https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 91 - 107

Diseño e implementación de una solución híbrida para pruebas de desempeño a formularios dinámicos de software libre: metodología, arquitectura y herramientas

Información del reporte:

Para citar este reporte técnico:

Rangel Cano et al. (2025). Diseño e implementación de una solución híbrida para pruebas de desempeño a formularios dinámicos de software libre: metodología, arquitectura y herramientas. *Cuadernos Técnicos Universitarios de la DGTIC, 3* (3) páginas(91 - 107).

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123

Liliana Rangel Cano
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
lilianarc@unam.mx
ORCID: 0009-0001-4100-4011

Cristhian Eder Alavez Barrita
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
calavez@comunidad.unam.mx
ORCID: 0009-0003-7408-2432

Resumen

El presente reporte técnico describe una experiencia en la aplicación de pruebas de desempeño a un formulario generado en la herramienta LimeSurvey por una entidad de la Universidad Nacional Autónoma de México. Las pruebas fueron aplicadas mediante una metodología conformada por las fases: planeación, diseño, aplicación y cierre, alineadas a actividades basadas en buenas prácticas. Durante el desarrollo de las actividades, se encontraron desafíos significativos en la automatización de pruebas, como son: el manejo de archivos adjuntos, de *tokens* de seguridad y la lógica condicional en los formularios. Para superar algunos de estos obstáculos, se desarrolló una solución que integró múltiples tecnologías: JMeter, WebDriver Sampler y Selenium WebDriver. Pese a ciertas limitaciones técnicas, la falta de información detallada de los componentes y del funcionamiento interno del

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 92 - 107

software, la solución fue viable para cumplir el objetivo de evaluar el desempeño de la aplicación al ajustar la estrategia con los recursos disponibles. Este reporte destaca la importancia de mantener un enfoque flexible al aplicar pruebas de desempeño de aplicaciones web complejas, lo cual sugiere que las estrategias híbridas pueden ser efectivas en ciertos contextos.

Palabras clave:

Pruebas de desempeño, scripts de automatización, LimeSurvey, Selenium WebDriver Sampler, JMeter.

Abstract

This technical report describes an experience applying performance testing on a form generated with the LimeSurvey tool by an entity at the National Autonomous University of Mexico. The tests were applied using a methodology comprised of the following phases: planning, design, implementation, and closure, aligned with activities based on best practices. During the development of the activities, significant challenges were encountered in test automation, such as the handling of attachments, security tokens, and conditional logic in forms. To overcome some of these obstacles, a solution was developed that integrated multiple technologies: JMeter, WebDriver Sampler, and Selenium WebDriver. Despite certain technical limitations and the lack of detailed information about the components and the internal workings of the software, the solution proved viable to achieve the objective of evaluating the application's performance by adjusting the strategy to the available resources. This report highlights the importance of maintaining a flexible approach when applying performance testing to complex web applications, suggesting that hybrid strategies can be effective in certain contexts.

Keywords:

Performance testing, automation scripts, LimeSurvey, Selenium WebDriver Sampler, JMeter.

1. INTRODUCCIÓN

En el panorama digital actual, es imprescindible considerar las características de calidad (atributos que determinan si una aplicación es rápida, confiable y usable) al evaluar un software, ya que éstas son fundamentales para el éxito de los proyectos. Entre ellas, se encuentra la eficiencia de desempeño, que mide qué tan rápido responde el software y cuántos recursos del sistema consume para realizar sus funciones. Esta característica es importante porque los usuarios esperan interacciones fluidas y tiempos de carga rápidos; cualquier retraso puede disminuir la satisfacción del cliente. Un retraso de tan solo un segundo en la respuesta de una página puede reducir esta satisfacción en un 16% (Mărcuță, 2024).

Las pruebas de desempeño son fundamentales para garantizar la calidad y eficiencia del software, ya que permiten evaluar su comportamiento bajo condiciones de estrés y uso intensivo de recursos, además de que aseguran un desempeño óptimo dentro de parámetros específicos de tiempo y rendimiento (International Organization for Standardization & International Electrotechnical Commission, 2011). De acuerdo con Legramante *et al.* (2020), su importancia radica en la capacidad para simular escenarios de carga y estrés, lo que permite prever posibles fallos y mejorar la experiencia del usuario, evitando problemas como tiempos de respuesta lentos. Por su parte, ImpactQA (s. f.) señala que aplicarlas ayuda a prevenir la disminución de ingresos, frustración en los usuarios y a mitigar riesgos reputacionales.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 93 - 107

Además, organismos internacionales como Micro Focus, Sogeti y Capgemini (World Quality Report 2023-24, 2023) destacan que las pruebas de desempeño son esenciales para garantizar la calidad técnica y la satisfacción del usuario.

LimeSurvey es una herramienta de encuestas en línea que, de acuerdo a lo indicado en su página, cuenta con más de 1.5 millones de usuarios a nivel mundial (Limesurvey, s. f.) para recolectar datos mediante una plataforma que permite a los usuarios diseñar encuestas personalizadas; su naturaleza de código abierto favorece la personalización, el desarrollo colaborativo y el soporte comunitario. En la Universidad Nacional Autónoma de México (UNAM), LimeSurvey es utilizada en actividades académicas y administrativas, evaluaciones estudiantiles, investigaciones y recolección de datos en proyectos sociales (Holguín, 2020).

En este marco, una entidad de la UNAM determinó utilizar LimeSurvey Community Edition versión 6.5.9 para atender el registro de propuestas de proyectos mediante un formulario, y así proyectar un escenario de 100 usuarios concurrentes al cierre del registro.

Para garantizar la respuesta a los usuarios que realizaron el registro de sus propuestas de proyecto de manera concurrente, generalmente en la última hora de la fecha límite del registro, y recopilar adecuadamente la información asociada de los proyectos, la entidad universitaria, basada en el conocimiento de la importancia de las pruebas de desempeño, identificó la necesidad de realizar pruebas de este tipo, que tenían como objetivo evaluar la estabilidad de la plataforma ante un escenario de 100 usuarios en carga sostenida y conocer los límites de atención con los recursos de infraestructura disponibles. Por lo anterior, solicitó el apoyo técnico para su realización y proporcionó los insumos necesarios para las pruebas (ambiente tecnológico de pruebas), entre los que se encontraba el apoyo de especialistas con disponibilidad para aclarar dudas estructurales del software, así como para atender cualquier eventualidad.

