



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería Electrónica y Eléctrica

Escuela Profesional de Ingeniería Electrónica

**Diseño e implementación de pruebas automatizadas
para procesos de autenticación y flujos funcionales en
aplicaciones web**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero Electrónico

AUTOR

José Ignacio VALENCIA BERNA

ASESOR

José VIDAL HUARCAYA

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Valencia, J. (2026). *Diseño e implementación de pruebas automatizadas para procesos de autenticación y flujos funcionales en aplicaciones web*. [Trabajo de Suficiencia Profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Escuela Profesional de Ingeniería Electrónica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	José Ignacio Valencia Berna
Tipo de documento de identidad	DNI
Número de documento de identidad	70986204
URL de ORCID	No Aplica
Datos de asesor	
Nombres y apellidos	José Vidal Huarcaya
Tipo de documento de identidad	DNI
Número de documento de identidad	09450859
URL de ORCID	https://orcid.org/0000-0002-7702-8162
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Teresa Esther Nuñez Zuñiga
Tipo de documento	DNI
Número de documento de identidad	08006778
Miembro del jurado 1	
Nombres y apellidos	Jean Carlos Malca Fernandez
Tipo de documento	DNI
Número de documento de identidad	45535181
Miembro del jurado 2	
Nombres y apellidos	Raul Sixto Hinojosa Sanchez
Tipo de documento	DNI
Número de documento de identidad	08605661
Datos de investigación	
Línea de investigación	C.0.3.22. Ingeniería de Software
Grupo de investigación	No Aplica

Agencia de financiamiento	No Aplica
Ubicación geográfica de la investigación	Departamento: Lima Provincia: Lima Distrito: San Isidro Latitud: -12.092294 Longitud: -77.034612
Año o rango de años en que se realizó la investigación	2024 - 2025
URL de disciplinas OCDE	Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
Teléfono 619-7000 Anexo 4226
Calle Germán Amezaga 375 – Lima 1 – Perú



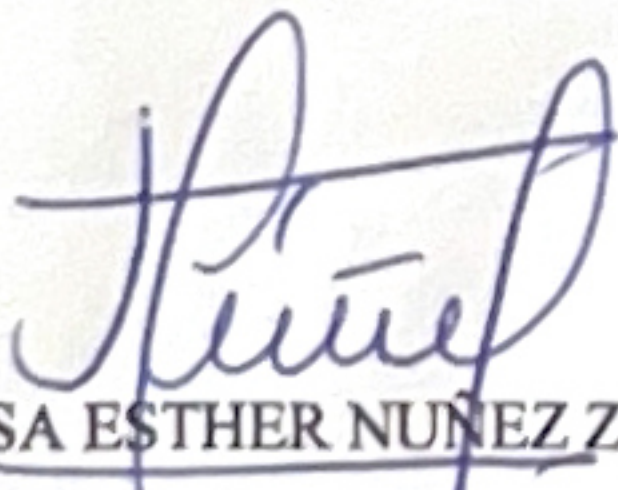
ACTA DE SUSTENTACIÓN
TRABAJO DE
SUFICIENCIA PROFESIONAL
ACTA N°18/FIEE-PPP/2026

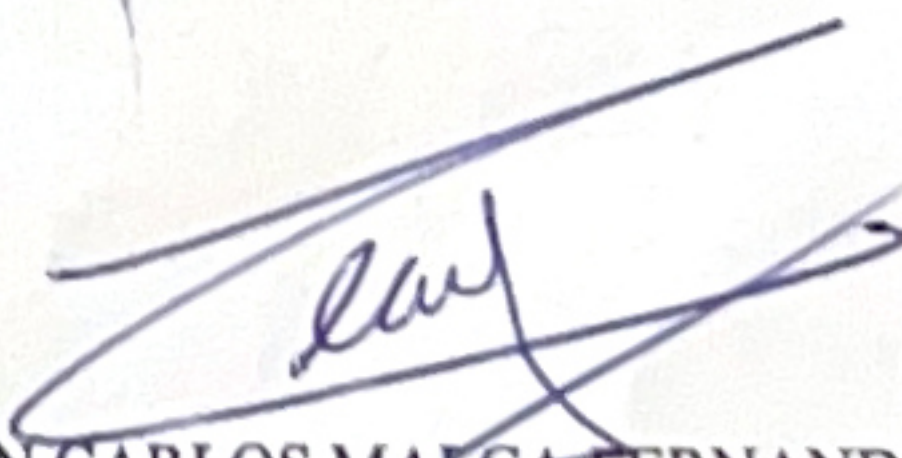
Los suscritos Miembros del Jurado, docentes permanentes de las Escuelas Profesionales de la Facultad de Ingeniería Electrónica y Eléctrica, reunidos en la fecha 10 de marzo del 2026, como presidente de Jurado la **DRA. TERESA ESTHER NUÑEZ ZUÑIGA**, integrado por el Miembro de Jurado el **MG. JEAN CARLOS MALCA FERNANDEZ**, el Miembro de Jurado el **MG. RAUL SIXTO HINOJOSA SANCHEZ** y Miembro Asesor el **ING. JOSÉ VIDAL HUARCAYA**.

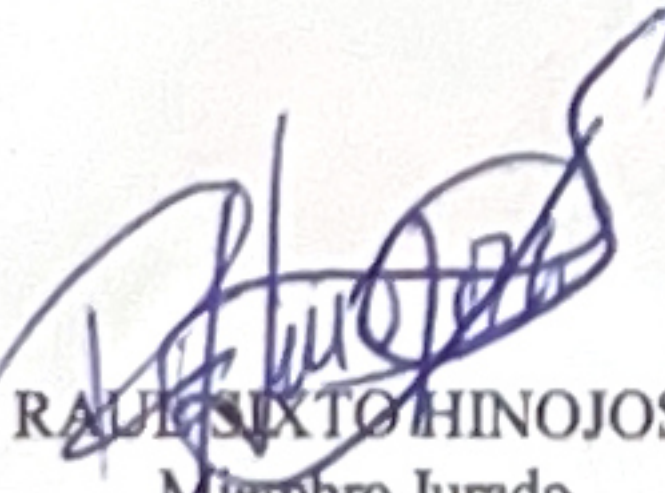
Después de escuchar la Sustentación Presencial del Trabajo de Suficiencia Profesional del Bachiller **JOSÉ IGNACIO VALENCIA BERNAL** con código 17190140 que para optar el Título Profesional de Ingeniero Electrónico sustentó el Trabajo de Suficiencia Profesional titulado **DISEÑO E IMPLEMENTACIÓN DE PRUEBAS AUTOMATIZADAS PARA PROCESOS DE AUTENTICACIÓN Y FLUJOS FUNCIONALES EN APLICACIONES WEB**.

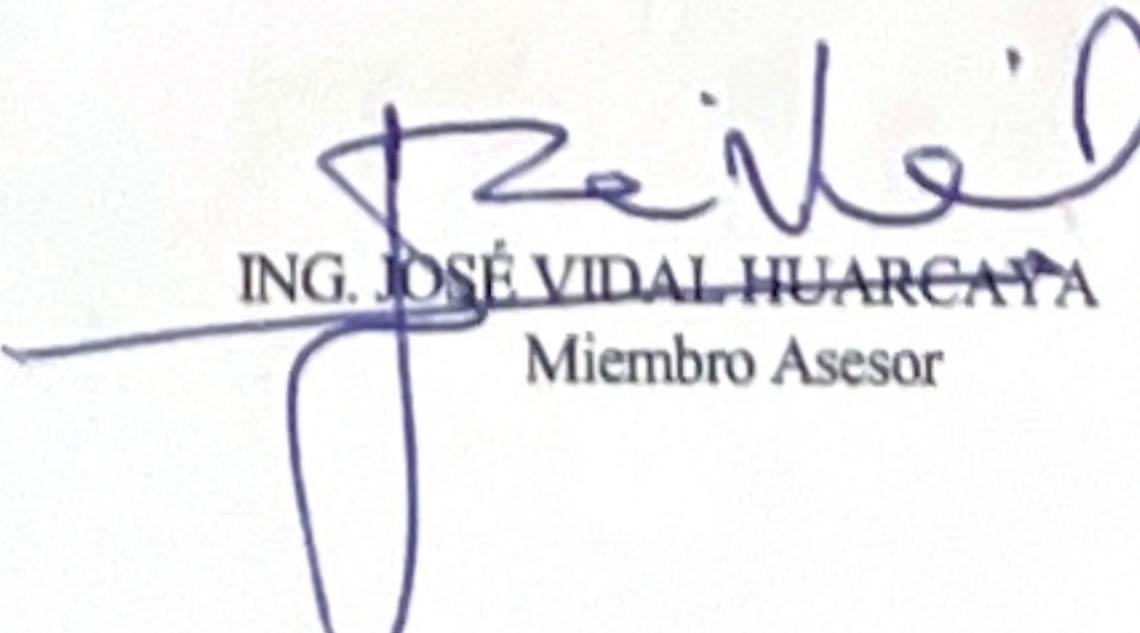
El jurado examinador procedió a formular las preguntas reglamentarias y, luego de una deliberación en privado, decidió otorgarle el calificativo de 16 (Dieciseis).

Ciudad Universitaria, 10 de marzo del 2026


DRA. TERESA ESTHER NUÑEZ ZUÑIGA
Presidente Jurado


MG. JEAN CARLOS MALCA FERNANDEZ
Miembro Jurado


MG. RAUL SIXTO HINOJOSA SANCHEZ
Miembro Jurado


ING. JOSÉ VIDAL HUARCAYA
Miembro Asesor



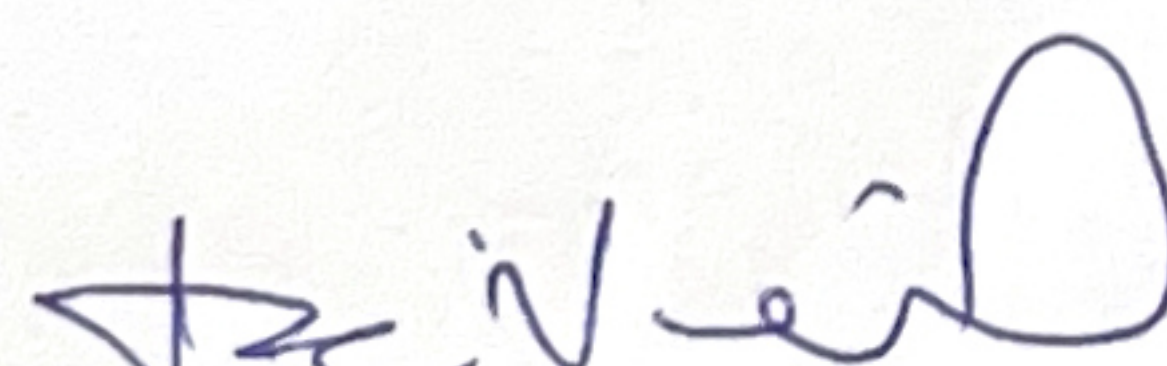
UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, **JOSÉ VIDAL HUARCAYA** en mi condición de asesor acreditado con el Dictamen N°15/FIEE-PPP/2026 del Trabajo de Suficiencia Profesional titulado **DISEÑO E IMPLEMENTACIÓN DE PRUEBAS AUTOMATIZADAS PARA PROCESOS DE AUTENTICACIÓN Y FLUJOS FUNCIONALES EN APLICACIONES WEB**, presentado por el bachiller JOSÉ IGNACIO VALENCIA BERNA, para optar al título profesional de Ingeniero Electrónico. CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 4% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su publicación en el repositorio institucional.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título de Ingeniero Electrónico.

Lima, 10 de Marzo del 2026


Firma del Asesor

DNI: 09450859.

Nombres y apellidos del asesor:
JOSÉ VIDAL HUARCAYA

DEDICATORIA

Dedico este trabajo a mi familia, quienes me acompañaron en cada etapa de mi camino profesional; a mi pareja, que creyó en mí incluso cuando yo dudaba; a mi mascota, que con su compañía silenciosa me brindó lealtad y amor incondicional; y, sobre todo, a mi madre, que desde el cielo sigue siendo mi fuente de motivación y fortaleza para alcanzar cada una de mis metas.

AGRADECIMIENTOS

Agradezco profundamente a mis familiares, pareja y amigos por su constante aliento y apoyo para alcanzar este logro tan importante. También a mi asesor, por su tiempo, orientación y compromiso durante el desarrollo de este trabajo, y a la empresa que me brindó la oportunidad de llevar a cabo este proyecto, contribuyendo significativamente a mi crecimiento profesional y personal.

RESUMEN

El presente trabajo de suficiencia se desarrolla en un pequeño equipo de desarrollo encargado de uno de los muchos proyectos de la empresa consultora de Tecnología de la Información, Indra Perú.

El sistema de pruebas automatizadas tiene como objetivo optimizar la ejecución de pruebas y la recopilación de evidencias, tareas que anteriormente se realizaban manualmente por el equipo de desarrollo. Esto contribuye a reducir la carga laboral y los costos operativos asociados.

La metodología de desarrollo utilizada abarca la evaluación del contexto actual del equipo y las limitaciones para la implementación del sistema. A partir de este análisis, se plantea la creación de una aplicación de escritorio que permite ejecutar scripts de pruebas automatizadas, utilizando una arquitectura modular que facilita su escalabilidad y mantenimiento.

Palabras clave: Pruebas de software, automatización, autenticación, flujo funcional, aplicaciones web.

ABSTRACT

This sufficiency work is being carried out by a small development team responsible for one of the many projects of the Information Technology consulting firm Indra Peru.

The automated testing system aims to optimize test execution and evidence collection, tasks that were previously performed manually by the development team. This contributes to reducing workload and associated operating costs.

The development methodology used includes assessing the team's current context and the constraints on system implementation. Based on this analysis, the creation of a desktop application is proposed that allows the execution of automated test scripts, using a modular architecture that facilitates scalability and maintenance.

Keywords: Software testing, authentication, automation, functional flow, web applications.

