DESDE 2023
https://rictrevista.org

RICT Revista de Investigación Científica, Tecnológica e Innovación

Publicación Semestral RICT Vol. 4 No. 7 (2026) P. 29 – 33

ISSN: 2992-7315

Cypress en formación universitaria para testing web
Cypress in University Training for Web Testing

Israel Antonio Marín Castañeda[a]

[a]TecNM/TESE, Maestría en Ingeniería en Sistemas Computacionales, Ecatepec de Morelos, Estado de México, México.

Resumen

La digitalización del sector público exige portales web confiables, accesibles y sostenibles; sin embargo, la formación universitaria en ingeniería de software todavía presenta brechas prácticas en aseguramiento de calidad y testing automatizado. Este artículo analiza la pertinencia de Cypress como herramienta de entrada para capacitar estudiantes universitarios en pruebas web, en comparación con Selenium y Playwright. Se desarrolló una revisión documental narrativa, complementada con una matriz comparativa, una rúbrica de evaluación y una priorización MoSCoW aplicada a competencias de QA. Los resultados, derivados del análisis documental y no de un experimento con estudiantes, muestran que Cypress ofrece ventajas pedagógicas en configuración inicial, depuración visual y retroalimentación inmediata. Playwright destaca en escenarios E2E avanzados y Selenium conserva valor por su madurez e interoperabilidad. Se concluye que Cypress es adecuado para formación inicial, siempre que se enseñen también sus limitaciones y métricas objetivas de evaluación.

Palabras clave: aseguramiento de calidad; Cypress; educación en ingeniería; sector público; testing automatizado

Abstract

Public-sector digitalization requires reliable, accessible, and sustainable web portals; however, university software engineering education still shows practical gaps in quality assurance and automated testing. This article analyzes the relevance of Cypress as an entry-level tool for training university students in web testing, compared with Selenium and Playwright. A narrative documentary review was conducted, supported by a comparative matrix, an evaluation rubric, and a MoSCoW prioritization applied to QA competencies. The results are derived from documentary analysis rather than from a student experiment. They show that Cypress provides pedagogical advantages in initial setup, visual debugging, and immediate feedback. Playwright stands out for advanced end-to-end scenarios, while Selenium remains valuable because of its maturity and interoperability. The paper concludes that Cypress is suitable for initial training, provided that its limitations and objective evaluation metrics are explicitly taught.

Keywords: automated testing; Cypress; engineering education; public sector; quality assurance

1. Introducción

La calidad del software en los portales gubernamentales incide directamente en la continuidad del servicio, la accesibilidad y la confianza institucional. Una falla en autenticación, formularios, pagos o seguimiento de trámites no solo representa un defecto técnico: puede interrumpir el acceso ciudadano a servicios públicos y deteriorar la percepción de eficacia del gobierno digital. Por ello, el aseguramiento de calidad (QA) y el testing automatizado se han convertido en prácticas necesarias para sostener aplicaciones web sometidas a cambios frecuentes.

En la formación universitaria, el problema no se limita a enseñar conceptos de pruebas. La literatura sobre educación en testing muestra que aún existe una brecha entre los contenidos impartidos y las prácticas que demanda la industria, especialmente cuando los cursos no incorporan herramientas reales, proyectos auténticos y criterios medibles de desempeño (Garousi et al., 2020). Esta brecha resulta crítica en contextos públicos, donde los equipos técnicos suelen enfrentar restricciones presupuestales, heterogeneidad tecnológica y alta presión por mantener servicios disponibles.

Cypress se ha posicionado como una herramienta relevante para pruebas web modernas debido a su ejecución dentro del navegador, su retroalimentación visual, su configuración inicial relativamente simple y su integración con flujos de desarrollo basados en JavaScript. Sin embargo, una adopción académica responsable no debe presentarlo como solución universal. Su pertinencia debe compararse con Selenium, que conserva madurez e interoperabilidad mediante WebDriver, y con Playwright, que ofrece capacidades sólidas para escenarios end-to-end avanzados, aislamiento por contextos y depuración mediante trazas.

