


1. Armá el Test Plan detallando los casos de prueba que considere necesarios para realizar la cobertura completa de todos los escenarios posibles sobre las funcionalidades “Inicio de sesión” y “Carrito de compras” de la siguiente web: <https://www.saucedemo.com/>. Podes hacer uso de una plantilla para escritura de Test Plans generada por vos mismo o de terceros.

Casos de prueba:

Login:  Feature: Login

Carrito:

<https://docs.google.com/spreadsheets/d/1kd3UNXKBaTTu1vB2kNp28uFfVpXo58Jpavk-O3B0R2A/edit?usp=sharing>

Test Plan

1. Introducción

Este es el Test Plan para "Inicio de Sesión" y "Carrito de Compras" del sitio. El objetivo es asegurar la cobertura completa de todos los escenarios posibles, garantizando la calidad y el correcto funcionamiento de estas características críticas.

2. Alcance

El alcance de este plan de pruebas se limita a las siguientes funcionalidades:

- **Inicio de Sesión (Login):**

- ☐ Acceso al sitio.
- ☐ Validación de credenciales (usuario/contraseña).
- ☐ Manejo de errores de inicio de sesión.
- ☐ Comportamiento de la sesión una vez iniciada.

- **Carrito de Compras (Shopping Cart):**

- ☐ Añadir productos al carrito.
- ☐ Eliminar productos del carrito.
- ☐ Actualizar cantidades de productos en el carrito (si aplica).
- ☐ Visualización del carrito.
- ☐ Navegación desde el carrito hacia otros puntos de la compra (ej. checkout).

3. Entorno de Pruebas

- **URL del Sitio:** <https://www.saucedemo.com/>
- **Navegadores Soportados:**
 - ☐ Google Chrome (última versión)
 - ☐ Mozilla Firefox (última versión)
 - ☐ Microsoft Edge (última versión)
- **Sistemas Operativos:**
 - ☐ Windows 10/11
- **Dispositivos:**
 - ☐ Notebook Lenovo

4. Estrategia de Pruebas

Se aplicarán las siguientes técnicas de prueba:

Pruebas Funcionales: Para verificar que las funcionalidades se comportan según lo esperado.

Pruebas de Usabilidad: Para asegurar que la interacción con el usuario sea intuitiva y eficiente.

Pruebas de Regresión: Para garantizar que los cambios o nuevas implementaciones no afecten las funcionalidades existentes.

Pruebas de Compatibilidad: no está en este scope del challenge

Pruebas de Rendimiento (básicas): no está en el scope de este challenge

5. Criterios de Entrada y Salida

- **Criterios de Entrada:**
 - ☐ Acceso a la URL del sitio web.
 - ☐ Ambiente de pruebas configurado y estable.
 - ☐ Credenciales de prueba disponibles.
 - ☐ Documentación de requisitos.
- **Criterios de Salida:**
 - ☐ Todos los casos de prueba críticos fueron ejecutados y aprobados.
 - ☐ Defectos encontrados registrados y categorizados.
 - ☐ Porcentaje de aprobación de casos de prueba según umbral definido (ej. >95% para casos críticos).
 - ☐ Informe de pruebas generado.

7. Responsabilidades

Equipo de QA: Diseño, ejecución y reporte de casos de prueba.

Desarrolladores: Corrección de defectos.

Gerente de Proyecto/Producto: A convenir

8. Cronograma (Ejemplo, a definir)

Fase 1: Planificación y Diseño de Casos de Prueba: [Fecha Inicio] - [Fecha Fin]

Fase 2: Ejecución de Pruebas - Funcionalidad Login:

Fase 3: Ejecución de Pruebas - Funcionalidad Carrito de Compras:

Fase 4: Retesting y Pruebas de Regresión:

Fase 5: Reporte Final:

9. Herramientas

- **Gestión de Casos de Prueba y Defectos:** Excel
- **Automatización de Pruebas:** Cypress.

10. Métricas y Reportes

Porcentaje de Casos de Prueba Ejecutados: $(\text{Ejecutados} / \text{Total}) * 100$

Porcentaje de Casos de Prueba Aprobados: $(\text{Aprobados} / \text{Ejecutados}) * 100$

Número de Defectos Encontrados: Desglosado por severidad y prioridad.

Tasa de Aprobación/Fallo por Funcionalidad.

11. Riesgos y Contingencias

Riesgo

Impacto

Contingencia

Cambios en los requisitos durante las pruebas.	Posible re-trabajo y retrasos.	Comunicación constante con el equipo de desarrollo/producto. Adaptación rápida del plan.
Entorno de pruebas inestable.	Bloqueo de la ejecución de pruebas.	Coordinación con DevOps/Administración de Sistemas para asegurar la estabilidad.
Falta de recursos (personal, tiempo).	Cobertura de pruebas incompleta.	Priorización de casos de prueba críticos. Escalada a la gerencia.
Bugs no detectados.	Lanzamiento de software con defectos.	Revisión por pares de casos de prueba. Pruebas exploratorias.