ÍNDICE GENERAL

CARÁTULA	i
RESUMEN	iv
ABSTRACT	v
ÍNDICE GENERAL	vi
LISTA DE FIGURAS	viii
LISTA DE TABLAS	ix
CAPITULO I: INTRODUCCIÓN	1
1.1 Objetivos del informe	1
1.2 Estructura del informe	1
CAPITULO II: INFORMACIÓN DEL LUGAR DONDE SE DESARROLLÓ LA ACTIVIDAD	3
2.1 Institución – Actividad que desarrolla	3
2.2 Periodo de duración de la actividad.	3
2.3 Finalidad y objetivos de la entidad	3
2.4 Razón social	3
2.5 Dirección postal	3
2.6 Nombre y correo electrónico del profesional a cargo.	4
CAPITULO III: DESCRIPCIÓN DE LA ACTIVIDAD	5
3.1 Organización de la actividad	5
3.2 Finalidad y objetivos de la Actividad	6
3.2.1 Finalidad	6
3.2.2 Objetivos	6
3.2.2.1 Objetivo General	6
3.2.2.2 Objetivos Específicos	7
3.3 Problemática	7
3.3.1 Problema General	8

3.3.2	Problemas Específicos	8
3.3.3	Justificación e importancia de la investigación	8
3.4	Metodología y Procedimiento	9
3.4.1	Metodología	9
3.4.1.1	Bases teóricas	9
3.4.1.2	Marco conceptual	11
3.4.2	Procedimiento	24
3.4.2.1	Contexto de la situación actual	24
3.4.2.2	Stack tecnológico	25
3.4.2.3	Definición del alcance	26
3.4.2.4	Estructura del proyecto	27
3.4.2.5	Diseño de la propuesta	29
3.5	Resultado de la actividad	35
CAPITULO IV: DISCUSIÓN DE RESULTADOS		37
4.1	Descripción de la implementación	37
CAPITULO V: CONCLUSIONES		39
CAPITULO VI: RECOMENDACIONES		41
CAPITULO VII: BIBLIOGRAFÍA		42
CAPITULO VIII: ANEXOS		47

LISTA DE FIGURAS

Figura 1 Organigrama Referencial Indra Perú.	5
Figura 2 Técnicas para pruebas de software.	15
Figura 3 Pirámide de pruebas en aplicaciones web.	17
Figura 4 Arquitectura del sistema.	28
Figura 5 Lógica de negocio.	30
Figura 6 Automatización de caso de prueba.	32
Figura 7 Comparación del tiempo de ejecución de pruebas manuales y automatizadas.	36

LISTA DE TABLAS

Tabla 1 Comparación de herramientas de automatización de pruebas	21
Tabla 2 Comparación lenguajes de programación más utilizados	24

CAPITULO I: INTRODUCCIÓN

Un aspecto fundamental que siempre debe estar garantizado en el desarrollo de software es la calidad. Por esta razón, es importante realizar pruebas sobre las aplicaciones web para asegurar que cumplan con sus requerimientos y ofrezcan al usuario una experiencia estable y confiable. Sin embargo, en muchos equipos de desarrollo, la ejecución de estas pruebas es realizada de forma manual, lo que las hace repetitivas, costosas y propensas a errores humanos. Para afrontar estos problemas, la automatización se presenta como una solución que puede aportar eficiencia al reducir el tiempo de ejecución y validaciones posteriores, así como de errores por impericia del validador.

1.1 Objetivos del informe

En ese contexto, la presente investigación tuvo como objetivo diseñar un sistema de pruebas automatizadas para los procesos de autenticación y flujo funcional en la validación de aplicaciones web utilizadas en una empresa de telecomunicaciones. Específicamente, se centró en diseñar pruebas automatizadas para cada proceso mencionado y también en la integración de ambos tipos de pruebas.

1.2 Estructura del informe

Esta investigación se estructuró por capítulos para poder desarrollar de forma ordenada y secuencial el trabajo; en el segundo capítulo se ha descrito el contexto organizacional de la empresa en la cual se desarrolló el trabajo; en

el tercer capítulo se detalló la finalidad, los objetivos y la problemática que motivó su desarrollo, así como los antecedentes y fundamentos teóricos necesarios para comprender los conceptos aplicados, y se detalló el procedimiento seguido y los resultados obtenidos; en el cuarto capítulo se realizó la discusión de resultados, respecto a la implementación realizada y resultados obtenidos en investigaciones similares; y en el quinto y sexto capítulo se presentaron las conclusiones y recomendaciones derivadas del trabajo, así como consideraciones orientadas a posibles mejoras futuras.

CAPITULO II: INFORMACIÓN DEL LUGAR DONDE SE DESARROLLÓ LA ACTIVIDAD

2.1 Institución – Actividad que desarrolla

Indra Perú es una empresa líder en transformación digital y Tecnologías de la Información. Su modelo de negocio está basado en una oferta integral de productos propios, con un enfoque end-to-end, de alto valor y con un elevado componente de innovación.

2.2 Periodo de duración de la actividad.

El presente trabajo se desarrolló durante el periodo diciembre 2024 – julio 2025.

2.3 Finalidad y objetivos de la entidad

Impulsar la transformación de los negocios y la sociedad mediante soluciones y servicios innovadores, poniendo a las personas en el centro.

2.4 Razón social

INDRA PERÚ S.A.C.

2.5 Dirección postal

Av. Basadre No 233 San Isidro
Lima 27 15073 Lima Perú

2.6 Nombre y correo electrónico del profesional a cargo.

Victor Rios Campos

vrisc@indracompany.com

CAPITULO III: DESCRIPCIÓN DE LA ACTIVIDAD

3.1 Organización de la actividad

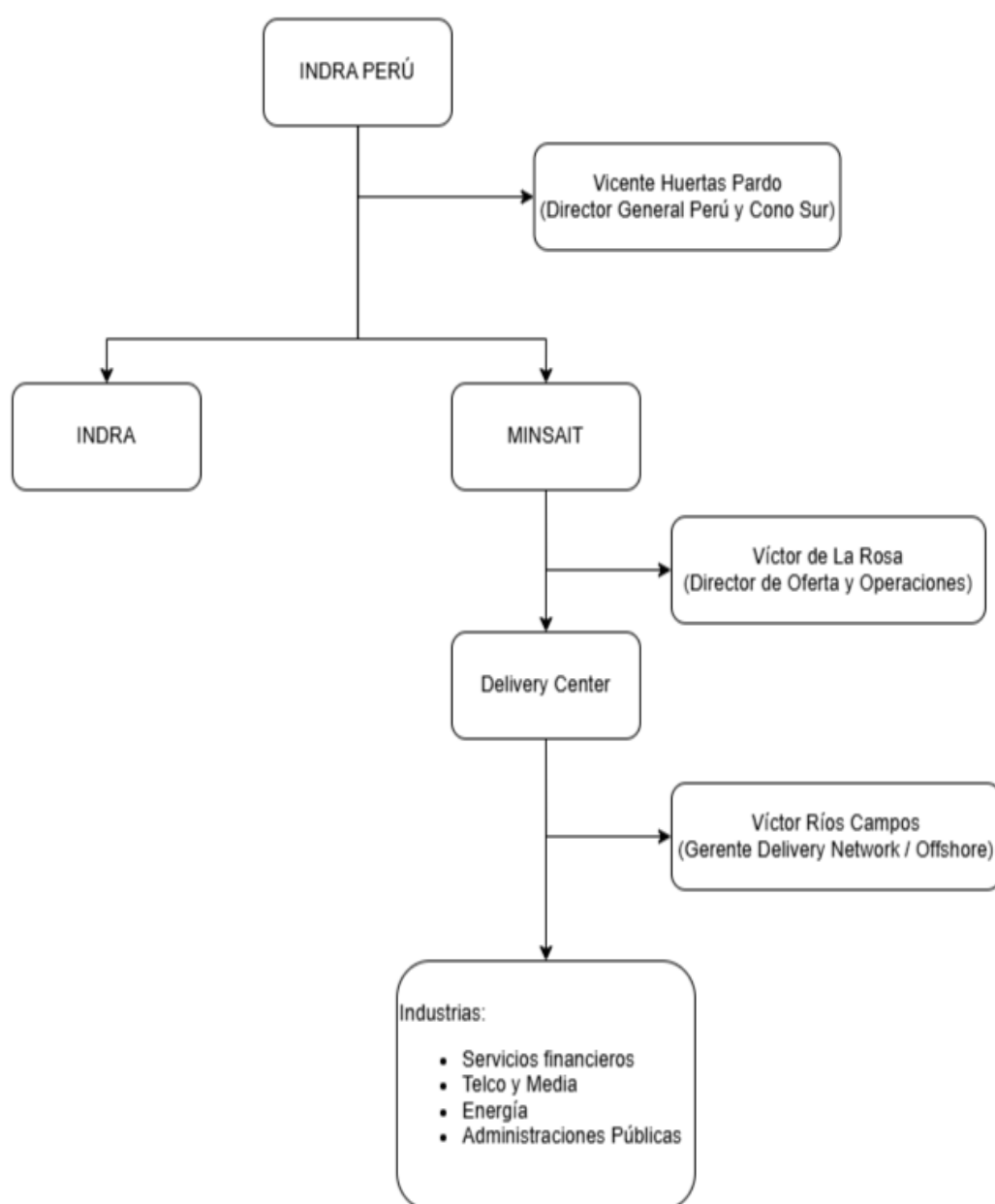


Figura 1 Organigrama Referencial Indra Perú. Fuente: Elaboración propia

La actividad se desarrolla a través de la empresa Minsait, una compañía de Indra, dentro de su Delivery Center, área encargada de la ejecución y entrega de servicios tecnológicos de forma estandarizada y a gran escala.

Desde el Delivery Center se asignan equipos multidisciplinarios a distintos proyectos de los diversos mercados en los que opera la compañía, tales como banca, telecomunicaciones, energía y sector público.

En este caso, el equipo de trabajo se encuentra asignado a un proyecto del mercado de telecomunicaciones (Telco), donde se realizan labores de desarrollo y mantenimiento de aplicaciones.

Desde el punto de vista organizativo, el responsable jerárquico a nivel local es Víctor Ríos Campos (Gerente del Delivery Network). La ejecución de las actividades se realiza en coordinación con el equipo del proyecto, siguiendo los lineamientos y la planificación definidos para la entrega del servicio.

Debido a políticas internas de confidencialidad, la compañía no proporciona un organigrama oficial de su estructura local. Por ello, se ha elaborado un organigrama referencial basado en información pública disponible en la página web de Indra Group (IndraGroup, 2026)

3.2 Finalidad y objetivos de la Actividad

3.2.1 Finalidad

Aportar al desarrollo de sistemas que permitan automatizar tareas repetitivas en el ciclo de vida del desarrollo de software, optimizando su tiempo de duración y mitigando en gran medida los errores que puede producir la impericia humana.

3.2.2 Objetivos

3.2.2.1 Objetivo General

Diseñar un sistema de pruebas automatizadas para la autenticación y flujo funcional en la validación de aplicaciones web en una empresa de

telecomunicaciones

3.2.2.2 *Objetivos Específicos*

- Diseñar pruebas automatizadas para la validación de autenticación en servicios web en una empresa de telecomunicaciones.
- Diseñar las pruebas automatizadas para la validación del flujo funcional en servicios web en una empresa de telecomunicaciones.
- Integrar las pruebas de autenticación y flujo funcional para la validación de servicios web en una empresa de telecomunicaciones.

3.3 Problemática

Las compañías de telecomunicaciones hacen uso de aplicaciones web para mejorar la gestión de sus operaciones. Estas herramientas pueden ayudar a cumplir funciones clave como monitoreo de equipos, manejo de nóminas, inventario de materiales o seguimiento de obras. No obstante, para mantener su correcto funcionamiento y responder a la evolución del negocio, estos programas requieren modificaciones, ya sea por incorporación de mejoras, solución de vulnerabilidades, cambios de infraestructura, entre otros factores que pueden afectar su funcionamiento.

Después de realizar cambios en el entorno de producción, es de suma importancia hacer pruebas inmediatamente para asegurar que los cambios no afecten las funcionalidades ya existentes de las aplicaciones. Es común que estas actividades se programen fuera del horario laboral para evitar interrupciones en los servicios, generando así fatiga y estrés en los trabajadores porque deben realizar horas extras para completar estos trabajos.

Dentro de las tareas adicionales que deben asumir los equipos técnicos se encuentran las pruebas y las validaciones posteriores para confirmar si los

cambios pueden permanecer o se necesitan revertir. Sin embargo, estas actividades suelen realizarse de forma manual y repetitiva, lo que incrementa la posibilidad de errores humanos en los resultados de la validación, especialmente en contextos de cansancio o presión. Estas fallas pueden retrasar la implementación de mejoras por algún cambio revertido que se deba reprogramar y a su vez aumenta costos operativos ya que implica movilizar personal y recursos para llevar a cabo los cambios.

Considerando lo expuesto, se propone el diseño e implementación de un sistema de pruebas automatizadas para validar el funcionamiento de las aplicaciones web utilizadas en una empresa de telecomunicaciones con el fin de optimizar procesos de aseguramiento de la calidad del software.

3.3.1 Problema General

¿Cómo diseñar un sistema de pruebas automatizadas para la autenticación y flujo funcional en la validación de servicios web en una empresa de telecomunicaciones?

3.3.2 Problemas Específicos

- ¿Cómo diseñar las pruebas automatizadas para la autenticación en la validación de servicios web en una empresa de telecomunicaciones?
- ¿Cómo diseñar las pruebas automatizadas para el flujo funcional en la validación de servicios web en una empresa de telecomunicaciones?
- ¿Cómo integrar las pruebas de autenticación y flujo funcional en la validación de servicios web en una empresa de telecomunicaciones?

3.3.3 Justificación e importancia de la investigación

Actualmente, muchos equipos de desarrollo aún dependen en gran medida de la intervención humana para la validación, monitoreo y aplicación de cambios en sus sistemas. La transición hacia procesos de integración y despliegue continuos puede ser compleja, pero es posible automatizar ciertas tareas

clave para reducir costos operativos y minimizar errores humanos sin necesidad de una reestructuración completa del flujo de trabajo.

En este marco, el presente trabajo de suficiencia es relevante porque muestra el diseño de un sistema de pruebas automatizadas para la validación de aplicaciones web en una empresa de telecomunicaciones, ofreciendo una mejora en los procesos de validación de cambios, principalmente al acelerar los tiempos de ejecución; minimizando la dependencia de infraestructura externa y asegurando un entorno controlado para las pruebas funcionales definidas.

3.4 Metodología y Procedimiento

3.4.1 Metodología

3.4.1.1 Bases teóricas

- Bons et al. (2023) analizaron las pruebas automatizadas en interfaces gráficas desde los enfoques scripted y scriptless. Para ello, implementaron pruebas utilizando Selenium, como herramienta de generación de scripts, y Testar, para el segundo enfoque. Ambas herramientas se aplicaron sobre el sistema Yoobi de la empresa E-Dynamics. El objetivo de este artículo fue explorar los beneficios complementarios de combinar el scriptless testing con Testar y el scripting con Selenium dentro de la compañía mencionada que, hasta ese momento, solo realizaba pruebas manuales. Se evaluaron la efectividad, eficiencia y satisfacción de los participantes con ambos enfoques. Los resultados mostraron que los scripts montados con Selenium son necesarios para probar si los escenarios de prueba comunes están bien implementados puesto que permite reproducirlos dentro de un script de pruebas específico para cada uno, mientras que Testar destacó por su capacidad para detectar fallos visuales y comportamientos inesperados, gracias a su exploración autónoma de la interfaz. El estudio concluyó que las pruebas scripted y scriptless son

complementarias, y que su uso combinado permite una mayor cobertura y detección de fallos, permitiendo optimizar procesos manuales existentes dentro de un entorno industrial.