*Autor para la correspondencia: 202510217@tese.edu.mx
Correo electrónico: 202510217@tese.edu.mx (Israel-Antonio Marín-Castañeda).
Historial del manuscrito: recibido el 07/03/2026, última versión-revisada recibida el 09/5/2026, aceptado el 12/05/2026, publicado el 13/05/2026. **DOI:** https://doi.org/10.5281/zenodo.20142501

El objetivo de este artículo es analizar la pertinencia de Cypress como herramienta de entrada para la capacitación universitaria en testing web, comparándolo con Selenium y Playwright, y proponiendo criterios de evaluación aplicables a QA en portales gubernamentales o escenarios equivalentes del sector público mexicano.

1.1 Estado del arte y problema de investigación

El testing automatizado ha sido estudiado como una estrategia para reducir retrabajo, ejecutar regresiones de manera repetible y detectar defectos en etapas tempranas del ciclo de vida del software (Pressman & Maxim, 2014; Jorgensen, 2013). En paralelo, los estudios de educación en testing han señalado que muchos cursos universitarios mantienen una orientación conceptual o fragmentada, con menor presencia de experiencias reales y evaluación mediante métricas de calidad (Garousi et al., 2020).

Desde la perspectiva de herramientas, Selenium mantiene vigencia por su alineación con WebDriver, que permite controlar navegadores de forma interoperable y cercana al comportamiento de usuario (Selenium, n.d.). Playwright amplía las posibilidades de automatización moderna mediante contextos de navegador aislados y herramientas de traza que facilitan el diagnóstico de fallos en integración continua (Playwright, n.d.-a, n.d.-b). Cypress, por su parte, destaca por su experiencia visual de ejecución y por reducir fricción inicial en proyectos frontend, aunque su documentación reconoce trade-offs relevantes, como la imposibilidad de controlar más de un navegador abierto al mismo tiempo y restricciones específicas en pruebas cross-origin (Cypress, n.d.-b, n.d.-c).

La brecha de investigación se ubica en la falta de modelos formativos universitarios que articulen herramientas de automatización actuales, criterios objetivos de QA y escenarios relacionados con portales públicos mexicanos. Por tanto, el análisis no se plantea como un experimento de rendimiento absoluto, sino como una revisión comparativa y propuesta metodológica para orientar la incorporación de Cypress en cursos de ingeniería de software.

2. Materiales y Método

Se realizó una revisión documental narrativa con énfasis comparativo. El corpus incluyó literatura académica sobre educación en testing, libros de ingeniería de software, documentación oficial de Cypress, Selenium y Playwright, así como reportes sectoriales vinculados con gobierno digital. La revisión se delimitó principalmente al periodo 2014-2024 para literatura académica, complementándose con documentación oficial vigente de las herramientas consultada en 2026.

El diseño del estudio fue no experimental. No se aplicó un curso piloto ni se recolectaron datos de estudiantes; por consiguiente, los resultados no se presentan como evidencia empírica de aprendizaje ni como mediciones de desempeño con muestra n>50. La aportación del artículo consiste en una síntesis comparativa, una rúbrica analítica y una propuesta de indicadores para una futura validación empírica.

Los criterios de análisis fueron: curva de aprendizaje, facilidad de configuración, depuración visual, estabilidad esperada de ejecución, cobertura de navegadores y escenarios, integración con CI/CD, mantenibilidad de pruebas y capacidad de detección/diagnóstico de fallos. Cada criterio se valoró en escala ordinal de 1 a 5, donde 1 representa baja adecuación para formación inicial y 5 representa alta adecuación. La rúbrica no mide tiempos absolutos de ejecución; sintetiza evidencia documental y pertinencia pedagógica.

Para atender la necesidad de métricas estandarizadas, se incorporó una matriz MoSCoW aplicada a QA. Esta técnica prioriza requisitos en categorías Must have, Should have, Could have y Won't have this time, y se usa para clarificar qué capacidades deben formar parte de una primera implementación (Agile Business Consortium, n.d.). También se definieron indicadores propuestos de evaluación técnica y educativa, alineados con criterios de calidad de software y con el modelo ISO/IEC 25010 (International Organization for Standardization, 2011).