Dado que la plataforma fue desarrollada externamente a las área universitarias involucradas en la aplicación de las pruebas de desempeño, la automatización de las actividades a simular presentó un desafío técnico significativo, debido a la falta de información detallada de los componentes y del funcionamiento interno del software, lo que dificultó la identificación de los procesos clave y la realización de ajustes.

El presente reporte técnico tiene como objetivo describir tanto la metodología utilizada como las actividades realizadas para la aplicación de las pruebas de desempeño al cuestionario, así como indicar la manera en que se resolvió el problema técnico en la automatización de las pruebas.

2. DESARROLLO TÉCNICO

Las pruebas de desempeño permiten realizar una evaluación al software tanto para medir su rendimiento bajo diversas condiciones de concurrencia, carga, volumen y/o estrés, como para identificar el comportamiento del software en términos de velocidad, estabilidad y capacidad de respuesta, ya que son un tipo de prueba para determinar la eficiencia de rendimiento de un componente o sistema (Bath et al., 2018). Se comprendió la aplicación de la siguiente metodología establecida (Molyneaux, 2014), la cual puede definirse como un conjunto estructurado de actividades y técnicas que se utilizan para evaluar la eficacia en la simulación de 100 usuarios concurrentes al resolver un formulario compuesto de diversos elementos de captura.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 94 - 107

Consideraciones éticas

Antes de hablar de la metodología empleada, se destaca que las pruebas de desempeño deben realizarse únicamente a solicitud y con autorización expresa de los responsables del proyecto, con objetivos claros y definidos para el beneficio del mismo, debido a los riesgos asociados con su aplicación, tales como: la consulta o generación de información sensible, la posible degradación del rendimiento o la caída tanto del servicio evaluado como de otros servicios que coexisten en la misma infraestructura o con los que se comunica. En caso de que el software evaluado se comunique con otros sistemas o servicios, también debe existir la aprobación de los responsables de dichos servicios.

Las pruebas deben llevarse a cabo en un ambiente tecnológico específico para tal fin; si se realizan en el ambiente productivo, es indispensable contar con respaldos completos y un plan de mitigación de riesgos para evitar impactos negativos.

Los especialistas encargados de las pruebas deben informar sobre el proceso, los riesgos y las implicaciones de estas actividades, planificándolas cuidadosamente para incluir respaldos de información, seleccionar momentos oportunos que minimicen el impacto tanto en el servicio evaluado como en los que comparten infraestructura, y establecer mecanismos de seguridad compensatorios. Además, se debe evitar comprometer la integridad y confidencialidad de la información, haciendo uso exclusivo de datos de prueba cuando sea posible.

Metodología

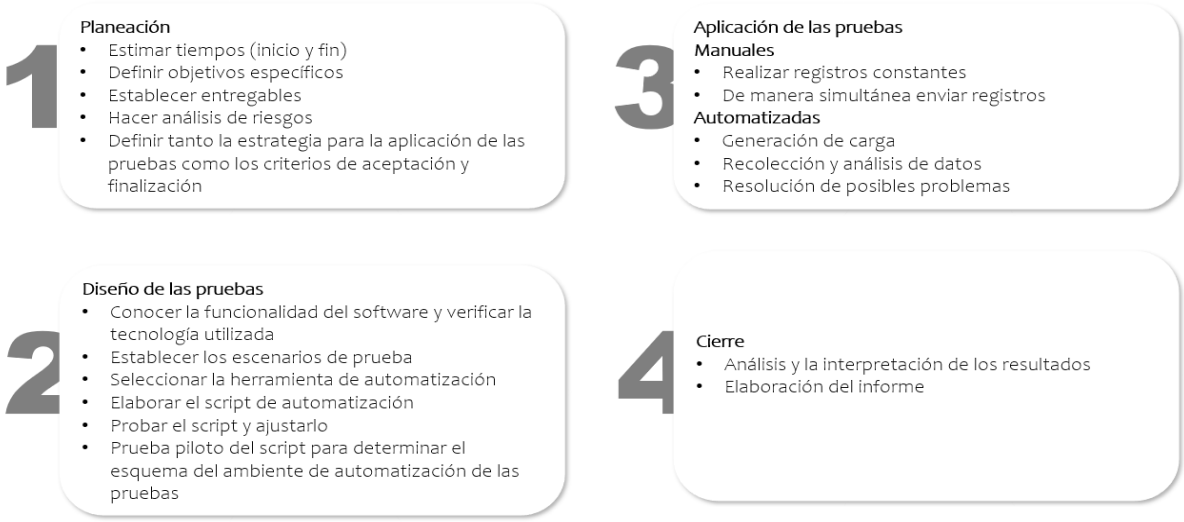
A continuación, como se muestra en la Figura 1, se presentan las fases y buenas prácticas que se llevaron a cabo en la aplicación de pruebas de desempeño del formulario generado en la herramienta LimeSurvey, con énfasis en las actividades que presentaron mayores desafíos. Aunque las fases no se denominan exactamente como en el *Foundation Level Specialist Syllabus Performance Testing* del ISTQB (Bath et al., 2018), las tareas se alinean con las definidas en el *syllabus*: "Planeación", "Análisis, diseño e implementación", "ejecución" y "análisis y reporte de resultados".[1]

1 Las actividades del *Syllabus* denominadas "Planeación", "Análisis, diseño e implementación", "Ejecución" y "Análisis y reporte de resultados" corresponden respectivamente a las fases de "Planeación", "Diseño de pruebas", "Aplicación de las pruebas" y "Cierre" en la metodología propuesta.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 Vol. 3, Núm. 3. julio-septiembre 2025, págs. 95 - 107

Figura 1

Metodología aplicada en las pruebas de desempeño



Entre el 10 y el 28 de junio de 2024, se llevó a cabo el proceso de pruebas de desempeño, en el cual, la etapa de diseño ocupó la mayor parte del tiempo disponible, representando un 80% del total (12 de los 15 días hábiles). Esta fase fue clave para identificar y desarrollar la solución tecnológica adecuada, centrada en la automatización de las pruebas. Durante este periodo, se llevaron a cabo actividades como la búsqueda de información relevante, la selección de herramientas apropiadas, así como la elaboración y configuración del *script* de automatización.

2.1 PLANEACIÓN

La planeación de las pruebas de desempeño fue tratada como un proyecto, específicamente, en actividades de estimación de tiempos (inicio y fin), definición de objetivos específicos, asignación de recursos y establecimiento de entregables.

