**Estrategias de pruebas.**

1. **Introducción:**

A través de presente documento, se quiere definir la estrategia de pruebas para las funcionalidades seleccionadas de la aplicación <https://opensource-demo.orangehrmlive.com/web/index.php/auth/login> y la aplicación <https://www.amazon.com/>. Se especifica los tipos de pruebas a realizar, herramientas seleccionadas y plan para llevar a cabo dichas pruebas.

1. **Alcance**

Este plan de pruebas contemplara las siguientes funcionalidades:

2 scripts de automatización unitarias de inicio de sesión unitarias con intellij IDE y karate (1 positiva y otra negativa)

2 scripts de automatización de integración con Intellij IDE y selenium (1 positiva y otra negativa).

I1 scripts de regresión de pruebas con Intellij IDE y selenium

Se realizarán pruebas automatizadas (90%) y manuales (10%), incluyendo pruebas unitarias, integración y de regresión.

**3 Tipos de prueba**

# Pruebas unitarias

# Herramientas: intellij IDE y karate

# Justificación: compatible con múltiples lenguajes y múltiples navegadores.

# Pruebas Integración

# Herramientas: Java Testing with Selenium

# Justificación: compatible con múltiples lenguajes y múltiples navegadores.

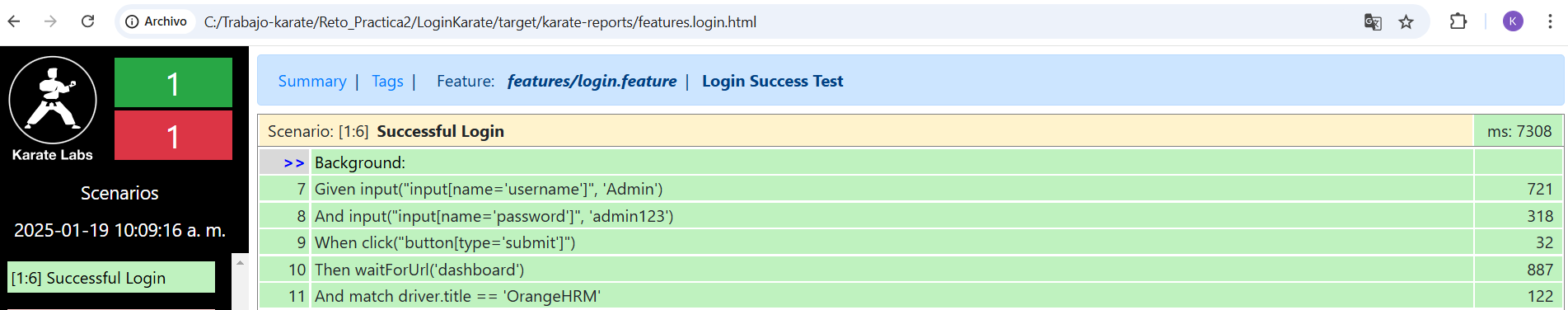
# Pruebas Regesión

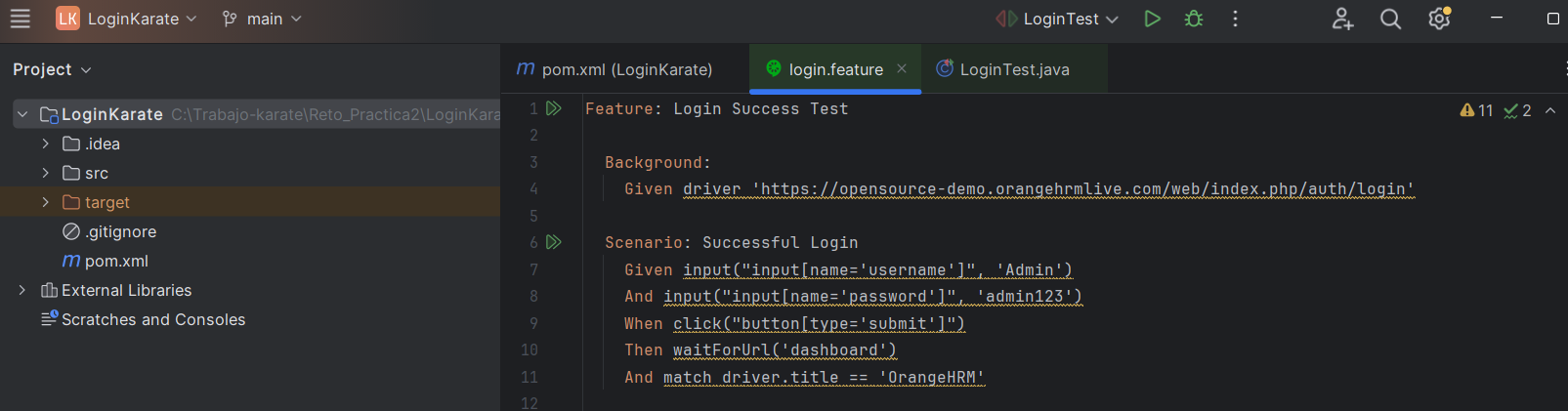
# Herramientas: Java Testing with Selenium

