Playwright + POM + CI/CD

Playwright + POM + CI/CD (GitHub Actions)

Aplicar mejores prácticas en la automatización de pruebas usando Playwright, implementar el patrón de diseño Page Object Model (POM), y configurar un pipeline CI/CD con GitHub Actions que permita ejecutar las pruebas de forma automática en un entorno profesional.

1. Introducción

Este taller está diseñado para llevar a los estudiantes a un nivel más profesional en el desarrollo de pruebas automatizadas.
Luego de aprender a automatizar una aplicación web como Swag Labs , ahora aprenderán:

- A estructurar sus proyectos de forma profesional
- Implementar Page Object Model (POM)
- Integrar su repositorio con GitHub Actions
- Ejecutar pruebas automáticamente con CI/CD
- Comprender la importancia de estos procesos en ambientes empresariales

La guía utiliza conceptos oficiales de la documentación de Playwright:
Installation | Playwright

2. Estructura recomendada del proyecto

La estructura del proyecto debe organizarse de forma clara para permitir escalabilidad.

```
/tests
   login.spec.ts
/pages
   LoginPage.ts
   ProductsPage.ts
/playwright.config.ts
/package.json
/.github/workflows
   ci.yml

```

3. Page Object Model (POM)

¿Qué es?

Es un patrón de diseño que organiza la lógica de interacción en clases llamadas "Pages".
Cada página contiene sus selectores y funciones, permitiendo:

- Mantener el código limpio y ordenado
- Simplificar la lectura de test cases
- Facilitar el mantenimiento
- Centralizar los selectores
- Escalar la automatización sin duplicaciones

4. Implementación de POM (Ejemplo funcional)

4.1 LoginPage.ts

```ts
import { Page } from '@playwright/test';

export class LoginPage {
  readonly page: Page;
  readonly usernameInput;
  readonly passwordInput;
  readonly loginButton;

  constructor(page: Page) {
    this.page = page;
    this.usernameInput = page.locator('#user-name');
    this.passwordInput = page.locator('#password');
    this.loginButton = page.locator('#login-button');
  }

  async goto() {
    await this.page.goto('https://www.saucedemo.com/v1/index.html');
  }

  async login(username: string, password: string) {
    await this.usernameInput.fill(username);
    await this.passwordInput.fill(password);
    await this.loginButton.click();
  }
}

```

4.2 ProductsPage.ts

```ts
import { Page } from '@playwright/test';

export class ProductsPage {
  readonly page: Page;
  readonly title;

  constructor(page: Page) {
    this.page = page;
    this.title = page.locator('.product_label');
  }

  async verifyIsOnProductsPage() {
    await this.title.waitFor();
  }
}

```

4.3 Test con POM

```ts
import { test, expect } from '@playwright/test';
import { LoginPage } from '../pages/LoginPage';
import { ProductsPage } from '../pages/ProductsPage';

test('Login exitoso en SauceDemo con POM', async ({ page }) => {
  const loginPage = new LoginPage(page);
  const productsPage = new ProductsPage(page);

  await loginPage.goto();
  await loginPage.login('standard_user', 'secret_sauce');

  await productsPage.verifyIsOnProductsPage();

  await expect(productsPage.title).toHaveText('Products');
});

```

5. Configuración Playwright

Archivo `playwright.config.ts`:

```ts
import { defineConfig } from '@playwright/test';

export default defineConfig({
  testDir: './tests',
  retries: 1,
  use: {
    headless: true,
    screenshot: 'only-on-failure',
    video: 'retain-on-failure',
  },
});

```

6. Integración CI/CD con GitHub Actions

Crear el workflow

Ruta:

```
.github/workflows/ci.yml

```

Contenido:

```yaml
name: Playwright Tests

on:
  push:
    branches: ["main"]
  pull_request:
    branches: ["main"]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 18

      - name: Install dependencies
        run: npm install

      - name: Install Playwright Browsers
        run: npx playwright install --with-deps

      - name: Run tests
        run: npx playwright test --reporter=line

```

7. Reportería en Playwright

Playwright incluye herramientas de reporte:

- HTML Reporter
- Screenshots
- Videos de errores

Generar reporte html:

```
npx playwright test --reporter=html
npx playwright show-report

```

Se pueden subir como artifacts en GitHub Actions.

8. Buenas prácticas recomendadas

Selectores oficiales recomendados

- `getByRole`
- `getByLabel`
- `getByTestId`

Evitar selectores frágiles o basados en XPath.

Pruebas independientes

Cada test debe poder ejecutarse solo sin depender de otro.

Reutilización con POM

Mantener las reglas de UI dentro de las Page Classes.

CI/CD en modo headless

Asegura estabilidad y eficiencia.

10. Importancia del CI/CD en grandes empresas

El CI/CD permite:

- Aumentar la velocidad del desarrollo
- Reducir errores en ambientes de producción
- Mantener un estándar de calidad uniforme
- Automatizar validaciones críticas
- Integración continua entre equipos

Es una habilidad altamente solicitada en QA y Desarrollo.

11. ROI en automatización

El retorno de inversión (ROI) de la automatización se ve reflejado en:

- Menos tiempo invertido en pruebas repetitivas
- Menor cantidad de bugs en producción
- Menores costos operativos
- Mayor velocidad en los ciclos de desarrollo
- Equipos más eficientes

Un framework con POM + CI/CD aumenta el ROI de forma considerable.

12. Ventajas y desventajas

Ventajas

- Código organizado y profesional
- Ejecuciones automáticas y constantes
- Facilita escalabilidad
- Incrementa el valor profesional del estudiante

Desventajas

- Tiempo inicial de aprendizaje
- Mantenimiento constante del framework
- Dependencia de pipelines externos

13. Bibliografía y enlaces recomendados

- Playwright Docs: Installation | Playwright
- Locators: https://playwright.dev/docs/test-locators
- GitHub Actions Docs: GitHub Actions documentation - GitHub Docs
- POM guía oficial: Page object models | Playwright

14. Conclusión

Con esta guía, los estudiantes pueden:

- Implementar un proyecto Playwright con buenas prácticas
- Utilizar POM de forma profesional
- Configurar CI/CD con GitHub Actions
- Entender su impacto real en la industria

Powered by Ing. Gustavo Molina