GESTIÓN DEL PROCESO DE DESARROLLO DE

SOFTWARE

ME. ROBERTO CARLOS VAQUERA FAVELA

Herramientas de automatización de pruebas

GRUPO 10°B

PRESENTA:

Guillermo Romero colunga

Lucio Rafael Salazar Veliz

Francisco Javier Mota Ontiveros

Yoshep Emmanuel Garza

Padilla

20/10/2025

Informe Completo sobre Herramientas de Automatización de Pruebas

La automatización de pruebas es el uso de software para ejecutar casos de prueba, comparar resultados esperados con resultados reales, generar informes de prueba y gestionar los datos de prueba.

A continuación, se presenta una investigación detallada sobre las herramientas de automatización.

1. Herramientas de Automatización de Pruebas Fundamentales

Estas herramientas se utilizan para escribir y ejecutar los scripts que interactúan directamente con la aplicación (Web, Móvil, API, Escritorio).

Herramienta	Tipo	Propósito Principal (Para qué sirve)	Funcionamiento Clave (Cómo funciona)
Selenium	Código Abierto (Web)	Automatización de pruebas funcionales de extremo a extremo (E2E) para aplicaciones web en cualquier navegador.	Se compone de: **WebDriver** (API que interactúa directamente con los navegadores a través de sus drivers nativos), **IDE** (grabación y reproducción), y **Grid** (ejecución paralela).
Cypress	Código Abierto (Web)	Proporcionar un marco de pruebas rápido, fiable y específico para JavaScript (front-end).	Se ejecuta *en el mismo bucle de ejecución* que la aplicación, lo que permite una comunicación más rápida y un mejor control sobre el DOM. Está limitado principalmente a JavaScript (Node.js).

Playwright	Código Abierto (Web, API)	Automatización E2E moderna y robusta, con soporte nativo para múltiples navegadores (Chromium, Firefox, WebKit) y multiplataforma (Web, Mobile Web, API).	Utiliza un protocolo de automatización fuera de proceso (out-of-process) para interactuar con los navegadores, lo que lo hace muy rápido y evita problemas de *flakiness* (pruebas inestables).
Appium	Código Abierto (Móvil)	Automatización de pruebas para aplicaciones móviles (nativas, híbridas y web) en plataformas iOS y Android.	Utiliza el protocolo **WebDriver** (el mismo de Selenium) para interactuar con las interfaces de usuario de iOS (a través de XCUITest) y Android (a través de UiAutomator/Espresso). Es un servidor que traduce comandos.
Katalon Studio	Licencia (Todo-en-uno)	Automatización integral de pruebas (Web, API, Móvil, Escritorio) para equipos de todos los tamaños, ofreciendo soluciones *low-code* y basadas en código.	Combina un IDE para la creación de scripts (basado en Groovy/Java) con funciones de grabación/reproducción y gestión de proyectos integrada. Está diseñado para simplificar el flujo de trabajo de automatización.
Postman	Freemium (API)	Diseño, desarrollo y prueba de APIs (REST, SOAP).	Permite enviar peticiones HTTP/S a las APIs, validar las respuestas (código de estado, cuerpo, cabeceras) y organizar las peticiones en colecciones. Su componente **Newman** permite la ejecución desde la línea de comandos (CI/CD).

Apache JMeter	Código Abierto (Performance)	Pruebas de rendimiento, carga y estrés para medir el comportamiento de aplicaciones web y servicios API bajo diferentes condiciones de carga.	Simula un gran número de usuarios y peticiones concurrentes para evaluar la escalabilidad y la capacidad de respuesta de un servidor.

2. Diferencias Clave entre Herramientas

Las diferencias radican principalmente en el alcance, la tecnología y la curva de aprendizaje:

Característica	**Selenium**	**Cypress**	**Playwright**	**Appium**
Alcance	Web (Solo la interfaz gráfica)	Web (Front-end JS)	Web, Mobile Web, API	Móvil (Nativo, Híbrido, Web)
Lenguajes	Múltiples (Java, Python, C#, JS, Ruby, etc.)	JavaScript / TypeScript	Múltiples (JS, Python, Java, .NET)	Múltiples (Según el lenguaje de la prueba)
Arquitectura	Controlador Externo (WebDriver)	Se ejecuta en el navegador	Out-of-process, Protocolo Nativo	Servidor que traduce comandos
Velocidad/Estabilidad	Puede ser lento y más propenso a	Muy rápido y estable para JS.	Extremadamente rápido y moderno.	Depende de la plataforma móvil (iOS/Android).

	inestabilidad.			
Uso de CI/CD	Requiere configuración de Grid/Drivers.	Nativo y sencillo (requiere Node.js).	Nativo y sencillo (similar a Cypress).	Requiere infraestructura de dispositivos/emuladores.