- Leotta et al. (2023) analizaron los principales desafíos que enfrentan los equipos al implementar soluciones de automatización de pruebas con Selenium en entornos reales de desarrollo de software. El objetivo del estudio fue identificar y comprender los retos del testing end-to-end con Selenium, así como presentar las estrategias, técnicas y enfoques adoptados para superarlos de manera efectiva. La investigación se basó en una encuesta hecha a 78 profesionales con experiencia en pruebas punto a punto con Selenium, contactados vía online a través de diferentes redes sociales. Los resultados revelaron trece problemas comunes con los cuales los participantes estuvieron de acuerdo y ordenaron por importancia, siendo los tres más relevantes el manejo de contenido asíncrono, la fragilidad de los scripts y su comportamiento impredecible en ciertas ocasiones. Entre las soluciones más efectivas identificadas destacan la implementación de esperas dentro de los scripts para interactuar con el contenido una vez cargue, coordinar con equipos de desarrollo y calidad para que los cambios no afecten la ubicación de elementos clave, y también hacer reintentos para cubrir fallos inesperados. El estudio concluye que su importancia recae en darle herramientas al lector para afrontar los problemas mencionados utilizando estrategias aplicadas en la industria y avaladas por los profesionales encuestados.
- Jimenez Vargas & Fartolino Guerrero (2023), en su trabajo de suficiencia profesional, tuvieron como principal objetivo disminuir la duración de la ejecución de pruebas hasta en un 50% a través de la automatización. Para lograrlo, analizaron herramientas y marcos de trabajo de código abierto, siendo estas las bases de la arquitectura lógica de la solución. Además, analizaron la arquitectura física que presenta la empresa para poder implementar un flujo de pruebas integrado, teniendo en cuenta los entornos disponibles para realizar la automatización, sin afectar los ambientes de producción. Finalmente,

validaron los resultados mediante una encuesta a especialistas del área de TI. El trabajo logra cumplir con el objetivo planteado inicialmente indicando que superaron la reducción de tiempos de ejecución en más del 50%, consiguiendo implementar una arquitectura escalable y flexible, habiendo seleccionado estratégicamente las herramientas que reflejaron una mejora en la eficiencia y la calidad del software en la empresa. Concluyen también que la automatización de pruebas es una mejora significativa al proceso manual que se venía utilizando en el entorno bancario para la validación de la calidad del software, calificando esta solución implementada como factible y provechosa.

- Delgado (2024), en su trabajo de suficiencia profesional, aborda la optimización de los tiempos de ejecución de pruebas y la mejora de la calidad del software en una entidad bancaria. El objetivo principal del autor fue implementar pruebas automatizadas utilizando la metodología de trabajo Scrum y la herramienta de automatización Selenium, además de apegarse a los estándares que propone el ISTQB, desarrollando así un marco de trabajo de automatización con el cual pueda optimizar la ejecución de pruebas que hasta el momento el banco hacía de forma manual. Durante el desarrollo el autor crea sus propios casos de prueba a partir de historias de usuario, estos casos son automatizados y a través de un proceso iterativo de programación, ejecución, evaluación de resultados y retroalimentación va incorporando mejoras y afinando la automatización. Finalmente se llega a la conclusión que la automatización de pruebas ofrece una mejora en tiempo y eficiencia en contraste con las ejecuciones manuales. Además, la metodología planteada permitió ajustarse de manera rápida a los cambios presentados durante la implementación de la solución y el apego a los estándares ISQTB aseguró un nivel alto de fiabilidad que cumplió con las expectativas de la entidad donde se desarrolló la experiencia

3.4.1.2 Marco conceptual

Metodologías de desarrollo de software

La metodología es el procedimiento establecido en el desarrollo de software que describe los pasos necesarios a seguir para lograr un proyecto exitoso. Según Delgado Olivera & Díaz Alonso (2021), su importancia radica en asegurar sistemas eficientes y bien documentados, garantizando la entrega a tiempo de algún desarrollo.

Las metodologías se desarrollan entonces, con el objetivo de resolver problemas en el proceso de desarrollo de software. Narang & Mittal (2022) mencionan que algunos de estos desafíos son entregas fuera de la fecha límite y costos por encima del presupuesto inicial. Mientras que, Delgado Olivera & Díaz Alonso (2021) también destacan cambios costosos y expectativas incumplidas a usuarios.

Es por esto por lo que, para afrontar los retos mencionados, a lo largo del tiempo los expertos han ido creando y afinando las metodologías propuestas y sus aplicaciones. Una de las primeras en aparecer son las metodologías tradicionales, que buscan llevar la gestión de proyectos al desarrollo de software, estableciendo etapas muy bien definidas. No obstante, Santos et al. (2025) afirman que son muy rígidas puesto que no se adaptan a los cambios rápidos en el mercado ni de los usuarios.

Ante estas limitaciones surgen las metodologías ágiles que introducen, según Morales-Carrillo et al. (2022), métodos iterativos y de poca duración que buscan proporcionar piezas pequeñas de software en poco tiempo para cumplir con las solicitudes de los usuarios.

Finalmente, la metodología DevOps incorpora la automatización de procesos de software y, según Narang & Mittal (2022), reduce la brecha de comunicación entre los equipos de desarrollo y operaciones, una de las principales dificultades de las metodologías ágiles.

a. Metodologías tradicionales

Son las primeras metodologías propuestas y, como se mencionó anteriormente, tienen su base en los principios de la gestión de proyectos.

Morales-Carrillo et al. (2022) refuerzan esta idea al afirmar que se basan en un control de procesos definiendo las actividades involucradas, el alcance, los entregables, las herramientas y el tipo de seguimiento a utilizar en su aplicación.

A pesar del auge de las nuevas tendencias, no han sido desplazadas completamente ya que su fortaleza recae en la gestión integral del proyecto. Por esta razón, son una opción válida para desarrollos grandes con requerimientos bien definidos. Narang & Mittal (2022) destacan que su enfoque secuencial permite manejar proyectos a gran escala de manera más exitosa. Además, Morales-Carrillo et al. (2022) señalan que su aplicación es más común para organizaciones con altos ingresos, gran cantidad de empleados y que gestionan proyectos con altos presupuestos y de criticidad alta.

b. Modelo en cascada

Es un modelo secuencial y cíclico. No termina una vez llegado a la última etapa, al contrario, vuelve a evaluar etapas anteriores según las necesidades del software desarrollado.

Según Delgado Olivera & Díaz Alonso (2021), el modelo en cascada consta de cinco fases: definición de requisitos, diseño de software, implementación y pruebas unitarias, integración y pruebas del sistema, y operación y mantenimiento. En la última etapa, además de corregir errores descubiertos, se implementan mejoras y se identifican nuevos requerimientos por parte del usuario.

Pruebas de software

Las pruebas de software son un conjunto de actividades destinadas a evaluar la calidad de una aplicación en distintas fases del ciclo de vida del desarrollo. También son un método rentable para reducir defectos (ISTQB, 2023). El objetivo principal de las pruebas es identificar fallos o errores y confirmar que el software es apto para su uso (Umar, 2020).

Su correcta puesta en práctica asegura el buen desempeño del software, ya que lo mantiene en un proceso de mejora continua porque a medida que se encuentran fallos, es posible corregirlos durante las etapas de desarrollo o mantenimiento del sistema. Como consecuencia, se reducen costos al identificar y corregir errores lo más temprano posible. Además, garantizan la seguridad de la información del usuario (Kumar, 2023).

a. Tipos de pruebas

Los tipos de pruebas son grupos de técnicas enfocadas en examinar características específicas de la calidad del software (ISTQB, 2023).

En la literatura existen diferentes categorizaciones para tipos de pruebas de software, pero a continuación se mencionan las más comunes:

- **Pruebas funcionales**

Evalúan las funciones del software, que se definen como lo que el objeto de prueba debería hacer (ISTQB, 2023). Esta evaluación valida que el programa cumpla con los requerimientos funcionales específicos (Olaoye & Idowu, 2024).

- **Pruebas de rendimiento**

Evalúan la capacidad de respuesta y estabilidad del software bajo una carga de trabajo particular (Umar, 2020).

- **Pruebas de seguridad**

Identifican vulnerabilidades y riesgos de seguridad (Olaoye & Idowu, 2024), por lo que protegen los datos y mantienen la funcionalidad del software (Umar, 2020).

- **Pruebas de usabilidad**

Evalúan qué tan fácil de usar es el software desarrollado (Umar, 2020)

b. Técnicas para pruebas de software

Las técnicas de prueba constituyen un apoyo fundamental para los

desarrolladores, ya que permiten analizar y diseñar los casos de prueba de manera sistemática (ISTQB, 2023). Según Umar (2020), estas técnicas proporcionan estrategias que permiten incrementar la cobertura de las pruebas; no obstante, alcanzar una cobertura del 100% resulta imposible. Por ello, su adecuada aplicación busca lograr la mayor efectividad posible en la validación del software. En la Figura 2 se presenta una clasificación de las técnicas de prueba más utilizadas con este propósito.

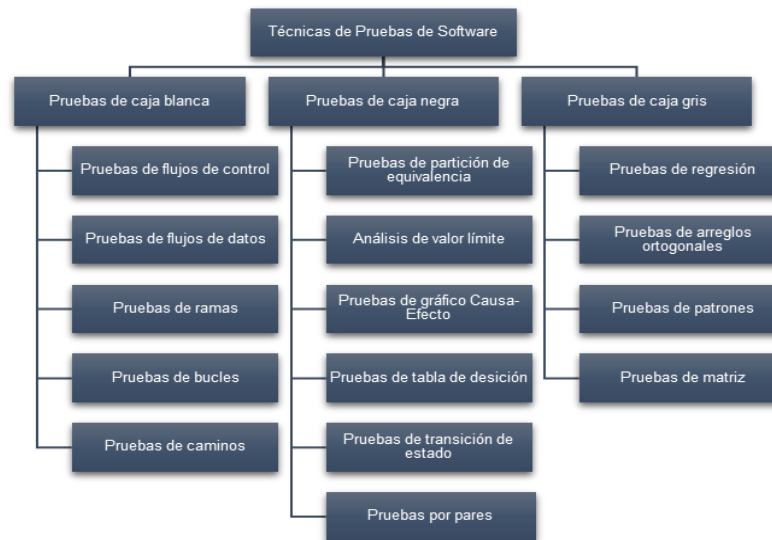


Figura 2 Técnicas para pruebas de software. Fuente: Adaptado de Umar (2020)

- **Técnicas de caja negra**

Técnica en la cual la estructura interna del software no es conocida por el evaluador (Umar, 2020). Por lo tanto, los casos de prueba son independientes de cómo ha sido implementado el programa o aplicación (ISTQB, 2023). Las pruebas basadas en estas técnicas se encargan de validar la salida esperada de acuerdo con una entrada conocida.

- **Técnicas de caja blanca**

Técnicas en las que sí se conoce la implementación interna del software. Para aplicar estas técnicas es necesario el conocimiento del código fuente porque los casos de prueba se generan a partir de una visión interna del sistema y las habilidades de programación del evaluador (Umar, 2020). Por lo tanto, estos casos de prueba solo pueden ser creados después del diseño o

implementación del objeto de prueba (ISTQB, 2023).

- Técnicas de caja gris

Combinan la técnica sencilla de prueba de caja negra con los sistemas dirigidos por código en la prueba de caja blanca (Umar, 2020), puesto que, si bien los casos de prueba están enfocados en probar funcionalidad, se guía su construcción en base al conocimiento parcial de la implementación interna.

c. Casos de prueba

Un caso de prueba determina el éxito y madurez de las pruebas. Es el conjunto de tres elementos: entrada, estado del programa y resultado esperado. (Mukherjee & Patnaik, 2021). Pueden derivar de historias de usuarios, casos de uso, requerimientos de negocio, criterios de rendimiento, políticas de seguridad y estándares de experiencia de usuario (Olaoye & Idowu, 2024).

Pruebas en aplicaciones Web

Una aplicación web es un programa accesible mediante un navegador que genera contenido de manera dinámica basado en las interacciones del usuario (Li et al., 2025).

Este tipo de aplicaciones se implementan en un servidor, y su despliegue o actualizaciones se realizan en un único conjunto de máquinas servidoras. Su arquitectura se basa en tecnologías web y el protocolo HTTP requiriendo conexión a Internet para su funcionamiento, pero ofreciendo disponibilidad desde cualquier ubicación al estar alojadas en servidores (Bubenko et al., 2023).

Las aplicaciones web modernas son complejas y se utilizan para tareas de máxima importancia (Leotta et al., 2023), por lo que las pruebas juegan un papel vital en garantizar la funcionalidad, seguridad y fiabilidad de estas (Li et al., 2025). Estas aplicaciones pueden tener componentes frontend con integración a servicios backend que requieren pruebas rigurosas, ya que

funciones como autenticación, pagos y actualizaciones en tiempo real pueden

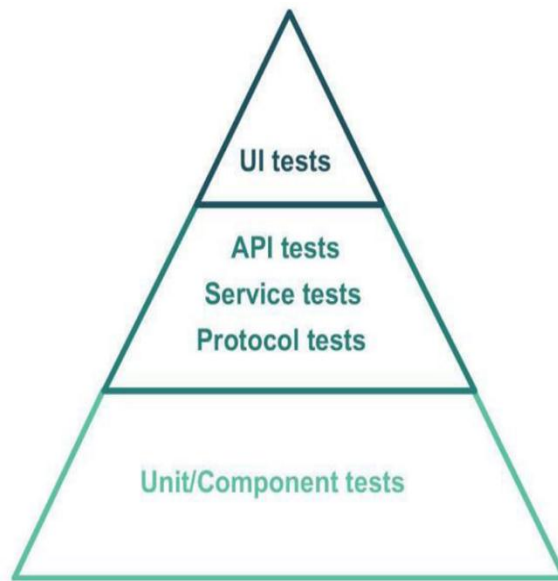


Figura 3 Pirámide de pruebas en aplicaciones web. Fuente: Delgado (2024)

afectar la experiencia del usuario y las operaciones comerciales si no se validan correctamente (Talakola, 2024).

Existen diferentes niveles de prueba que se pueden hacer en aplicaciones web. Estas pueden abarcar la interfaz gráfica de usuario (GUI), APIs, servicios o protocolos (Delgado, 2024).

De la figura 3, podemos identificar que los niveles más bajos se vinculan estrechamente con el código o la implementación, lo que presupone el uso de técnicas de caja blanca para la materialización de las pruebas en estos niveles. A medida que vamos subiendo de escalón, el conocimiento de la implementación no resulta tan necesario, más allá del conocimiento del funcionamiento final, lo que permite aplicar técnicas de caja negra, por ejemplo, para evaluar las funcionalidades de la interfaz de usuario.

a. Pruebas de autenticación

Las pruebas de autenticación tienen como objetivo validar que el proceso de inicio de sesión funcione sin errores. Esta verificación es crucial, puesto que asegura la fiabilidad de una funcionalidad que impacta la experiencia del usuario y la seguridad general del sistema (Zhou et al., 2025).