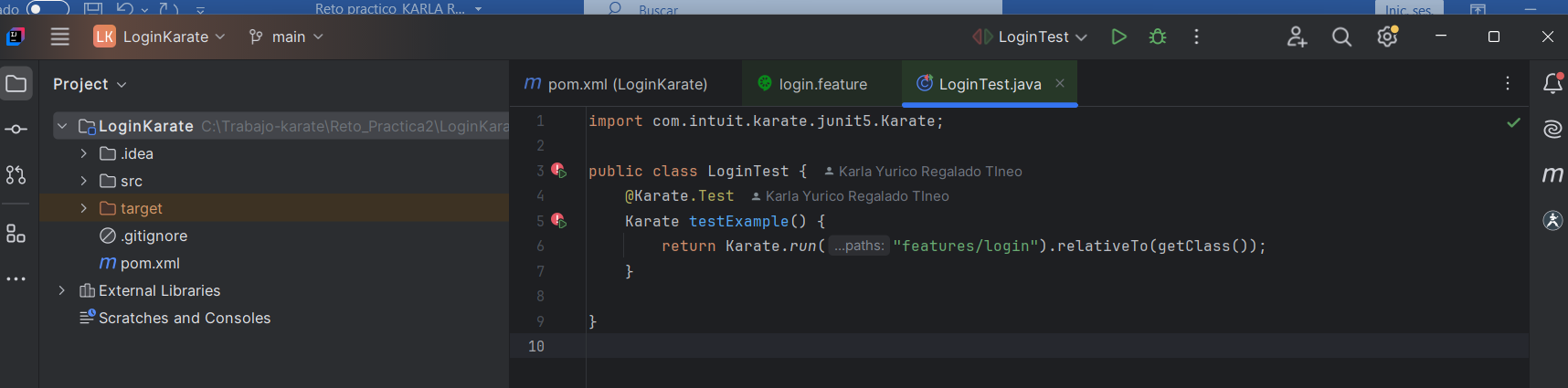
# Justificación: compatible con múltiples lenguajes y múltiples navegadores.

1. **casos de prueba**
2. **Prueba unitaria**

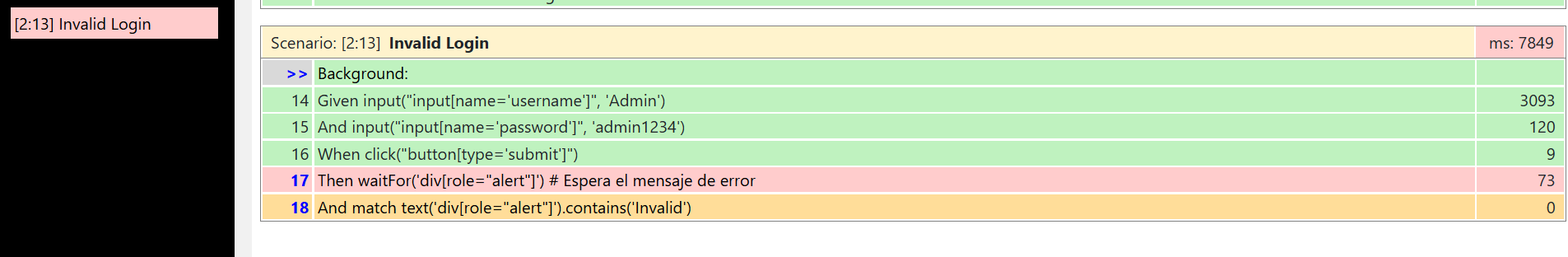
* **Caso prueba 01: Validación del login correcto.**

