**Conclusiones del Proyecto E2E**

El desarrollo e implementación del proyecto de pruebas **End-to-End (E2E)** permitió validar de forma integral el correcto funcionamiento del sistema desde la perspectiva del usuario final, garantizando la interacción coherente entre los diferentes módulos de la aplicación.

Durante el proceso se aplicaron principios de diseño y buenas prácticas que fortalecieron la arquitectura, la mantenibilidad y la escalabilidad del código. A continuación, se detallan las principales conclusiones:

**✅ Aplicación de los Principios SOLID**

La adopción de los **principios SOLID** resultó fundamental para mantener un código **modular, flexible y fácil de mantener**:

* **S (Single Responsibility):** Cada clase y componente fue diseñado con una única responsabilidad, evitando dependencias innecesarias y facilitando su reutilización.
* **O (Open/Closed):** El código se mantuvo abierto a extensiones pero cerrado a modificaciones, permitiendo agregar nuevos casos de prueba o flujos sin alterar la lógica existente.
* **L (Liskov Substitution):** Se garantizó que las clases derivadas pudieran sustituir a sus clases base sin afectar el comportamiento del sistema, reforzando la consistencia de las pruebas.
* **I (Interface Segregation):** Se crearon interfaces específicas y ligeras, evitando la sobrecarga de métodos innecesarios.
* **D (Dependency Inversion):** Se promovió la inversión de dependencias para desacoplar los módulos de alto y bajo nivel, facilitando la inyección de datos y la simulación de componentes externos.

**Ventajas obtenidas:**

* Reducción de la complejidad del código.
* Mayor facilidad para realizar mantenimiento y refactorización.
* Incremento en la reutilización de componentes de prueba.
* Mejor escalabilidad ante nuevos escenarios E2E.

**📊 Uso de Datos en Archivos CSV**

El uso de **archivos CSV** como fuente de datos de prueba proporcionó una manera práctica, flexible y eficiente de manejar grandes volúmenes de información.

**Ventajas principales:**

* Permite la **parametrización de pruebas**, evitando la codificación estática de datos.
* Facilita la **actualización de datos** sin modificar el código fuente.
* Mejora la **legibilidad y organización** de los casos de prueba.
* Compatible con la **automatización y ejecución masiva** de escenarios E2E.

Este enfoque ayudó a mantener los datos separados de la lógica de negocio, fortaleciendo la limpieza del proyecto y reduciendo errores por manipulación manual.

**🧱 Programación Orientada a Objetos (POO)**

La **programación orientada a objetos** fue la base estructural del proyecto, permitiendo organizar las pruebas E2E mediante **clases, herencia, encapsulamiento y polimorfismo**.

**Beneficios obtenidos:**

* **Modularidad:** Cada flujo de prueba se encapsuló en clases independientes, facilitando su mantenimiento y comprensión.
* **Reutilización:** Las clases base y componentes comunes (por ejemplo, páginas, controladores o helpers) fueron reutilizados en múltiples pruebas.
* **Extensibilidad:** Fue posible ampliar funcionalidades sin alterar el código ya probado.
* **Abstracción:** Se logró separar la lógica de automatización del detalle técnico, simplificando el desarrollo de nuevos casos.

En conjunto, la aplicación de la POO fortaleció la estructura del framework E2E, asegurando una arquitectura sólida, escalable y de fácil evolución.

**🚀 Conclusión General**

La integración de **principios SOLID**, **POO** y **manejo de datos mediante CSV** dentro del proyecto de pruebas **E2E** demostró ser una combinación altamente efectiva para construir soluciones **robustas, limpias y mantenibles**.  
Estas prácticas no solo mejoraron la calidad técnica del código, sino también la **eficiencia en la ejecución de pruebas**, la **facilidad de mantenimiento** y la **capacidad de adaptación** ante cambios futuros en la aplicación.

En síntesis, este enfoque permitió asegurar la calidad del producto desde una perspectiva técnica y funcional, alineando la automatización con los estándares de desarrollo profesional.

**En cuanto a la página se encontró siguiente**

Encontramos las siguientes ventajas:

**Ventajas**

1. **Uso de una plataforma robusta (OpenCart)**

OpenCart es un CMS / tienda online bien conocido y maduro, con comunidad, extensiones y soporte.

Permite gestionar productos, categorías, métodos de pago, diseño, plugins, etc., de forma estándar.

1. **Diseño estructurado y navegación clara**

Las categorías están bien visibles (“Desktops”, “Laptops & Notebooks”, “Components”, etc.).

En la página principal se exhiben productos “Featured” lo cual ayuda a visibilidad.

Menú con enlaces estratégicos (“About Us”, “Contact Us”, “Privacy Policy”, etc.).

1. **Internacionalización / manejo de monedas**

Tiene selector de moneda (Euro, Libra esterlina, Dólar) — útil para usuarios de diferentes regiones.

Esto facilita que compradores de distintos países vean precios en su moneda local.

1. **Información legal y de servicio visible**

Se incluyen enlaces como “Privacy Policy”, “Terms & Conditions”, “Contact Us”, “Returns” — mejora confianza.

**Desventajas**

**Experiencia de usuario limitada en algunos flujos**

* Algunas funciones comunes de tiendas modernas (zoom sobre imágenes, filtros avanzados, recomendaciones personalizadas) podrían no estar presentes o estar poco desarrolladas.
* Si muchos productos tienen stock cero o precios genéricos, el usuario puede no tener incentivos reales para navegar.