Para cumplir con este propósito, es necesario realizar pruebas específicas que verifiquen elementos clave como la autenticación multifactor, el tiempo de la sesión activa o que luego de un intento de inicio de sesión, el programa se comporte según lo esperado dando acceso a los recursos si es correcto o bloqueando el acceso si es incorrecto (Talakola, 2024). Estas comprobaciones buscan confirmar que el acceso esté limitado solo a los usuarios correctos y que la aplicación refleje correctamente su estado autenticado.

Es necesario conocer el proceso de autenticación dentro de una aplicación web, para poder probarlo. Zhou et al. (2025) dividen este proceso en tres fases: pre-login, login y post-login. La primera, comprueba si existen sesiones activas para un usuario específico y si no se encuentran se solicitan sus credenciales. En la fase de login, se validan las credenciales ingresadas por el usuario. Finalmente, se otorgan los permisos y se habilita el acceso a los recursos de la aplicación disponibles para el usuario.

Conocer estas etapas permite establecer pruebas y enfoques específicos para los puntos críticos del proceso. En concreto, la etapa de post-login se puede abordar con pruebas de tipo funcional, ya que un inicio correcto de sesión se puede verificar con la carga de los recursos específicos. Talakola (2024) resalta el post-login como una etapa crítica del proceso de autenticación al porque los errores en el manejo de sesiones o inconsistencias en el estado del sistema pueden comprometer aspectos como la funcionalidad, la seguridad y, por lo tanto, la confiabilidad de la aplicación.

En resumen, las pruebas de autenticación no solo deben centrarse en verificar el ingreso de credenciales correctas, también se debe verificar que el programa, tras darle acceso al usuario, responda de manera coherente y segura, lo que hace de esta funcionalidad un componente esencial en la validación de cualquier aplicación web.

b. Pruebas de flujo funcional

Las pruebas de flujo funcional, puesto que se realizan desde la Interfaz Gráfica de Usuario (GUI), también llamadas pruebas GUI, se ejecutan interactuando

con los elementos disponibles (botones, inputs, etc.) en la interfaz de usuario (Bons et al., 2023).

Su objetivo principal es asegurar que la interfaz permanezca estable en términos de experiencia y de funcionalidad en los dispositivos y navegadores para los que está destinada la aplicación (Li et al., 2025). Estas pruebas permiten validar que la lógica de negocio definida se muestre correctamente en el flujo de la interfaz puesta a prueba y que los elementos tengan consistencia visual (Nass et al., 2021).

Una prueba GUI se compone de un flujo determinado, conformado por eventos de forma secuencial, es decir, interacciones del usuario con la interfaz (como hacer clic, rellenar formularios, etc.), y de criterios de validación que permiten verificar el estado esperado de la interfaz tras cada acción (Mariani et al., 2021).

No obstante, este tipo de pruebas suele enfrentarse a diversas dificultades, especialmente debido al carácter dinámico de las aplicaciones web. En muchos casos, los componentes de la interfaz se generan de forma automática a partir del framework utilizado en el desarrollo, lo que complica su identificación y obliga al profesional responsable del proceso de pruebas a realizar un análisis detallado antes de poder definir y ejecutar adecuadamente los casos correspondientes. También, la propia ejecución de una prueba puede modificar el estado del sistema, ya que puede establecer condiciones previas que influyen en las pruebas posteriores y por lo tanto es necesario establecer un orden específico de ejecución (Nass et al., 2021).

En síntesis, las pruebas sobre interfaces gráficas resultan fundamentales para asegurar una experiencia de usuario sólida y adecuada, en particular en entornos complejos y cambiantes, donde fallos en la interfaz pueden afectar de manera significativa la usabilidad y la confiabilidad del sistema.

Automatización de pruebas de software

La automatización de pruebas de software consiste en la ejecución de casos de prueba mediante el uso de herramientas especializadas o scripts, con una intervención manual mínima o inexistente (Wang et al., 2025). Esta práctica

permite optimizar el proceso de validación del software, ya que acelera la identificación de errores y disminuye el esfuerzo operativo que implican las pruebas realizadas de forma manual.

En consecuencia, su aplicación contribuye a una gestión más eficiente del tiempo y los recursos del proyecto, al tiempo que incrementa la precisión, la cobertura y la repetibilidad del proceso de pruebas (Mustafa & Handayani, 2024). Sin embargo, a pesar de sus ventajas, la automatización no reemplaza totalmente la intervención humana, ya que sigue siendo necesaria para la creación, diseño y mantenimiento de las pruebas, así como para la configuración de su ejecución (Ateşoğulları & Mishra, 2020).

Por lo tanto, aunque la automatización supone una mejora significativa en la eficiencia de la ejecución de pruebas, no sustituye por completo el trabajo manual, sino que lo complementa.

a. Herramientas de automatización

Son instrumentos que se utilizan para automatizar el proceso de pruebas de software desde la creación y ejecución, hasta el análisis de resultados (Bons et al., 2023). Existen muchas herramientas presentes en la literatura. A continuación, se presentan algunas de las más comunes:

- **Selenium**

Selenium es una de las más utilizadas para automatización de pruebas en aplicaciones web. Soporta múltiples lenguajes de programación (Java, Python, C#, entre otros) y navegadores, y es completamente gratuita, lo que la hace ideal para proyectos con presupuestos limitados (Fatima et al., 2023; Majeed et al., 2021).

Requiere habilidades de programación para su implementación efectiva (Fatima et al., 2023).

- **TestComplete**

TestComplete es una herramienta comercial que ofrece una experiencia accesible a usuarios no técnicos. Incluye soporte para grabación y reproducción de pruebas, integración con múltiples lenguajes de scripting y

una interfaz gráfica más intuitiva (Tibell & Kholi, 2023).

Aunque muestra una curva de aprendizaje menor comparada con otras herramientas, puede presentar tiempos de ejecución altos y una cantidad de falsos positivos considerable (Tibell & Kholi, 2023).

- Katalon Studio

Katalon Studio combina lo mejor de Selenium con una interfaz gráfica amigable. Permite grabar pruebas, modificarlas con lenguaje Groovy o Java, y ejecutar pruebas de manera sencilla. También admite pruebas web, móviles, y API (Majeed et al., 2021).

- TestProject

TestProject es una herramienta open source moderna que admite pruebas para aplicaciones web y móviles, sin requerir instalaciones complejas. Permite a usuarios sin conocimientos técnicos grabar y ejecutar pruebas, incluyendo funciones como inspección de elementos, generación de scripts y colaboración en la nube (Majeed et al., 2021).

En la Tabla 1, se realiza una comparación de las herramientas de automatización descritas anteriormente.

Tabla 1 Comparación de herramientas de automatización de pruebas

Herramienta	Licencia	Lenguajes soportados	Plataformas	Grabación y reproducción	Facilidad de uso	Ideal para	Fuente
Selenium	Open Source	Java, Python, C#, JavaScript, etc.	Solo Web	Parcial (solo en IDE)	Baja (requiere programación)	Desarrolladores técnicos	Tibell & Kholi, 2023; Majeed et al., 2021
TestComplete	Comercial	JavaScript, Python, VBScript	Web, Desktop	Sí	Alta (interfaz intuitiva)	Testers novatos	Tibell & Kholi, 2023
Katalon Studio	Open Source	Groovy, Java	Web, móvil, API	Sí	Media (GUI + scripting)	QA semi-técnicos	Majeed et al., 2021
TestProject	Open Source	Java, C#	Web, móvil	Sí	Alta (muy amigable)	QA sin experiencia técnica	Majeed et al., 2021
QTP / UFT	Comercial	VBScript	Windows, Desktop	Sí	Alta (no requiere programación)	Empresas con presupuesto elevado	Fatima et al., 2023

Fuente. Elaboración propia

b. Enfoques de implementación de automatización de pruebas

Se refieren a la forma de creación y mantenimiento de los casos de prueba automatizados. En la literatura revisada encontramos dos enfoques:

- Automatización de pruebas basadas en scripts (Scripted Test Automation)

Enfoque en el que los validadores crean scripts manualmente por cada una de la secuencia de prueba deseadas antes de la ejecución de los casos de prueba, los cuales se pueden utilizar para automatizar la ejecución de las pruebas (Bons et al., 2023).

Ofrece flexibilidad al poder implementar la orquestación de las pruebas de manera personalizada, lo cual permite alcanzar un alto grado de precisión en escenarios end-to-end (Choubey, 2025).

Sin embargo, requiere conocimientos técnicos avanzados y los scripts suelen ser frágiles frente a cambios en la interfaz gráfica o los requisitos del sistema (Mohan & Kumar, 2021).

- Automatización de pruebas sin scripts (Scriptless Test Automation)

Enfoque que requiere del uso de herramientas con las cuales se elimina la necesidad de programación, ya que estas generan automáticamente las secuencias de prueba mediante la detección de elementos de la interfaz y la selección de acciones posibles (Bons et al., 2023).

Al no tener que programar, reduce la curva de aprendizaje para nuevos validadores que no tengan habilidades o experiencia previa. Pero por otro lado, presentan limitaciones en escenarios de alta complejidad o que requieren un control exhaustivo, así como cierta dependencia de las capacidades específicas de cada herramienta (Choubey, 2025).

c. Marcos de trabajo para automatización de pruebas

Ateşoğulları & Mishra (2020) recogen los siguientes enfoques de desarrollo

para pruebas automatizadas en su trabajo:

- Marco de Pruebas Modular

Basado en la programación orientada a objetos. Se separa la aplicación en un conjunto de módulos. Por cada módulo se crea un script y finalmente cuando se agrupan todos de manera secuencial se forma un flujo más complejo representado por un script más largo.

- Marco de pruebas basado en datos

Se utilizan diferentes combinaciones de datos en una sola prueba, lo que evita la redundancia. Finalmente se comparan los resultados obtenidos con los esperados.

- Marco de pruebas basado en Palabras clave

Se utilizan palabras clave para referenciar acciones, las cuales son mapeadas a través de un script que ejecuta una prueba específica por cada palabra clave.

Lenguajes de programación en la automatización de pruebas

Un lenguaje de programación es un conjunto de normas que permiten expresar instrucciones a un computador mediante código, el cual es interpretado o compilado para ejecutar tareas específicas (Gallego Izquierdo, 2024).

La elección del lenguaje para desarrollar scripts de automatización resulta crucial, ya que condiciona la selección de la herramienta y el enfoque de automatización basado en scripts (Dalampis, 2021).

En el Anexo C se presentan los lenguajes de programación más populares según la encuesta de Stack Overflow (2025). En ella, JavaScript ocupa el primer lugar, Python el cuarto y Java el séptimo. (Los puestos intermedios faltantes no se consideran ya que son lenguajes de marcado o de scripting). Tal como se evidencia en la Tabla 1, estos tres lenguajes destacan por su uso en la automatización de pruebas a través de diversas herramientas.

La Tabla 2 muestra una comparativa entre los 3 lenguajes

Tabla 2 Comparación lenguajes de programación más utilizados

Característica	Java	Python	JavaScript
Paradigma y tipo	Lenguaje compilado, orientado a objetos y con tipado fuerte; altamente portable gracias a la JVM (Cuellar-Juarez & Mamani-Vilca, 2024).	Interpretado, multiparadigma y de tipado dinámico; destaca por su sintaxis clara y legible (Cuellar-Juarez & Mamani-Vilca, 2024).	Interpretado, multiparadigma y dinámico; pieza clave en frontend y backend con Node.js (Gallego Izquierdo, 2023).
Facilidad de aprendizaje	Sintaxis más compleja pero sólida en fundamentos de POO; recomendada para formación estructurada (Oracle, 2020).	De fácil aprendizaje y rápida implementación, ideal para principiantes (Oracle, 2020).	Requiere aprendizaje progresivo en asincronía y DOM, aunque es ampliamente adoptado (Gallego Izquierdo, 2023).
Rendimiento	Más eficiente en aplicaciones de gran escala y alto rendimiento (Cuellar-Juarez & Mamani-Vilca, 2024).	Menos veloz en ejecución, pero rápido en desarrollo y despliegue (Oracle, 2020).	Rendimiento intermedio, dependiente de V8; menor en cálculos intensivos (Gallego Izquierdo, 2023).
Usos principales	Aplicaciones empresariales, móviles (Android) y sistemas críticos (Cuellar-Juarez & Mamani-Vilca, 2024).	Desarrollo web con Django/Flask, ciencia de datos, IA y prototipado ágil (Cuellar-Juarez & Mamani-Vilca, 2024).	Desarrollo web frontend, APIs backend con Node.js y aplicaciones en tiempo real (Gallego Izquierdo, 2023).
Ecosistema y librerías	Ecosistema amplio en múltiples áreas, incluidas librerías para IA/ML (Oracle, 2020).	Gran ecosistema en ciencia de datos y web, con librerías como NumPy y TensorFlow (Cuellar-Juarez & Mamani-Vilca, 2024).	Ecosistema diverso para frontend (React, Angular, Vue) y backend (Express, Next.js) (Gallego Izquierdo, 2023).
Escalabilidad	Altamente escalable en entornos empresariales complejos (Cuellar-Juarez & Mamani-Vilca, 2024).	Escalable en proyectos pequeños y medianos, pero menos robusto en alta carga (Cuellar-Juarez & Mamani-Vilca, 2024).	Muy escalable en arquitecturas distribuidas como microservicios y serverless (Gallego Izquierdo, 2023).

Fuente. Elaboración propia.

3.4.2 Procedimiento

3.4.2.1 Contexto de la situación actual

El proyecto tiene lugar dentro de un equipo de desarrollo encargado del mantenimiento de aplicaciones de una empresa de telecomunicaciones extranjera. El equipo tiene a su cargo más de 10 aplicaciones web, las cuales

se encuentran en constante mejora y monitorización.

Al realizar cambios dentro de las aplicaciones mantenidas, es necesario realizar pruebas para validar que estas no hayan sufrido daños y que los cambios estén correctamente integrados.

Añadido a lo anteriormente descrito, existen cambios dentro de la infraestructura que las pueden afectar. Estos cambios son los más críticos, puesto que afectan simultáneamente a varias de estas aplicaciones.

Dependiendo de la posible afectación de los cambios, se acordó en el proyecto que se pueden realizar dos tipos de pruebas: rápidas y completas. Las pruebas rápidas se concentran en comprobar el acceso a la aplicación y la correcta carga de la página de inicio. Mientras las pruebas completas validan que tanto la autenticación como el flujo funcional del sistema operen de manera adecuada. Además, finalizando las pruebas es necesario enviar un correo al equipo encargado con un documento que contenga evidencias de las pruebas realizadas con capturas de pantalla y el estado de cada una. Finalmente, tomamos de referencia que toda la actividad relacionada con validación de pruebas rápidas ya descrita, dura en promedio 10 minutos por prueba, mientras que las pruebas completas duran un poco más del doble, en promedio 25min.

