A blue and yellow logo with white text

Description automatically generated  
*Universidad Don Bosco*

*Facultad de Ciencias de la Computación*

*Entrega de Investigación 02*

*Desarrollo de Aplicaciones Web con Software*

*Interpretado en el Cliente   
Enrique Alexander Solano López SL223188*

*Norman Enmanuel Quijano Amaya QA240640*

*Manuel Ezequiel guerrero granados GG241501*

**Introducción**

En el desarrollo de aplicaciones web, las herramientas de Testing juegan un papel crucial para asegurar la calidad y funcionalidad de los sistemas. Especialmente en el contexto de aplicaciones web con JavaScript, estas herramientas permiten realizar pruebas automáticas para detectar errores, verificar el comportamiento esperando y asegurar la interacción correcta unitarias, de integración, de aceptación, funcionales y rendimiento, que pueden realizarse en las aplicaciones web.

Selenium es una de tantas herramientas mas conocidas para la automatización de pruebas en aplicaciones web. Esta diseñado para simular el comportamiento de un usuario interactuando con la interfaz grafica del navegador, proporcionando una forma robusta de realizar pruebas funcionales en diferentes entornos y plataformas.

Descripción detallada

Selenium es una herramienta de software que se utiliza para automatizar navegadores web. Ya que permite realizar pruebas automáticas a aplicaciones web, pero también se puede usar para tareas repetitivas de interacción con un navegador, como la extracción de datos, la automatización de formularios y la simulación de interacciones de usuario.

Sus usos más comunes de Selenium son los siguientes:

* Pruebas automatizadas: se usa Selenium para probar aplicaciones web, asegurando que funcionen correctamente después de realizar cambios en el código.
* Web Scraping: Selenium se puede usarse para extraer información de sitios web, en especial en sitios web en donde el contenido se carga dinámicamente mediante JavaScript.
* Automatización de tareas: puede automatizar tareas repetitivas, como la navegación web por sitios web, la descarga de archivos, la interacción con formularios, etc.

Selenium cuenta con diferentes componentes principales que ayudan en la automatización. Tales como:

* Selenium Web Driver: herramienta principal que permite controlar el navegador de manera programática. Puedes escribir scripts en leguajes como java, Python, c#, Ruby, etc. Para interactuar con el navegador, simular clics, rellenar formularios, etc. También es compatible con pruebas móviles y permite una mayor flexibilidad para ejecutar pruebas más complejas.
* Selenium IDE: es un entorno de desarrollo integrado para Selenium, que permite grabar, editar y depurar pruebas en el navegador. Es una herramienta sencilla para crear casos de prueba sin necesidad de escribir código.
* Selenium Grid: permite distribuir y ejecutar pruebas en múltiples maquinas y entornos en paralelo, lo cual es útil para realizar pruebas en diferentes navegadores y sistemas operativos al mismo tiempo.

Selenium cuenta con diferentes tipos de pruebas que puede soportar tales como:

* Pruebas funcionales: verifican que las funcionalidades de la aplicación se comporten correctamente, simulando la interacción del usuario don la interfaz (clics, entrada de texto, navegación etc.).
* Pruebas de regresión: aseguran que nuevas actualizaciones o cambios en el código no afecten funcionalidades ya existentes.
* Pruebas de integración: verifican la correcta integración entre componentes de la aplicación web.

Cuenta con las siguientes integraciones disponibles como:

* Jenkins: Integración continua para ejecutar pruebas automáticas después de cada despliegue.
* Junit/TestNG: Frameworks de testing en Java para estructurar y ejecutar casos de prueba.
* BrowserStack y Sauce Labs: Plataformas de testing en la nube que permiten ejecutar pruebas en una variedad de dispositivos y navegadores.
* Allure: generación de reportes detallados de ejecución de pruebas.