Diferencias de Paradigma (Low-Code vs. Código)

- **Herramientas basadas en Código (Selenium, Cypress, Playwright, Appium):** Requieren sólidos conocimientos de programación. Ofrecen la máxima flexibilidad, control y personalización.
- **Herramientas Low-Code/AI (Katalon, TestRigor, Mabl):** Permiten a los testers crear pruebas con mínima o nula codificación, a menudo mediante grabación o lenguaje natural. Suelen tener un costo de licencia, pero reducen el tiempo de mantenimiento y la barrera de entrada.

3. Plataformas CI/CD y Servicios en la Nube

A. Plataformas de Integración y Entrega Continua (CI/CD)

Las herramientas CI/CD son esenciales para ejecutar automáticamente las pruebas cada vez que se realiza un cambio en el código, garantizando una **estrategia de Prueba Continua (Continuous Testing)**.

Plataforma	Tipo	Función en la Automatización	Ejemplos de Uso
Jenkins	Servidor de Automatización (Open Source)	Orquestar el pipeline de CI/CD. Ejecuta scripts de automatización de pruebas (Selenium, JMeter) en servidores dedicados tras cada *commit*.	Uso de *plugins* para disparar la ejecución de pruebas, generar informes (Allure), y notificar resultados.
GitLab CI/CD	Nativo (Basado en Git)	Integración continua integrada	Define pipelines de prueba mediante archivos `.gitlab-`

		directamente en el repositorio de código.	`ci.yml`. Ejecuta pruebas en paralelo utilizando *runners* y soporta despliegues condicionales.
GitHub Actions	Nativo (Basado en Git)	Automatización de flujos de trabajo (workflows) para cualquier evento de GitHub (push, pull request).	Crea flujos de trabajo de prueba usando YAML, que pueden ejecutarse en ambientes Linux, Windows o macOS.
Azure DevOps / CircleCI / Bamboo	Servicios Comerciales	Plataformas CI/CD con integración nativa de sus propios ecosistemas (Azure, Atlassian) para construir, probar e implementar software.	Ofrecen plantillas y fácil integración con servicios de *cloud testing* para ejecutar pruebas a escala.

B. Servicios de Automatización en la Nube (Cloud Testing)

Estos servicios proporcionan la infraestructura (máquinas virtuales, navegadores, dispositivos móviles) necesaria para ejecutar pruebas a escala, eliminando la necesidad de gestionar un *Test Lab* local.

- **BrowserStack / LambdaTest:** Son líderes en la ejecución de pruebas **Cross-Browser y Cross-Device**.
    - **Propósito:** Permiten ejecutar scripts de Selenium, Cypress, Playwright o Appium en una parrilla (Grid) de miles de combinaciones de navegadores y sistemas operativos (incluyendo dispositivos móviles reales o emulados).
    - **Funcionamiento:** El script de prueba se ejecuta localmente o en un pipeline de CI/CD, pero envía los comandos de ejecución a los dispositivos remotos provistos por el servicio en la nube.
- **Mabl / CloudQA (SaaS AI-Powered):**
    - **Propósito:** Soluciones SaaS de automatización *low-code* impulsadas por IA que se encargan de la ejecución y el mantenimiento de las pruebas automáticamente en su propia infraestructura en la nube.
    - **Funcionamiento:** El usuario crea la prueba (a menudo grabando la interacción o usando lenguaje natural), y la plataforma se encarga de ejecutarla, gestionar la infraestructura y aplicar la IA para auto-sanar (mantener) la prueba cuando la UI cambia ligeramente.

4. Herramientas de Apoyo para la Automatización

Estas herramientas no ejecutan las pruebas directamente, sino que gestionan el proceso de calidad, el ciclo de vida del código o la generación de informes.

Herramienta de Apoyo	Categoría	Para qué sirve
Jira / ClickUp	Gestión de Proyectos / Bugs	Seguimiento de defectos (bugs), planificación de sprints y gestión de historias de usuario. La automatización se integra para reportar fallos directamente en estos sistemas.
TestRail / Zephyr Scale	Gestión de Casos de Prueba (TCM)	Centralizar los casos de prueba manuales y automatizados. Mapean los resultados de la ejecución automatizada con los casos definidos, proporcionando métricas de cobertura.
Git / Bitbucket	Control de Versiones	Almacenar y gestionar el código fuente de los scripts de automatización. Es fundamental para la colaboración y el trabajo en CI/CD.
Maven / Gradle	Herramientas de Construcción (Build)	Gestionar dependencias (librerías de prueba), compilar el código de las pruebas y empaquetar los resultados. Son esenciales para proyectos basados en Java/Groovy/Kotlin.
Cucumber / Gherkin	BDD (Behavior Driven Development)	Fomentar la colaboración al permitir escribir pruebas en un lenguaje natural (Gherkin) que es legible por el negocio, el QA y el desarrollo.
Allure TestOps	Gestión de Informes y Orquestación	Generar informes detallados, visuales y centralizados de los resultados de pruebas manuales y automatizadas, mejorando la colaboración del equipo.

Este panorama muestra que la elección de una herramienta principal (como Playwright o Appium) depende del objetivo de la prueba (Web o Móvil), mientras que su eficiencia operativa requiere la integración con plataformas CI/CD y herramientas de apoyo para la gestión y la escalabilidad.