Esta es la razón por la que se propuso, como proyecto interno, un sistema que permita automatizar el proceso de validación, permitiendo reducir tiempos en la ejecución de pruebas y la generación del documento con las evidencias respectivas.

3.4.2.2 Stack tecnológico

Las tecnologías seleccionadas para el desarrollo del proyecto incluyen:

- Lenguaje de programación:
 - **Python:** Se selecciona Python como lenguaje de programación debido a su sintaxis sencilla, clara y legible, lo que facilita la creación y mantenimiento de scripts de prueba. Además, favorece el prototipado ágil y un desarrollo más rápido
- Herramientas de automatización:

- **Selenium:** De acuerdo con el análisis comparativo presentado en el marco conceptual de herramientas de automatización, se selecciona Selenium al ser una herramienta Open Source y estar enfocada a la automatización de aplicaciones Web, además de su compatibilidad con el lenguaje de programación elegido.
- Librerías adicionales:

Ya elegidos el lenguaje de programación y la herramienta de automatización, se seleccionan algunas librerías necesarias para la generación de reportes y la interfaz gráfica del sistema, relacionadas con el lenguaje seleccionado.

 - Tkinter, para la interfaz gráfica de la aplicación de escritorio.
 - Openpyxl, para la generación y manejo de archivos Excel.
 - Pillow, para gestionar las capturas de pantalla como evidencia del resultado de las pruebas.
 - Cryptography, para gestionar encriptación y desencriptación de credenciales.
- Navegador web
 - Firefox, por su compatibilidad con aplicaciones web antiguas.

3.4.2.3 Definición del alcance

Durante el desarrollo se identificaron las siguientes restricciones y desafíos:

- **Alojamiento de la aplicación:** Inicialmente, se propuso desplegar la solución como una aplicación web en una máquina disponible dentro de la red del proyecto. Sin embargo, dado que las pruebas automatizadas requieren un navegador instalado, el equipo de infraestructura denegó esta opción por razones de seguridad.
- **Uso de contenedores:** Se evaluó la posibilidad de dockerizar la aplicación para incluir el navegador dentro de la imagen y ejecutarla en un contenedor. No obstante, esta alternativa fue descartada, ya que las máquinas disponibles estaban dedicadas exclusivamente a tecnologías

específicas.

- **Infraestructura dedicada:** Se consideró la adquisición de un servidor exclusivo para el despliegue de la aplicación, pero el costo debía ser asumido por el proyecto, lo que no era viable dentro del presupuesto asignado.

Como solución a estas restricciones, se optó por un enfoque de ejecución local, en el que cada desarrollador del equipo puede replicar el entorno de pruebas en su propio ordenador, siguiendo las indicaciones detalladas para configurar correctamente su entorno de desarrollo. Teniendo en cuenta que existe un equipamiento estándar provisto por la organización para los desarrolladores del equipo, las especificaciones de hardware en las que se desarrolló la solución se detallan en el Anexo D. Estas características aseguran un rendimiento fluido durante la ejecución de las pruebas automatizadas.

3.4.2.4 Estructura del proyecto

Debido a las restricciones ya mencionadas, el proyecto se ha diseñado como una aplicación de escritorio. Este consta de dos módulos: la interfaz gráfica (GUI) y el módulo de automatizaciones. Finalmente, los resultados de cada automatización son creados y guardados en una carpeta diferente para que puedan ser revisados por el validador.

En la figura 4 podemos observar el diagrama de la arquitectura del sistema que detalla los módulos y conexiones del desarrollo.

A continuación, se describen los componentes que conforman la arquitectura del sistema

- **Interfaz gráfica (GUI):** Permite la ejecución y gestión de pruebas, proporcionando una forma visual de interactuar con el sistema. Se desarrolló con Python y la librería Tkinter. Además, se convirtió en un ejecutable para que pueda utilizarse como aplicación de escritorio y el usuario no dependa de un IDE con el cual ejecutar las pruebas.

- **Módulo de automatizaciones:** Desarrollado con Python y Selenium, contiene un componente independiente por cada aplicación a automatizar. No se incluyó en el ejecutable, ya que debe tener una estructura dinámica en la que se puedan modificar o añadir componentes. Esto permite poder actualizar la aplicación sin tener que compilarla nuevamente por cada cambio realizado.
- **Resultados:** Los resultados son almacenados en una carpeta dentro del equipo del usuario. Un archivo de log que almacena el registro de la ejecución, y otro de tipo Excel, que recopila los casos de prueba, el estado final de la prueba y la evidencia como captura de pantalla.

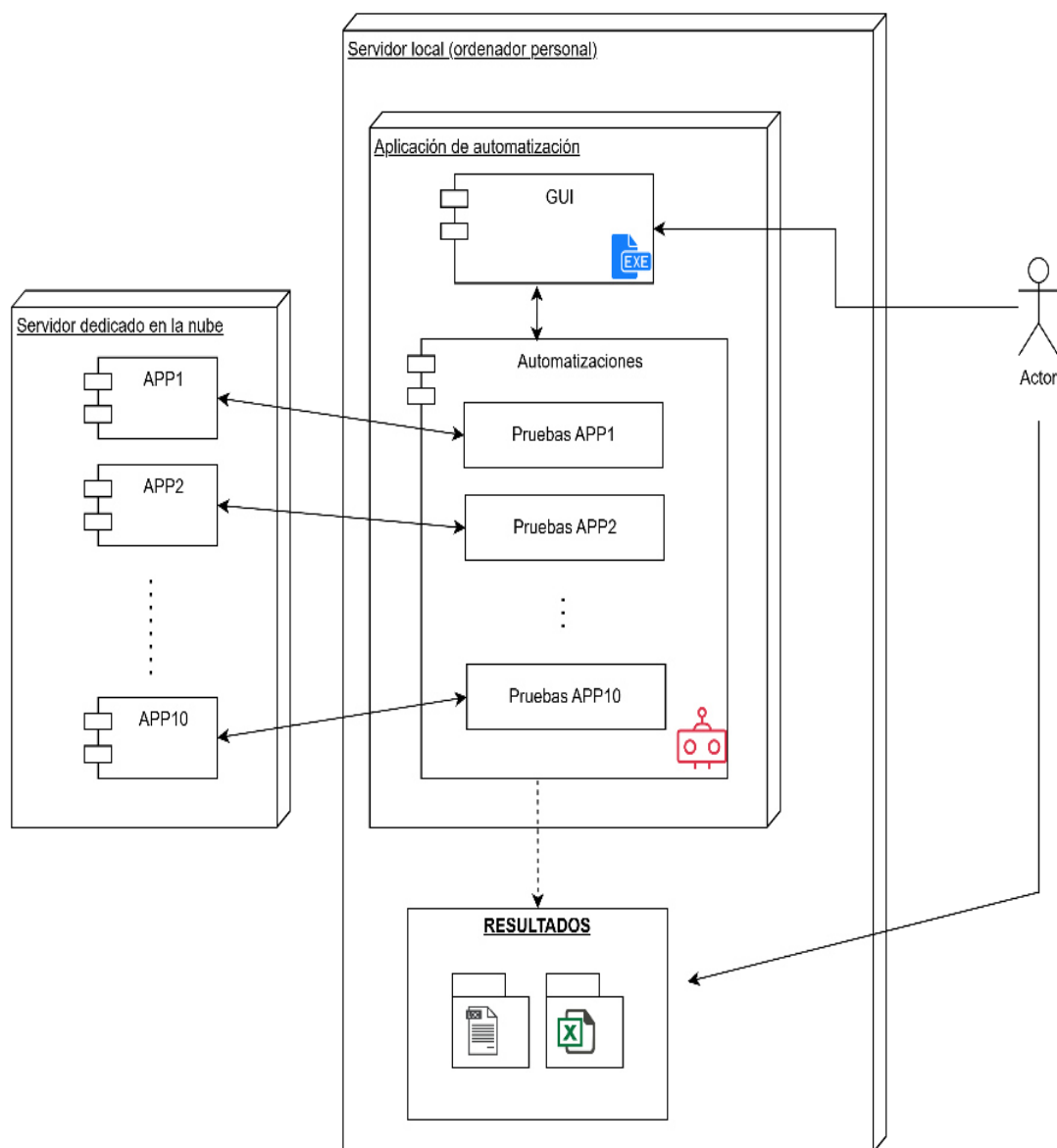


Figura 4 Arquitectura del sistema. Fuente: Elaboración propia.

3.4.2.5 Diseño de la propuesta

Desarrollo de la lógica de negocio

La automatización de pruebas se estructura utilizando clases en Python y aplicando principios de Programación Orientada a Objetos (POO), para mejorar la modularidad y reutilización del código. La lógica del negocio se encuentra dentro del módulo de automatizaciones de la aplicación. A continuación, se muestra un diagrama de componentes:

- **Paquete de utilidades:**

En este paquete se encuentran clases comunes que se pueden reutilizar en los demás módulos.

- Controlador: Puesto que se busca que los scripts de automatización puedan ser creados por los desarrolladores de forma declarativa, se programa una clase controlador donde se contemplan las posibles acciones que se pueden realizar en una interfaz web como puede ser rellenar un campo, dar click a un elemento, validar que un texto aparezca, validar que una tabla tenga contenido, etc. Estas acciones se reflejan en métodos en los cuales se utilizan internamente funciones proporcionadas por Selenium.
- Gestor Excel: Clase con métodos necesarios para generar un archivo Excel y modificarlo para mostrar las validaciones necesarias. Se hace uso de la librería openpyxl en su mayoría.
- Gestor de credenciales: Clase con métodos necesarios para encriptar y desencriptar nombres y contraseña de usuarios para las diferentes aplicaciones. Se hace uso de la librería cryptography
- Gestor de archivos: Clase con métodos necesarios para gestionar el nombre, la lectura y escritura de archivos. Se utiliza la librería os para trabajar con rutas y la librería json para deserializar este tipo de archivos.

- **Paquete de automatización:**

Este paquete se puede repetir tantas veces como aplicaciones se deban probar. Contiene toda la lógica de automatización.

- Plan de pruebas: Esta clase crea un método por cada caso de prueba contemplado de una aplicación. Además, incluye métodos para abrir el navegador, terminar la prueba en caso de errores y cerrar la aplicación una vez terminado el flujo. Un solo caso de prueba puede tener diferentes acciones a realizar en una plataforma web de manera secuencial, por lo que se trata de emular este procedimiento en cada método por medio de la clase Controlador y se utiliza Gestor Excel para registrar los resultados en el archivo de validación.

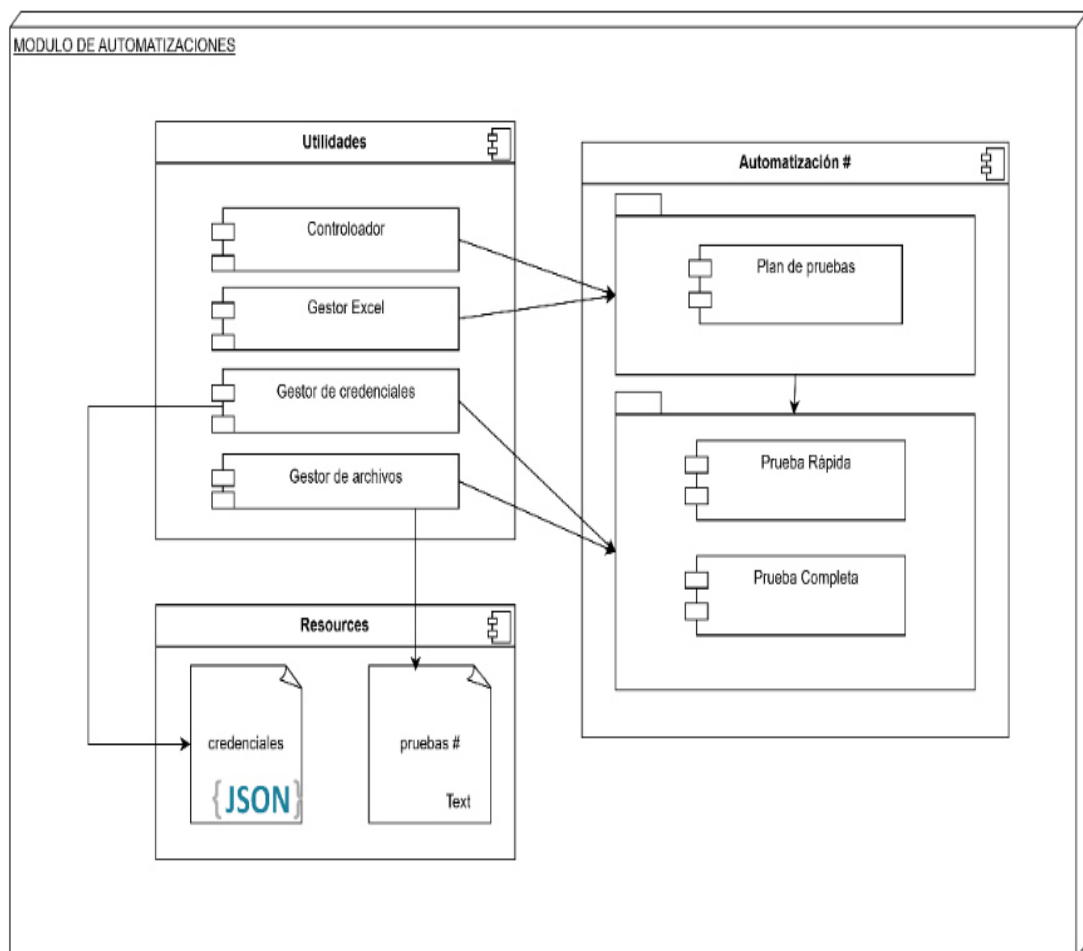


Figura 5 Lógica de negocio. Fuente: Elaboración propia.

- Prueba rápida: Este es un módulo que define una función para poder ejecutar el primer método de la clase Plan de pruebas. Por medio de Gestor de credenciales obtiene el nombre de usuario y contraseña para autenticarse en la aplicación y con Gestor de Archivos obtiene la descripción de la prueba.
- Prueba Completa: Este módulo define la función que orquesta todos los métodos de la clase Plan de pruebas para ejecutarlos secuencialmente. También necesita todas las clases que se utilizan en el módulo anteriormente descrito.

Estrategia y Directrices para la Automatización de Pruebas

Dentro del proyecto ya existía creado un plan de pruebas por aplicación. Estos han sido diseñados en base a la experiencia del desarrollador, donde se han identificado las funciones más críticas de cada sistema y a partir de estas se han establecido los casos de prueba.