En el caso concreto de las pruebas de desempeño al formulario, la actividad con mayor relevancia, debido al desarrollo y los resultados de éstas, fue la definición tanto de la estrategia para su aplicación como de los criterios de aceptación y finalización de las mismas.

Conforme a las buenas prácticas del sector y a las recomendaciones del Programa de estudios *Certified Performance Tester* con *JMeter* (Echeverria et al., 2019), se establecieron los criterios de aceptación para las pruebas de desempeño. Según este programa, es importante definir uno o más criterios u objetivos a alcanzar, los cuales pueden basarse en la cantidad de operaciones a ejecutar en un intervalo de tiempo, al consumo de recursos del sistema o al tiempo de respuesta de cada transacción, medidos bajo condiciones de carga definidas por el escenario correspondiente. Una vez que se cumplen los criterios de aceptación, las pruebas se dan por finalizadas.

Un desacierto que se cometió al definir la estrategia fue considerar como viable la automatización del flujo funcional de las pruebas, debido a que este proceso fue factible en proyectos previos con diversas

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 96 - 107

tecnologías. Por lo anterior, la estrategia tuvo que ser ajustada, conforme se desarrollaron las actividades, al plantear un esquema que incluyó la aplicación de pruebas manuales para adjuntar archivos y pruebas automatizadas para el ingreso de la información.

Un acierto que debe mantenerse en futuros proyectos es puntualizar la definición de los criterios de aceptación y finalización de las pruebas, que incluyan los escenarios favorables y desfavorables, para evitar prolongar las actividades sin obtener beneficios adicionales y/o cambios en los resultados de las pruebas.

Como lecciones aprendidas que se podrán considerar en proyectos que involucren pruebas de desempeño, se identificaron principalmente las siguientes:

- Considerar en el análisis de riesgos la inviabilidad de automatizar todos los movimientos realizados por el usuario (aun cuando se hayan realizado en proyectos previos); la manera de implementar y estructurar el código fuente del software pueden ser un obstáculo importante en la automatización.
- En la asignación de recursos humanos, contemplar como factor clave de éxito del proyecto, involucrar y establecer como compromiso la disponibilidad y participación del equipo de desarrollo en la automatización.
- En la estimación de tiempo, tomar en consideración que, debido a que cada proyecto es diferente, la automatización está asociada a la complejidad, tecnología utilizada y la estructura del código fuente, por lo que la duración de esta actividad puede variar incluso semanas.

2.2 DISEÑO DE LAS PRUEBAS

En las siguientes secciones, se presentan las actividades de preparación para la aplicación de pruebas que se contemplaron durante el diseño de las mismas:

2.2.1 CONOCER LA FUNCIONALIDAD DEL SOFTWARE Y VERIFICAR LA TECNOLOGÍA UTILIZADA

El formulario bajo pruebas fue generado en la plataforma de software libre y de código abierto LimeSurvey, la cual permite crear y administrar formularios en línea. Dicha plataforma posibilita diseñar cuestionarios, guardar las respuestas de los usuarios y analizar los resultados, a la vez que permite la personalización de los cuestionarios creados con preguntas condicionadas y múltiples tipos de respuestas; además, brinda la posibilidad de exportar los datos obtenidos.

El formulario se organizó en 6 páginas, cada una incluyendo entre 1 y 10 preguntas, con la información distribuida en un mínimo de 38 preguntas. Cada una solicitaba el ingreso de información mediante elementos como listas desplegables, opciones de selección única, casillas de verificación, campos para capturar datos y la posibilidad de adjuntar archivos.

Para definir las pruebas y determinar la manera de abordarlas, fue necesario conocer las condiciones en las que se encontraba el software, asegurar tener una versión estable sin errores de funcionamiento en el flujo principal y/o en los movimientos a realizar por el usuario e identificar las validaciones en los datos solicitados para evitar fallos en la generación del *script* de automatización o en la ejecución de las pruebas a causa de ingresar datos inválidos.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 97 - 107

2.2.2 ESTABLECER LOS ESCENARIOS DE PRUEBA A AUTOMATIZAR

El objetivo de las pruebas era conocer el comportamiento del formulario al ser contestado de manera concurrente, por lo que el único escenario seleccionado fue simular el flujo para contestarlo.

Se descartó la aplicación de pruebas relacionadas con la funcionalidad de LimeSurvey para gestionar cuestionarios o usuarios, obtención de resultados y estadísticas, como cualquier otra función complementaria. Esto se debió a que el objetivo principal no era evaluar la herramienta en su totalidad, sino centrarse exclusivamente en el comportamiento del formulario.

2.2.3 SELECCIÓN DE LA HERRAMIENTA DE AUTOMATIZACIÓN

Se eligió esta herramienta para la automatización de las pruebas debido a la experiencia en el uso de la herramienta JMeter para capturar información y adjuntar archivos, además de que cuenta con una comunidad y material de apoyo extenso.

JMeter es una aplicación de código abierto diseñada para realizar pruebas de carga y medir el rendimiento de aplicaciones web (Apache JMeter, s. f.). Su versatilidad permite simular múltiples usuarios y generar cargas significativas en el sistema, lo que resulta ideal para evaluar cómo responde el software bajo condiciones específicas.

2.2.4 ELABORAR EL SCRIPT DE AUTOMATIZACIÓN

Una vez seleccionada la herramienta *JMeter*, se creó un primer *script* de automatización[2] basado en las prácticas de grabación y ejecución para *scripts* avanzados, como la parametrización, tiempos de espera, controladores lógicos, aserciones y depuración del *script*, propuestas en el programa de estudios PtU (Echeverria et al., 2019). Este *script* se generó mediante la grabación y reproducción de las interacciones de los usuarios con el formulario de manera tradicional. Durante este proceso, se identificó que los elementos dinámicos, como algunos que utilizan *JavaScript*, no se integran adecuadamente.

