

Universidad Mariano Gálvez de Guatemala

Facultad de Ingeniería en Sistemas

Curso: Aseguramiento de Calidad de Software

Catedrático: Ing. Carmelo Estuardo Mayen Monterroso

Semestre: Decimo



Contenido: Tipos de Prueba

Alumno: Sergio Otoniel Crocker Cristales

Carnet: 1790.21.8869

Chiquimulilla Santa Rosa 2025

Introducción

El presente trabajo tiene como finalidad realizar una investigación detallada sobre tres tipos fundamentales de pruebas de software: pruebas funcionales, pruebas de rendimiento y pruebas de seguridad. Estas pruebas son esenciales en el ciclo de vida del desarrollo de software (SDLC) porque permiten validar que el producto cumpla con las especificaciones, se desempeñe adecuadamente bajo condiciones reales y esté protegido contra posibles ataques.

En la actualidad, el mercado exige aplicaciones de alta calidad, con interfaces confiables, tiempos de respuesta óptimos y mecanismos de seguridad robustos. Implementar estas pruebas desde las primeras etapas del desarrollo reduce costos, evita fallos en producción y aumenta la satisfacción del usuario final.

Pruebas Funcionales

Definición y Objetivo:

Las pruebas funcionales se centran en verificar que cada función del software opere conforme a las especificaciones definidas en los requisitos. Evalúan la interacción del usuario con el sistema, el flujo de datos y las salidas generadas.

Son esenciales para garantizar la correcta implementación de las funcionalidades críticas.

Tipos comunes:

- Pruebas de humo (Smoke Testing)
- Pruebas de regresión
- Pruebas de integración
- Pruebas de aceptación del usuario (UAT)

Herramientas:

1. Selenium: Herramienta open-source para automatizar navegadores. Compatible con múltiples lenguajes (Java, Python, C#) y navegadores (Chrome, Firefox, Edge). Permite ejecutar pruebas en entornos locales o en la nube.
2. TestComplete: Herramienta comercial de SmartBear que soporta aplicaciones web, móviles y de escritorio. Ofrece grabación de acciones, scripts y ejecución distribuida.

Comparativa:

- Selenium: Gratuito, personalizable, ideal para equipos con experiencia técnica.
- TestComplete: Pago, más amigable para usuarios sin amplia experiencia en programación.

Aplicaciones prácticas:

Se usan para validar flujos completos, como un proceso de compra online, registro de usuario o gestión de transacciones bancarias.

Pruebas de Rendimiento

Definición y Objetivo:

Las pruebas de rendimiento analizan la capacidad de respuesta, estabilidad y uso de recursos del software bajo diferentes cargas de trabajo. Su objetivo es identificar cuellos de botella y optimizar el comportamiento antes de que el producto llegue al usuario final.

Tipos de pruebas de rendimiento:

- Pruebas de carga: Miden el rendimiento con usuarios esperados.
- Pruebas de estrés: Evalúan la respuesta bajo cargas extremas.
- Pruebas de estabilidad (soak testing): Determinan la fiabilidad durante periodos prolongados.
- Pruebas de volumen: Analizan el manejo de grandes volúmenes de datos.
- Pruebas de picos: Observan el comportamiento ante incrementos súbitos de usuarios.

Herramientas:

1. Apache JMeter: Open-source, soporta múltiples protocolos como HTTP, FTP, JDBC. Permite ejecutar pruebas distribuidas y generar reportes gráficos.
2. LoadRunner: Herramienta comercial que simula miles de usuarios virtuales y ofrece análisis detallados del rendimiento.

Comparativa:

- JMeter: Gratuito, flexible, requiere mayor configuración.
- LoadRunner: Pago, interfaz más intuitiva para pruebas a gran escala.

Aplicaciones prácticas:

Se emplean para validar el tiempo de respuesta de una API, evaluar un sistema de reservas en línea o probar la escalabilidad de un e-commerce durante eventos como Black Friday.

Pruebas de Seguridad

Definición y Objetivo:

Las pruebas de seguridad buscan identificar vulnerabilidades que puedan ser explotadas por atacantes, comprometiendo la confidencialidad, integridad o disponibilidad del sistema.

Principales amenazas detectadas:

- Inyección SQL
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Configuraciones inseguras
- Fugas de información

Herramientas:

1. OWASP ZAP: Herramienta open-source que incluye escaneo automatizado y manual, proxy interceptador y soporte para scripting.
2. Burp Suite: Herramienta comercial muy utilizada por pentesters, con módulos como escáner, intruder y repetidor.