Para automatizar los casos de prueba es necesario identificar las acciones a realizar dentro de la página web y los elementos con los cuales se debe interactuar. Selenium es una librería que permite interactuar con la estructura HTML de la página, facilitando la identificación de elementos mediante diferentes atributos, como la etiqueta, el id, la clase, el nombre, el XPATH, entre otros. Además, nos permite controlar el navegador, navegar entre ventanas, abrirlas o cerrarlas, e incluso tomar capturas de pantalla

Con esto en cuenta se han propuesto las siguientes directrices que nos permiten utilizar la librería mencionada para programar cada caso:

- Cada caso de prueba puede tener n acciones por realizar.
- Automatizar una acción requiere diferente cantidad de código dependiendo de su dificultad.
- Si el elemento con el que se desea interactuar no se puede identificar por un atributo específico, se puede obtener una lista de elementos comunes sobre la que podemos iterar hasta dar con elemento necesario para la prueba.
- Para validar el resultado del caso de prueba automatizado

podemos usar diferentes técnicas: verificar la presencia de un texto, a priori inmutable, dentro de algún elemento de la página o verificar que una tabla cargada contenga elementos, dependiendo del tipo de caso de prueba.

Para que las automatizaciones se puedan programar de manera declarativa y sea fácil modificarlas o crear nuevos casos de prueba, se identificaron las acciones más repetitivas que podemos encontrar al querer interactuar con una página web y se convirtieron en métodos, agrupados en una clase específica, en los cuales, dependiendo del tipo de acción.

Las acciones identificadas son las siguientes:

- Rellenar un campo
- Dar click a un elemento
- Escoger algún valor de un Select
- Validar que un elemento tenga un texto específico
- Validar que una tabla tenga elementos
- Cambiar de ventana
- Abrir página de descargas
- Mantener el puntero sobre un elemento

Finalmente, estos métodos son invocados dentro de una función, que representa un caso de prueba, y luego el paquete respectivo las orquesta para generar la prueba automatizada. Se aplica, entonces, un marco de Pruebas modular, descrito anteriormente en las bases teóricas.

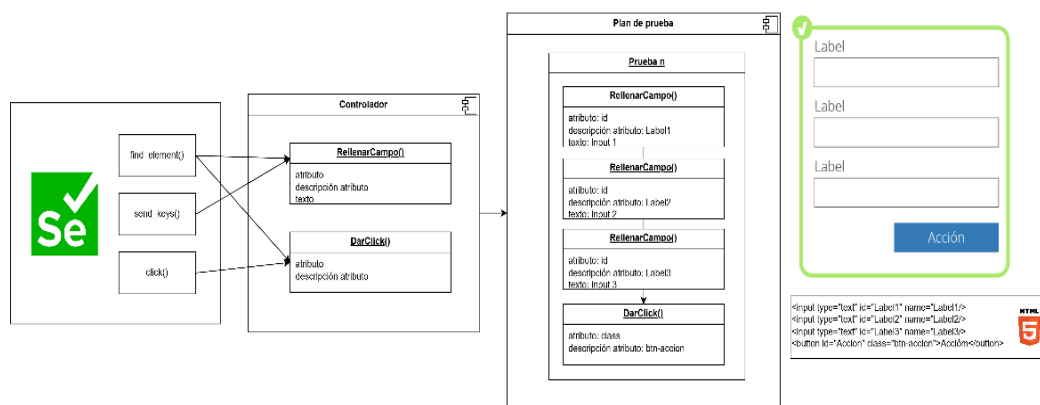


Figura 6 Automatización de caso de prueba. Fuente: Elaboración propia

En la figura 6 se muestra la automatización del envío de un formulario web. Inicialmente, se analiza la estructura HTML de la vista a automatizar. Luego, se proceden a identificar las acciones a realizar. En este caso, hay tres inputs de texto y un botón de envío. Las acciones por realizar, por lo tanto, son rellenar el campo 1, luego el campo 2, seguir con el campo 3 y finalmente dar click al botón Acción.

En Prueba n, se utilizan los métodos de la clase controlador que emulan las acciones ya reconocidas. En esta ocasión, serían `RellenarCampo()` y `DarClick()`, las cuales se valen de funciones propias de Selenium como `find_element()`, `send_keys()`, `click()`, para identificar los elementos e interactuar con estos.

Al utilizar alguno de los métodos elegidos, es necesario indicarle el atributo del elemento a interactuar (atributo), el valor de este atributo (descripción atributo) y, para rellenar un input, también indicarle el texto a ingresar (texto). Todos estos argumentos son obtenidos del análisis hecho inicialmente. En el caso del campo 1 por ejemplo, el atributo elegido es el id, el valor es "Label1" y el texto que se desea ingresar es "Input1". Algo parecido también para los otros campos del formulario. En el caso del botón, el atributo para identificarlo es class y el valor de este es "btn-accion".

Así es como, al ejecutar la "Prueba n" los campos se rellenarán automáticamente con el texto deseado y una vez llenos, el botón será presionado automáticamente sin que el validador haya tenido que realizar alguna otra acción más que enviar a ejecutar el caso de prueba.

Diseño de la automatización de pruebas

La automatización de pruebas se centra en la ejecución de los casos de prueba previamente establecidos en el plan de pruebas del proyecto. Se implementan scripts automatizados para validar el comportamiento de la aplicación de acuerdo con los criterios definidos.

- **Pruebas de autenticación:**

- Objetivo: Automatizar la ejecución de los casos de prueba

relacionados con el ingreso a la aplicación establecidos en el plan de pruebas de cada aplicación.

- Alcance: Validación de inicio de sesión con credenciales y carga correctas de la pantalla de inicio.
- Proceso de automatización:
 1. Preparar y descryptar credenciales desde el archivo de configuración o gestor de secretos.
 2. Simular la introducción de credenciales en los campos de usuario y contraseña.
 3. Enviar el formulario de autenticación.
 4. Capturar y registrar la respuesta del sistema (mensaje, vista cargada, código de estado).
 5. Validar que la vista resultante corresponda a la pantalla inicial mediante la verificación del encabezado esperado.
 6. Generar reportes de ejecución con evidencias (logs, Excel con evidencias).

- **Pruebas de flujo funcional:**

- Objetivo: Automatizar la ejecución de los casos de prueba relacionados con las funcionalidades de la aplicación definidos en cada plan de pruebas.
- Alcance: Validación de la correcta ejecución de funcionalidades clave y navegación dentro de la aplicación
- Funcionalidades clave a validar:
 - Descargar un documento
 - Realizar búsquedas a través de formularios con filtros definidos.
 - Ordenar resultados alfabéticamente o por antigüedad.
 - Navegación entre vistas.
 - Visualización de datos en tablas.
- Proceso de automatización:

1. Simular las acciones del usuario correspondientes a cada caso de prueba (descarga, búsqueda, ordenamiento, navegación).
2. Capturar y registrar la respuesta del sistema para cada acción automatizada.
3. Validar los resultados esperados por cada funcionalidad (archivo descargado correcto, búsqueda filtrada, orden aplicado, datos consistentes en tablas).
4. Generar reportes de ejecución con evidencias (logs, Excel con evidencias).

3.5 Resultado de la actividad

Luego de implementar el sistema de pruebas automatizadas y ejecutarlo en diversas ocasiones dentro del proyecto para el cual fue diseñado, se obtuvieron resultados observables en tres áreas principales: autenticación, flujo funcional e integración end-to-end.

En el caso de las pruebas de autenticación, se registró un tiempo promedio de ejecución de aproximadamente un minuto por aplicación. Este tipo de pruebas, consideradas como pruebas rápidas, mostraron una reducción significativa en el tiempo de validación, pasando de 10 minutos a solo 1 minuto por prueba, lo que representa una disminución del 90%. Durante algunas ejecuciones se presentaron fallos, aunque no de manera recurrente. Estos errores estuvieron relacionados principalmente con el comportamiento del driver de Selenium, especialmente en momentos como la carga inicial de la página o la apertura del navegador. Para reducir el impacto de estos eventos, se incorporó una lógica de reintentos que permite volver a ejecutar automáticamente la prueba en caso de error.

Respecto a las pruebas de flujo funcional, el tiempo de ejecución varió según la complejidad de cada aplicación, con un promedio de 30 segundos por caso de prueba. Los fallos observados en esta categoría se asociaron a tiempos de carga variables en las vistas de las aplicaciones. Para abordar esta situación, se implementaron esperas explícitas que permiten pausar la ejecución

durante un intervalo prudente, asegurando que todos los elementos de la interfaz estén completamente cargados antes de continuar con la validación. Finalmente, al integrar las pruebas en un flujo end-to-end, que incluye tanto la autenticación como los flujos funcionales, se observó que la duración total de cada ejecución depende directamente de la aplicación evaluada. Las ejecuciones más extensas alcanzaron una duración de hasta cinco minutos, mientras que las más breves, correspondientes a aplicaciones con pocos casos de prueba, se completaron en menos de un minuto. En este tipo de pruebas, consideradas como pruebas completas, se registró una reducción del tiempo de validación del 80%, pasando de 25 minutos a 5 minutos por ejecución. Un ejemplo representativo de esta comparación se presenta en el Anexo E, donde se muestran los tiempos de ejecución de pruebas manuales y automatizadas sobre una misma aplicación.

En la figura 7 se muestran gráficos en los que podemos observar los tiempos de ejecución de las pruebas realizadas manualmente en comparación con las pruebas automatizadas.

Cabe aclarar que los resultados presentados en esta sección se calcularon en base a 15 aplicaciones web. A manera de ejemplo, en el Anexo F se muestran los tiempos obtenidos en 5 aplicaciones web.

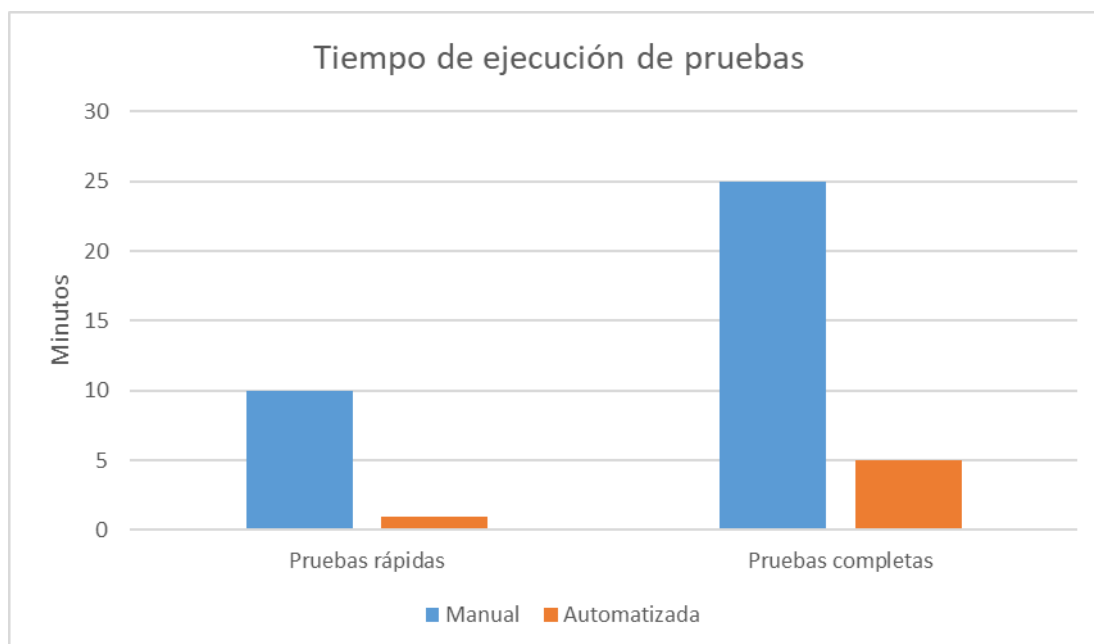


Figura 7 Comparación del tiempo de ejecución de pruebas manuales y automatizadas. Fuente: Elaboración propia.

CAPITULO IV: DISCUSIÓN DE RESULTADOS

4.1 Descripción de la implementación

La implementación del sistema de pruebas automatizadas fue el resultado de un proceso de evolución progresiva, orientado a optimizar su estructura, eficiencia y mantenibilidad. Al igual que Delgado (2024), el proceso de desarrollo fue iterativo, considerando mejoras a medida que se implementaban nuevas automatizaciones. En la etapa inicial, se adoptó un enfoque sencillo consistente en desarrollar un script independiente para cada aplicación web, donde los casos de prueba se programaban de forma secuencial utilizando Selenium y Python. Si bien este método cumplía su propósito funcional, presentaba importantes limitaciones: los scripts resultaban extensos, poco legibles y difíciles de mantener, debido principalmente a la ausencia de una separación clara entre la lógica de ejecución de pruebas y el proceso de validación, que incluía la generación de evidencias en archivos Excel.

Con el objetivo de superar estas restricciones, se procedió a rediseñar la arquitectura del sistema. Los scripts fueron reemplazados por clases, cada una dedicada a una aplicación específica, en las que cada caso de prueba se implementó como un método independiente. Estas clases se integraron en la estructura general del sistema de forma que cada una es utilizada por dos módulos distintos: uno encargado de orquestar las pruebas rápidas y otro de orquestar las pruebas completas para la misma aplicación. Este enfoque permitió mantener un flujo de ejecución más ordenado, escalable y fácil de mantener, adaptándose a distintos niveles de profundidad en la validación.

Durante la optimización de la arquitectura, se identificó la necesidad de reducir la duplicación de código y centralizar las tareas comunes. Para ello, se

incorporaron componentes que permitieron unificar funciones recurrentes y concentrar en un solo módulo la generación de evidencias. Esta reorganización mejoró la claridad del código y facilitó que nuevos desarrolladores pudieran integrarse rápidamente al proyecto.

En la fase final, se desarrolló una interfaz gráfica que actúa como centro de control del sistema. Mediante esta GUI, cuyo modelo se muestra en el Anexo A, es posible visualizar todas las aplicaciones con pruebas automatizadas disponibles y seleccionar entre ejecuciones rápidas o completas. Las pruebas rápidas se enfocan en validar únicamente el acceso correcto a la aplicación, mientras que las pruebas completas abarcan una verificación más amplia de sus funcionalidades. Para cada ejecución, el sistema genera automáticamente un archivo Excel con las evidencias correspondientes, cuyo formato y contenido se presentan en el Anexo B.

Este proceso de evolución técnica permitió consolidar una herramienta robusta, eficiente y escalable, manteniendo como principios fundamentales la claridad en la implementación, la reutilización de componentes y la facilidad de mantenimiento. Estos principios coinciden con los objetivos de escalabilidad y flexibilidad mencionados por Jimenez & Fartolino (2023), quienes desarrollaron su propuesta en un contexto tecnológico y rubro distintos a los abordados en el presente trabajo; no obstante, esta concordancia permite evidenciar que los objetivos asociados al desarrollo de herramientas de automatización de pruebas pueden ser extrapolados y aplicados en diferentes entornos organizacionales.

