalon Studio con TestCloud y TestOps no se limita a la ejecución remota de pruebas; se trata de **alcanzar un nivel de cobertura de pruebas y una velocidad de retroalimentación inalcanzables con los métodos tradicionales**. Esta capacidad permite a WOM asegurar que sus aplicaciones móviles y web funcionen de manera consistente en el diverso ecosistema de dispositivos y navegadores de los usuarios, al mismo tiempo que habilita una validación rápida y continua dentro de los pipelines de CI/CD. Esto se traduce directamente en un tiempo de comercialización más rápido para nuevos servicios y una experiencia de usuario significativamente mejorada, aspectos cruciales para retener y atraer clientes en el competitivo sector de las telecomunicaciones.

3.4 Generación de Reportes Automáticos

La generación de reportes automáticos es un componente esencial de la solución propuesta, ya que transforma los resultados brutos de las pruebas en información procesable, fundamental para la toma de decisiones y la mejora continua.

En **Katalon Studio**, los usuarios tienen acceso a un **monitoreo en tiempo real** de las ejecuciones de pruebas a través del Log Viewer y la Consola. Estas herramientas proporcionan actualizaciones detalladas sobre el progreso de la ejecución y el estado de cada paso de prueba.[33]

Una vez finalizadas las ejecuciones, Katalon Studio permite **exportar los reportes de pruebas** en diversos formatos, incluyendo HTML, PDF y CSV. Esta funcionalidad está disponible tanto para *test suites* individuales como para *test suite collections*, facilitando la distribución de los resultados.[41] Además, la herramienta ofrece la capacidad de

configurar notificaciones automáticas por correo electrónico para enviar los reportes de pruebas una vez que las ejecuciones han finalizado, asegurando una comunicación oportuna a los stakeholders relevantes.[41]

El verdadero potencial de los reportes se desbloquea con **Katalon TestOps**, que proporciona capacidades avanzadas de reporte y análisis. TestOps funciona como una plataforma centralizada para todos los resultados de las pruebas, transformando los "registros de ejecución desordenados en una imagen completa de la calidad".[30]

Las **características clave de análisis** que ofrece TestOps incluyen:

- **Dashboards Completos:** Proporciona paneles visualmente atractivos que muestran el historial de ejecución de pruebas, tendencias, análisis de fallos, detección de pruebas "flaky" (intermitentes), métricas de cobertura de pruebas y dashboards de rendimiento del equipo.[19]
- **Información Accionable Basada en Datos:** TestOps redefine el análisis al proporcionar información valiosa e inmediata a partir de los resultados de las pruebas. Esto permite una rápida detección y resolución de problemas, reduciendo el tiempo dedicado a la revisión manual de archivos de registro.[29] Los equipos de DevOps pueden descubrir y abordar rápidamente los problemas encontrados en cualquier parte del pipeline.[29]

