**Herramientas**

Tabla – Evaluación de Herramientas

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Criterio** | **Peso** | **Selenium**  **(0-10)** | **Po Selenium** | **Playwright**  **(0-10)** | **Po Playwright** | **Cypress**  **(0-10)** | **Po Cypress** |
| Funcionalidad/características deseadas o requeridas | 5 | 4 | 20 | 4 | 20 | 3 | 15 |
| Facilidad de uso | 5 | 1 | 5 | 4 | 20 | 4 | 20 |
| Curva de aprendizaje | 4 | 1 | 4 | 4 | 16 | 4 | 16 |
| Compatibilidad/Integración a otras herramientas y aplicaciones | 5 | 4 | 20 | 4 | 20 | 3 | 15 |
| Costo/Licenciamiento | 3 | 4 | 12 | 4 | 12 | 2 | 6 |
| Escalabilidad | 5 | 4 | 20 | 4 | 20 | 3 | 15 |
| Flexibilidad/Capacidad de personalización | 3 | 4 | 12 | 4 | 12 | 4 | 12 |
| Lenguajes de programación soportados | 3 | 4 | 12 | 3 | 9 | 2 | 6 |
| Navegadores compatibles | 3 | 4 | 12 | 2 | 6 | 3 | 9 |
| Ponderado |  |  | 117 |  | 135 |  | 114 |

Tabla – Justificación de la Evaluación

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Selenium [1]** | **Playwright [2]** | **Cypress [3]** |
| Funcionalidad/características deseadas o requeridas | Soporta pruebas automatizadas de múltiples tipos, incluyendo pruebas funcionales en diferentes navegadores y sistemas operativos. Puede manejar múltiples pestañas, ventanas, y trabajar con elementos dinámicos. Muy potente para pruebas complejas. | Puede hacer pruebas de interacción con APIs. Se pueden realizar pruebas paralelas y en múltiples navegadores al mismo tiempo. Permite subir archivos, manejar ventanas emergentes, cuadros de diálogo, etc. Permite pruebas en modo headless (sin interfaz) o con interfaz visible. | Tiene funciones modernas para pruebas de interfaz y testing de APIs, y permite ver paso a paso lo que ocurre en cada prueba. Sin embargo, su soporte para múltiples pestañas o ventanas es limitado. |
| Facilidad de uso | No es tan intuitivo como otras herramientas más modernas. Puede requerir configuración adicional y manejo de drivers de navegador. | Fácil de usar. El proceso de instalación es sencillo y posee una API moderna y limpia. | Destaca por su interfaz visual muy amigable, que permite ver las pruebas ejecutándose. Es intuitivo para usuarios con poca experiencia. |
| Curva de aprendizaje | La curva puede ser moderada a alta, especialmente si se requiere configuración avanzada o integración con otros frameworks. | La curva es baja. La herramienta es intuitiva y con ejemplos claros. | Es fácil comenzar a usar Cypress incluso sin mucha experiencia previa en testing, lo que reduce la curva de aprendizaje. |
| Compatibilidad/Integración a otras herramientas y aplicaciones | Alta compatibilidad con herramientas de CI/CD, frameworks de testing (JUnit, TestNG, etc.) y lenguajes de programación. Se integra fácilmente en ecosistemas complejos. | Funciona con frameworks como Pytest, Mocha, Jest y se integra nativamente con CI/CD (GitHub Actions, Azure DevOps, Jenkins) | Tiene integración con herramientas de CI/CD, pero algunas integraciones requieren planes pagos si se usan en su plataforma oficial (Dashboard). |
| Costo/Licenciamiento | Completamente gratuito y open source. | Completamente gratuito y open source. | Es open source y gratuito, pero algunas funciones avanzadas como el Dashboard y reportes automáticos tienen planes pagos. |
| Escalabilidad | Muy escalable, apto para proyectos pequeños, medianos y grandes en entornos empresariales. | Soporta ejecución en paralelo y containers por lo que tiene muy buena escalabilidad. | Funciona bien en proyectos medianos, pero su arquitectura basada en un solo navegador y algunas limitaciones técnicas. |
| Flexibilidad/Capacidad de personalización | Muy flexible, permite personalizaciones profundas en flujos de prueba y configuración. | Permite automatizar escenarios diversos y complejos con una API más simple. | Es flexible, aunque al estar más centrado en una estructura propia, puede ser menos adaptable a ciertas configuraciones externas. |
| Lenguajes de programación soportados | Java, Python, C#, Ruby, JavaScript, entre otros. | Python, JavaScript, TypeScript, C# y Java. | Está centrado en JavaScript y TypeScript. No soporta otros lenguajes. |
| Navegadores compatibles | Soporta una amplia gama de navegadores: Chrome, Firefox, Safari, Edge, Opera, entre otros, con soporte completo para pruebas cruzadas. | Firefox, Edge, Chrome y WebKit. | Funciona bien en Chrome, Firefox y Edge, pero aún no tiene soporte oficial completo para Safari y tiene limitaciones en pruebas cruzadas entre navegadores. |

**Conclusión**

Playwright es la mejor opción porque combina la potencia de Selenium con la facilidad de uso de Cypress. Ofrece una instalación sencilla, una API moderna y clara, soporte para pruebas paralelas, aunque no tiene tanto soporte de navegadores como Selenium, es compatible con los navegadores más populares. Comparativamente posee una curva de aprendizaje baja y una alta facilidad de uso y se integra nativamente con CI/CD. Esto lo hace más eficiente, escalable y versátil, cubriendo tanto proyectos simples como entornos empresariales sin las limitaciones técnicas de Cypress ni la complejidad de configuración de Selenium.

**Referencias**

[1] APIdog, “Selenium vs Playwright: ¿Cuál es mejor para tus pruebas automáticas?”, *APIdog Blog*, [En línea]. Disponible en: <https://apidog.com/es/blog/selenium-vs-playwright-3/>.

[2] *Playwright*, “Playwright Documentation”, *DevDocs.io*, [En línea]. Disponible en: <https://devdocs.io/playwright/>.

[3] Itequia, “Cypress: La herramienta que automatiza tus pruebas y garantiza la calidad de tus proyectos”, *Itequia Blog*, [En línea]. Disponible en: <https://itequia.com/es/cypress-la-herramienta-que-automatiza-tus-pruebas-y-garantiza-la-calidad-de-tus-proyectos/>.