3. Resultados

El hallazgo principal del análisis fue que Cypress constituye una opción sólida para formación universitaria inicial en testing web por su baja fricción de entrada y su depuración visual, pero no reemplaza a Selenium ni a Playwright en todos los escenarios. Su adopción debe plantearse como parte de una ruta formativa gradual y medible, no como una superioridad absoluta.

3.1 Adopción reportada y contexto profesional

En la encuesta State of JavaScript 2024, dentro de la pregunta sobre herramientas usadas en contexto profesional, Playwright registró 3,674 menciones, Cypress 3,603 y Selenium 1,130, sobre 11,667 respuestas a esa pregunta (State of JS, 2024). Esto equivale aproximadamente a 31.5 %, 30.9 % y 9.7 %, respectivamente. Estas cifras no representan cuota global de mercado, pero sí muestran que Cypress se mantiene como herramienta relevante en la comunidad JavaScript profesional.

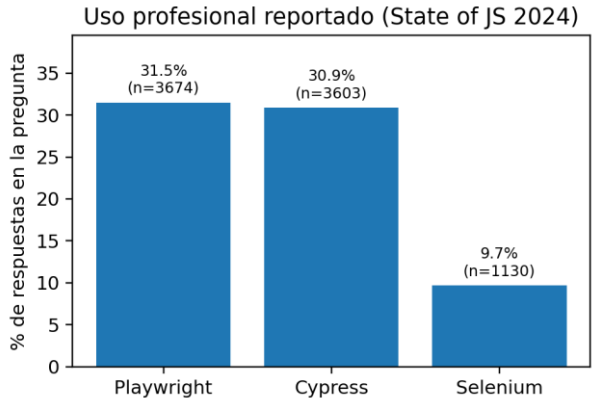

Figura 1. Uso profesional reportado de Playwright, Cypress y Selenium.

Fuente: elaboración propia con datos de State of JS (2024).

1.2. 3.2 Comparación técnica y pedagógica

La comparación muestra perfiles complementarios. Selenium ofrece madurez, interoperabilidad y amplia

experiencia acumulada, aunque su configuración inicial puede resultar menos accesible para estudiantes. Playwright destaca por su modelo moderno, sus contextos aislados y su Trace Viewer, lo que lo hace conveniente para escenarios E2E avanzados. Cypress sobresale en la etapa de formación inicial por su ejecución observable, mensajes de error comprensibles, recarga rápida y facilidad para relacionar script, DOM y comportamiento de interfaz.

Tabla 1. Matriz comparativa numérica de herramientas para capacitación QA.

Criterio (1=bajo, 5=alto)	Selenium	Playwright	Cypress	Justificación sintética
Curva de aprendizaje	2	4	5	Cypress reduce fricción inicial; Selenium exige más configuración.
Facilidad de configuración inicial	3	4	5	Cypress y Playwright se integran rápido con npm; Selenium requiere drivers.
Depuración visual y diagnóstico	2	4	5	Cypress y Playwright ofrecen mejor observabilidad visual que Selenium básico.
Estabilidad/sincronización	3	4	4	Los tres pueden ser estables con buenas prácticas; Cypress sincroniza comandos.
Cobertura de navegadores y escenarios	5	5	3	Selenium y Playwright cubren mejor escenarios amplios y multi-contexto.
Integración con CI/CD	4	4	4	Los tres pueden ejecutarse en pipelines modernos.
Mantenibilidad de pruebas	3	4	4	Playwright/Cypress favorecen patrones modernos; Selenium depende más del stack.
Detección y explicación de fallos	3	4	5	Cypress facilita diagnóstico inmediato; Playwright aporta trazas robustas.
Pertinencia para formación inicial	3	4	5	Cypress resulta más didáctico para primeras prácticas.
Ajuste a portales públicos simples/medios	3	4	4	Cypress es viable para flujos web; Playwright crece en complejidad avanzada.