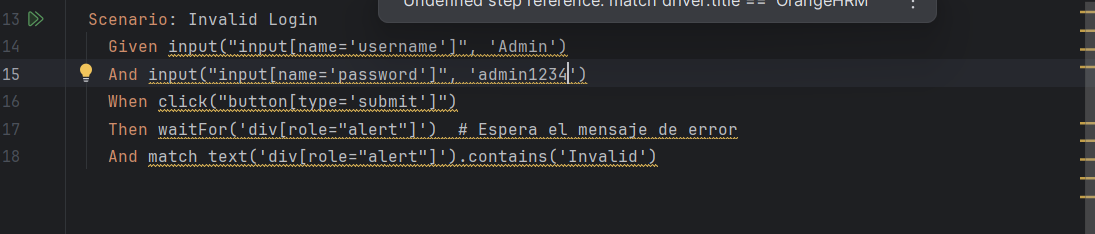
****

****

****

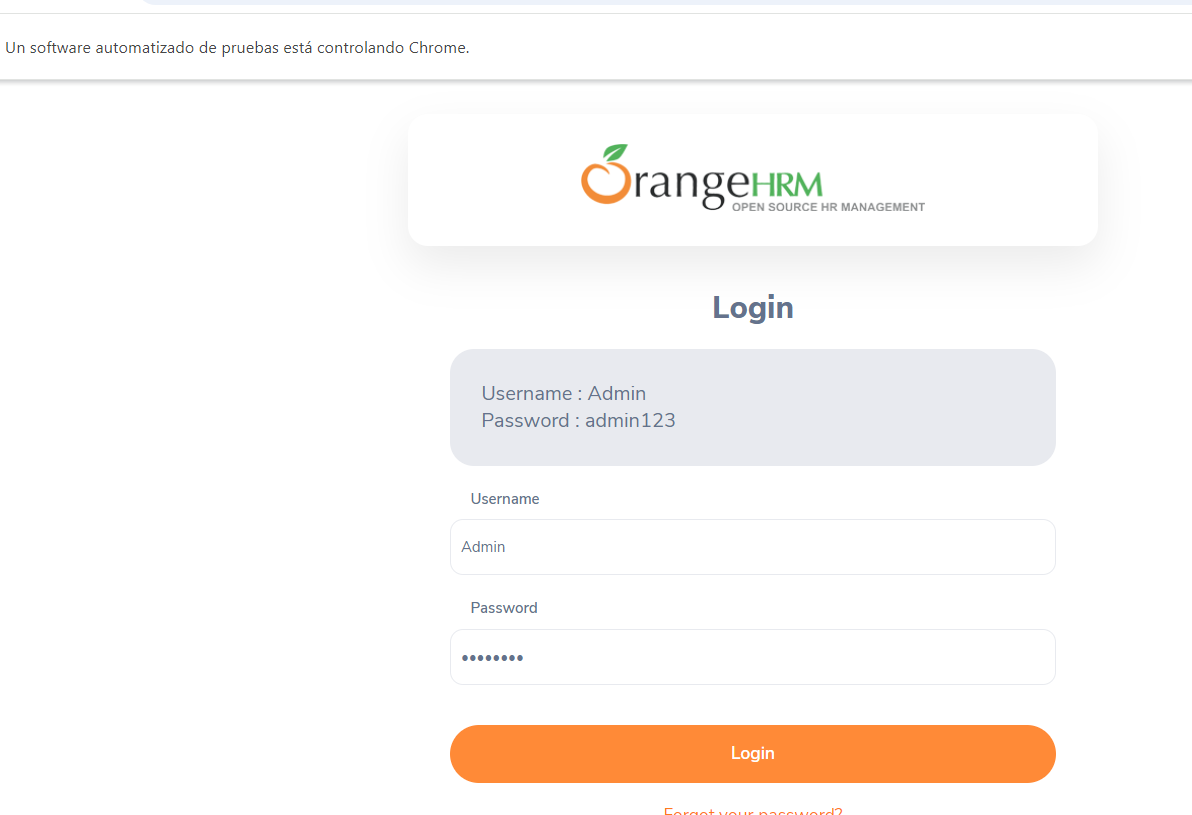
* **Caso prueba 02: Validación del login incorrecto.**