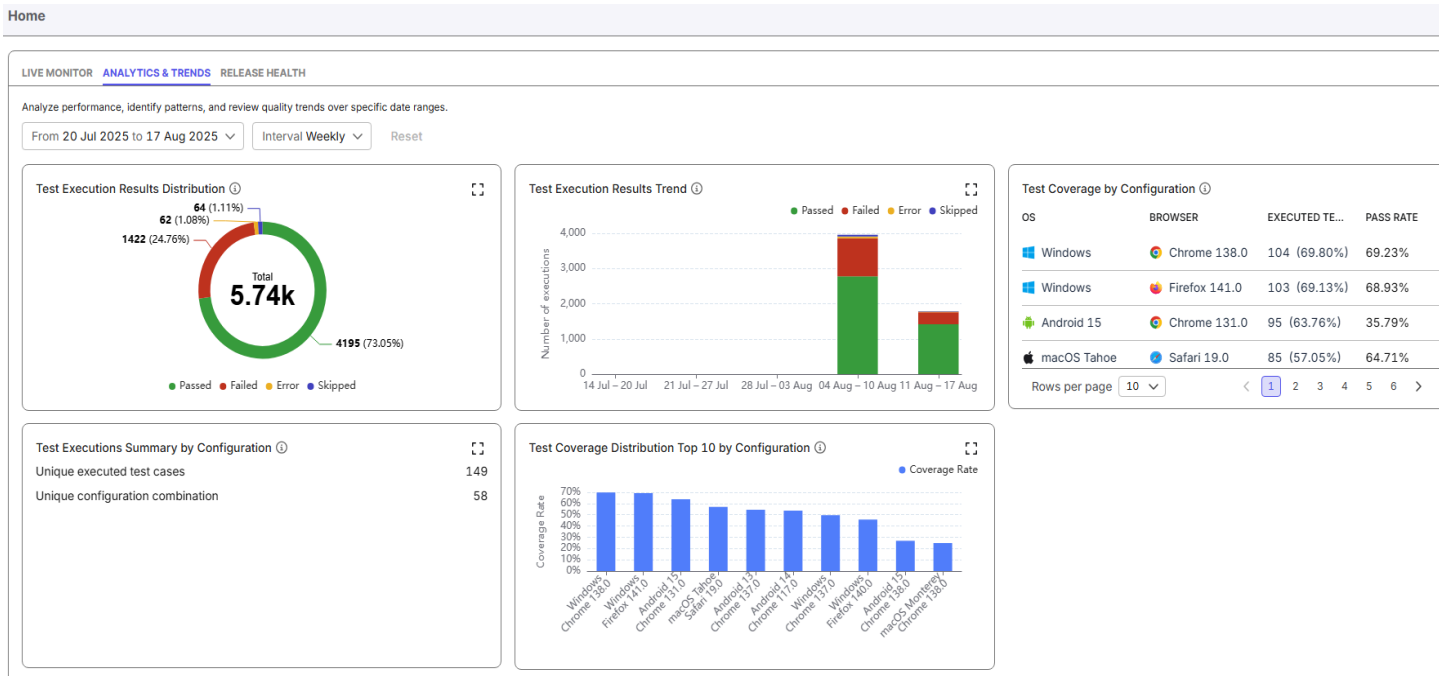

Figura 6:Dashboard TestOps.

Fuente: Elaboración Propia

La **integración entre Katalon Studio y TestOps** es crucial para obtener una visión completa. Una vez habilitada, todas las ejecuciones de pruebas realizadas en Katalon Studio se cargan

automáticamente a TestOps, asegurando que todos los resultados contribuyan a la plataforma centralizada de análisis.[41]

Las robustas capacidades de reporte y análisis de Katalon TestOps no se limitan a la presentación de datos; se trata de **habilitar un ciclo de mejora continua** tanto para el producto de software como para el proceso de QA. Al proporcionar información clara y basada en datos sobre el rendimiento de las pruebas, los patrones de defectos y las tendencias de calidad, estos reportes empoderan a los equipos de QA, desarrolladores y stakeholders del negocio para tomar decisiones informadas. Para una empresa de telecomunicaciones, esto significa identificar y abordar proactivamente los problemas de calidad del servicio, optimizar la asignación de recursos y demostrar el valor tangible de la automatización, fomentando así una cultura de calidad y alineando los esfuerzos técnicos con los objetivos de negocio.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

Este capítulo tiene como objetivo validar empíricamente la solución propuesta de automatización de pruebas. Se presentarán los escenarios de prueba seleccionados, las métricas obtenidas durante la ejecución y un análisis detallado de los resultados para demostrar la efectividad del framework implementado.

4.1 Escenarios de Prueba

La selección de escenarios de prueba representativos es fundamental para validar la efectividad del framework de automatización. Se han elegido escenarios clave que cubren las funcionalidades críticas y los flujos de usuario más importantes tanto para las aplicaciones móviles como web de WOM.

Los escenarios de prueba se clasifican en las siguientes categorías:

- **Escenarios de Pruebas Funcionales:** Estos escenarios verifican que las funcionalidades de la aplicación operen de acuerdo con los requisitos esperados.
    - **Autenticación de Usuario:** Incluye la verificación de la funcionalidad de inicio y cierre de sesión con credenciales válidas e inválidas, así como la recuperación de contraseña y la autenticación multifactor, si aplica.[45]
    - **Navegación de Servicios Centrales:** Asegura que los menús sean visibles y accesibles, y que los usuarios puedan navegar sin problemas a través de las funcionalidades principales, como la consulta de saldo, la visualización de planes o la gestión de servicios.[45]