Nota. La tabla sintetiza adecuación relativa para docencia y proyectos públicos de complejidad baja a media; no corresponde a una medición experimental de velocidad absoluta.

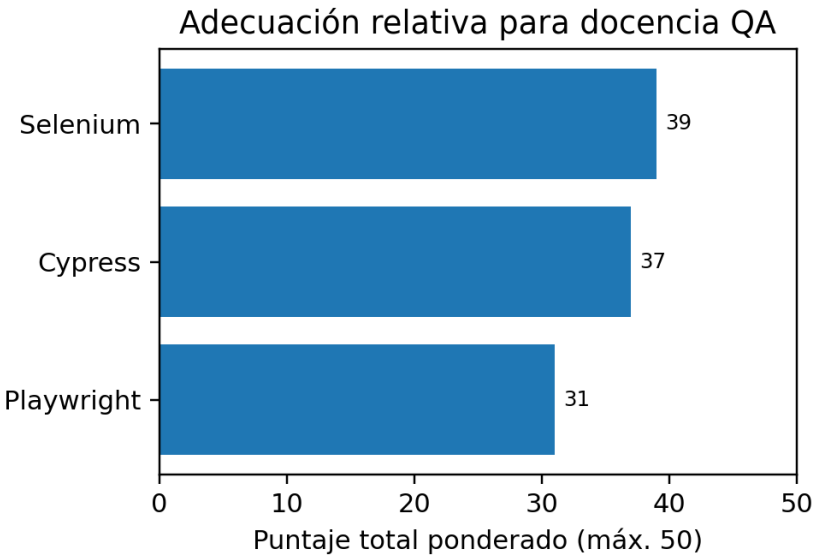

Figura 2. Puntaje total ponderado de adecuación relativa para docencia QA.

Fuente: elaboración propia con base en documentación oficial y criterios de la rúbrica.

El puntaje total fue de 39/50 para Cypress, 37/50 para Playwright y 31/50 para Selenium. La diferencia entre Cypress y Playwright es estrecha; por tanto, el resultado no debe interpretarse como una jerarquía universal, sino como una preferencia contextual para cursos introductorios de testing web.

3.3 Matriz MoSCoW para una primera implementación formativa

La priorización MoSCoW permitió definir qué capacidades deben incluirse en una primera versión del modelo de capacitación y cuáles pueden reservarse para fases avanzadas. Esta matriz evita sobredimensionar el curso inicial y permite alinear competencias con objetivos verificables.

Tabla 2. Priorización MoSCoW aplicada a competencias de QA con Cypress.

Categoría	Capacidades priorizadas	Criterio de decisión
Must have	Diseño de casos funcionales, selectores robustos, assertions, ejecución local, evidencia de fallos.	Sin estas capacidades no existe una práctica mínima viable de QA automatizado.
Should have	Integración básica con CI/CD, fixtures, comandos personalizados, reporte de defectos y cobertura funcional.	Aumenta sostenibilidad y trazabilidad, aunque puede añadirse tras la primera práctica.
Could have	Pruebas visuales, mocks avanzados, pruebas de componentes, monitoreo sintético y dashboards.	Mejora madurez técnica, pero no es indispensable para una primera cohorte.
Won't have this time	Escenarios multi-navegador simultáneo, pruebas móviles nativas, pruebas de carga y automatización de escritorio.	Quedan fuera del alcance inicial porque requieren otras herramientas o mayor complejidad.

3.4 Indicadores propuestos de evaluación

Debido a que el artículo no reporta una intervención con estudiantes, los indicadores se presentan como propuesta metodológica para un piloto posterior. Su función es evitar afirmaciones no sustentadas y preparar una evaluación replicable.

Tabla 3. Indicadores propuestos para validar el modelo en una fase piloto.