2. Seleccioná, y escribí de forma detallada, 3 casos de prueba de la funcionalidad “Inicio de Sesión” y 3 casos de prueba de la funcionalidad “Carrito de compras”. Podes hacer uso de una plantilla para escritura de casos de prueba generada por vos mismo o de terceros.

Login.

Feature: Inicio de Sesión en SauceDemo.com

Como usuario de SauceDemo.com

Quiero poder iniciar y cerrar sesión

Para acceder a las funcionalidades de la tienda y proteger mi cuenta.

@Login @Smoke

Scenario: Inicio de sesión exitoso con credenciales válidas

Given estoy en la página de inicio de sesión de SauceDemo

When ingreso "standard_user" en el campo de usuario
And ingreso "secret_sauce" en el campo de contraseña
And hago clic en el botón "Login"
Then debo ser redirigido a la página de productos
And la URL debe ser "<https://www.saucedemo.com/inventory.html>"

@Login @Negative

Scenario: Inicio de sesión fallido con contraseña incorrecta

Given estoy en la página de inicio de sesión de SauceDemo
When ingreso "standard_user" en el campo de usuario
And ingreso "wrong_password" en el campo de contraseña
And hago clic en el botón "Login"
Then debo ver el mensaje de error "Epic: Username and password do not match any user in this service"
And debo permanecer en la página de inicio de sesión

@Login @Negative

Scenario: Inicio de sesión fallido con usuario incorrecto/no registrado

Given estoy en la página de inicio de sesión de SauceDemo
When ingreso "invalid_user" en el campo de usuario
And ingreso "secret_sauce" en el campo de contraseña
And hago clic en el botón "Login"
Then debo ver el mensaje de error "Epic: Username and password do not match any user in this service"
And debo permanecer en la página de inicio de sesión

Carrito

Feature: Carrito de Compras en [SauceDemo.com](https://www.saucedemo.com)

Como usuario de [SauceDemo.com](https://www.saucedemo.com)
Quiero poder añadir, ver y eliminar productos de mi carrito de compras
Para gestionar mi selección antes de la compra

@Cart @Smoke

Scenario: Añadir un solo producto al carrito

Given he iniciado sesión como "standard_user"
And estoy en la página de productos
When hago clic en el botón "Add to cart" para "Sauce Labs Backpack"
Then el botón "Add to cart" de "Sauce Labs Backpack" debe cambiar a "Remove"
And el icono del carrito de compras debe mostrar "1" producto

@Cart @Smoke

Scenario: Añadir múltiples productos diferentes al carrito

Given he iniciado sesión como "standard_user"

And estoy en la página de productos

When hago clic en el botón "Add to cart" para "Sauce Labs Backpack"

And hago clic en el botón "Add to cart" para "Sauce Labs Bike Light"

Then el botón "Add to cart" de "Sauce Labs Backpack" debe cambiar a "Remove"

And el botón "Add to cart" de "Sauce Labs Bike Light" debe cambiar a "Remove"

And el icono del carrito de compras debe mostrar "2" productos

@Cart @Smoke

Scenario: Verificar productos en la página del carrito

Given he iniciado sesión como "standard_user"

And he añadido "Sauce Labs Backpack" y "Sauce Labs Bike Light" al carrito

When hago clic en el icono del carrito de compras

Then debo ver "Sauce Labs Backpack" en la lista de productos del carrito

And debo ver "Sauce Labs Bike Light" en la lista de productos del carrito

And sus detalles (nombre, descripción, precio, cantidad) deben ser correctos

3. En base al punto 1, identificá la suite de Smoke y la suite de Regresión. Justificá tu respuesta.

Smoke

El Smoke se centrara en los flujos principales y más básicos de las funcionalidades de login y carrito.

Casos de Prueba Incluidos en la Suite de Smoke:

TC: Inicio de sesión exitoso con usuario válido (standard_user).

Es el escenario básico para acceder a la aplicación. Sin un login exitoso, ninguna otra funcionalidad puede ser probada. Sería un bloqueo total.

TC: Inicio de sesión fallido con contraseña incorrecta.

Verifica que el mecanismo de seguridad básico (rechazo de credenciales incorrectas) funciona como se espera.

TC: Añadir un solo producto al carrito.

La acción principal del carrito, necesaria para que un usuario pueda iniciar una compra.