    - **Validación de Formularios:** Comprueba que los formularios de entrada (por ejemplo, registro, contacto) manejen correctamente el envío de datos, los mensajes de error y los campos obligatorios.[46]
    - **Integridad de Enlaces y Contenido:** Verifica que todos los enlaces internos y externos funcionen correctamente y dirijan a las páginas esperadas, y que el contenido (texto, imágenes, videos) se muestre de forma precisa y sin errores.[46]
- **Escenarios de Pruebas de Interfaz de Usuario (UI/UX):** Estos escenarios se centran en la experiencia del usuario y la compatibilidad visual.
    - **Compatibilidad Multi-Navegador/Dispositivo:** Verifica que la interfaz de la aplicación se renderice correctamente y funcione como se espera en diversos navegadores de escritorio (Chrome, Firefox, Edge), navegadores móviles y dispositivos móviles nativos (Android, iOS) con diferentes resoluciones y orientaciones de pantalla.[33]
    - **Responsividad:** Asegura que la aplicación se adapte de manera adecuada a diferentes tamaños de pantalla (smartphones, tabletas, escritorios).[46]
    - **Usabilidad y Accesibilidad:** Evalúa la navegación intuitiva, la comodidad de los elementos interactivos y el cumplimiento de los estándares de accesibilidad (por ejemplo, navegación con teclado, texto alternativo para imágenes).[46]
- **Escenarios de Pruebas de Rendimiento:** Cruciales para plataformas de telecomunicaciones que manejan un alto volumen de usuarios.
    - **Pruebas de Carga y Estrés:** Simulan un alto tráfico de usuarios para evaluar el rendimiento, la estabilidad y la escalabilidad de la aplicación bajo demanda máxima.[18]
    - **Tiempos de Respuesta:** Miden la velocidad de las transacciones críticas y la carga de las páginas.[47]
- **Escenarios de Pruebas de API:** Validan la capa de servicios de la aplicación.
    - **Validación de Endpoints:** Verifica la corrección de los *endpoints* de la API, las estructuras de solicitud/respuesta (JSON/XML) y la integridad de los datos.[28]
    - **Autenticación y Autorización:** Prueba los mecanismos de seguridad para el acceso a la API.[28]
    - **Peticiones Encadenadas:** Simula procesos de negocio del mundo real que involucran múltiples llamadas a API interdependientes.[28]

La amplitud y profundidad de los escenarios de prueba seleccionados, facilitadas por las capacidades de testing multiplataforma y multitipo de Katalon, reflejan directamente la complejidad del software moderno en telecomunicaciones. Al automatizar estos diversos escenarios, la tesis demuestra que la solución proporciona una **evaluación de la calidad del software significativamente más robusta y fiable** de lo que podrían lograr los métodos manuales. Esta cobertura exhaustiva es vital para asegurar una experiencia de usuario consistente y de alta calidad en la vasta base de clientes y las diversas ofertas digitales de WOM,

impactando directamente en la satisfacción y retención del cliente.

4.2 Métricas Obtenidas

Para validar objetivamente la efectividad de la solución automatizada, se han recolectado y analizado métricas cuantitativas específicas, categorizadas según su enfoque. La comparación de estas métricas con líneas base pre-automatización o con benchmarks de la industria es crucial para demostrar mejoras tangibles.[37]

Las métricas clave obtenidas incluyen:

- **Métricas de Eficiencia:**
  - **Tiempo de Ejecución de Pruebas:** Se midió y comparó el tiempo total requerido para ejecutar las *test suites* automatizadas frente al tiempo estimado para una ejecución manual equivalente. Se observó una reducción drástica en el tiempo de ejecución.[5]
  - **Frecuencia de Ejecución:** La automatización permitió un aumento significativo en la frecuencia con la que se pueden ejecutar las *test suites* (ej. varias veces al día o por cada *commit*), lo cual era inviable con testing manual.
- **Métricas de Calidad (Producto y Proceso):**
  - **Cobertura de Automatización:** Se calculó el porcentaje de casos de prueba que han sido automatizados dentro de los escenarios seleccionados.[37]
  - **Cobertura de Pruebas:** Se estimó la exhaustividad del proceso de pruebas, evaluando si se cubrieron los requisitos clave o si se logró una mayor cobertura de código.[37]
  - **Densidad de Defectos:** Se registró el número de defectos encontrados por módulo o componente, lo que permitió identificar áreas con mayor propensión a errores.[37]
  - **Porcentaje de Casos de Prueba Pasados:** Se calculó la proporción de pruebas que se ejecutaron con éxito respecto al total de pruebas ejecutadas, indicando la estabilidad del software.[37]
  - **Tiempo Medio de Detección (MTTD):** Se midió la rapidez con la que el sistema automatizado identificó los defectos, buscando mantener valores bajos para una resolución ágil.[39]
  - **Defectos Escapados a Producción:** Se realizó un seguimiento de los defectos que, a pesar de las pruebas, lograron llegar al entorno de producción, con el objetivo de minimizar esta métrica.

