Retroalimentación sobre el proceso de automatización con Playwright

Durante el desarrollo del flujo automatizado para **crear una Nota de Venta con Playwright**, me encontré con una serie de desafíos tanto en la configuración del entorno local como en la ejecución de los tests.
En mi experiencia previa, había tenido una asignatura de **automatización de pruebas con Eclipse, Cucumber y Java**, donde al inicio me costó bastante comprender los conceptos de automatización y el manejo de frameworks de testing.
Sin embargo, esa base previa me ayudó a entender más rápido el flujo en **Playwright con TypeScript**, aunque igualmente me enfrenté a nuevos retos técnicos.

Uno de los primeros problemas que tuve fue que **npx no era reconocido** en la consola de Visual Studio Code. Descubrí que esto se debía a que **Node.js no estaba configurado en el PATH**, así que reinstalé Node.js (versión LTS), marqué la opción *"Add to PATH"* durante la instalación, y verifiqué que todo funcionaba con node -v y npx -v.
Después de esto, ejecuté npm install para reinstalar las dependencias del proyecto y pude volver a correr los tests con npx playwright test --headed.

En las pruebas automatizadas, uno de los problemas principales fueron los **selectores dinámicos**, especialmente en los campos que usan select2.
Estos componentes generan IDs dinámicos que cambian en cada carga de página, por lo que comandos como page.click o page.fill fallaban constantemente.
Consulté con la IA y entendí que debía trabajar con **selectores parciales** usando expresiones de atributos como [id^="prefix_"][id$="_suffix"], además de incluir **waitForSelector con timeouts más largos** para asegurar que los elementos estuvieran listos antes de interactuar.

Otro problema frecuente fueron los **timeouts por elementos invisibles**.
Esto lo solucioné utilizando **await expect(locator).toBeVisible()** en cada paso clave, y agregando pausas con waitForTimeout únicamente cuando era necesario.
Para depurar mejor, implementé **test.step con descripciones claras** y agregué mensajes de console.log para saber exactamente qué paso se estaba ejecutando y en qué parte fallaba la prueba.

También tuve un error en **GitHub Actions**, ya que el pipeline fallaba por incompatibilidad de versiones de Node y dependencias.
Lo solucioné actualizando el workflow para que usara **Node 20** y el comando npx playwright install --with-deps.

Además, incorporé la acción upload-artifact@v4 para que los **reportes HTML de Playwright** se generen y suban automáticamente al repositorio.

En cuanto a la lógica de la prueba, logré crear un flujo **100% estable y dinámico**, que inicia sesión, navega a la sección de Notas de Venta, selecciona sucursal, bodega, cliente, contacto, canal de venta, moneda, producto y cantidad sin fallar.
Finalmente, ejecuté todas las pruebas y pasaron con éxito, dejando el proyecto totalmente funcional y con integración continua lista en GitHub.

Lo que más me costó fue **entender y manejar los selectores dinámicos**, ya que era la primera vez que me enfrentaba a este tipo de comportamiento.
Sin embargo, fue también lo que más aprendí y me ayudó a mejorar mis habilidades con Playwright.
Pese a los errores iniciales y al tiempo invertido en resolver problemas técnicos, logré terminar el proyecto dentro del plazo.

```
C:\Users\serdo\Desktop\relke-qa-challenge\relke-qa-challenge>npx playwright test --headed

Running 1 test using 1 worker

  ok 1 tests\nota_de_venta.spec.ts:3:5 › Login y navegación inicial a Nota de Venta (13.5s)
Paso 1: Login Completado
Paso 2: Navegación a Notas de Venta completada
Paso 3: Página de Nota de Venta cargada
Paso 4: Documento Tributario "Boleta Electrónica" seleccionado
Paso 5: Sucursal "Casa Matriz" seleccionada
Paso 6: Bodega seleccionada "Principal"
Paso 7: Cliente seleccionado "FALABELLA"
Paso 8: Agregar Contacto
Paso 9: Canal de Venta "Web" seleccionado
Paso 10: Moneda "Pesos" seleccionada
Paso 11: Producto agregado con éxito
Paso 12: Cantidad de Producto agregada con éxito

  1 passed (16.2s)
```

Log de Aprendizaje - Proyecto QA Automatización (Playwright)

1. **Configuración del entorno:** Resolví problemas con npx y Node.js reinstalando la versión LTS, configurando el PATH y reinstalando dependencias con npm install.
2. **Selectores dinámicos:** Aprendí a manejar IDs variables usando selectores parciales ([id^="prefix_"][id$="_suffix"]) y waitForSelector para evitar fallos por elementos no disponibles.
3. **Mejoras en estabilidad:** Implementé test.step con console.log para depuración y validaciones con expect(locator).toBeVisible() para sincronizar acciones con la carga del DOM.
4. **CI/CD con GitHub Actions:** Configuré un pipeline con **Node 20**, instalación de dependencias Playwright y generación automática de reportes HTML mediante upload-artifact@v4.
5. **Evolución personal:** Apliqué conocimientos previos de Cucumber y Java, pero con Playwright el flujo fue más intuitivo; resolví errores complejos y completé el proyecto dentro del plazo.