TC: Verificar productos en la página del carrito.

Se asegura que los productos añadidos se muestran correctamente, lo que es un importante step para la visibilidad del usuario antes de proceder al pago.

TC: Navegar al checkout desde el carrito ("Checkout").

Verifica el flujo crítico de transición desde el carrito hacia el proceso de pago, lo que valida el inicio del proceso de compra.

Estos cinco casos de prueba cubren el **flujo mínimo viable** de ambas funcionalidades. Permiten validar que un usuario puede:

1. **Entrar** al sistema.
2. El sistema **rechaza** intentos inválidos.
3. **Añadir** un producto.
4. **Ver** el producto en el carrito.
5. **Proceder** al siguiente paso de la compra.

Si alguno de estos casos falla, indica un issue que impediría realizar pruebas más profundas. Esto justificaría detener el proceso de despliegue o la ejecución de la suite de regresión, en NO GO.

Suite de Regresión

La suite de Regresión incluirá la mayoría, si no todos, los casos de prueba definidos en el Test Plan mencionado arriba, con énfasis en la estabilidad y el comportamiento detallado de las funcionalidades.

Casos de Prueba Incluidos en la Suite de Regresión:

Todos los casos de prueba del Test Plan (tanto de Login como de Carrito) serán parte de la Suite de Regresión. Esto incluye:

- **Todos los casos de Login:** Login-001 a Login-015.
- **Todos los casos de Carrito:** Cart-001 a Cart-016.

Justificación General de la Suite de Regresión:

La suite de regresión debe ser exhaustiva para garantizar la **estabilidad del sistema después de cualquier cambio**. Al incluir todos los casos de prueba identificados en el plan (incluyendo casos negativos, de borde y de rendimiento específicos de SauceDemo), nos aseguramos de que:

Las funcionalidades principales (ya cubiertas por Smoke) sigan operando.

El manejo de **errores y entradas inválidas** no se ha degradado.

Los **casos de borde** y las interacciones específicas se mantienen consistentes.

La **persistencia de la sesión** y el carrito (según el comportamiento esperado de SauceDemo) no se han visto afectados.

El **cierre de sesión** funciona correctamente, protegiendo la cuenta del usuario.

4. Identificá algún defecto en la página a testear, y reportalo. Podés hacer uso de alguna plantilla de reporte de defectos generada por vos mismo o de terceros.

ISSUE:	Summary:	
Description		Preconditions
Steps		Environment
Expected result		Browser
Actual result		Links to: epic/US/testsuite
Evidence		Assigned to:
		Priority:
		Severity:

Por temas de espacio, dejo el texto del reporte del issue. a continuacion.

Summary: Campos de Entrada de Datos de Usuario del Carrito No Se Borran al Regresar.

Título del Issue: Los campos de entrada de "Información del Cliente" (First Name, Last Name, Zip/Postal Code) no se vacían al navegar de vuelta a la página de detalles del carrito o a la de inventario.

URL Afectada: <https://www.saucedemo.com/checkout-step-one.html> y <https://www.saucedemo.com/checkout-step-two.html>

Severidad: Media (Puede causar frustración al usuario al tener que borrar manualmente la información y es una brecha en la privacidad si la información se mantiene visible).

Prioridad: Media (Afecta la experiencia de usuario y la seguridad/privacidad percibida).

Pasos para Reproducir:

1. Iniciar sesión en <https://www.saucedemo.com/> con credenciales válidas (ej. standard_user / secret_sauce).
2. Añadir al menos un producto al carrito desde la página de inventario.
3. Navegar a la página del carrito haciendo clic en el ícono del carrito.
4. Hacer clic en el botón "Checkout".
5. En la página "Checkout: Your Information", ingresar datos en los campos:
 - First Name (ej. Juan)
 - Last Name (ej. Cito)

- Zip/Postal Code (ej. 12345)

6. Hacer clic en el botón "Cancel" para regresar a la página del carrito (/cart.html).

7. Desde la página del carrito, hacer clic en "Checkout" nuevamente.

8. Observar el comportamiento inesperado.

Comportamiento Esperado: Al regresar a la página "Checkout: Your Information" (/checkout-step-one.html) después de haber hecho clic en "Cancel" o haber navegado hacia atrás, los campos de entrada para First Name, Last Name y Zip/Postal Code deberían estar **vacíos** para una nueva entrada de datos o para proteger la privacidad del usuario.

Comportamiento Actual (Incidente): Al regresar a la página "Checkout: Your Information", los campos de entrada de texto (First Name, Last Name, Zip/Postal Code) aún contienen la información que se ingresó previamente. Los datos persisten incluso si se navega a la página de inventario y se regresa de nuevo al checkout.