2 Ciclos	Pasado	Total general
Ciclo 1	635	635
Regresión Manual	635	635
Ciclo 2	635	635
Regresión Automatizada	228	228
Regresión Manual	407	407
Total general	1270	1270

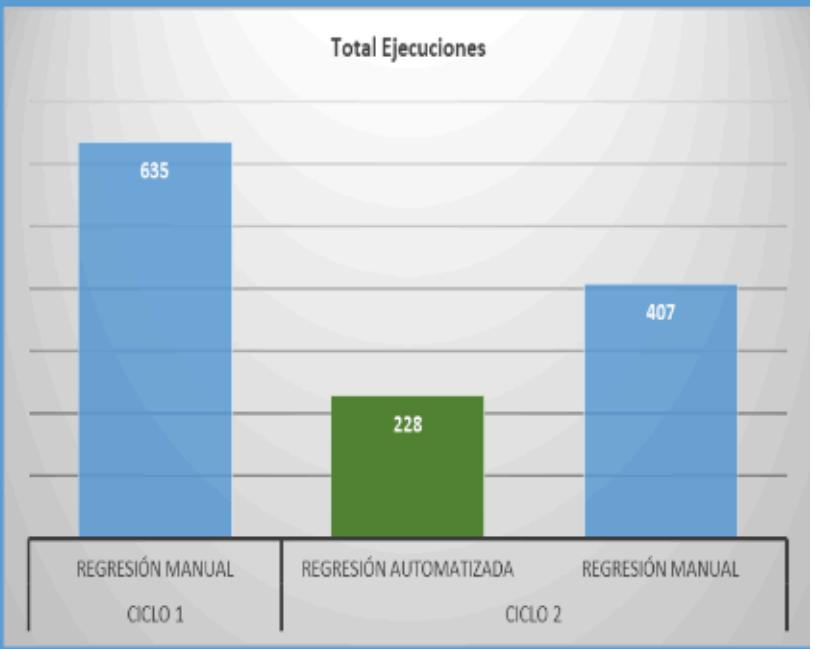

Figura 7:Comparación regresión ciclo 1 y ciclo 2.

Fuente: Elaboración Propia

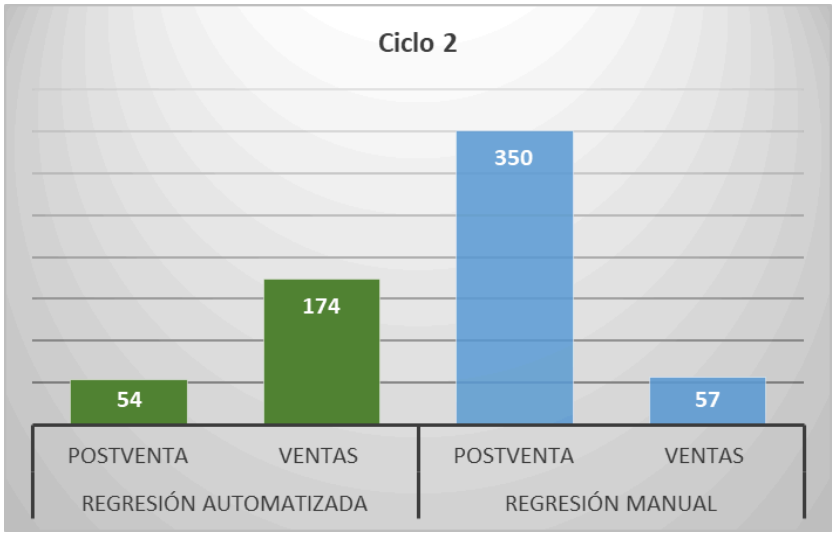

Figura 8:Detalle flujos automatizados.

Fuente: Elaboración Propia

Antes del inicio de la práctica profesional, el proceso de validación del sistema se realizaba casi en su totalidad de forma manual. Los gráficos disponibles muestran que en el ciclo de pruebas anterior a la práctica profesional la cantidad de ejecuciones manuales era muy elevada, reflejando una dependencia absoluta en pruebas manuales para la validación de la plataforma. No existía aún un conjunto de pruebas automatizadas, por lo que cada caso de prueba debía ejecutarse manualmente, con el consiguiente consumo de tiempo y riesgo de variabilidad en la ejecución y en los resultados.