Comparativa:

- ZAP: Gratuita, ideal para integrarse en pipelines DevSecOps.
- Burp Suite: Pago, más completa y precisa para auditorías profundas.

Aplicaciones prácticas:

Se utilizan para auditar portales web gubernamentales, sistemas bancarios o plataformas de comercio electrónico, detectando y corrigiendo vulnerabilidades antes de que sean explotadas.

Conclusiones

Implementar pruebas funcionales, de rendimiento y de seguridad es fundamental para garantizar la calidad de un software. La elección de la herramienta adecuada dependerá del presupuesto, la complejidad del proyecto y la experiencia del equipo. Las pruebas continuas reducen riesgos, mejoran la experiencia del usuario y fortalecen la reputación de la organización.

Recomendaciones

- Realizar una evaluación previa de las necesidades del proyecto antes de seleccionar la herramienta.
- Integrar las pruebas en cada fase del ciclo de vida del desarrollo.
- Capacitar continuamente al equipo de QA y desarrollo.
- Documentar y analizar los resultados para tomar decisiones de mejora.

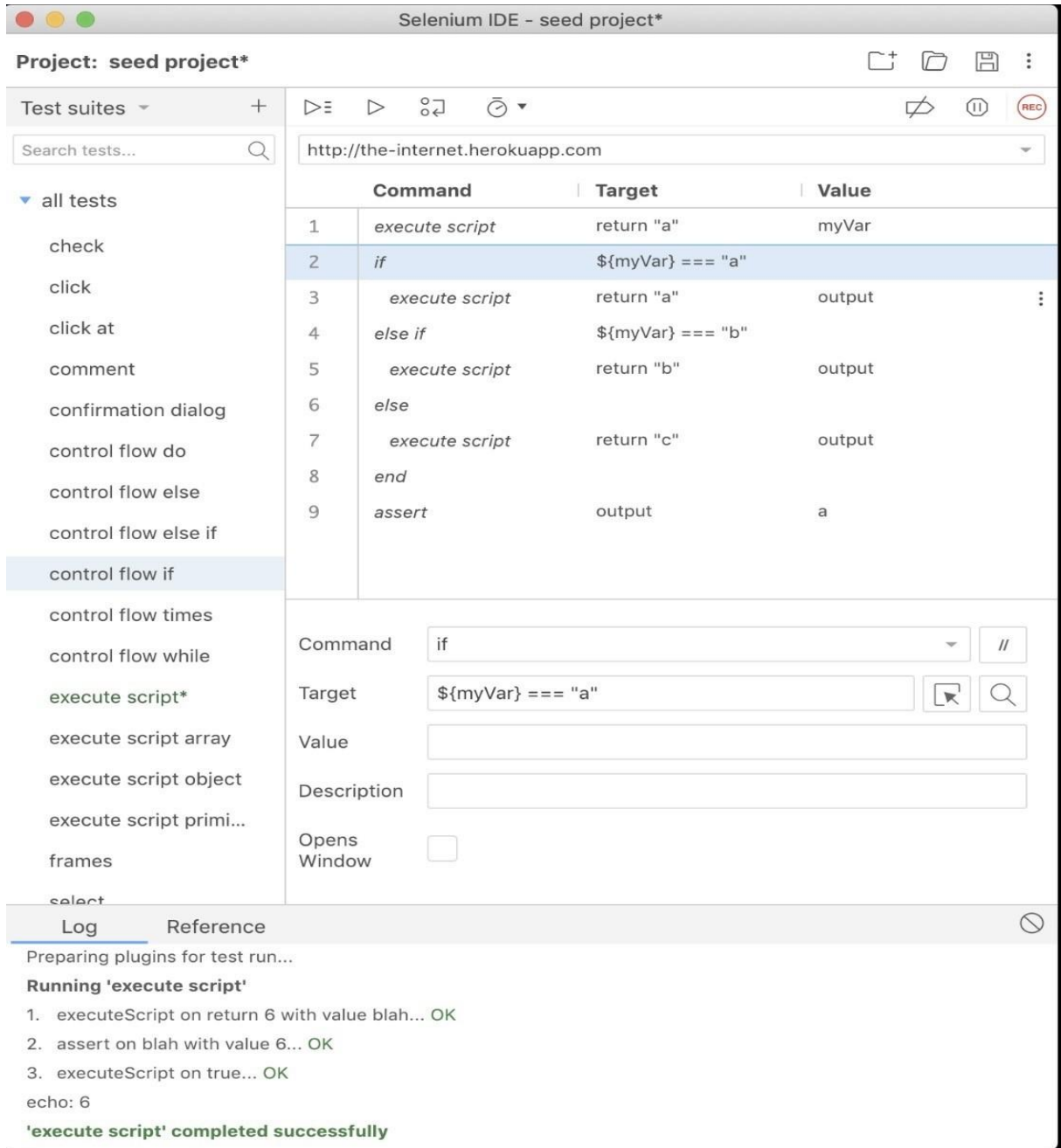
Bibliografía

<https://applitools.com/blog/functional-testing-guide/>

<https://www.browserstack.com/guide/performance-testing>

Anexos

Interfaz de Selenium IDE mostrando un flujo de comandos en un proyecto de prueba.