Indicador	Cálculo sugerido	Uso interpretativo
Cobertura funcional	Escenarios automatizados / escenarios críticos definidos x 100	Mide alcance real de la suite.
Tasa de fallos detectados	Casos fallidos / casos ejecutados x 100	Permite identificar estabilidad del sistema bajo prueba.
Defect leakage	Defectos posprueba / defectos totales x 100	Evalúa qué fallos escaparon de la fase de QA.
Flakiness	Ejecuciones inconsistentes / repeticiones totales x 100	Mide confiabilidad de la suite automatizada.
MTTR	Tiempo promedio entre reporte, corrección y reverificación	Relaciona calidad del reporte con velocidad de solución.
Aprendizaje técnico	Diferencia entre pretest y postest o rúbrica de desempeño	Mide mejora formativa sin inventar resultados previos.

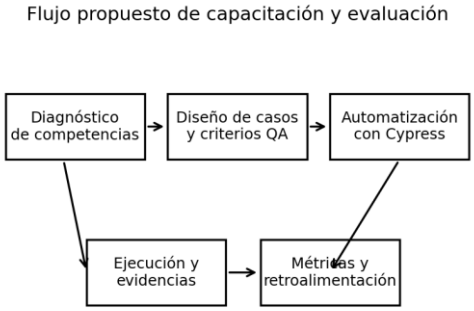

Figura 3. Flujo propuesto de capacitación y evaluación del modelo.

Fuente: elaboración propia.

3.5 Limitaciones de Cypress identificadas

La evaluación crítica muestra que Cypress no debe presentarse como herramienta suficiente para todos los escenarios empresariales. Su documentación oficial indica que no controla más de un navegador abierto al mismo tiempo, por lo que casos de colaboración en tiempo real, múltiples sesiones simultáneas o flujos estrictamente multi-usuario requieren estrategias complementarias o herramientas adicionales (Cypress, n.d.-c).

También existen restricciones en pruebas cross-origin. Cypress puede manejar distintos orígenes mediante cy.origin(), pero navegar entre dominios dentro del mismo test requiere una estructura específica; además, los iframes cross-origin no están soportados de forma general (Cypress, n.d.-b). En portales públicos, esto es relevante cuando existen integraciones con pasarelas de pago, autenticación externa o servicios federados.

Otra limitación es su dependencia natural del ecosistema JavaScript. Aunque esto es una ventaja para cursos frontend, puede ser una barrera en programas centrados en Java, .NET o entornos legacy. Asimismo, el mantenimiento de selectores sigue siendo un riesgo: una suite Cypress mal diseñada puede volverse frágil si se basa en selectores visuales o estructuras de DOM inestables. Por ello, la formación debe incluir buenas prácticas de selectores, datos de prueba y diseño de pruebas mantenibles.

4. Discusión

Los resultados responden a las observaciones de revisión en tres sentidos. Primero, se incorporaron figuras y tablas que hacen visible la comparación entre herramientas y el flujo de capacitación. Segundo, se agregó cuantificación, pero delimitada como datos de encuesta o como rúbrica documental, no como resultado empírico inventado. Tercero,

se desarrolló una evaluación explícita de las limitaciones de Cypress, evitando presentar la herramienta como solución universal.

Cypress obtiene una ventaja pedagógica clara para formación inicial porque permite que el estudiante observe la relación entre código, interfaz, DOM, ejecución y evidencia de fallo. Esta característica es relevante para cursos donde el objetivo es desarrollar criterio de QA antes que cubrir todos los escenarios complejos de automatización. En ese sentido, su valor no radica en superar siempre a Selenium o Playwright, sino en facilitar una primera apropiación de la lógica del testing automatizado.

Playwright debe considerarse un complemento natural para fases avanzadas. Su Trace Viewer, sus contextos de navegador y su cobertura multi-navegador lo hacen especialmente pertinente para escenarios de mayor complejidad. Selenium conserva vigencia cuando se requiere interoperabilidad amplia, integración con infraestructuras existentes y compatibilidad histórica. Por tanto, un currículo equilibrado podría iniciar con Cypress, incorporar criterios de QA y migrar progresivamente a Playwright o Selenium según el tipo de sistema.

La principal limitación del artículo es metodológica: al no existir todavía un piloto con estudiantes, no se reportan porcentajes de aprendizaje, tiempos de ejecución medidos en laboratorio ni reducción real de defectos en un portal gubernamental. Esta decisión fortalece la integridad del manuscrito, porque diferencia con claridad los resultados documentales de los indicadores que deberán medirse en una fase empírica posterior.