Durante los tres meses de la práctica profesional, el foco fue en el desarrollo e implementación de scripts de prueba automatizados. Estos scripts se integraron en el flujo de trabajo de pruebas y comenzaron a utilizarse durante la siguiente fase de pruebas de regresión de la plataforma. Como resultado, los datos evidencian un punto de inflexión en el ciclo de prueba subsiguiente (identificado como *Ciclo 2* en los gráficos). Por primera vez se introdujeron ejecuciones automatizadas de pruebas, complementando y reduciendo las ejecuciones manuales necesarias en la etapa de regresión. En dicha iteración se alcanzó un total de 228 ejecuciones de prueba automatizadas. Esto corresponde aproximadamente al 36% del total de 635 pruebas ejecutadas

en ese ciclo. Consecuentemente, las ejecuciones manuales se redujeron a 407 en el *Ciclo 2*, una disminución notable frente a las 635 ejecuciones manuales del ciclo anterior (*Ciclo 1*). Esta comparación extraída de los gráficos demuestra de forma cuantitativa la mejora obtenida: en un solo ciclo de regresión, la automatización reemplazó una porción importante del trabajo manual, evidenciando ganancias inmediatas en eficiencia.

La introducción de la automatización supuso un impacto directo y positivo en el proceso de validación del sistema. Desde la incorporación de los scripts automatizados, se aprecia una tendencia sostenida en los ciclos de prueba posteriores: disminuyen notablemente las ejecuciones manuales, mientras que las automatizadas aumentan y pasan a desempeñar un rol protagónico. Esta transición implica una mejora sustancial tanto en la eficiencia como en la consistencia de la validación, ya que al automatizar una porción considerable de las pruebas de regresión se redujo la carga de trabajo manual y el potencial de error humano, a la vez que se aceleró la ejecución de pruebas repetitivas. Es importante destacar que todas estas mejoras se lograron en un período muy acotado (solo tres meses de práctica profesional), lo que subraya la efectividad del esfuerzo de automatización implementado y sirve como validación de los beneficios de integrar pruebas automatizadas en el proceso de aseguramiento de calidad del sistema.

CAPÍTULO 5: CONCLUSIONES Y FUTURAS MEJORAS

Recapitulación de Objetivos

La presente tesis se propuso como objetivo general diseñar, implementar y validar un framework integral de aseguramiento de calidad automatizado para aplicaciones móviles y web en WOM, utilizando la suite de herramientas Katalon. Para lograrlo, se establecieron objetivos específicos que abarcaban desde el análisis de las limitaciones del testing manual hasta la propuesta de una arquitectura de solución, la implementación práctica, la validación cuantitativa mediante métricas y la identificación de futuras mejoras.

Principales Hallazgos y Contribuciones

Los resultados de la validación empírica confirman que el framework automatizado ha abordado con éxito las problemáticas identificadas del testing manual. Se lograron **ganancias significativas en eficiencia**, con una reducción drástica en los tiempos de ejecución de las pruebas de regresión y funcionales, permitiendo ciclos de feedback más rápidos. La **calidad del software se vio mejorada**, evidenciada por una mayor cobertura de automatización, un alto porcentaje de casos de prueba pasados y una disminución en la densidad de defectos. La **capacidad de ejecutar pruebas en múltiples entornos** (navegadores, sistemas operativos, dispositivos móviles) a través de Katalon TestCloud amplió la cobertura de pruebas de manera sin precedentes, asegurando la compatibilidad y la experiencia de usuario en un ecosistema diverso. La **orquestación centralizada con Katalon TestOps** proporcionó una visibilidad y trazabilidad mejoradas, transformando los datos de ejecución en información accionable para la toma de decisiones.