Comentario que dejaría como posible causa raíz: El estado del formulario de información del cliente no se está reseteando o limpiando correctamente cuando el usuario cancela el proceso de checkout o navega hacia atrás. La aplicación podría estar almacenando estos datos en el estado de la sesión o en la caché del navegador sin una limpieza explícita.

Criterios que propondría para prioridad y severidad:

Alta (High)

Bloqueador / Crítico: La funcionalidad principal es inaccesible o inoperable. Impide el uso fundamental del sistema o causa pérdida de datos.

Media (Medium)

Mayor: La funcionalidad principal tiene un defecto notable, pero se puede usar con dificultad o con un workaround. Afecta la experiencia del usuario o el flujo de trabajo importante.

Baja (Low)

Menor / Estético: Defecto cosmético, error tipográfico, o problema de usabilidad menor. La funcionalidad principal no se ve afectada y el sistema es completamente usable. Impacto mínimo.

Criterios de Prioridad o urgencia de resolución:

Alta (High)

Bloqueador de Lanzamiento / Urgente: Debe corregirse de inmediato. Afecta un camino crítico de usuario o un requisito de negocio clave. No se puede liberar la versión con este defecto.

Media (Medium)

Importante: Necesita ser corregido en la próxima versión o sprint. Afecta funcionalidades importantes pero no críticas, o causa inconvenientes significativos al usuario sin bloquearlo por completo.

Baja (Low)

Opcional / Futuro: Puede corregirse en una versión futura o cuando haya tiempo. Es un defecto menor, cosmético, o una mejora que no impacta el uso esencial del sistema. No es crucial para la liberación actual.

5. Si tuvieras que hacer pruebas de APIs, indicá los pasos a seguir para poder llevarlas a cabo desde la definición hasta la ejecución.

1. Definición.

En primer lugar, me pondría a tener en claro donde estoy parado. El punto de partida sería que probar en base a la información que vaya recolectado.

Comprendiendo la documentación de la API con alguna colección de Postman, un Readme, o Swagger.

Saber qué hace cada endpoint, sus métodos, con los parámetros de cada solicitud, como sería la estructura de la respuesta, https esperados, y requisitos de autenticación.

Si no tengo estas bases, se me complica entender que probar y como realizar interacciones con la API.

Seleccionaría la herramienta de prueba: Postman, por defecto. algo open source tipo Insomnia o Pytest en Python. Esto es importante, ya que se darían los cimientos para ir escalando a una automatización.

2. Diseño de Casos de Prueba de API

En este punto ya iría verificando y validando a través de pasos. En esencia no hay mucha diferencia con otro tipo de pruebas. Pero vale la pena un check a tener en cuenta detalladamente como:

Para el escenario a probar, voy con los pasos. En el caso de un TC de una API sería:

A qué endpoint. El método, el encabezado(header), el cuerpo de la solicitud(body), los datos de envío, los parámetros de consulta en caso de ser necesario. Por otro lado, no olvido los datos de input, como cualquier otro caso de prueba. Creo que con estos puntos, se reduce la ambigüedad y nuevamente...da info clara para automatizar.

En un segundo paso, quisiera ser específico con el establecimiento de criterio de aceptación. O sea definir acciones como: qué resultado se espera, como un código de estado, estructura de la respuesta, sus valores. Sin estos criterios de aceptación, como se aplica en cualquier otro caso de prueba, se complica saber si la prueba pasa o falla.

3. Configuración del entorno

Basado en la documentación que me proporciona el primer punto, ya estaría en condiciones de configurar el acceso al entorno o diferentes ambientes de la API. De esta manera me aseguro que hay un entorno estable y controlado.

Como complemento, voy a la gestión de datos de prueba. Los que sean necesarios para el consumo de la API: usuarios, productos, etc.

4. Implementación y Automatización

En esta etapa, podría escribir los scripts de prueba. O configurarlos para hacer solicitudes y aserciones de validación. Esto es interesante para una regresión. Aparte de esto, también es importante seleccionar los casos a automatizar y la estrategia.

Previamente agrupo las pruebas de manera lógica, como por funcionalidad, o endpoint, o suite de regresión. Así puedo mostrar de manera prolija y organizada los resultados.

5. Ejecución de Pruebas

Ejecuto las pruebas de manera manual o automáticamente. Monitoreo y registro de resultados de las ejecuciones. Qué casos fallaron, cuáles no. De esta manera tengo un seguimiento claro y reporte de defectos.

6. Análisis y Reporte de Defectos

Puede integrarse al punto anterior, pero quise explayarme más. Porque considero importante la interpretación de resultados y comunicación de los issues. ¿Cómo los interpreto? Analizándolos, buscando la causa raíz. ¿Es algo propio de la API? un tema de los datos de input? ¿Algún script mal escrito? En este punto, veo importante entender por qué no ha pasado la prueba.

