Análisis de Caso

Escenarios de testing cross-browser

Ashley Rodriguez.

Situación Inicial.

TechSoft, una empresa de desarrollo de aplicaciones web, ha detectado que su aplicación presenta comportamientos diferentes en varios .Se requiere implementar pruebas automatizadas para garantizar que la interfaz y la funcionalidad sean consistentes en cada entorno. El equipo utilizará Visual Studio Code y librerías gratuitas (Selenium WebDriver, TestNG, y WebDriverManager) para configurar y ejecutar pruebas cross-browser en modo headless.

1. Análisis de ventajas y limitaciones

*Cómo ayuda el cross-browser testing:

- Permite detectar diferencias de renderizado entre navegadores.
- Valida que la funcionalidad (formularios, botones, alertas, etc.) sea la misma.
- Ayuda a asegurar una experiencia uniforme para todos los usuarios.

Ventajas:

- Mayor cobertura
- Detección temprana de errores
- Automatización eficiente

Limitaciones:

- Mantenimiento de múltiples drivers
- Posibles incompatibilidades
- Tiempo extra de ejecución

2. Implementación de pruebas

Crea una clase de prueba en Java que abra la aplicación web en navegadores, ejecute una acción sencilla (por ejemplo, verificar la carga de la página) y cierre el navegador.

```java
public class CrossBrowserEdgeTest {

    WebDriver driver;

    @BeforeMethod
    public void setup() {
        // Configurar EdgeDriver local (actualiza la ruta si es diferente)
        System.setProperty(key:"webdriver.edge.driver", value:"C:\\Program Files\\WebDriver\\Edge\

        // Configuración de Edge en modo headless (opcional, puedes descomentar)
        EdgeOptions options = new EdgeOptions();
        // options.addArguments("--headless");  // descomenta si quieres que no se abra la ventana
        options.addArguments(...arguments:"--disable-gpu");
        options.addArguments(...arguments:"--window-size=1920,1080");

        driver = new EdgeDriver(options);
    }

    @Test
    public void testAbrirGoogle() {
        // Abrir Google
        driver.get(url:"https://www.google.com");

        // Verificar título
        String title = driver.getTitle();
        System.out.println("Título de la página: " + title);
        Assert.assertTrue(title.contains(s:"Google"), "El título no contiene 'Google'");
    }

    @AfterMethod
```

c. Utiliza TestNG para parametrizar la ejecución de la prueba en ambos navegadores mediante un archivo XML

```xml
ng.xml > ...
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="EdgeSuite">
    <test name="EdgeTest">
        <classes>
            <class name="com.ejemplo.tests.CrossBrowserEdgeTest"/>
        </classes>
    </test>
</suite>
```

3. Uso de suites de prueba

Se ha creado la suite 'EdgeSuite' en TestNG, que agrupa las pruebas de la clase CrossBrowserEdgeTest.

Ventajas de usar suites:

1. Permite ejecutar varias pruebas de manera organizada.

2. Facilita la automatización y la integración continua.

3. Se pueden configurar prioridades, dependencias y distintos entornos o navegadores.

Reflexión Final

lo aprendido:

Aprendí a configurar Selenium y TestNG, parametrizar pruebas, validar resultados y usar suites. Y esto se puede aplicar en cualquier proyecto web para automatizar pruebas repetitivas y garantizar calidad.