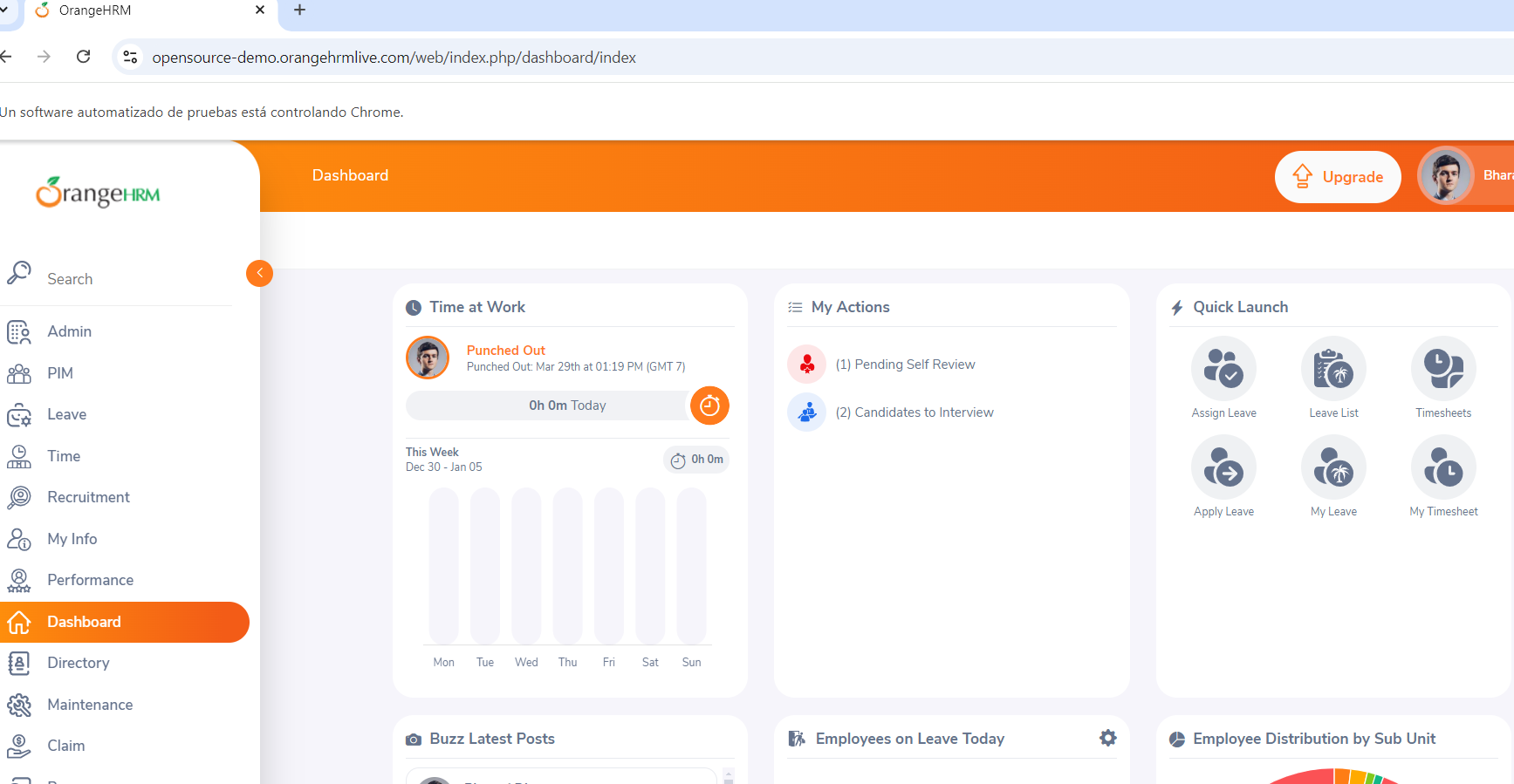
****

****

# Pruebas Integración

* **Caso prueba 01: Validación del login correcto con Java Testing and Selenium**