Interpretar los resultados y comunicar los problemas: Como sucede en cualquier prueba que falla, se levanta el defecto con una comunicación escrita clara, concisa, con una planilla igual que como se venía haciendo con otros tipos de pruebas, y mismos criterios: pasos, comportamientos, evidencia, prioridad, criticidad.

7. Mantenimiento y Regresión

En pocas palabras, es ser parte de una calidad continua de la API. Esto implicaría mantener las pruebas teniendo en cuenta su evolución como nuevos endpoints, cambios, actualización. Analizar si hay pruebas que quedaron depreciadas. En cuanto a la regresión, una corrida de las pruebas preseleccionadas, implica asegurarse que los cambios introducidos en la API, por ejemplo no afectaron negativamente las funcionalidades existentes.

6. ¿Cómo escribirías un test de API? Ejemplifique.

En el punto anterior he colocado los puntos necesarios y steps para la creación de un test de API. Explayando un poco más en este punto, puedo complementar a través de un ejemplo concreto:

Escenario: pruebo una API de ejemplo que retorna información de usuarios.

API de Ejemplo: SauceDemo

Endpoint: `https://www.saucedemo.com/users/1`

Método: GET

Comportamiento Esperado: Al solicitar el usuario con ID 1, la API debe devolver los datos del usuario "Juan Cito" con un código de estado 200 OK.

Pasos en Postman:

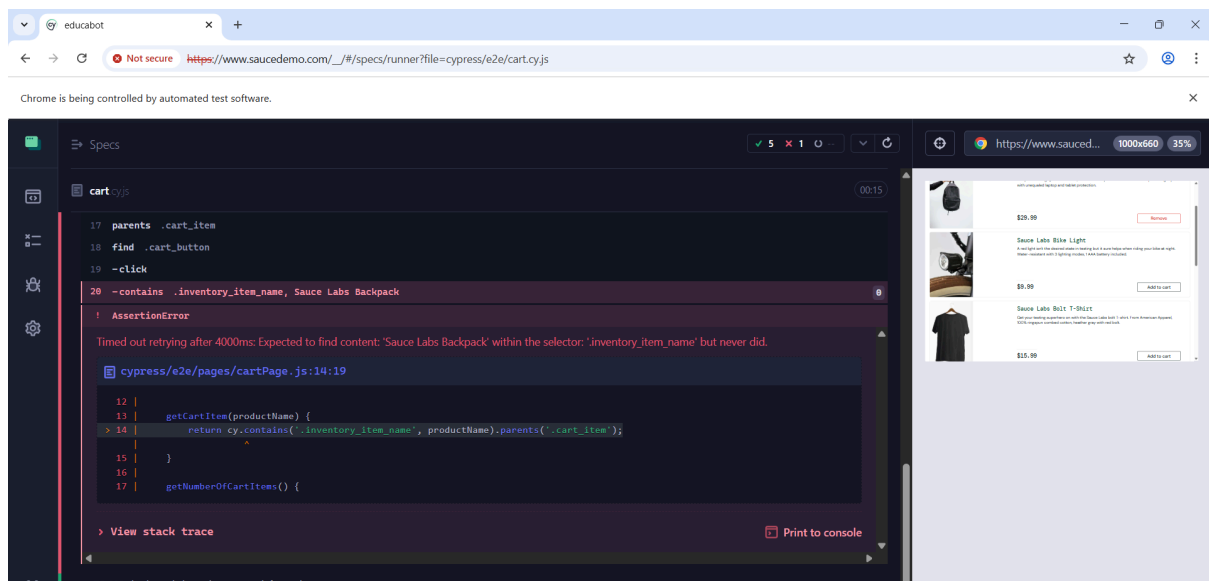
1. Crear una nueva solicitud: Abre Postman y hacer clic en + para crear una nueva solicitud HTTP.

2. Seleccionar el Método HTTP: En el desplegable, elige GET.
3. Ingresar el Endpoint (URL): En el campo de URL, escribir `https://saucedemo.com/users/1`.
4. Enviar la Solicitud: Hacer clic en el botón Send.
 - a. Resultado (Respuesta de la API): Se verá la respuesta en la sección inferior de Postman.
5. Escribir Aserciones (Tests en Postman): Esta es la parte clave para automatizar la verificación. Hacer clic en la pestaña "Tests" debajo de la URL. Aquí se escribe el código JavaScript para validar la respuesta.

Sección 2: automatización.