Dado que se identificó una complejidad difícil de acotar, se solicitó apoyo a la entidad universitaria en:

- Proporcionar formularios con preguntas con elementos de captura específicos, para abordar la automatización de las pruebas de desempeño y con el propósito de analizar de manera aislada la manera de automatizar cada pregunta; se utilizaron formularios simples y estructurados con preguntas básicas, lo que permitió centrarse en la funcionalidad de cada campo de manera individual, de modo que se acotaron los puntos y variables involucrados en la funcionalidad del sistema.
- Acompañamiento en la solución de los problemas presentados en los *scripts* de automatización mediante asesoría respecto a la forma en que se encontraba estructurado el código fuente en ciertas funciones, apoyo para realizar ajustes en el código y en el monitoreo de *logs* para poder solucionar los problemas asociados.
- El apoyo proporcionado por la entidad universitaria se enfocó en comprender la funcionalidad LimeSurvey para identificar posibles soluciones a los errores presentados en el *script* de automatización, actividad que involucró tiempo considerable.

2 *Script* de automatización es un código que contiene un conjunto de instrucciones para realizar una tarea, que reproducen la actividad de los usuarios al utilizar el sistema.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 98 - 107

- En la Figura 2, se presenta un resumen de las incidencias identificadas durante el proceso de automatización de las pruebas, así como las soluciones implementadas para cada una. Estos aspectos se describen con mayor detalle en las secciones posteriores.

Figura 2

Desafíos y la solución generada en la elaboración del script de automatización

Desafío	Descripción	Solución
LimeSurveyutiliza JavaScript	Los elementos dinámicos, como algunos que utilizan JavaScript, no se integran adecuadamente	Integrar Selenium
Manejo de archivos adjuntos	LimeSurvey utiliza un modal para cargar archivos	Se ajustó la estrategia y se hicieron pruebas manuales
Gestión de sesiones y token de seguridad	LimeSurvey, implementa tokens CSRF que cambian constantemente durante las interacciones con el formulario	Parametrización del script
Formularios dinámicos	LimeSurvey permite incluir lógica condicional (preguntas que solo se muestran de acuerdo con las respuestas anteriores)	Establecer un flujo de información estático para todos los usuarios

En los formularios proporcionados por la entidad universitaria, que contenían elementos de captura específicos, se presentaron los siguientes problemas durante la generación de *scripts*:

Manejo de archivos adjuntos (Carga de archivos)

- LimeSurvey utiliza un modal para cargar archivos, lo que es un desafío al realizar pruebas de carga con herramientas como JMeter, ya que requiere interacción con la interfaz del formulario de manera dinámica para llevar a cabo la carga.

Gestión de sesiones y *tokens* de seguridad

- LimeSurvey implementa medidas de seguridad como los *tokens* CSRF para evitar ataques, estos *tokens* cambian constantemente durante las interacciones con el formulario, lo que dificulta la automatización de las pruebas de carga, ya que cada sesión o solicitud debe incluir un *token* válido. Si no se manejan correctamente, los *scripts* de prueba pueden fallar, ya que no pueden obtener ni mantener los *tokens* actualizados entre las interacciones.
- Si la plataforma no está configurada para mantener sesiones abiertas el tiempo suficiente para completar las pruebas, o si las sesiones expiran rápidamente debido a la inactividad, se pueden generar errores debido al "tiempo de sesión expirado".

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 99 - 107

Complejidad de formularios con condicionalidades y lógica

- LimeSurvey permite incluir lógica condicional (preguntas que sólo se muestran de acuerdo con las respuestas anteriores), lo que hace que los formularios sean dinámicos, esto aumenta la complejidad al simular múltiples usuarios.
- Las interacciones del usuario, como la selección de opciones de respuestas o la validación de campos, dependen de *scripts* JavaScript o elementos dinámicos; estas interacciones no siempre son susceptibles de automatización.

Compatibilidad con la herramienta de prueba de desempeño JMeter

- Al usar herramientas de pruebas como *JMeter*, simular múltiples usuarios simultáneos y su interacción con LimeSurvey es más complejo si el formulario depende de *JavaScript* o tiene interacciones dinámicas, debido a que no simulan la interfaz de usuario.
- LimeSurvey no está diseñado para pruebas de desempeño como prioridad, lo que significa que su integración con herramientas externas de prueba de carga o automatización (como *JMeter*) requiere trabajo exhaustivo.

Para dar solución a los problemas identificados, principalmente la complejidad en la simulación de interacciones con formularios dinámicos y la dificultad para manejar elementos como la carga de archivos, se llevó a cabo una revisión exhaustiva en diversos espacios de Internet con el objetivo de identificar soluciones y prácticas para llevar a cabo las pruebas de desempeño al formulario con *JMeter*. El proceso incluyó la consulta de recursos académicos, manuales oficiales de la herramientas LimeSurvey, artículos técnicos, así como de foros y discusiones en comunidades especializadas.

Como resultado de la revisión, se probaron diversas propuestas de solución planteadas para JMeter, sin obtener una solución que atendiera los problemas mencionados. Sin embargo, se identificaron trabajos como el de (Tufegdžić et al., 2021), en el que se aborda un marco híbrido para pruebas automatizadas a una aplicación de publicidad basada en web, con *Selenium* como componente central.

En este contexto, se decidió explorar soluciones alternativas para superar los desafíos presentados y *WebDriver* surgió como una solución viable, tras integrar el componente con *JMeter*, realizar pruebas y obtener resultados satisfactorios en la capacidad de interacción con los diferentes tipos de campos y formularios, superando la mayoría de las limitaciones previas. Esta estrategia híbrida permitió combinar la capacidad de *JMeter* para generar carga concurrente con la habilidad de *Selenium WebDriver* para controlar un navegador real, ejecutar *JavaScript* y manejar elementos dinámicos de la interfaz de usuario, para simular de forma más fiel la interacción del usuario final.

2.2.4.A IMPLEMENTACIÓN TÉCNICA

Para preparar el esquema del ambiente de automatización, se integraron diversas soluciones tecnológicas que abordaron las problemáticas presentadas en el funcionamiento de cada componente, como se muestra en la Figura 3.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 100 - 107

A continuación se detallan las soluciones implementadas:

Linux Mint

Como base del ambiente, se utilizó el sistema operativo Linux Mint 21.3, seleccionado debido a que su diseño y gestor de software permiten agilizar la instalación de paquetes, dependencias y aplicaciones.