****

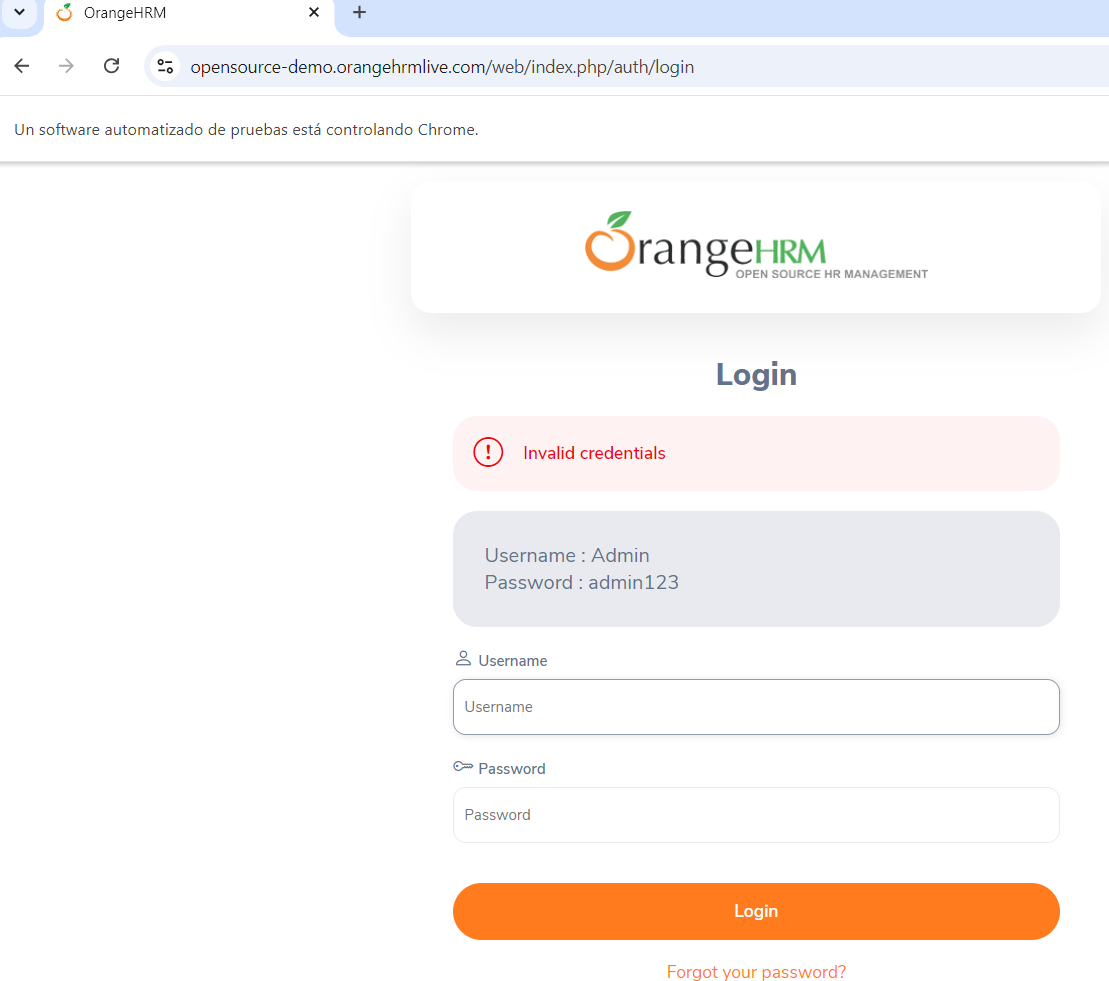
****

package seleniumtest;  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.testng.Assert;  
import org.testng.annotations.AfterClass;  
import org.testng.annotations.BeforeClass;  
import org.testng.annotations.Test;  
  
public class LoginTest {  
 WebDriver driver;  
  
 @BeforeClass  
 public void SetUp()  
 {  
 driver = new ChromeDriver();  
 driver.manage().window().maximize();  
 driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  
 }  
 @AfterClass  
 public void tearDown()  
 {  
 driver.quit();  
 }  
  
 @Test  
public void testLoggingIntoApplication() throws InterruptedException {  
 Thread.*sleep*(2000);  
 WebElement username= driver.findElement(By.*name*("username"));  
 username.sendKeys("Admin");  
  
 var password = driver.findElement(By.*name*("password"));  
 password.sendKeys(("admin123"));  
  
 driver.findElement(By.*tagName*("button")).click();  
 Thread.*sleep*(2000);  
 String actualResult=driver.findElement(By.*tagName*("h6")).getText();  
 String expectedResult="Dashboard";  
 Assert.*assertEquals*(actualResult,expectedResult);  
  