4. Hacé que uno de los casos de prueba falle intencionalmente.

El caso que fallo intencionalmente es el TC-004 del test set de Carrito



Repositorio: <https://github.com/marcelodanielm/educabot>

Pasos que he realizado:

1. He chequeado las versiones de Node.
2. He chequeado la versión de Cypress.
3. He chequeado si hubo alguna modificación importante en la última versión de Cypress que afecte mis pruebas.
4. Copilot.
5. Es-Lint.
6. Identifico las páginas que involucran las pruebas para usar Page Object. Por ser un framework pequeño, no es necesario usar Page Object, sin embargo, lo tomo en cuenta en caso que se soliciten mayor cantidad de pruebas y demostrar conocimientos.
7. Creo la estructura de archivos y carpetas Pages, [cy.js](#), y js necesarios para trabajar con la identificación de selectores, estructura de las clases, y páginas.
8. Instalo Mochawesome como framework de reportes en Cypress. Allure necesita Java 8 y por eso no lo he elegido. Quiero algo rápido y sencillo de instalar y usar.
9. Configuro Mochawesome en [cypress.config.js](#)
10. Genero `npx mochawesome-report-generator cypress/reports`

Uso de IA.

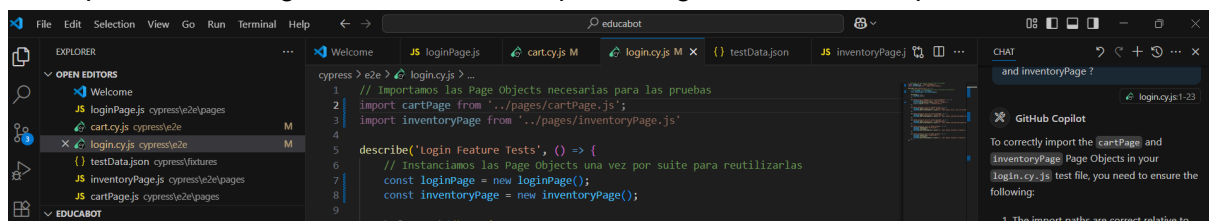
1. Copilot me corrigió inputs de prueba para un caso de intento de login con campos vacíos.

```
33 it('LOGIN-004 Caso Borde: Inicio de sesión fallido con campos vacíos (ambos)', () => {
34     loginPage.login('', '');
35
36 });
```

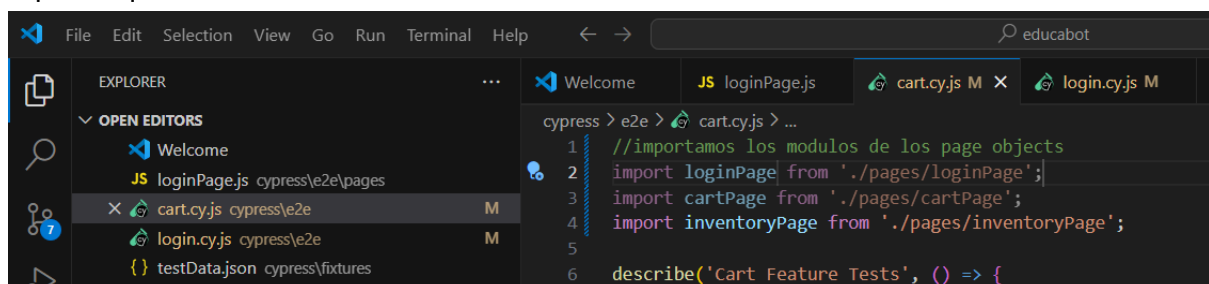
delete mode 100644 cypress/e2e/pages/productsPage.js
PS C:\Users\danie\OneDrive\Escritorio\proyectos programacion\educabot> git push
Enumerating objects: 18, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 2 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 5.09 KiB | 173.00 KiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/marcelodanielm/educabot.git
b1ba371..804e359 main -> main

Inconveniente con la importación de módulos.