OpenJDK

OpenJDK (*Open Java Development Kit*) es una implementación de código abierto de la plataforma Java que incluye el compilador, la máquina virtual (JVM) y las bibliotecas estándar. Proporciona un entorno libre y gratuito que cumple con los estándares de la plataforma Java y es ampliamente utilizado tanto en servidores como en aplicaciones de escritorio. Este componente permitió sustituir a JDK (software propietario de Oracle), que es requisito para el funcionamiento de *JMeter.*

WebDriver Sampler

El *WebDriver Sampler* (*Documentation :: JMeter-Plugins.org*, s. f.) permite automatizar la ejecución y recolección de métricas de desempeño en el navegador (lado del cliente), lo cual puede resolver algunos de los factores que complican la automatización mediante peticiones entre los que se encuentran:

- Ejecución de JavaScript en el lado del cliente, por ejemplo, AJAX y plantillas JS.
- Transformaciones CSS, por ejemplo, transformaciones de matriz 3D, animaciones.
- Plugins de terceros.

Este componente especializado actuó como un puente entre *JMeter* y *Selenium WebDriver*, lo que permitió integrar las capacidades de automatización de navegador de *Selenium* dentro del marco de pruebas de desempeño de *JMeter.*

Selenium WebDriver

Selenium (*The Selenium Browser Automation Project*, s. f.) es una suite de herramientas de código abierto diseñada para la automatización de pruebas en aplicaciones web. Permite simular la interacción de un usuario con una página web, al ejecutar pruebas en navegadores reales de forma automática. Soporta múltiples lenguajes de programación como Java, Python, C# y *JavaScript*, lo que lo convierte en una opción flexible para diferentes ambientes de desarrollo. Con *Selenium*, es posible automatizar acciones como hacer clic en botones, rellenar formularios, navegar entre páginas y verificar el comportamiento de la interfaz de usuario.

WebDriver es una interfaz que permite la introspección y el control de agentes de usuario. Proporciona un protocolo de comunicación independiente de la plataforma y el lenguaje para que los programas fuera de proceso instruyan remotamente el comportamiento de los navegadores web. Se ofrece un conjunto de interfaces para descubrir y manipular elementos del DOM en documentos web y controlar el comportamiento de un agente de usuario (*WebDriver*, s. f.).

Según Leotta *et al*. (2023), *Selenium WebDriver* se ha consolidado como la biblioteca de facto para desarrollar pruebas funcionales de extremo a extremo (E2E) de aplicaciones web.

Esta herramienta brindó la interfaz para controlar los navegadores mediante código al simular las interacciones reales de los usuarios en el navegador.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 101 - 107

Driver **(controlador)**

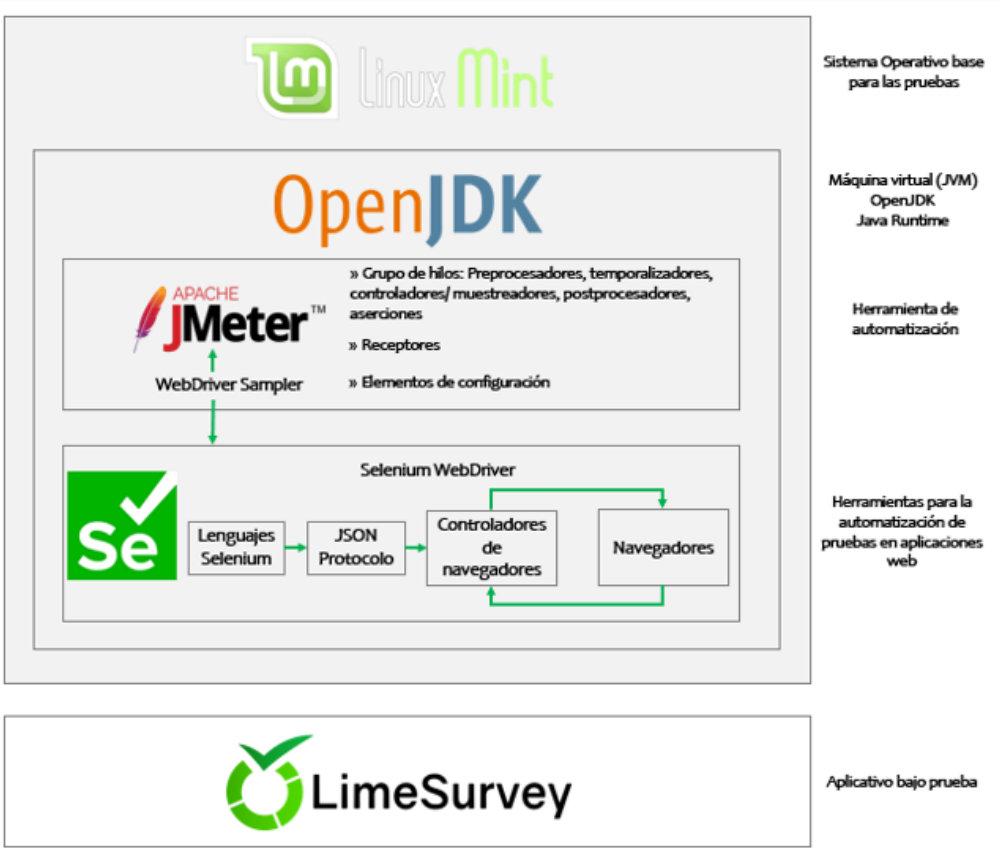
Para funcionar, *Selenium WebDriver* requiere un componente intermedio llamado *driver*, que controla el navegador y permite la interacción con la aplicación web. Cada proveedor de navegadores ofrece un *driver* específico para cada versión del navegador, lo que implica que los desarrolladores deben mantener alineadas las versiones del navegador y del *driver* para asegurar la compatibilidad (Leotta et al., 2023).

En este sentido, el *driver* desempeñó un papel esencial como intermediario entre *Selenium WebDriver* y el navegador. Este controlador implementó el protocolo W3C WebDriver y permitió que *Selenium WebDriver* se comunicara con Firefox, tradujo los comandos en acciones que el navegador pudo ejecutar, y proporcionó una capa de abstracción que facilitó la compatibilidad y estabilidad de la automatización.

Con la integración de estas tecnologías, se generó un ambiente de pruebas robusto y versátil; Linux Mint facilitó la instalación de OpenJDK, requisito de *JMeter*; el *WebDriver Sampler* proporcionó la infraestructura dentro de *JMeter*; *Selenium WebDriver* ofreció las capacidades de automatización del navegador; y el *driver* aseguró la comunicación con el navegador.

Figura 3

Arquitectura del ambiente para las pruebas de desempeño al formulario en LimeSurvey



https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 102 - 107

Una vez integrada la solución tecnológica para abordar la problemática presentadas en el *script* de automatización, se presentó un nuevo desafío significativo: aprender a codificar para *WebDriver*.