Selenium IDE - seed project*

Project: seed project*

Test suites +

Search tests...

all tests

- check
- click
- click at
- comment
- confirmation dialog
- control flow do
- control flow else
- control flow else if
- control flow if**
- control flow times
- control flow while
- execute script*
- execute script array
- execute script object
- execute script primi...
- frames
- select

http://the-internet.herokuapp.com

	Command	Target	Value
1	execute script	return "a"	myVar
2	if	\${myVar} === "a"	
3	execute script	return "a"	output
4	else if	\${myVar} === "b"	
5	execute script	return "b"	output
6	else		
7	execute script	return "c"	output
8	end		
9	assert	output	a

Command: if

Target: \${myVar} === "a"

Value:

Description:

Opens Window: ☐

Log Reference

Preparing plugins for test run...

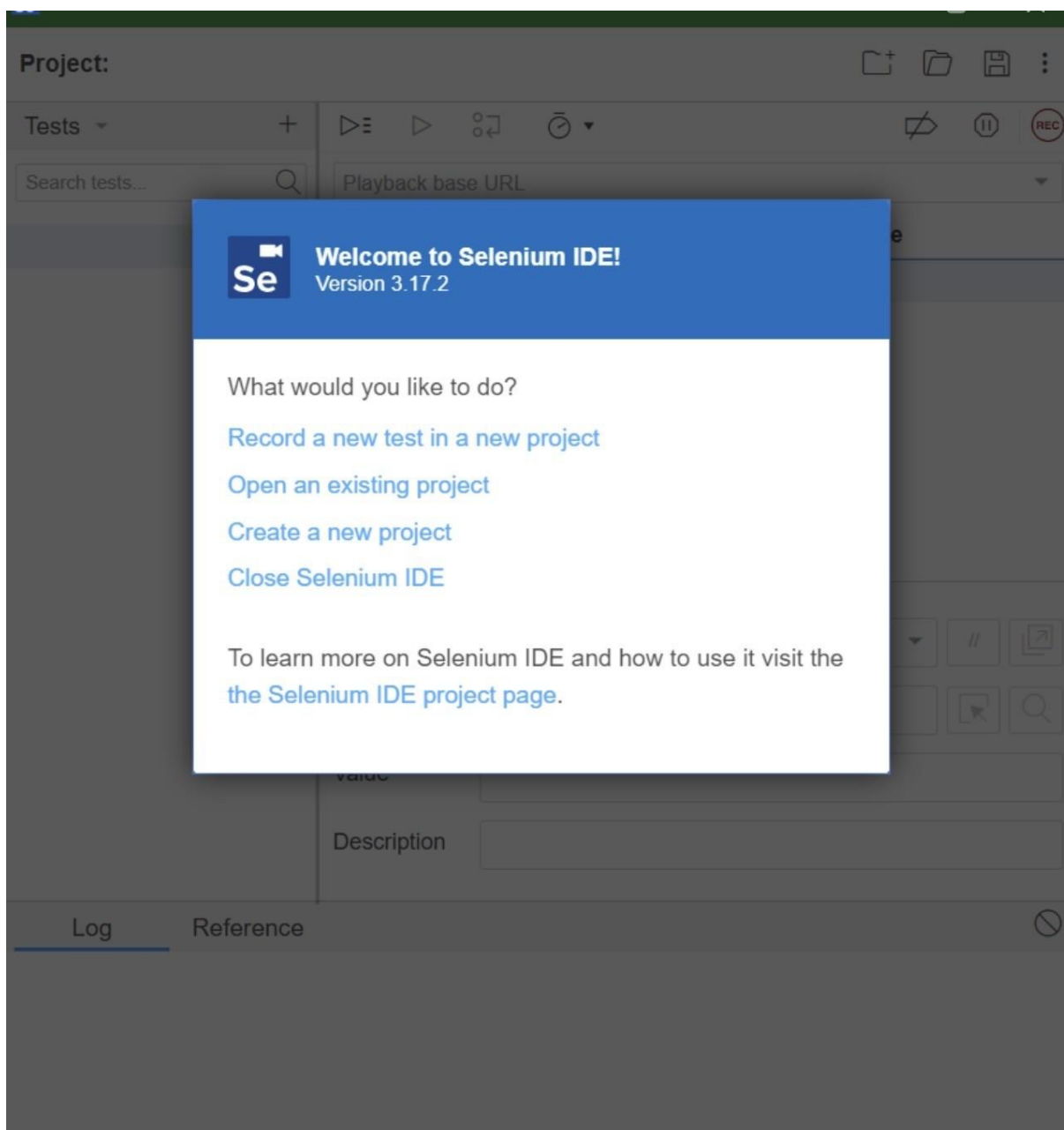
Running 'execute script'