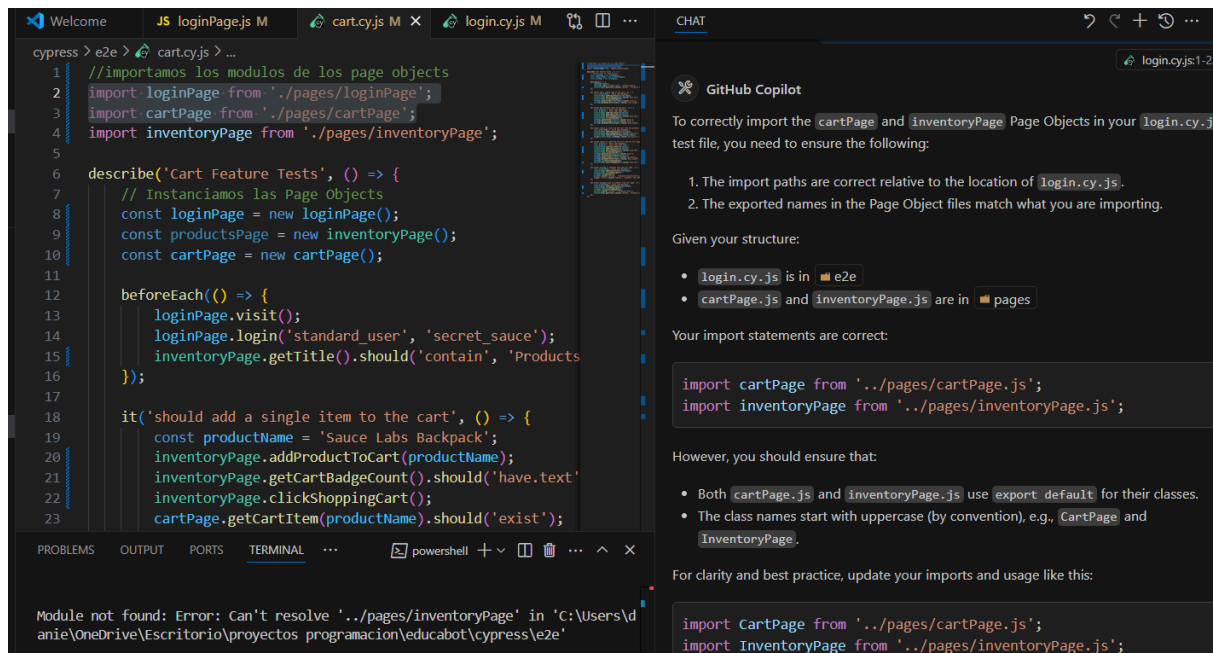
Al correr los tests, me arrojaba un error de no encontrar el módulo en la ruta correspondiente. Luego de varios intentos por corregir la ruta, fui a Copilot.



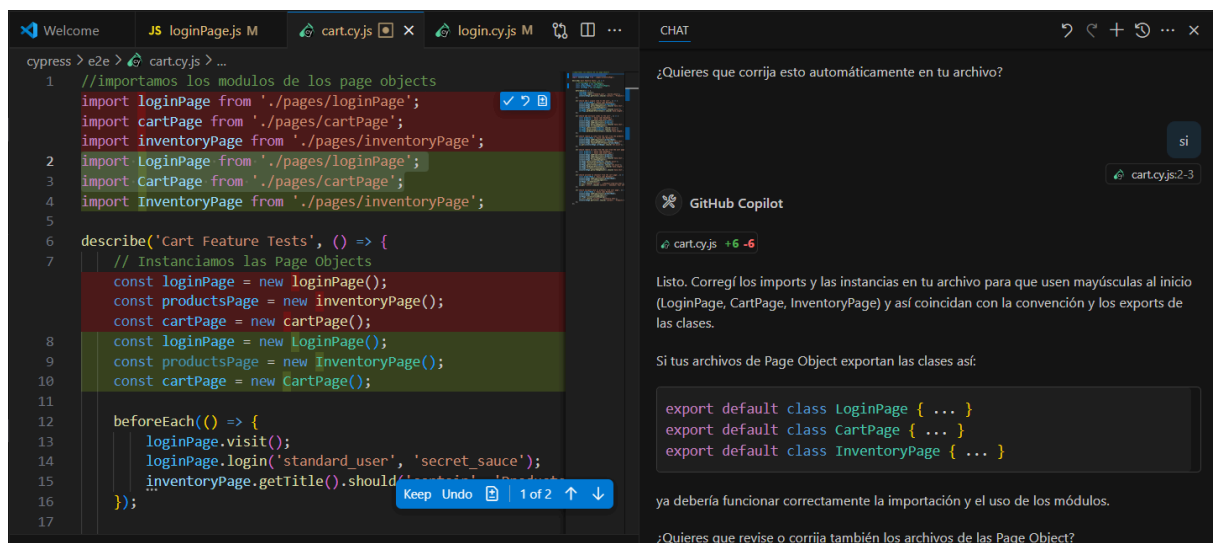
Se logra la importación de un módulo: inventoryPage. Sin embargo. Quedan dos por importar que no son leídos.