2.2.4.B CODIFICACIÓN

Otro de los retos en este proceso fue aprender a codificar en la interfaz, ya que requería la comprensión de su estructura y cómo interactuar con los navegadores. Para superar esta barrera, se dedicó tiempo a revisar diversas fuentes de información, como documentación oficial, tutoriales en línea, foros de desarrolladores y ejemplos prácticos. Si bien existen fuentes formales del tema, los métodos y recomendaciones no son exactos debido al cruce de tecnologías, por tal motivo fue necesario adoptar un enfoque de prueba y error[3], mismo que permitió experimentar directamente con el código para identificar la lógica de programación y los métodos para llevar a cabo las interacciones necesarias, así como adaptarlo al flujo de trabajo establecido.

Para la codificación, se determinó utilizar *Groovy*, el cual es un lenguaje de programación dinámico que se ejecuta sobre la máquina virtual de Java (JVM), que simplifica la escritura de *scripts* debido a su sintaxis concisa y fácilmente legible.

A partir del cambio de enfoque, la construcción de *scripts* cambió drásticamente, por lo que fue necesario escribir el código para cada interacción en el aplicativo. Esto implicó una nueva revisión del formulario para recolectar información de cada uno de los elementos involucrados en el flujo de trabajo: nombres, etiquetas, identificadores, argumentos y rutas, entre otros; así como los diversos eventos requeridos para replicar la funcionalidad: clics, desplazamientos, ingreso de datos, validaciones, entre otros movimientos.

Con los insumos identificados y registrados, la generación de *scripts* se llevó a cabo con la incorporación secuencial de los movimientos de cada sección del formulario, por lo que fue necesario validar y, en su caso, buscar varios elementos de interacción en tiempo de ejecución.

Es importante mencionar que esta solución resolvió únicamente los problemas asociados a elementos de captura solicitados como parte de la misma página, sin solventar la funcionalidad para adjuntar archivos.

Finalmente, se añadieron espacios de tiempo entre los movimientos a realizar, con la finalidad de que los elementos del formulario se encontraran presentes al interactuar con ellos.

2.2.5 PROBAR EL SCRIPT Y AJUSTARLO

A causa de los problemas presentados y a las soluciones generadas, en conjunto con la entidad universitaria, se acordó como estrategia para abordar las pruebas: cambiar el alcance de la automatización a la resolución del cuestionario y omitir los reactivos asociados a la importación de archivos.

3 La *prueba y error* es un proceso no lineal que implica experimentar repetidamente y explorar diferentes operaciones para aprender y utilizar aplicaciones de software complejas. Este método se basa en un ciclo de ejecución, evaluación y recuperación, permite a los usuarios identificar dificultades y explorar nuevas rutas para alcanzar sus objetivos, (Barbosa, 2022).

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 103 - 107

2.2.6 PRUEBA PILOTO DEL SCRIPT PARA DETERMINAR EL ESQUEMA DEL AMBIENTE DE AUTOMATIZACIÓN DE LAS PRUEBAS

Una vez que el *script* de automatización funcionó adecuadamente con el nuevo alcance, se llevaron a cabo un conjunto de ejecuciones de éste para identificar la cantidad máxima de usuarios que se permitía simular desde el esquema del ambiente de automatización.

El máximo de usuarios que se pudieron simular por equipo fueron 20. Para lograr el objetivo esperado en las pruebas, se tuvo que replicar el esquema del ambiente de automatización en cinco equipos de cómputo.

Con la finalidad de poder distribuir las pruebas en diversos equipos, se generó una máquina virtual con la estructura del ambiente de pruebas, la cual integró las herramientas de automatización de pruebas de desempeño.

2.3 APLICACIÓN DE LAS PRUEBAS: MANUALES Y AUTOMATIZADAS

La aplicación de las pruebas estuvieron basadas y apoyadas en la planeación y en los productos de trabajo de control preparados en el diseño, para lo cual fue necesario que el formulario se encontrara en un ambiente análogo al productivo en cuanto a infraestructura y configuración se refiere.

Alineados a la nueva estrategia para realizar las pruebas de desempeño, en seguida se describen las actividades para la aplicación de las pruebas manuales y automatizadas.

2.3.1 APLICACIÓN DE LAS PRUEBAS MANUALES

Debido a la inviabilidad de automatizar las acciones para adjuntar archivos y a que el formulario presenta ventanas modales, tanto web como de sistema operativo, para esta tarea, se aplicó la prueba de manera manual, en conjunto con el personal de la entidad universitaria, la cual consistió en que, durante una hora, diversos usuarios realizaron registros de manera constante al formulario completo, incluida la importación de archivos de diferente extensión y tamaño. Para complementar la prueba y revisar la concurrencia, 10 usuarios ingresaron la información solicitada y, de manera relativamente simultánea, enviaron el registro.

Estas pruebas permitieron observar, desde la perspectiva del usuario final, el comportamiento del formulario e identificar el número de usuarios concurrentes que responde la plataforma. Adicionalmente, al personal de la entidad universitaria le permitió identificar acciones de respuesta en caso de materializarse el riesgo de que el formulario no responda adecuadamente a todos los usuarios por cuestiones de concurrencia.

2.3.2 APLICACIÓN DE LAS PRUEBAS AUTOMATIZADAS

Las pruebas automatizadas se realizaron a un formulario que omitió únicamente la funcionalidad para adjuntar archivos.

La aplicación de estas pruebas se llevó a cabo en un ciclo iterativo de *generación de carga*, *recolección y análisis de los datos* y *resolución de los posibles problemas*.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 104 - 107

En la *generación de carga,* se definió y aplicó la prueba con las características acordadas para cada prueba respecto al número de usuarios en concurrencia, tiempos de espera, entre otros elementos establecidos.

En la *recolección y análisis de los datos*, durante la ejecución de cada prueba, se monitoreó el comportamiento del sistema desde la interfaz visual de la herramienta de automatización, se observaron tanto las respuestas proporcionadas a los usuarios, como el registro en el *log* de la herramienta de automatización; al concluir la prueba, se verificó el registro de la información mediante la misma herramienta LimeSurvey. Adicionalmente, apoyados de personal de la entidad universitaria y en constante comunicación, se compartió la información del consumo de recursos de red, CPU y memoria RAM. Al término de cada prueba, se compartieron los resultados y, en conjunto, se determinó si la prueba fue exitosa o fallida[4].