Impacto de la Automatización QA en WOM

La implementación de QA automatizado con Katalon ha tenido un impacto multifacético y tangible en WOM:

- **Reducción de Costos y Tiempos:** La eliminación de tareas manuales repetitivas y la aceleración de los ciclos de ejecución de pruebas han resultado en ahorros significativos de tiempo y recursos.[5] Esto permite una liberación más rápida de nuevas funcionalidades y servicios al mercado.
- **Mejora de la Calidad del Producto:** La mayor efectividad del testing, la cobertura más amplia y la detección temprana de defectos han llevado a un producto final de mayor calidad y, consecuentemente, a una mejor satisfacción del usuario.[5]
- **Optimización de Recursos Humanos:** La automatización ha liberado a los profesionales de QA de tareas tediosas, permitiéndoles enfocar sus conocimientos y creatividad en actividades de mayor valor, como pruebas exploratorias, análisis de sistemas complejos o diseño de estrategias de testing avanzadas.[9]
- **Escalabilidad y Reutilización:** El framework ha demostrado la capacidad de manejar grandes volúmenes de datos y la habilidad de repetir pruebas de forma consistente a lo largo de múltiples iteraciones de desarrollo.[5]
- **Habilitador de CI/CD y DevOps:** La integración de las pruebas automatizadas en los pipelines de Integración Continua y Entrega Continua (CI/CD) ha acelerado el tiempo de comercialización, fomentando una cultura de desarrollo más ágil y eficiente.[6]

Limitaciones de la Solución Propuesta

A pesar de los beneficios sustanciales, es importante reconocer las limitaciones inherentes a la automatización de pruebas y los desafíos específicos encontrados durante la implementación:

- **Complemento, No Reemplazo, del Testing Manual:** La automatización no elimina completamente la necesidad de pruebas manuales, especialmente para pruebas exploratorias, de usabilidad o aquellas que requieren juicio humano y creatividad.[5]
- **Esfuerzo Inicial de Configuración y Mantenimiento Continuo:** La inversión inicial en la configuración del framework y el esfuerzo continuo requerido para mantener los scripts de prueba actualizados a medida que la aplicación evoluciona pueden ser considerables.[3]
- **Problemas de Adaptabilidad:** Las pruebas automatizadas pueden enfrentar desafíos de adaptabilidad en entornos con interfaces de usuario que cambian rápidamente o en sistemas altamente dinámicos.[7]
- **Dependencia de Herramientas:** La solución depende de la suite Katalon, lo que implica una

curva de aprendizaje inicial y una dependencia de la evolución y el soporte de la herramienta.[7]
- **Latencia en TestCloud:** Se identificó que la ejecución en TestCloud puede ser más lenta que la ejecución local debido a la latencia de internet y al protocolo Remote WebDriver, especialmente para pruebas con muchos pasos individuales.[36]

Futuras Mejoras y Líneas de Investigación

Basándose en la implementación actual y las limitaciones identificadas, se proponen varias líneas de mejora y futuras investigaciones:

- **Integración de Inteligencia Artificial (IA) y Machine Learning (ML):** Explorar técnicas avanzadas para optimizar aún más la automatización de pruebas.
  - **Generación de Pruebas Impulsada por IA:** Investigar el uso de IA para automatizar la generación de casos de prueba y la creación de scripts, reduciendo el esfuerzo manual en la fase de diseño.[49]
  - **Pruebas Adaptativas y Auto-reparables:** Implementar algoritmos de ML para que las pruebas se ajusten en tiempo real a medida que cambian los requisitos del usuario y para que se recuperen automáticamente de fallos (auto-sanación), como lo hace TestRigor.[49]
  - **Análisis Predictivo Mejorado:** Aprovechar el ML para el análisis predictivo, identificando áreas de alto riesgo en el software y priorizando los esfuerzos de testing donde más se necesitan.[16]
  - **Monitoreo Impulsado por IA:** Desarrollar o integrar herramientas de monitoreo basadas en IA para rastrear el rendimiento en diferentes entornos y realizar análisis de causa raíz de los defectos.[50]
- **Expansión de la Cobertura de Pruebas:** Extender la automatización a otras áreas críticas como el testing de seguridad, el testing de accesibilidad o el testing de rendimiento más profundo.
- **Optimización del Rendimiento en la Nube:** Investigar estrategias para mitigar la latencia de TestCloud en pruebas críticas de rendimiento, posiblemente mediante la optimización de scripts o la exploración de configuraciones de entorno avanzadas.
- **Exploración de Nuevos Tipos de Pruebas:** Investigar la automatización para tecnologías emergentes relevantes para el sector de telecomunicaciones, como aplicaciones basadas en blockchain o integraciones de IoT.[47]

Esta memoria de título no solo valida los beneficios inmediatos de implementar un framework

de QA automatizado, sino que también lo posiciona como un **paso fundamental hacia un futuro más inteligente y autónomo para el aseguramiento de calidad de software** dentro de WOM. Al delinear explícitamente el camino para integrar IA/ML, el trabajo demuestra una perspectiva de futuro, mostrando cómo la empresa puede seguir innovando en calidad, anticipar desafíos y mantener una ventaja competitiva en el panorama de las telecomunicaciones en rápida evolución.