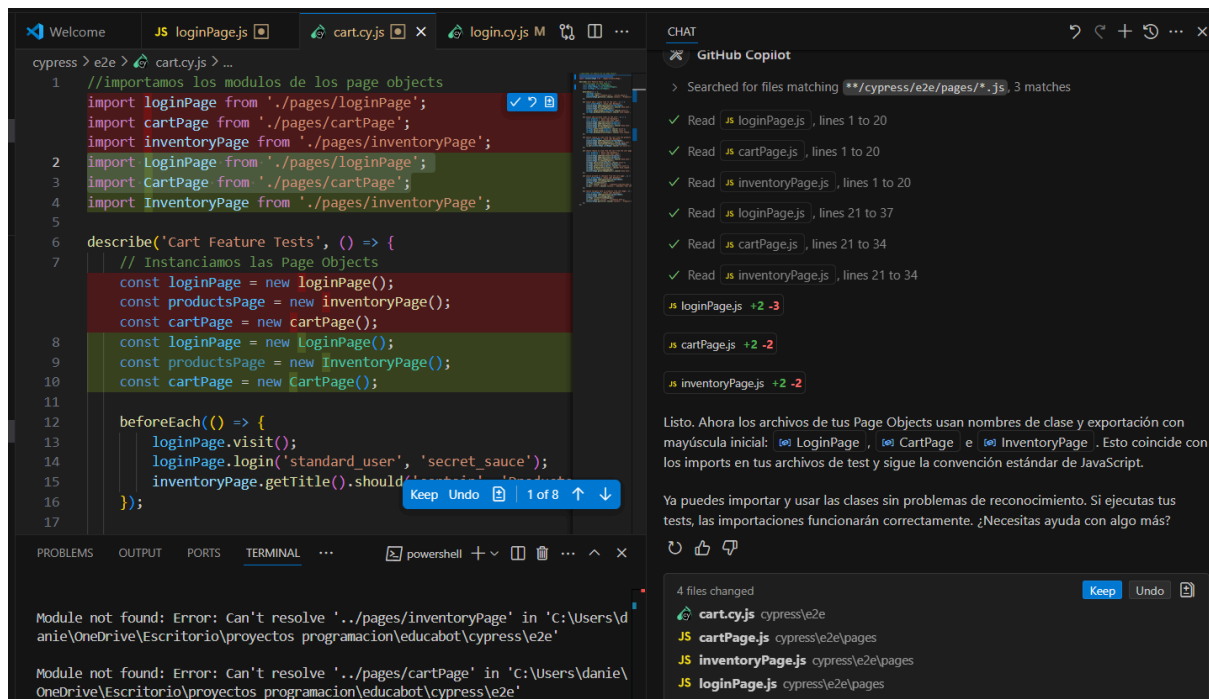
Copilot sugiere corregir los nombres de clase. He cometido el error de no haber comenzado con Uppercase por convención.



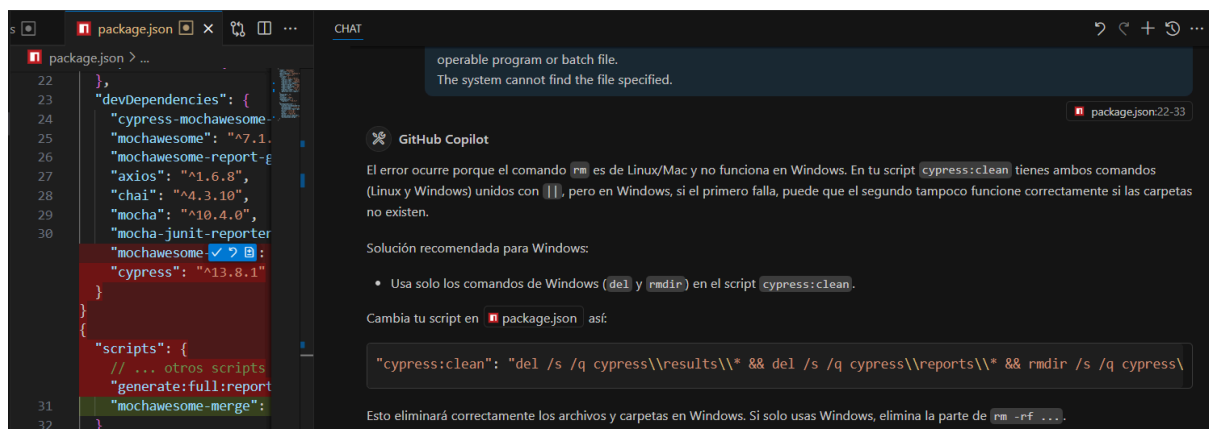
Chequeo las correcciones que hizo Copilot.



Realizo una nueva iteración de corrección de los archivos en Page Object.



No se ejecutaba el comando de reportes con Mochawesome. Use Copilot para conocer la causa raíz y corregir package.json.



Por último. En el texto de este documento, he solicitado correcciones gramaticales y de estilo a Gemini.