5. Conclusiones

Cypress es una herramienta pertinente para la capacitación universitaria inicial en testing web automatizado por su facilidad de configuración, depuración visual y retroalimentación inmediata. Estas características favorecen el aprendizaje activo y la construcción de competencias prácticas de QA en estudiantes que comienzan a trabajar con automatización de pruebas.

La comparación con Selenium y Playwright muestra que la selección de herramienta debe responder al objetivo formativo y al contexto técnico. Cypress resulta adecuado para introducir pruebas funcionales web y razonamiento de QA; Playwright es fuerte para escenarios E2E avanzados; y Selenium mantiene valor por su madurez, interoperabilidad y presencia histórica en automatización web.

El artículo aporta una matriz comparativa numérica, una priorización MoSCoW y una propuesta de indicadores técnicos y educativos que pueden servir como base para una implementación piloto. Como trabajo futuro, se recomienda aplicar el modelo con una cohorte estudiantil, documentar una muestra suficiente, medir cobertura, flakiness, defect leakage, tiempo de ejecución y aprendizaje antes/después, y validar la propuesta en un portal público real o en un entorno equivalente controlado.

6. Agradecimientos

El autor agradece al Tecnológico Nacional de México/TESE y a los revisores académicos cuyas observaciones permitieron fortalecer el enfoque metodológico, la trazabilidad de fuentes y la claridad de los resultados presentados.

Referencias

Agile Business Consortium. (n.d.). MoSCoW prioritisation. Recuperado el 9 de mayo de 2026, de https://www.agilebusiness.org/dsdm-project-framework/moscow-prioritisation.html

Cypress. (n.d.-a). Cross browser testing. Recuperado el 9 de mayo de 2026, de https://docs.cypress.io/app/guides/cross-browser-testing

Cypress. (n.d.-b). Cross origin testing. Recuperado el 9 de mayo de 2026, de https://docs.cypress.io/app/guides/cross-origin-testing

Cypress. (n.d.-c). Trade-offs in Cypress. Recuperado el 9 de mayo de 2026, de https://docs.cypress.io/app/references/trade-offs

Cypress. (n.d.-d). Why Cypress? Recuperado el 9 de mayo de 2026, de https://docs.cypress.io/app/get-started/why-cypress

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. Proceedings of the National Academy of Sciences, 111(23), 8410-8415. https://doi.org/10.1073/pnas.1319030111

Garousi, V., Rainer, A., Lauvås, P., & Arcuri, A. (2020). Software-testing education: A systematic literature mapping. Journal of Systems and Software, 165, 110570. https://doi.org/10.1016/j.jss.2020.110570

International Organization for Standardization. (2011). ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. https://www.iso.org/standard/35733.html

Jorgensen, P. C. (2013). Software testing: A craftsman's approach (4th ed.). CRC Press.

Mobaraya, F., & Ali, S. (2019). Technical analysis of Selenium and Cypress as functional automation framework for modern web application testing. In Proceedings of the 9th International Conference on Computer Science, Engineering and Applications (pp. 27-46). https://doi.org/10.5121/csit.2019.91803

OECD. (2020). Digital Government Index: 2019 results. OECD Publishing. https://doi.org/10.1787/4de9f5bb-en

Playwright. (n.d.-a). BrowserContext. Recuperado el 9 de mayo de 2026, de https://playwright.dev/docs/api/class-browsercontext

Playwright. (n.d.-b). Trace viewer. Recuperado el 9 de mayo de 2026, de https://playwright.dev/docs/trace-viewer

Pressman, R. S., & Maxim, B. R. (2014). Software engineering: A practitioner's approach (8th ed.). McGraw-Hill.

Selenium. (n.d.). WebDriver. Recuperado el 9 de mayo de 2026, de https://www.selenium.dev/documentation/webdriver/

State of JS. (2024). State of JavaScript 2024: Testing. https://2024.stateofjs.com/en-US/libraries/testing/