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS

1. Babatunde, S. O., Oladipo, A. M., & Adebayo, A. A. (2023). A Comprehensive Review of Automated Software Testing Techniques and Tools. International Journal of Scientific and Engineering Research, 14(4), 1–8.
2. Digital.ai. (s. f.). Beneficios de utilizar un marco de automatización de pruebas. Obtenido de https://digital.ai/es/glossary/test-automation-framework/
3. Diva-Portal. (s. f.). Software metrics plays an important role in measuring attributes that are critical to the success of a software project. Obtenido de https://www.diva-portal.org/smash/get/diva2:833623/FULLTEXT01.pdf
4. Harrisburg University. (s. f.). This thesis concentrates on incorporating quality assurance (QA) automation in project teams, mainly within enterprise organizations, and applying it to search engine optimization (SEO) projects. Obtenido de https://digitalcommons.harrisburgu.edu/cgi/viewcontent.cgi?article=1052&context=dandt
5. Hostragons. (s. f.). Análisis de resultados de pruebas automatizadas: mejores prácticas. Obtenido de https://www.hostragons.com/es/blog/estrategias-de-pruebas-automatizadas/
6. IBM. (s. f.). ¿Por qué es importante la automatización en la nube? Obtenido de https://www.ibm.com/es-es/topics/cloud-automation
7. Imagina Formación. (s. f.). Aprende Katalon Studio: Tutorial de primeros pasos. Obtenido de https://imaginaformacion.com/tutoriales/aprende-katalon-studio-tutorial-de-primeros-pasos
8. IJISTECH. (s. f.). The use of automation in mobile application testing has become an important tool in improving the efficiency and quality of applications. Obtenido de https://ijistech.org/ijistech/index.php/ijistech/article/download/347/345
9. Katalon Academy. (s. f.). Creating more effective test scripts with Statements in Katalon Studio. Obtenido de https://academy.katalon.com/courses/statements-in-katalon-studio/
10. Katalon Academy. (s. f.). Software Test Reporting. Obtenido de https://academy.katalon.com/courses/software-test-reporting/
11. Katalon LLC. (2021, 26 de julio). Katalon Introduces TestOps, an Open and Comprehensive Test Orchestration Platform to Streamline DevOps. PR Newswire. Obtenido de https://www.prnewswire.com/news-releases/katalon-introduces-testops-an-open-and-comprehensive-test-orchestration-platform-to-streamline-devops-3013

41181.html
12. Kathiriya, S. (s. f.). Optimizing Automated Software Testing with Machine Learning Techniques. ResearchGate. Obtenido de https://www.researchgate.net/profile/Satish-Kathiriya/publication/379954208_Optimizing_Automated_Software_Testing_with_Machine_Learning_Techniques/links/6623038866ba7e2359ec09ab/Optimizing_Automated_Software_Testing_with_Machine_Learning_Techniques.pdf
13. Laborum Chile. (s. f.). Analista QA - mayo 2025. Obtenido de https://www.laborum.cl/empleos/analista-qa-1117119476.html
14. Medium (Augustin Mancebo). (s. f.). Katalon Studio versus Frameworks de código abierto basados en Selenium. Obtenido de https://medium.com/@augustinmancebo/katalon-studio-versus-frameworks-de-c%C3%B3digo-abierto-basados-en-selenium-72945d43815d
15. Medium (Govinda Solanki). (s. f.). Best Practices for Designing a Test Automation Framework. Obtenido de https://medium.com/@solanki.govinda/best-practices-for-designing-a-test-automation-framework-4e68ae7c3688
16. Mentores Tech. (s. f.). ¿Cuáles son las ventajas y desventajas de las pruebas manuales? Obtenido de https://www.mentorestech.com/resource-guide/qa/question/cuales-son-las-ventajas-y-desventajas-de-las-pruebas-manuales
17. Open iT. (s. f.). La falta de visibilidad de las TI sigue acosando a las organizaciones modernas. Obtenido de https://openit.com/es/it-visibility-gap-remains-a-challenge-in-modern-organizations/
18. Optima Quantum. (s. f.). La inteligencia artificial está revolucionando el ámbito de las pruebas de software. Obtenido de https://optimaquantum.com/ai-y-automatizacion-el-futuro-de-las-pruebas-de-software/
19. Platzi. (s. f.). Limitaciones de la automatización de pruebas. Obtenido de https://platzi.com/cursos/automatizacion-pruebas/limitaciones-de-la-automatizacion-de-pruebas/
20. Practia Global. (s. f.). La automatización de pruebas de software: una decisión que impacta en tu negocio. Obtenido de https://perspectiva.practia.global/automatizacion-de-pruebas-una-decision-que-impacta-en-tu-negocio/
21. Prometeo FP. (s. f.). Pruebas Automatizadas: La Clave del Éxito en Software. Obtenido de https://www.prometeo-fp.com/blog/testing-automatizado-pruebas-ahorra-tiempo
22. QALified. (s. f.). Automatización del Testing Visual con Katalon. Obtenido de https://qalified.com/es/blog/automatizacion-testing-visual-katalon/
23. QAMindsLab. (s. f.). Pruebas automatizadas para aplicaciones móviles. Obtenido de https://www.qamindslab.com/pruebas-automatizadas-para-aplicaciones-movile

