TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

PRUEBA Y CALIDAD DE SOFTWARE

SELENIUM

SELENIUM

1. ¿Qué es Selenium?

Selenium es un conjunto de herramientas de código abierto diseñado para la automatización de pruebas de aplicaciones web. Su propósito principal es simular el comportamiento del usuario en un navegador, lo cual permite verificar que las funcionalidades de una aplicación se comporten correctamente ante diversas entradas y escenarios.

Contrario a lo que se piensa, Selenium no es un framework completo, sino más bien una colección de herramientas con diferentes objetivos:

- **Selenium WebDriver.** Biblioteca que permite escribir scripts que interactúan con navegadores web reales.
- **Selenium IDE.** Extensión para navegadores (Firefox/Chrome) que permite grabar y reproducir acciones del usuario (limitado a pruebas simples).
- **Selenium Grid.** Sistema distribuido que permite ejecutar pruebas en múltiples máquinas y configuraciones de navegador.

En términos generales, Selenium WebDriver es el componente más poderoso y flexible del ecosistema, y es el que se utiliza profesionalmente en proyectos reales de QA y DevOps.

2. ¿Selenium es un marco de trabajo?

Selenium no es un framework cerrado, pero puede actuar como la base de un framework de pruebas automatizadas. En entornos reales, Selenium suele integrarse dentro de un marco más amplio que incluye:

- **Framework de ejecución.** JUnit, TestNG, Cucumber.
- **Gestor de dependencias y compilación.** Maven, Gradle.
- **Herramientas de CI/CD.** Jenkins, GitLab CI, GitHub Actions.
- **Gestión de datos de prueba.** Excel, CSV, JSON, bases de datos.
- **POM (Page Object Model).** patrón de diseño que permite separar la lógica de prueba del acceso a elementos de la interfaz.

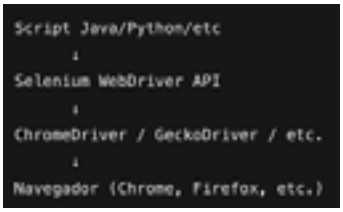
Así, Selenium es el núcleo de automatización de interacción con el navegador, pero su efectividad se potencia al formar parte de un ecosistema bien estructurado.

3. ¿Cómo funciona Selenium?

Selenium WebDriver interactúa directamente con el navegador, enviando comandos que replican acciones humanas (como escribir texto, hacer clic, seleccionar elementos, validar mensajes). Lo hace mediante un driver específico para cada navegador que actúa como puente de comunicación entre el script de prueba y el navegador real.

Diagrama lógico del funcionamiento:

Figura 1 . Diagrama Selenium funcionamiento



4. ¿Cuándo se debe usar Selenium?

Selenium debe emplearse cuando se desea automatizar:

- Pruebas funcionales en la capa de presentación (UI).
- Pruebas de regresión frecuentes.
- Validaciones en múltiples navegadores (cross-browser testing).
- Validación de rutas críticas (checkout, login, registro, búsquedas).
- Integración con pipelines de integración continua.

No se recomienda Selenium para:

- Pruebas de escritorio o móviles (a menos que se integre con Appium).
- Pruebas exploratorias o casos sin un flujo definido.
- Validaciones altamente dinámicas sin control de estado.

5. ¿Dónde se implementa Selenium?

a. Entornos de desarrollo local
   - Ideal para crear, depurar y validar scripts en tiempo real.
   - Requiere: JDK, Maven, Selenium, navegador y su driver.

b. Entornos de pruebas (QA/Staging)
   - Se integra en el proceso de pruebas de software antes de pasar a producción.
   - Suele ejecutarse de forma automática al hacer despliegues.

c. Pipelines de CI/CD
   - Selenium se conecta a Jenkins o GitHub Actions para ejecutarse en cada commit o pull request.
   - Permite detectar fallos antes de pasar a producción.

d. Infraestructura distribuida (Selenium Grid)
   - Útil en grandes empresas.
   - Permite ejecutar cientos de pruebas simultáneamente en múltiples entornos (combinación de navegadores y sistemas operativos).

6. Componentes técnicos claves de Selenium

Tabla 1. Componentes técnicos claves

Componente	Descripción
WebDriver	API que define las acciones de navegación/interacción.
ChromeDriver	Programa ejecutable que controla Chrome desde WebDriver.
By	Clase que permite buscar elementos por id, name, xpath, cssSelector.
Actions	Clase para realizar interacciones complejas (drag and drop, mouse over).
Waits	Mecanismo para esperar dinámicamente a que aparezca un elemento.

7. Ventajas de Selenium

- Código abierto y gratuito.
- Soporte para múltiples lenguajes y navegadores.
- Extensible. Puede combinarse con frameworks de pruebas, reportes, captura de errores.
- Compatible con arquitectura moderna de pruebas: pipelines CI/CD, contenedores (Docker), Selenium Grid.
- Alta fidelidad. Opera sobre el navegador real, no emulado.

8. Limitaciones de Selenium

- Solo para aplicaciones web.
- No puede automatizar pruebas de escritorio (a diferencia de herramientas como TestComplete).
- La mantenimiento de scripts puede volverse costoso si la UI cambia con frecuencia.
- La ejecución de pruebas puede ser más lenta que pruebas unitarias o de API.
- Requiere sincronización cuidadosa con elementos dinámicos (por ejemplo, usar WebDriverWait).

9. Ejemplo detallado paso a paso

Escenario. Verificar que un usuario puede registrarse en un formulario.

Estructura.

- Página de registro con campos: nombre, correo, contraseña, botón enviar.
- Resultado esperado: mensaje “Registro exitoso”.

Código Java.

Figura 2. Ejemplo prueba Selenium

```java
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import java.time.Duration;

public class RegistroTest {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        driver.get("https://ejemplo.com/registro");

        driver.findElement(By.id("nombre")).sendKeys("Carlos");
        driver.findElement(By.id("correo")).sendKeys("carlos@correo.com");
        driver.findElement(By.id("clave")).sendKeys("123456");
        driver.findElement(By.id("btnEnviar")).click();

        String mensaje = driver.findElement(By.id("resultado")).getText();
        if (mensaje.contains("Registro exitoso")) {
            System.out.println("Prueba PASÓ");
        } else {
            System.out.println("Prueba FALLÓ");
        }

        driver.quit();
    }
}
```

Selenium es una herramienta esencial en la caja de herramientas del profesional QA moderno. Aunque no es una solución completa por sí sola, su flexibilidad, madurez y compatibilidad la convierten en la mejor opción para la automatización de pruebas web. Combinada con buenas prácticas de programación y frameworks complementarios, permite implementar soluciones de prueba sólidas, sostenibles y eficientes a lo largo del ciclo de vida del software.