Entega 4: Suite de automatización funcional

Proyecto: Suite de pruebas E2E/Integración sobre wallet.keber.cl
Repositorio: github.com/keber/SeleniumTests
Reportes CI: keber.github.io/SeleniumTests/

Fecha: 03-ago-2025
Keber Flores – keberflores@gmail.com

1) Resumen ejecutivo

Se implementó una suite de pruebas automatizadas funcionales enfocada en los flujos críticos de Inicio de sesión y (prototipado de) Registro de la aplicación Wallet. La solución utiliza Java 17 + Maven + Selenium WebDriver + TestNG con Page Object Model (POM), se ejecuta cross-browser (Chrome/Firefox) y corre en GitHub Actions, publicando reportes de ejecución.

El objetivo fue reducir el tiempo de validación, mejorar la cobertura y generar evidencias consistentes para futuras regresiones.

2) Alcance y objetivos

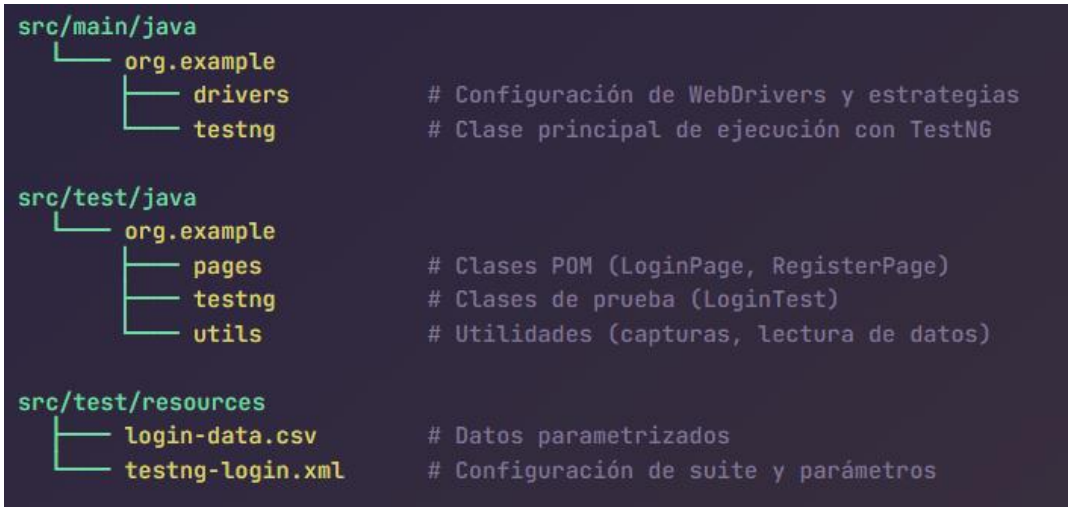
De acuerdo con la consigna, se implementaron pruebas automatizadas con Selenium WebDriver en Java, utilizando TestNG como framework de orquestación y Page Object Model (POM) para la organización del código.

Las pruebas cubrieron:

- Registro exitoso y con errores (campos obligatorios y formatos inválidos).
- Login exitoso y login fallido (incluyendo combinaciones múltiples y casos de error).
- Ejecución en Chrome y Firefox (Cross-browser).
- Validación de mensajes de error y estados posteriores a la acción.
- Capturas automáticas en caso de error y como evidencia de ejecución.
- Fuera de alcance actual: bloqueo por 3 intentos fallidos.

3) Arquitectura y entorno

- Lenguaje/stack: Java 17, Maven.
- Frameworks: Selenium WebDriver, TestNG (integración), JUnit (demostraciones), Playwright (demostración).
- Patrón: Page Object Model (clase LoginPage).
- Ejecución paralela/parametrizada: via TestNG + testng-login.xml y testng-register.xml (parámetros browser, baseUrl, archivo de datos).
- CI/CD: GitHub Actions (instalación de Chrome/Firefox, ejecución por perfiles Maven, publicación de reportes en GitHub Pages).
- Evidencias: capturas en screenshots/ y reportes HTML generados por los runners.



- Drivers: Implementados mediante patrón Strategy y WebDriverManager para facilitar la ejecución cross-browser.
- POM: Separación de lógica de interacción (Page Objects) de la lógica de prueba.
- Datos externos: Uso de CSV con combinaciones de email/contraseña y resultado esperado, integrados mediante DataProvider.

4. Ejecución y CI/CD

- Las pruebas se ejecutan localmente y en un flujo CI/CD con GitHub Actions, el cual:
- Instala dependencias y navegadores.
- Ejecuta las pruebas Selenium/TestNG en Chrome y Firefox.
- Genera reportes HTML y Allure, publicados automáticamente en GitHub Pages: https://keber.github.io/SeleniumTests/

5. Métricas Alcanzadas

- Escenarios automatizados: 4 principales (8+ validaciones).
    - Login: éxito, error (mútiples casos)
    - Registro: éxito, error (múltiples casos)
- Navegadores: Chrome y Firefox.
- Combinaciones de datos: ≥6 desde CSV.
- Capturas de error: Automáticas en cada fallo.
- POM aplicado: 2 páginas (Login, Registro).
- Reportes: Allure + capturas + logs en repositorio.