En los casos donde los resultados de la prueba no fueron los esperados, en conjunto con el personal de la entidad solicitante y de un grupo de especialistas en redes y servidores, se verificó la causa-raíz del incidente y se determinaron e implementaron los pasos a seguir, tales como realizar ajustes pertinentes en la configuración en la infraestructura que soportaba el cuestionario.

Un factor de éxito fue llevar el control de las pruebas aplicadas, resultados obtenidos del monitoreo de recursos y el análisis de los mismos, lo que permitió comparar los resultados entre pruebas y determinar la mejor configuración para que la infraestructura responda adecuadamente al mayor número de usuarios en concurrencia.

2.4 CIERRE

Para concluir las pruebas, se llevó a cabo el análisis y la interpretación de los resultados obtenidos en cada una de las pruebas aplicadas y se documentaron en un informe que fue entregado a la entidad universitaria.

Como buenas prácticas, se realizó el *testware* del proyecto, el cual es el conjunto de productos de trabajo producidos durante el proceso de prueba para su uso en la planeación, diseño, ejecución, evaluación e informes sobre las pruebas. De acuerdo con Sosnówka (2013), el *testware* permite una gestión más eficiente de las pruebas, ya que proporciona una estructura organizada de todos los artefactos necesarios para planificar, diseñar y ejecutar pruebas, lo que facilita la identificación y resolución de problemas.

3. RESULTADOS

La aplicación de pruebas de desempeño al formulario desarrollado en la plataforma LimeSurvey presentó múltiples retos en la automatización, entre los cuales se destacan: la dificultad para abordar las pruebas debido al desconocimiento de la estructura del código fuente, ya que el desarrollo de la herramienta fue realizado por personal externo a los involucrados en las pruebas, y el tiempo necesario para encontrar una solución adecuada.

En la generación del *script* de automatización en *JMeter*, se presentaron problemas en el manejo de archivos adjuntos (carga de archivos), en la gestión de sesiones y uso *tokens* de seguridad, en las

4 Una *prueba exitosa* es en la que sus resultados corresponden a los esperados, mientras que una *prueba fallida* es aquella que no es concluida, presenta errores y/o no cumplen con los criterios de aceptación definidos.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 105 - 107

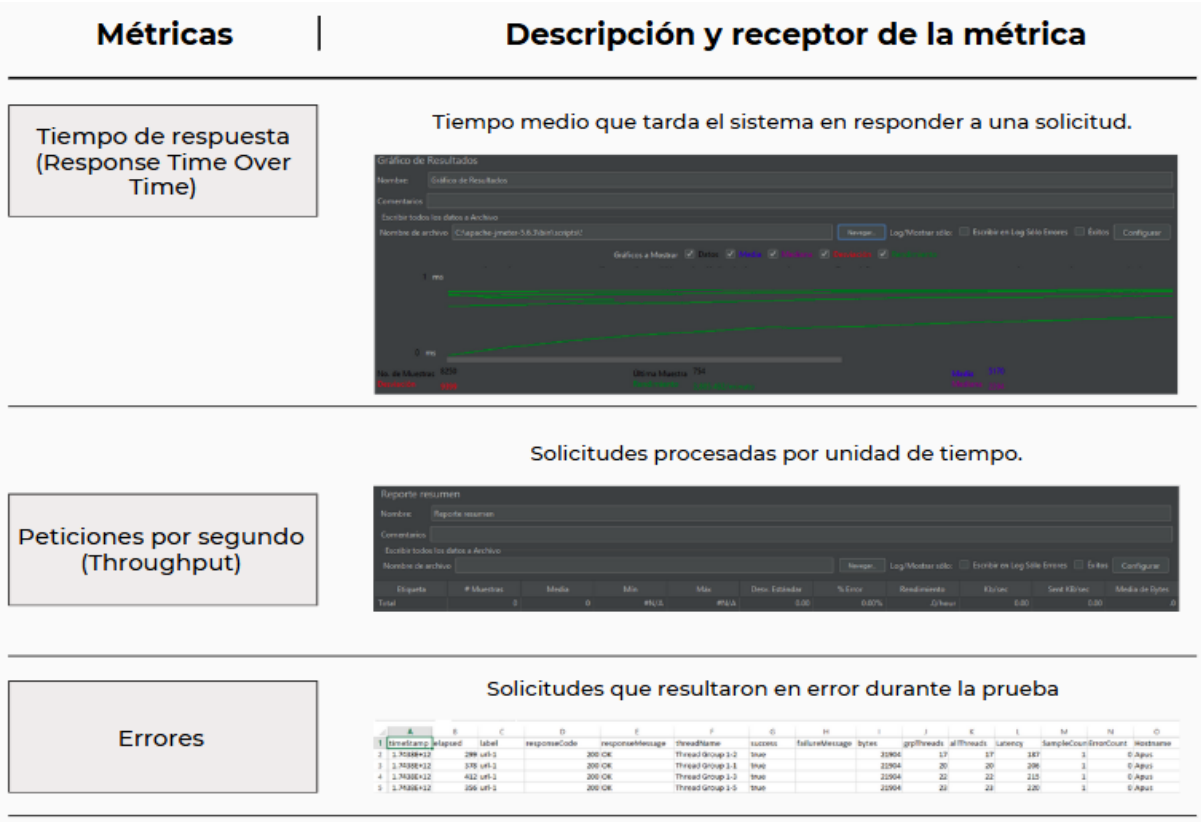
condiciones y lógica del formulario, así como en el funcionamiento interno de LimeSurvey que no era compatible con la herramienta de prueba de desempeño.

Aunque durante el proceso de preparación de las pruebas se presentaron algunos impedimentos que obstaculizaron la automatización, se abordaron la mayoría de problemas presentados al integrar diversas soluciones tecnológicas a *JMeter*, como *OpenJDK*, *WebDriver Sampler*, *Selenium WebDriver* y el *driver*. El único incidente que no fue resuelto fue la automatización en preguntas que requerían adjuntar archivos debido a que se solicitaban mediante ventanas modales dinámicas en *JavaScript*.

La Figura 4 presenta las principales métricas utilizadas para analizar los resultados obtenidos de las pruebas.