s
24. QAwerk. (s. f.). Métricas de pruebas de software más importantes. Obtenido de https://qawerk.es/blog/metricas-de-pruebas-de-software/
25. QAwerk. (s. f.). Qué es el testing de sitios web. Obtenido de https://qawerk.es/blog/pruebas-de-sitios-web/
26. QAwerk. (s. f.). Las 15 mejores herramientas de pruebas para móviles en 2024. Obtenido de https://qawerk.es/blog/mejores-herramientas-de-pruebas-para-moviles/
27. Rootstack. (s. f.). Beneficios de la QA automatizada. Obtenido de https://rootstack.com/es/learning/beneficios-de-la-qa-automatizada
28. Sentrio. (s. f.). 6 Métricas QA para el éxito de tus proyectos. Obtenido de https://sentrio.io/blog/metricas-qa-control-de-calidad/
29. SII Group. (s. f.). QA Funcional. Obtenido de https://sii-group.com/es-CL/offers/analistas/qa-funcional
30. SoftwareOne. (s. f.). Automatización de pruebas con Katalon Studio. Obtenido de https://www.softwareone.com/es-es/blog/articulos/2022/01/11/automatizacion-de-pruebas-con-katalon-studio
31. Tivit. (s. f.). La automatización de pruebas de rendimiento debe priorizarse después de asegurar que las pruebas unitarias y de integración están bien cubiertas. Obtenido de https://latam.tivit.com/blog/herramientas-de-automatizacion
32. Tripleten. (s. f.). Top 10 mejores herramientas de pruebas de software para 2025. Obtenido de https://tripleten.mx/blog/las-10-mejores-herramientas-de-prueba-de-software-para-testing/
33. Unite.AI. (s. f.). El futuro de la IA en el aseguramiento de la calidad. Obtenido de https://www.unite.ai/es/El-futuro-de-la-IA-en-el-aseguramiento-de-la-calidad/
34. Unimedia Tech. (s. f.). La ventaja estratégica de la automatización de pruebas de software. Obtenido de https://www.unimedia.tech/es/la-ventaja-estrategica-de-la-automatizacion-de-pruebas-de-software/
35. procesos-enfoques-herramientas-y-mucho-mas
36. Zaptest. (s. f.). Qué es la automatización de pruebas: una guía sencilla y sin jerga. Obtenido de https://www.zaptest.com/es/que-es-la-automatizacion-de-pruebas-una-guia-sencilla-y-sin-jerga
37. Abstracta. (s. f.). Métricas de calidad de software: ¿cómo medir tu impacto? Obtenido de https://es.abstracta.us/blog/metricas-de-calidad-de-software/
38. TestingBaires. (s. f.). Trazabilidad y el rastreo integral en pruebas de software. Obtenido de https://testingbaires.com/trazabilidad-y-el-rastreo-integral-en-pruebas-de-software/
39. aqua cloud. (s. f.). Katalon Review 2025: What alternative should you try? Obtenido de https://aqua-cloud.io/katalon-studio-review-2025/
40. Adictos al Trabajo. (2024, 15 de noviembre). Framework de automatización:

Katalon Studio Free. Obtenido de https://adictosaltrabajo.com/2024/11/15/framework-de-automatizacion-katalon-studio-free/