CAPITULO V: CONCLUSIONES

- El sistema desarrollado cumplió con el objetivo general al proporcionar una solución integral para la validación de servicios web que combina pruebas automatizadas de autenticación y de flujo funcional. Esta herramienta ofrece un método confiable y eficiente para garantizar la calidad de las aplicaciones en la empresa de telecomunicaciones.
- Se logró el diseño e implementación de pruebas automatizadas de autenticación adaptadas a las particularidades de cada aplicación, contemplando variaciones en credenciales y verificando el acceso correcto mediante la detección automática de elementos clave en la interfaz. Este avance asegura un punto de inicio válido para las pruebas funcionales posteriores.
- Se diseñaron e implementaron pruebas automatizadas de flujo funcional capaces de reproducir escenarios de uso reales y validar las funcionalidades críticas de cada aplicación. Este enfoque permitió verificar de forma consistente el correcto funcionamiento de los procesos más relevantes para la operación.
- Las pruebas de autenticación y de flujo funcional fueron integradas exitosamente en un módulo de prueba completa que orquesta su ejecución de manera coherente, permitiendo así una validación integral del comportamiento de las aplicaciones y facilitando la identificación de fallos o comportamientos inesperados.
- El sistema implementado optimizó significativamente la ejecución y la

recolección de evidencias de las pruebas de aplicaciones web, logrando una reducción del 90 % en el tiempo de validación de las pruebas rápidas —de 10 minutos a solo 1 minuto— y del 80 % en las pruebas completas —de 25 minutos a 5 minutos—, lo que incrementa sustancialmente la productividad en el ciclo de pruebas.

- Al contrastar los resultados obtenidos con el trabajo de Jimenez y Fartolino (2023), se observa que el presente sistema logró reducciones significativas en los tiempos de ejecución de las pruebas automatizadas. Esta diferencia puede explicarse, en gran medida, por el contexto y las decisiones adoptadas durante el desarrollo. Mientras que en el entorno bancario analizado por dichos autores la solución debía ajustarse a arquitecturas y tecnologías ya existentes, en este trabajo el sistema fue diseñado desde cero, lo que permitió una mayor libertad para seleccionar e integrar las tecnologías más adecuadas, orientando el desarrollo hacia la optimización del rendimiento y la eficiencia en la ejecución de las pruebas.
- La capacidad del sistema para controlar la mayoría de los errores y manejar automáticamente los casos excepcionales relacionados con el driver de automatización asegura la estabilidad y confiabilidad del proceso de validación. Esta característica reduce la necesidad de intervención manual y garantiza la continuidad de las pruebas, incluso ante incidencias esporádicas. En este sentido, durante el desarrollo del sistema también se evidenciaron problemas asociados a la fragilidad y al comportamiento impredecible de las automatizaciones, aspectos que han sido ampliamente señalados en la literatura especializada (Leotta et al., 2023). No obstante, la aplicación progresiva de técnicas como reintentos y esperas explícitas permitió gestionar estas situaciones de manera efectiva y consolidar una solución estable y confiable

CAPITULO VI: RECOMENDACIONES

- La implementación detallada y el procedimiento propuesto constituyen una base metodológica que puede ser utilizada como referencia para el diseño e implementación de sistemas de automatización de pruebas en aplicaciones web de otras organizaciones, favoreciendo la replicabilidad y adaptación del enfoque a distintos contextos empresariales.
- Se recomienda, en el futuro, migrar la aplicación a un entorno web, lo que facilitaría el acceso desde cualquier dispositivo, reduciría los problemas de mantenimiento en las estaciones de trabajo y permitiría una actualización centralizada y continua de la solución.
- El uso de Docker es fundamental en un proyecto de estas características, ya que permite contenerizar tanto el navegador como las configuraciones necesarias, ofreciendo un entorno de ejecución replicable. Esto facilita tanto el despliegue en un entorno web como su distribución para uso local en múltiples ordenadores, asegurando que todos los desarrolladores y testers trabajen con un entorno consistente.
- Una posible área de investigación futura es el uso de Large Language Models para la automatización dinámica de pruebas. Esto permitiría que los testers describan las acciones que desean realizar sobre una página web, y el LLM generaría automáticamente el código necesario para ejecutar esas pruebas, utilizando la misma estructura y validación del código ya existente en la aplicación. De esta manera, el tester no necesitaría programar manualmente cada nueva automatización, facilitando la creación de nuevas pruebas de manera rápida y eficiente.

CAPITULO VII: BIBLIOGRAFÍA

- Ateşoğulları, D., & Mishra, A. (2020). AUTOMATION TESTING TOOLS: A COMPARATIVE VIEW. *International Journal on Information Technologies & Security*, 12(4), 63-76.
- Bons, A., Marín, B., Aho, P., & Vos, T. E. J. (2023). Scripted and scriptless GUI testing for web applications: An industrial case. *Information and Software Technology*, 158, 107172. <https://doi.org/10.1016/j.infsof.2023.107172>
- Bubenko, M., Karpenko, N., Gerasimov, V., & Morozov, A. (2023). Analysis of web application testing methods. *System technologies*, 4(147), 82–90. <https://doi.org/10.37129/2707-2680-2023-4-147-7>
- Choubey, A. (2025). Comparing Script-Based and Script-Less Automation: A Practical Guide to Choosing the Right Approach for Testing. *International Journal of Research in Computer Applications and Information Technology*, 8(1), 3323–3341. https://doi.org/10.34218/IJRCAIT_08_01_239
- Cuellar-Juarez, L. N., & Mamani-Vilca, E. (2024). Análisis comparativo de lenguajes de programación: Python vs. Java en el desarrollo de aplicaciones web. *Revista Micaela*, 5(1), 74–81. Universidad Nacional Micaela Bastidas de Apurímac. <https://doi.org/10.57166/micaela.v5.n1.2024.143>
- Dalampis, N. D. (2021). A behavior-driven approach in automated software testing (Tesis de maestría, Aristotle University of Thessaloniki, School of Informatics). IKEE. <https://doi.org/10.26262/heal.auth.ir.337022>
- Delgado Mamani, I. J. (2024). Implementación de pruebas automatizadas

usando Selenium, Scrum y ISTQB para optimizar los tiempos de ejecución de pruebas en una entidad bancaria (Trabajo de suficiencia profesional). Universidad Nacional Mayor de San Marcos. Repositorio Cybertesis UNMSM. <https://hdl.handle.net/20.500.12672/22562>

Delgado Olivera, L. d., & Díaz Alonso, L. M. (31 de Marzo de 2021). Modelos de Desarrollo de Software. *Revista Cubana de Ciencias Informáticas*, 15(1), 37-51. Recuperado el 28 de Febrero de 2025, de SciELO Cuba: http://scielo.sld.cu/scielo.php?pid=s2227-18992021000100037&script=sci_arttext

Fatima, S., Nasim, S. F., Haider, N. G., Rasheed, M., & Akram, Z. (2023). Comparative study of software automation tools: Selenium and Quick Test Professional. *Journal of Independent Studies and Research: Computing*, 21(1), 52–54. <https://doi.org/10.31645/JISRC.23.21.1.6>

Gallego Izquierdo, V. (2024, 22 de enero). Análisis de rendimiento de lenguajes de desarrollo de aplicaciones y servicios web [Trabajo Final de Grado, Universitat Politècnica de Catalunya]. UPCommons. <https://hdl.handle.net/2117/407069>

IndraGroup. (2026). *Indra Group en Perú: Equipo*. Recuperado el 22 de enero de 2026, de <https://www.indragroup.com/es/america-latina/peru>

ISTQB. (2023). Certified Tester Foundation Level (CTFL) Syllabus v4.0.1. International Software Testing Qualifications Board. Obtenido de <https://www.istqb.org>

Izzat, S. K., & Saleem, N. N. (2023). Software Testing Techniques and Tools: A Review. *Journal of Education and Science*, 32(2), 30–44. <https://doi.org/10.33899/edusj.2023.137480.1305>

Jimenez Vargas, P. A., & Fartolino Guerrero, F. (2023). Solución de automatización de pruebas de calidad en empresa bancaria para reducir tiempos de certificación con herramientas tecnológicas de software [Trabajo de suficiencia profesional, Universidad Peruana de Ciencias Aplicadas (UPC)]. Repositorio Académico UPC. <http://hdl.handle.net/10757/670906>

- Kumar, S. (2023). Reviewing Software Testing Models and Optimization Techniques: An Analysis of Efficiency and Advancement Needs. *Journal of Computers, Mechanical and Management*, 2(1), 43-55. <https://doi.org/10.57159/gadl.jcmm.2.1.23041>
- Leotta, M., García, B., Ricca, F., & Whitehead, J. (2023). Challenges of End-to-End Testing with Selenium WebDriver and How to Face Them: A Survey. 2023 IEEE 16th International Conference on Software Testing, Verification and Validation, 16–20 April 2023, Dublin, Ireland: Proceedings, Piscataway, NJ: IEEE. Pp.: 339-350.
- Li, T., Huang, R., Cui, C., Towey, D., Ma, L., Li, Y.-F., & Xia, W. (2025). A survey on web application testing: A decade of evolution. *arXiv*. <https://arxiv.org/abs/2412.10476>
- Majeed, B., Chaudhary, M. N. A., Toor, S. K., & Majeed, K. (2021). Comparative study of open source automation testing tools: Selenium, Katalon Studio & Test Project. 2021 International Conference on Innovative Computing (ICIC). <https://doi.org/10.1109/ICIC53490.2021.9693066>
- Mariani, L., Mohebbi, A., Pezzè, M., & Terragni, V. (2021). Semantic matching of GUI events for test reuse: Are we there yet? In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21)* (pp. 177–190). Association for Computing Machinery. <https://doi.org/10.1145/3460319.3464827>
- Mohamed, K. N., & Kumar, R. P. (2021). Scriptless GUI Automation Testing Tool for Web Applications. *International Journal of Recent Technology and Engineering*, 10(1), 216–220. <https://doi.org/10.35940/ijrte.A5849.0510121>
- Morales-Carrillo, J., Cedeño-Valarezo, L., Bravo, J. S., & Calderón, J. G. (2022). Metodologías de desarrollo de software y su ámbito de aplicación: Una revisión sistemática. *Revista Ibérica de Sistemas e Tecnologías de Informação* (E47), 29-45. <https://www.proquest.com/openview/937c0ffcf92bf2c2761d76165f340>

90e/1?pq-origsite=gscholar&cbl=1006393

- Mukherjee, R., & Patnaik, K.S. (2021). A survey on different approaches for software test case prioritization. *Journal of King Saud University - Computer and Information Sciences*, 33(9), 1041-1054. <https://doi.org/10.1016/j.jksuci.2018.09.005>
- Mustafa, Y., & Handayani, L. (2024). Achieving Software Testing Efficiency Through the Implementation of Cutting-Edge Automation Technologies. *Applied Research in Artificial Intelligence and Cloud Computing*, 7(6), 83 - 107.
- Narang, P., & Mittal, P. (15 de Diciembre de 2022). Performance Assessment of Traditional Software Development Methodologies and DevOps Automation Culture. *Engineering, Technology & Applied Science Research*, 12(6), 9726–9731. <https://doi.org/10.48084/etasr.5315>
- Nass, M., Alégroth, E., & Feldt, R. (2021). Why many challenges with GUI test automation (will) remain. *Information and Software Technology*, 138, 106625. <https://doi.org/10.1016/j.infsof.2021.106625>
- Olaoye, G., & Idowu, M. (2024). Functional vs. Non-Functional Testing: Balancing Quality in Agile Teams. ResearchGate.
- Oracle. (2020). Java vs. Python: Which is best for students to learn? Oracle Academy. <https://academy.oracle.com/en/solutions-curriculum-java.html>
- Sawant, S., Patil, S., & Bhujbal, V. (2021). Implementation of Selenium automation & report generation using Selenium Web Driver & ATF. *International Research Journal of Engineering and Technology (IRJET)*, 8(10), 1044–1049. <https://www.irjet.net/archives/V8/i10/IRJET-V8I10209.pdf>
- Stack Overflow. (2025). *Stack Overflow Developer Survey 2025: Most popular technologies – Programming, scripting, and markup languages*. Stack Overflow. <https://survey.stackoverflow.co/2025/technology#most-popular-technologies-language>

- Talakola, S. (2024). Automated end to end testing with Playwright for React applications. *International Journal of Emerging Research in Engineering and Technology*, 5(1), 38-47. <https://doi.org/10.63282/3050-922X.IJERET-V5I1P106>
- Tibell, S., & Kholi, M. (2023). Choosing the Right Automated UI Testing Tool: A Comparative Study of Selenium and TestComplete [Bachelor's thesis].
- Umar, M. A. (2020). A Study of Software Testing: Categories, Levels, Techniques, and Types [Preprint]. *TechRxiv*. <https://www.techrxiv.org/doi/full/10.36227/techrxiv.12578714.v1>
- Wang, F., Arora, C., Tantithamthavorn, C., Huang, K., & Aleti, A. (2025). Requirements-Driven Automated Software Testing: A Systematic Review. <https://doi.org/10.48550/arXiv.2502.18694>
- Zhou, Z., Chen, R., Chen, J., Liu, Y., & Wei, L. (2025). Characterizing Bugs in Login Processes of Android Applications: An Empirical Study (arXiv:2502.04200v3). *arXiv*. <https://arxiv.org/abs/2502.04200>

CAPITULO VIII: ANEXOS

Anexo A: Modelo de Interfaz Gráfica implementada

Con el fin de facilitar la ejecución de las pruebas automatizadas, se desarrolló una interfaz gráfica sencilla. En la sección izquierda se listan las aplicaciones cuyas pruebas han sido automatizadas, mientras que en la parte derecha se muestra un listado dinámico que se actualiza conforme se seleccionan las automatizaciones a ejecutar.

Asimismo, la interfaz incluye selectores exclusivos que permiten elegir entre dos modalidades de prueba: rápida (solo autenticación) y completa (autenticación más flujo funcional). Finalmente, se incorpora el botón “Ejecutar”, que permite iniciar el proceso de automatización seleccionado.



Figura A1 Modelo de GUI implementada. Fuente: Elaboración propia.

Anexo B: Modelo de archivo Excel de evidencias

Con el fin de recopilar y organizar las evidencias de los resultados de las pruebas automatizadas, se estableció un modelo de archivo en formato Excel. Dicho archivo presenta la siguiente estructura:

- Primera columna: descripción del caso de prueba.
- Segunda columna: evidencia correspondiente (captura de pantalla).
- Tercera columna: estado de la prueba (OK o KO).

En las figuras B1 y B2 se muestran ejemplos de este formato, aplicados tanto a pruebas rápidas como a pruebas de flujo funcional.