Figura 4

Algunas métricas de JMeter consultadas para el análisis de resultados de las pruebas



Ante la complejidad técnica de la solución implementada, se ajustó la estrategia para combinar pruebas de desempeño manuales y automatizadas, lo que permitió cumplir con el objetivo de conocer el comportamiento de la aplicación bajo condiciones específicas de concurrencia. Aunque la solución integrada requería un elevado consumo de recursos tecnológicos para simular la concurrencia, fue favorable que el número de usuarios a simular fuera relativamente pequeño y, por tanto, compatible con los recursos disponibles.

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 106 - 107

Gracias a esta adaptación, se aplicaron las pruebas con los recursos tecnológicos y humanos disponibles, lo que garantizó resultados confiables y demostró que, incluso con restricciones técnicas, es posible alcanzar los objetivos planteados mediante una planificación flexible y eficiente.

La aplicación de pruebas de desempeño manuales permitieron observar el comportamiento del formulario al ser resuelto desde la perspectiva del usuario final, en carga de 27 usuarios al registrar información constante y adjuntar archivos de diversos tamaños y extensiones, mientras que las pruebas automatizadas mostraron el comportamiento del formulario en concurrencia de 100 usuarios.

4. CONCLUSIONES

Como elementos importantes a considerar que apoyarán a nuevos proyectos que involucren el desarrollo de pruebas de desempeño, destaca tener presente que automatizar toda la funcionalidad no siempre es la mejor solución para conocer el comportamiento del software. En este sentido, al considerar la automatización como única alternativa para las pruebas, es fundamental reconocer que cada funcionalidad puede requerir de un enfoque diferente y el uso de herramientas específicas.

La solución implementada para abordar las pruebas de desempeño al formulario en LimeSurvey, que integra *WebDriver* en *JMeter*, se enmarcó dentro de una metodología que permite una evaluación más precisa y detallada del comportamiento de la aplicación al simular las interacciones del usuario en un ambiente controlado de pruebas.

El esquema planteado permite replicar de manera fidedigna las acciones de los usuarios, como clics, desplazamientos y entradas de texto, lo cual es necesario al analizar aplicaciones que requieren una interacción dinámica. Además, es eficaz para aplicaciones que emplean tecnologías como *AJAX* o *JavaScript*, cuyas interacciones no pueden ser capturadas de manera adecuada en pruebas que sólo involucran solicitudes HTTP (como lo hace *JMeter* de manera independiente). Sin embargo, esta integración presenta limitaciones tecnológicas inherentes a su ejecución, la principal es que las pruebas basadas en *WebDriver* son considerablemente más costosas en términos de recursos y tiempo debido a la necesidad de interactuar con un navegador real, lo que aumenta la complejidad de la ejecución, especialmente cuando se requiere simular una gran cantidad de usuarios simultáneos.

El proyecto permitió adquirir nuevos conocimientos y destacó la importancia de considerar la integración de nuevas herramientas para la automatización de soluciones tecnológicas actuales. Esta experiencia no sólo amplió nuestra comprensión sobre las mejores prácticas en automatización, sino que también resaltó la necesidad de adaptar las herramientas a las características específicas de las aplicaciones.

REFERENCIAS

Apache JMeter—*Apache JMeter™*. (s. f.). Recuperado 23 de junio de 2025, de https://jmeter.apache.org/

Barbosa, S. (2022). *CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3491102

Bath, G., Black, R., Podelko, A., Pollner, A., & Rice, R. (2018). *Foundation Level Specialist Syllabus Performance Testing*.

Crear encuestas: LimeSurvey Herramienta de encuestas gratuita. (s. f.). Recuperado 23 de junio de 2025, de https://www.limesurvey.org/es

https://doi.org/10.22201/dgtic.30618096e.2025.3.3.123 **Vol. 3, Núm. 3.** julio-septiembre 2025, págs. 107 - 107

Documentation: JMeter-Plugins.org. (s. f.). Recuperado 23 de junio de 2025, de https://jmeter-plugins.org/wiki/WebDriverSampler/

Echeverria, D., Skrilec, G., Verma, R., Herrera, A., Fernández, A., Vivanco, A., Acevedo, Á. R., Brands, A., Acosta, B. M., Tolosa, D., Delgado, E. R., Sales, E. E., Henostroza, G., Sosa, G. M., Terrera, G., Revalcaba, H., Ortiz, J. P., Rios, J. P., Melendez, L., … Nane, S. (2019). *PtU Certified Performance Tester con JMeter (CPTJM)*.

Holguín Carrillo, R. (2020). Una caja de herramientas para medir el universo de protestas en México. *Revista Digital Universitaria*, *21*(3). https://doi.org/10.22201/codeic.16076079e.2020.v21n3.a2

International Organization for Standardization, & International Electrotechnical Commission. (2011). *ISO/IEC 25010:2011: Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models*. ISO/IEC.

Legramante, G., Bernardino, M., Rodrigues, E. M., & Basso, F. (2020). Systematic Literature Review on Web Performance Testing. *Anais Da IV Escola Regional de Engenharia de Software (ERES 2020)*, 285-295. https://doi.org/10.5753/eres.2020.13739

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 Conference on Software Testing, Verification and Validation (ICST)*, 339-350. https://doi.org/10.1109/ICST57152.2023.00039

Mărcuță, C. (2024, diciembre 5). *Understanding the Significance of Performance Testing and Why It is Essential for Your Software's Success*. https://moldstud.com/articles/p-understanding-the-significance-of-performance-testing-and-why-it-is-essential-for-your-softwares-success

Molyneaux, I. (2014). *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly Media.

Sosnówka, A. (2013). Testware Visualized—Visual Support for Testware Reorganization: *Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering*, 109-114. https://doi.org/10.5220/0004451001090114

The Selenium Browser Automation Project. (s. f.). Selenium. Recuperado 23 de junio de 2025, de https://www.selenium.dev/documentation/

Tufegdžić, M., Miodragović, G., & Aleksandrov, S. (2021). *Hybrid framework for automated testing of web application for advertisement*. Conference: Young science - Robotics and nano-technology of modern mechanical engineering, Donbass State Engineering Academy, Kramatprsk.

WebDriver. (s. f.). Recuperado 10 de abril de 2025, de https://www.w3.org/TR/webdriver1/

What is Performance Testing? The Complete Guide. (s. f.). *ImpactQA*. Recuperado 10 de abril de 2025, de https://www.impactqa.com/guides/performance-testing/

World Quality Report 2023-24. (2023, diciembre 8). *Capgemini*. https://www.capgemini.com/insights/research-library/world-quality-report-2023-24/