 }

}

# 

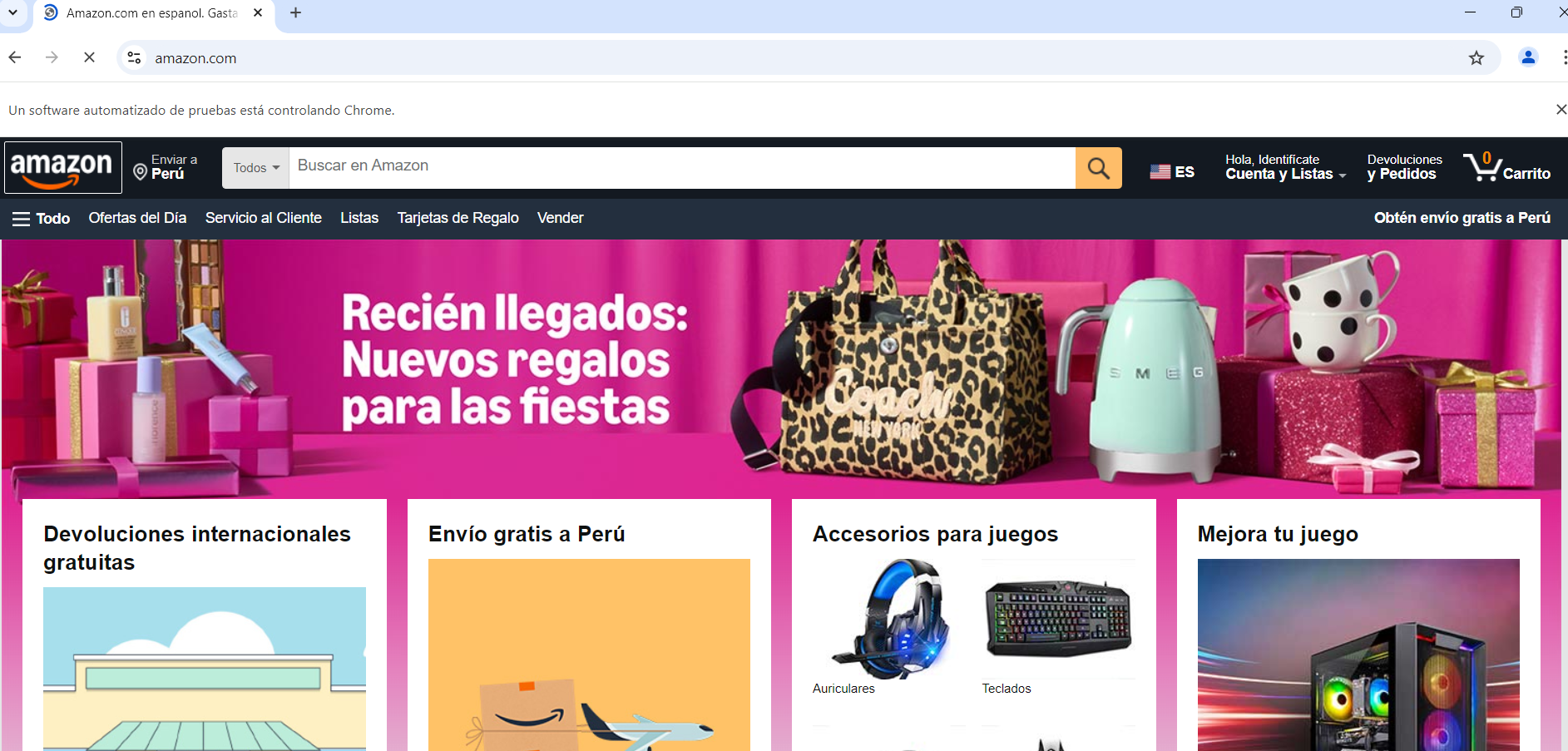
* **Caso prueba 02: Validación del login incorrecto con Java Testing and Selenium**