	A	B	C
2			
3	CASO DE PRUEBA	EVIDENCIA	ESTADO DE PRUEBA
4	Abrir la aplicación		OK
5	Hacer login		OK
6			

Figura B1 Modelo de Tabla en Excel con las validaciones realizadas de autenticación. Fuente: Elaboración propia

CASO DE PRUEBA	EVIDENCIA	ESTADO DE PRUEBA
Descargar documento		OK
Abrir información de un registro		OK

Figura B2 Modelo de Tabla en Excel con las validaciones realizadas de flujo funcional. Fuente: Elaboración propia.

Anexo C: Lenguajes de programación más populares en 2025 según Stack Overflow

Esta encuesta recopila las respuestas de 31 771 desarrolladores, quienes indicaron cuáles lenguajes de programación, scripting o de marcado utilizaron con mayor frecuencia entre 2024 y 2025. Se observa que los lenguajes de programación más populares son JavaScript, Python, TypeScript y Java.

En la figura C1 se puede revisar con mayor detalle los resultados de la encuesta mencionada.

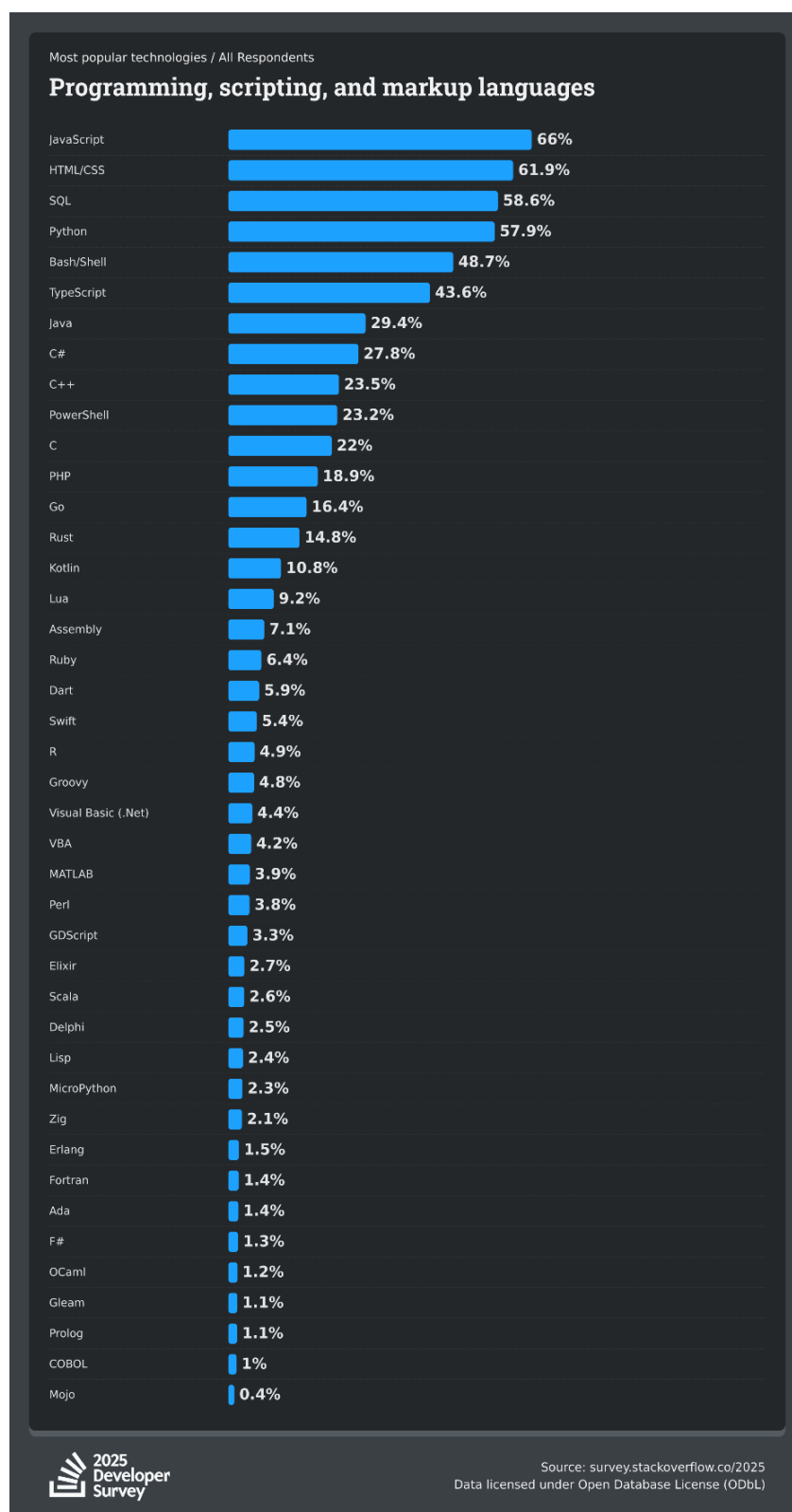


Figura C1 Lenguajes de programación más populares en 2025. Fuente: Tomado de Stack Overflow (2025).

Anexo D: Especificaciones técnicas de hardware de referencia

Con el objetivo de garantizar la repetibilidad de las pruebas y asegurar que los tiempos de respuesta del sistema automatizado no se vean afectados por limitaciones físicas, se describe a continuación el hardware estándar utilizado (estación de trabajo del desarrollador):

- Tipo de equipo: Laptop HP ProBook 440 G9.
- Procesador: Intel® Core™ i7-1260P (12va Generación).
- Frecuencia: 2.10 GHz (Frecuencia base) hasta 4.70 GHz (Turbo Boost).
- Núcleos / Hilos: 12 núcleos físicos y 16 hilos de procesamiento.
- Memoria RAM: 16 GB DDR4
- Almacenamiento: Unidad de Estado Sólido (SSD) de 512 GB NVMe M.2.
- Sistema Operativo: Windows 11 Pro (64 bits).

Anexo E: Comparación del tiempo de ejecución entre pruebas manuales y automatizadas: caso representativo

En el presente anexo se muestra un ejemplo representativo de la medición de tiempo de ejecución de pruebas aplicado sobre una de las aplicaciones web que evalúa el sistema.

Por cuestiones de confidencialidad, no es posible revelar el nombre de la aplicación, URLs, ni colores corporativos, refiriéndose a esta como APP1 y mostrando las capturas de pantalla en escala de grises.

a. Pruebas rápidas sobre APP1

Debido a que las pruebas manuales sobre APP1 no arrojan logs de tiempo automáticos, la trazabilidad de su duración se registra en un documento físico. Este archivo cuenta con la validación y firma del líder del proyecto, asegurando la veracidad de los tiempos de ejecución presentados en este

informe (ver figura E1).

ACTA DE EJECUCIÓN DE PRUEBA RÁPIDA MANUAL SOBRE APP1		
Hora Inicio: 15:30		
Hora Fin: 15:38		
ACCIÓN	EVIDENCIA	TIMESTAMP
Inicio de prueba (navegador abierto)		2026-01-15 15:30
Búsqueda de credenciales		2026-01-15 15:30
Carga de página de inicio		2026-01-15 15:31
Inserción de credenciales		2026-01-15 15:32
Ingreso a la aplicación		2026-01-15 15:33
Verificación de carga correcta		2026-01-15 15:33
Redacción de documento de evidencias		2026-01-15 15:34
Fin de prueba (documento de evidencias redactado)		2026-01-15 15:38


 Firma del líder de proyecto
(Josep Monje Darnau / Project Manager)

Figura E1 Acta de ejecución de pruebas rápidas manuales sobre APP1.

Fuente: Elaboración propia, validada por firma de líder de proyecto

En la Figura E1 se observa la hora de inicio y fin de las pruebas rápidas manuales:

- Hora inicio: 15:30
- Hora fin: 15:38
- Duración: 8 minutos

Con una duración total, por lo tanto, de 8min de ejecución manual para la prueba rápida sobre APP1.

Con respecto a las pruebas automatizadas, se puede observar su duración a través de los logs que genera después de su ejecución.



Figura E2 Archivo log de prueba rápida automatizada sobre APP1.

Fuente: Elaboración propia.

En la figura E2 se observa la hora de inicio y fin de ejecución de una prueba rápida automatizada de la que se obtiene los siguientes datos:

- Hora inicio: 15:58:22
- Hora fin: 15:59:06
- Duración: 44 segundos (0.73 minutos)

Con una duración total de 0.73 minutos en la ejecución automatizada.

b. Pruebas completas sobre APP1

En cuanto a las pruebas manuales completas, se sigue el mismo criterio que en las pruebas rápidas: el registro de tiempos se formaliza mediante un acta validada por el líder del proyecto. En la Figura E3 se detallan las acciones ejecutadas, el proceso de recolección de evidencias y la trazabilidad horaria, especificando tanto el inicio de la prueba sobre APP1 como su finalización tras la redacción del documento de evidencias.

- Hora inicio: 19:19
- Hora fin: 19:43
- Duración: 24 minutos

Con una duración total, por lo tanto, de 24 minutos de ejecución de pruebas manuales completas sobre la aplicación web APP1.

ACTA DE EJECUCIÓN DE PRUEBA COMPLETA MANUAL SOBRE APP1

Hora Inicio: 19:19

Hora Fin: 19:43

ACCIÓN	EVIDENCIA	TIMESTAMP
Inicio de prueba (navegador abierto)		2026-01-15 19:19
Búsqueda de credenciales		2026-01-15 19:20
Carga de página de inicio		2026-01-15 19:20
Prueba 1		2026-01-15 19:21
Prueba 2		2026-01-15 19:22
Prueba 3		2026-01-15 19:24
Prueba 4		2026-01-15 19:25
Prueba 5		2026-01-15 19:27
Prueba 6		2026-01-15 19:31
Redacción documento de evidencias		2026-01-15 19:32
Fin de prueba (documento de evidencias redactado)		2026-01-15 19:43

Firma del líder de proyecto
(Josep Monje Salmau / Project Manager)

Figura E3 Acta de ejecución de pruebas completas manuales sobre APP1. Fuente: Elaboración propia, validada por firma de líder de proyecto.

Para obtener los tiempos de ejecución de pruebas completas automatizadas sobre APP1 se puede revisar los logs en la figura E4.

De los logs se obtienen las horas de inicio y fin de la ejecución automatizada:

- Hora inicio: 20:39:40
- Hora fin: 20:43:11
- Duración: 3 minutos con 31 segundos (3.52 minutos)

```

pruebasCompletas APP1 log
Automatizaciones > Pruebas APP1 > resultados > pruebasCompletas APP1 log
1 2026-01-15 20:39:40,187 Inicio de pruebas
2 2026-01-15 20:39:54,938 Inicio de prueba 1
3 2026-01-15 20:40:22,959 Prueba 1 exitosa
4 2026-01-15 20:40:22,974 Inicio de prueba 2
5 2026-01-15 20:40:53,483 Prueba 2 exitosa
6 2026-01-15 20:40:53,483 Inicio de prueba 3
7 2026-01-15 20:41:39,368 Prueba 3 exitosa
8 2026-01-15 20:41:39,368 Inicio de prueba 4
9 2026-01-15 20:42:04,413 Prueba 4 exitosa
10 2026-01-15 20:42:04,413 Inicio de prueba 5
11 2026-01-15 20:42:46,427 Prueba 5 exitosa
12 2026-01-15 20:42:46,427 Inicio de prueba 6
13 2026-01-15 20:43:08,791 Prueba 6 exitosa
14 2026-01-15 20:43:11,970 Fin de pruebas APP1
15

```

Figura E4 Archivo log de prueba completa automatizada sobre APP1.
Fuente: Elaboración propia.

c. Comparación de tiempos de ejecución sobre APP1

De acuerdo con lo presentado anteriormente en este anexo, podemos hacer la comparación:

- Duración prueba rápida manual sobre APP1: 8 minutos
- Duración prueba rápida automatizada sobre APP1: 0.73 minutos
- Duración prueba completa manual sobre APP1: 24 minutos
- Duración prueba completa automatizada sobre APP1: 3.52 minutos

Con lo cual, podemos calcular la reducción de tiempos que ofrece la automatización en una aplicación web específica.

- Reducción de tiempo en prueba rápida: 90.88%
- Reducción de tiempo en prueba completa: 85.33%

Anexo F: Cálculo de tiempos promedios de pruebas

En el presente anexo se muestra la forma de calcular los tiempos promedios de los resultados de las pruebas.

Para ello, se tomaron 5 aplicaciones representativas que evalúa el sistema,

de las cuales se hicieron 3 mediciones de pruebas rápida y completa, de forma manual y automatizada, por cada aplicación. Se muestran los tiempos obtenidos en la Tabla F1.

Tabla F1 Medición de tiempo de pruebas en 5 aplicaciones web

Aplicación web evaluada	Prueba	Prueba 1	Prueba 2	Prueba 3	Promedio
APP3	Rápido Manual	7' 0"	8' 0"	8' 0"	7' 40"
	Rápido Auto	0' 38"	0' 40"	0' 33"	0' 37"
	Completa Manual	0' 40'	0' 39'	0' 43'	40' 40"
	Completa Auto	7' 27"	7' 41"	6' 52"	7' 20"
APP4	Rápido Manual	8' 0"	9' 0"	9' 0"	8' 40"
	Rápido Auto	0' 46"	0' 50"	0' 44"	0' 47"
	Completa Manual	15' 16"	17' 21"	15' 48"	16' 08"
	Completa Auto	2' 32"	2' 38"	2' 30"	2' 33"
APP1	Rápido Manual	8' 0"	10' 0"	7' 0"	8' 20"
	Rápido Auto	0' 44"	0' 45"	0' 44"	0' 44"
	Completa Manual	24' 0"	23' 0"	24' 0"	23' 40"
	Completa Auto	3' 31"	3' 32"	3' 36"	3' 33"
APP2	Rápido Manual	9' 0"	8' 0"	9' 0"	8' 40"
	Rápido Auto	0' 35"	0' 33"	0' 37"	0' 35"
	Completa Manual	37' 0"	41' 0"	40' 0"	39' 20"
	Completa Auto	5' 13"	5' 11"	5' 18"	5' 14"
APP8	Rápido Manual	10' 0"	8' 0"	9' 0"	9' 0"
	Rápido Auto	0' 52"	0' 51"	0' 56"	0' 53"
	Completa Manual	25' 0"	22' 0"	26' 0"	24' 20"
	Completa Auto	3' 22"	3' 20"	3' 25"	3' 22"

Fuente: Elaboración propia

En base a los datos de la Tabla F1 para calcular los tiempos se aplicó un promedio aritmético, de lo cual se obtuvo lo siguiente:

- Promedio pruebas rápidas manuales: 8' 28"
- Promedio pruebas rápidas automatizadas: 0' 43"
- Promedio pruebas completas manuales: 28' 42"
- Promedio pruebas completas automatizadas: 4' 24"