- executeScript on return 6 with value blah... OK
- assert on blah with value 6... OK
- executeScript on true... OK

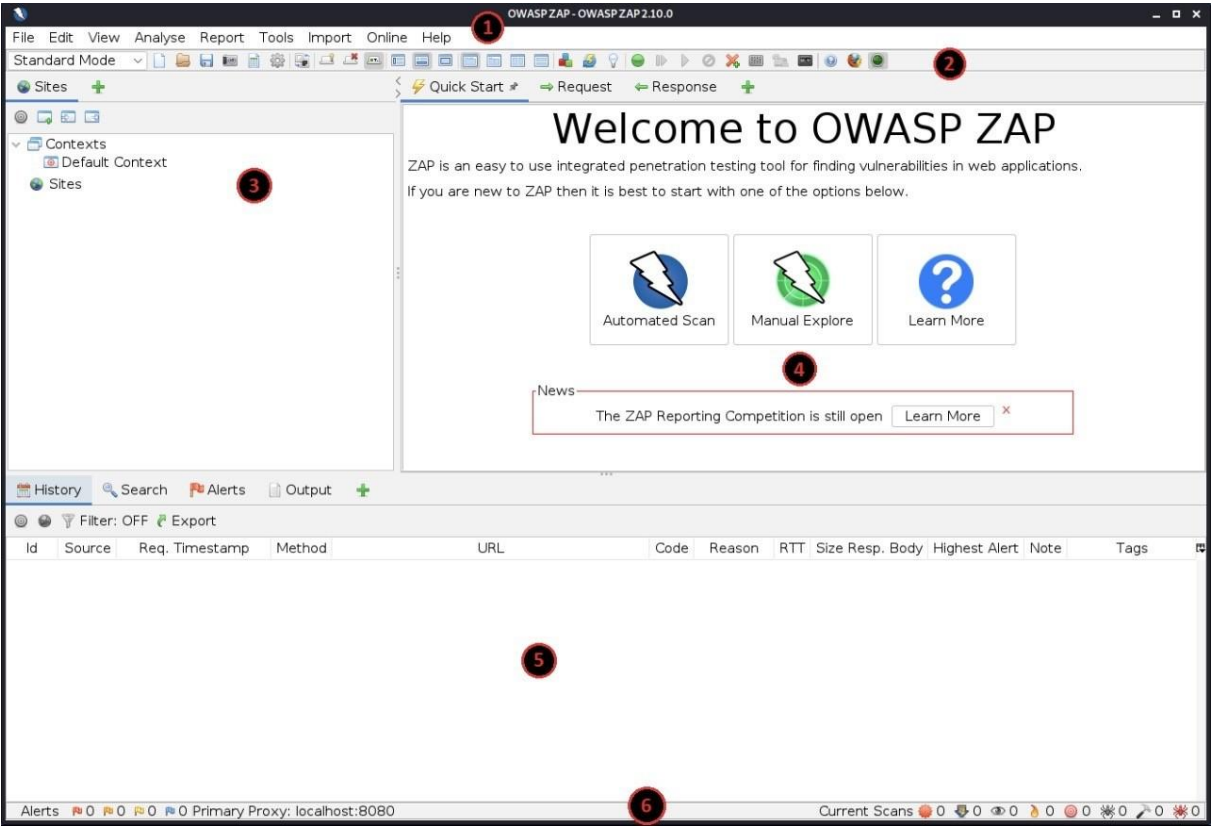
echo: 6

'execute script' completed successfully

Pantalla de bienvenida de Selenium IDE para iniciar o abrir proyectos de prueba.



Pantalla principal de OWASP ZAP con opciones de escaneo automatizado y manual.



Otra vista de OWASP ZAP mostrando su estructura de navegación y funciones.