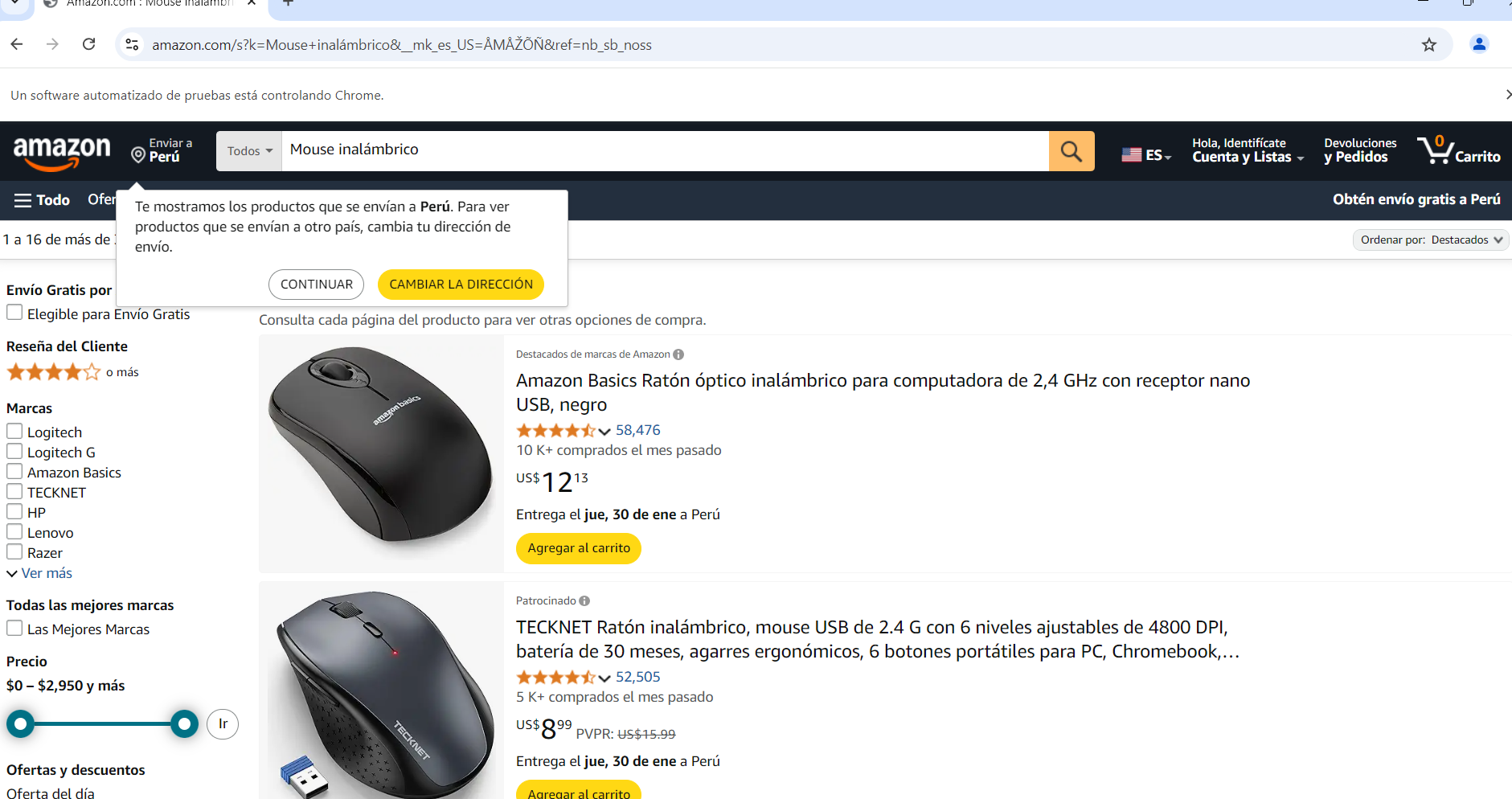


* package seleniumtest;  
  import org.openqa.selenium.By;  
  import org.openqa.selenium.WebDriver;  
  import org.openqa.selenium.WebElement;  
  import org.openqa.selenium.chrome.ChromeDriver;  
  import org.testng.Assert;  
  import org.testng.annotations.AfterClass;  
  import org.testng.annotations.BeforeClass;  
  import org.testng.annotations.Test;  
    
  public class LoginTestfallido {  
   WebDriver driver;  
    
   @BeforeClass  
   public void SetUp()  
   {  
   driver = new ChromeDriver();  
   driver.manage().window().maximize();  
   driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  
   }  
   @AfterClass  
   public void tearDown()  
   {  
   // driver.quit();  
   }  
    
   @Test  
  public void testLoggingIntoApplication() throws InterruptedException {  
   Thread.*sleep*(2000);  
   WebElement username= driver.findElement(By.*name*("username"));  
   username.sendKeys("Admin");  
    
   var password = driver.findElement(By.*name*("password"));  
   password.sendKeys(("admin1234"));  
   // Assert.assertEquals("admin1234","admin123");  
    
   driver.findElement(By.*tagName*("button")).click();  
   Thread.*sleep*(2000);  
    