6. Lecciones Aprendidas

- La parametrización con DataProvider y archivos CSV permitió aumentar la cobertura de forma sencilla.
- La ejecución cross-browser detectó diferencias sutiles en tiempos de carga, lo que obligó a afinar la sincronización con WebDriverWait.
- La integración de CI/CD garantizó reproducibilidad y facilitó la entrega de evidencias actualizadas en cada ejecución.
- El patrón POM hizo más mantenible el código y facilitó agregar nuevos escenarios.

7. **Conclusión**

El proyecto cumple con los objetivos planteados: se validaron los flujos críticos de la aplicación, se estableció una arquitectura escalable para futuras pruebas y se garantizó la ejecución automatizada en múltiples entornos. Esta suite servirá como base para pruebas de regresión en ciclos de desarrollo posteriores.

Enlaces

- Repositorio: https://github.com/keber/SeleniumTests
- Reporte de pruebas Allure: https://keber.github.io/SeleniumTests/
- Sitio probado: https://wallet.keber.cl

Imágenes ilustrativas

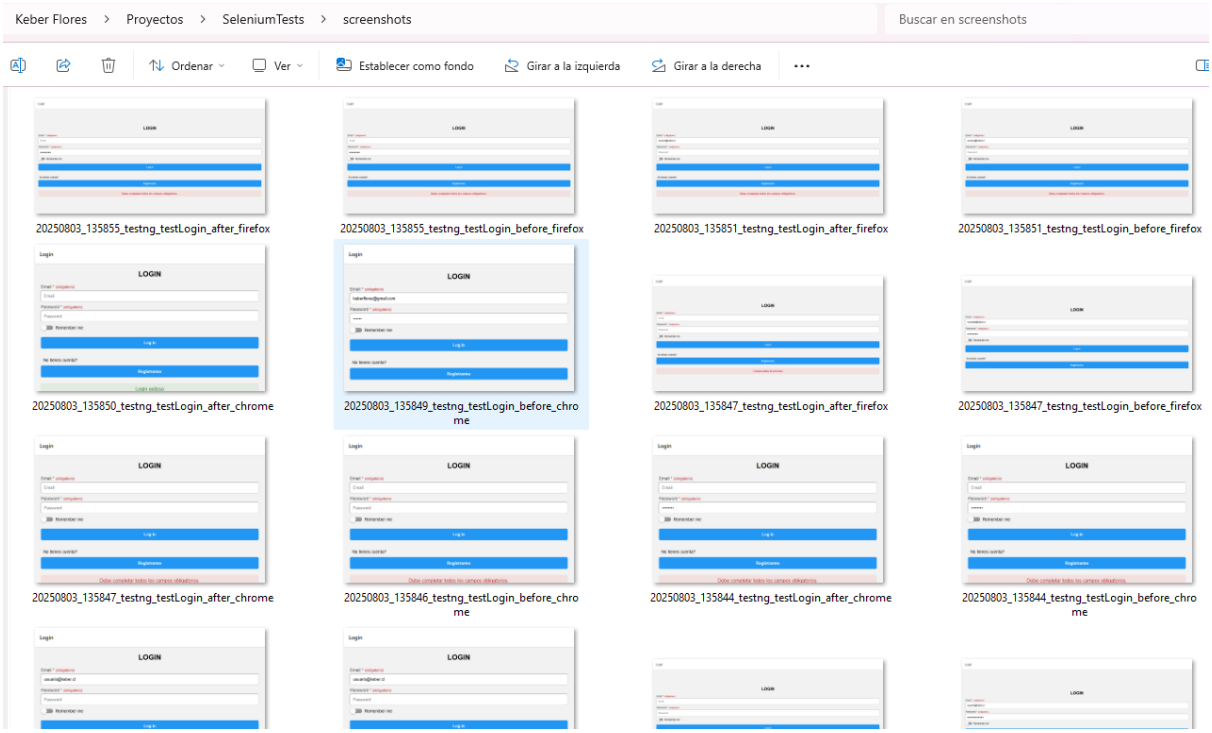


Ilustración 1: Screenshots de errores

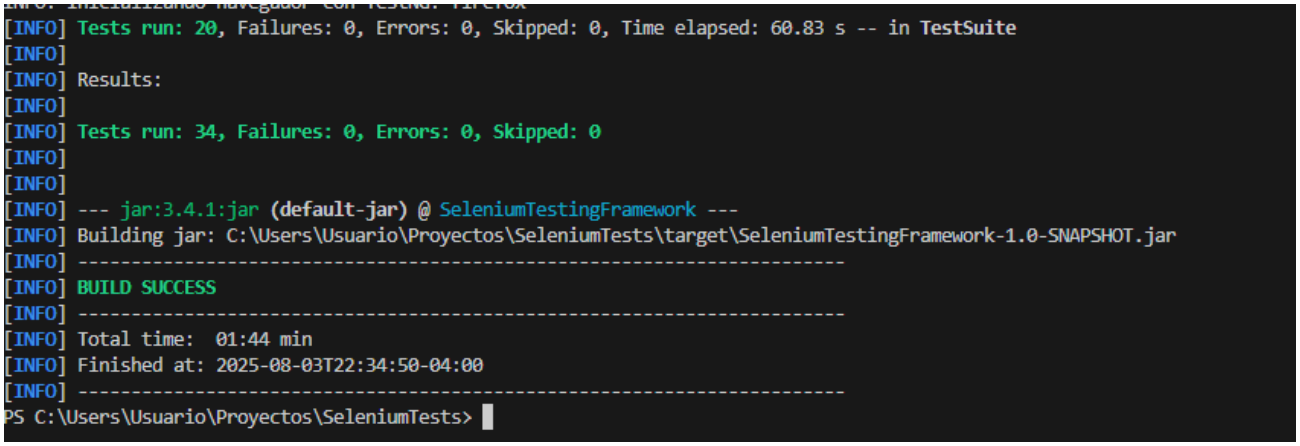


Ilustración 2: Ejecución de pruebas en desarrollo mediante mvn clean verify -P testng-tests

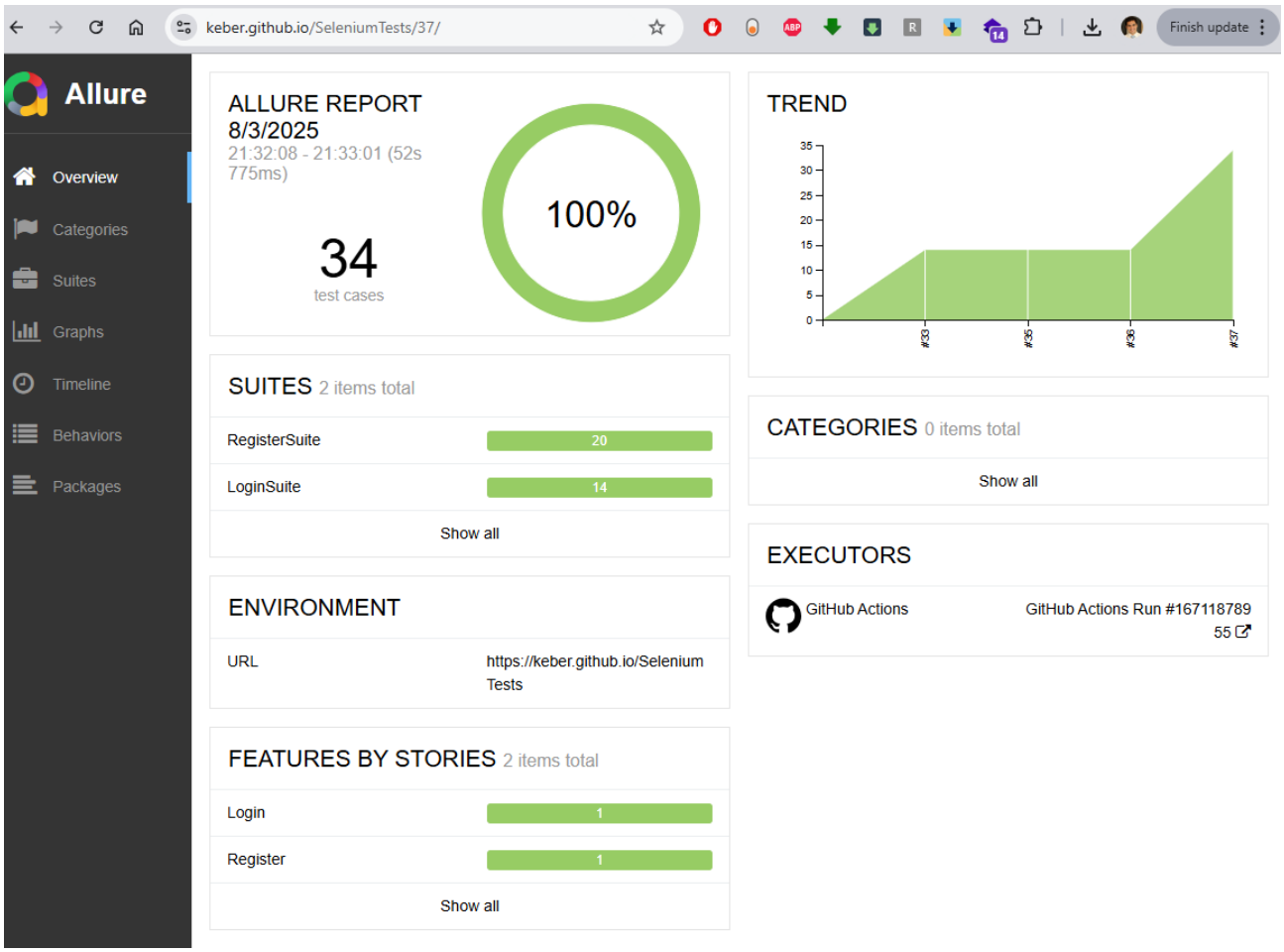


Ilustración 3: Reporte de ejecución Allure