**Ejemplos prácticos**

A continuación, un ejemplo básico de como realizar una prueba automatizada con Selenium web Driver en JavaScript usando node.js

Primero es la instalación de Selenium web driver:

npm install selenium-webdriver

una ves instalada, se procede a crear el siguiente código en JavaScript:

const {Builder, By, Key, until} = require('selenium-webdriver');

(async function example() {

let driver = await new Builder().forBrowser('chrome').build();

try {

await driver. Get('https://www.google.com');

await driver. find Element (By.name('q')). sendKeys ('Selenium', Key. RETURN);

await driver. wait(until. title Contains('Selenium'), 1000);

console.log ("Test Passed");

} finally {

await driver. quit();

}

})();

Ahora se inicia en un navegador predeterminado, se navega en Google, se busca Selenium, y se verifica que el titulo de la página contenga “Selenium”

Otro ejemplo básico con Selenium Web Driver podría involucrar la automatización de la carga de un sitio web, la localización de un elemento en la página (como un logotipo), y la verificación de su visibilidad. En un lenguaje como C#, se define un controlador (driver) y se escribe un método de prueba con las anotaciones correspondientes. Posteriormente, se usa una afirmación (assert) para validar si el elemento encontrado coincide con el esperado​.

Comparativa

Selenium tiene competidores notables en el ámbito de pruebas automatizadas para aplicaciones web, Selenium destaca frente a otras herramientas de automatización tales como:

* Cypress: Es una herramienta moderna diseñada específicamente para aplicaciones JavaScript. Tiene ventajas como una curva de aprendizaje más rápida, ejecución de pruebas más veloz y un entorno de desarrollo de pruebas más integrado. Sin embargo, tiene menos soporte para navegadores fuera de Chrome y Firefox y carece de soporte para pruebas distribuidas en múltiples máquinas.
* Puppeteer: Una biblioteca de automatización de navegador para Google Chrome y Chromium. Es más rápida y ligera que Selenium, pero está limitada a esos dos navegadores y es menos flexible para trabajar con otros lenguajes de programación.
* Playwright: Una herramienta de Microsoft que permite la automatización de navegadores como Chromium, Firefox y WebKit. Similar a Puppeteer pero con soporte multiplataforma y funciones más avanzadas como el manejo automático de múltiples pestañas y contexto de navegación.

Aunque herramientas como Cypress ofrecen una mayor simplicidad en la configuración y pruebas más rápidas debido a su arquitectura más moderna, Selenium sigue siendo la opción preferida para proyectos grandes que requieren pruebas en múltiples entornos.

Mejores prácticas y consideraciones

Al seleccionar una herramienta de pruebas automatizadas como Selenium, es importante considerar varios factores:

* Compatibilidad con múltiples navegadores: si la aplicación necesita ser probada en diversos navegadores, Selenium es una opción adecuada debido a su amplia compatibilidad.
* Complejidad del proyecto: esto es para proyectos simples o para aplicaciones JavaScript modernas, herramientas como Cypress o Playwright pueden ofrecer una experiencia más sencilla y rápida.
* Facilidad de integración en CI/CD: Si es crucial para tu equipo tener una integración sólida en un pipeline de integración continua (como Jenkins), Selenium es ampliamente soportado y probado en este ámbito.
* Curva de aprendizaje: Herramientas como Selenium requieren más configuración y experiencia técnica, mientras que Cypress o Selenium IDE son más accesibles para principiantes.
* Entornos distribuidos: Si necesitas ejecutar pruebas en paralelo en diferentes máquinas o navegadores, Selenium Grid ofrece soluciones avanzadas para pruebas distribuidas.

Selenium es ideal para proyectos que requieren alta personalización y compatibilidad entre múltiples lenguajes, pero herramientas como Cypress podrían ser más adecuadas para proyectos más pequeños con necesidades de pruebas rápidas y menos complejas.

Conclusiones

Selenium es una herramienta poderosa y versátil, adecuada para proyectos grandes que requieren compatibilidad multiplataforma y pruebas distribuidas, aunque puede no ser la opción más eficiente en términos de velocidad para proyectos más pequeños.

Bibliografías

* <https://www.indiumsoftware.com/blog/selenium-test-automation/>
* <https://iteo.com/blog/post/test-automation-with-selenium-basics-and-a-first-test/>
* <https://github.com/Manuelguerrero234/api-auth.git> (enlace al repositorio git)