    
   }  
  }

# Pruebas Regresión

* **Caso prueba 01: Validación compra de un articulo**

****

****

package seleniumtest;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.chrome.ChromeDriver;  
//import io.github.bonigarcia.wdm.WebDriverManager;  
  
public class AmazonPurchaseTest {  
 public static void main(String[] args) {  
 // Configurar el ChromeDriver  
 System.*setProperty*("webdriver.chrome.driver", "D:\\Utilitarios\\chromedriver-win64\\chromedriver.exe");  
 // Crear una instancia del navegador  
 WebDriver driver = new ChromeDriver();  
 try {  
 // Abrir Amazon  
 driver.get("https://www.amazon.com/-/es/ref=nav\_logo/session-id=132-5278641-6728915");  
 // Maximizar la ventana  
 // Encontrar el elemento por su XPath  
 driver.manage().window().maximize();  
 WebElement logoAmazon = driver.findElement(By.*xpath*("//\*[@id=\"c\"]"));  
  
 // Hacer clic en el elemento  
 logoAmazon.click();  
 // Buscar un producto  
 WebElement searchBox = driver.findElement(By.*id*("twotabsearchtextbox"));  
 searchBox.sendKeys("Mouse inalámbrico");  
 searchBox.submit();  
 // Seleccionar el primer producto  
 WebElement firstProduct = driver.findElement(By.*cssSelector*(".s-main-slot .sresult-item h2 a"));  
 firstProduct.click();  
 // Añadir al carrito  
 WebElement addToCartButton = driver.findElement(By.*id*("add-to-cart-button"));  
 addToCartButton.click();  
 // Validar que el producto está en el carrito  
 WebElement cart = driver.findElement(By.*id*("nav-cart"));  
 cart.click();  
 WebElement cartItem = driver.findElement(By.*cssSelector*(".sc-list-itemcontent"));  
 if (cartItem.isDisplayed()) {  
 System.*out*.println("El producto fue añadido al carrito correctamente.");  
 } else {  
 System.*out*.println("Error: el producto no está en el carrito.");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 // Cerrar el navegador  
 driver.quit();  
 }  
 }  
}

**Estrategia de pruebas para la aplicación**

Como estrategia de pruebas propuestas, se ha considerado abarcar 3 funciones especificas de dos aplicaciones. Pruebas con una orientación automática y pruebas manuales. Pruebas con una cobertura del 10% manuales y 90% de pruebas automatizadas.

1. **Objetivo de la estrategia de pruebas:**

El objetivo es garantizar que las pruebas, funciones correctamente, a nivel de pruebas funcionales como automatizadas. Verificar las pruebas unitarias y de integración y regresión

1. **Condiciones generales**

* **Porcentaje de pruebas:** 10% Pruebas manuales y 90% de pruebas automatizada.
* **Tipo de pruebas:** Puebas tipo web.
* **Herramientas de pruebas:** se trabajará con herramientas de pruebas unitarias, integración y regresion
* **Modelo de arquitectura de automatización:**

El modelo de arquitectura a utilizar es **POM, patrón diseñado para la automatización de software**, se utiliza particularmente en pruebas de interfaces gráficas de usuario (GUI) para aplicaciones web.

**Pasos:**

**En las pruebas unitarias trabajadas con karate y selenium**

1. Logintest: Clase que representa la página del login, con métodos para ingresar usuario y contraseña y hacer clic en el botón para logearse.
2. Logintestfallido: Clase que representa la página del login, con métodos para realizar una prueba fallida del logeo.

**En las pruebas de integración trabajadas con Java Testing and Selenium**

1. Logintest: Clase que representa la página del login, con métodos para ingresar usuario y contraseña y hacer clic en el botón para logearse.
2. Logintestfallido: Clase que representa la página del login, con métodos para realizar una prueba fallida del logeo.

**En las pruebas de regresión trabajadas con Java Testing and Selenium**

1. Busqueda de un artículo: Clase con métodos para realizar un flujo de búsqueda de un mouse inalámbrico.

* **Métricas y riesgos:** se identifican las siguientes métricas asociados a la ejecución:

**Tiempo de ejecución**: Mide el tiempo total que tarda en ejecutarse el conjunto completo de pruebas automatizadas.

**Riesgo**: Un tiempo de ejecución largo puede reducir los beneficios de la automatización, ya que puede volverse un cuello de botella en los ciclos de integración continua.