Sistema de Gestión de Inventario para Pequeños Negocios

**Funcionalidad:** Altas, bajas y movimientos de productos, reportes de stock.  
**Ideal para practicar:**

* Pruebas de regresión y automatización
* Diseño centrado en el usuario (UX)
* Integración con APIs (por ejemplo, para códigos de barras)
* Control de versiones y releases

¡Perfecto! Elegir un **sistema de gestión de inventario** es una excelente decisión para enfocarte en diseño de pruebas, porque tiene múltiples módulos funcionales, validaciones importantes, flujo de datos constante y muchos escenarios posibles que puedes testear.

Te propongo un enfoque modular para que puedas avanzar de forma progresiva, desarrollando y probando cada parte:

**Módulos principales del sistema**

1. **Gestión de productos**

* Alta, baja, modificación
* Categorías, stock mínimo, precios

2. **Entradas y salidas de inventario**

* Registro de movimientos por tipo: entrada (compra), salida (venta, merma)
* Actualización automática de stock

3. **Historial y reportes**

* Historial por producto y por usuario
* Reportes de stock bajo, movimientos recientes

4. **Usuarios y roles**

* Admin, vendedor, auditor (solo lectura)
* Control de accesos

5. **Dashboard**

* Visualización de KPIs: total productos, alertas, ventas, etc.
* **🧪 Enfoque en pruebas**

Podrías aplicar distintos tipos de pruebas en cada etapa del desarrollo:

|  |  |  |
| --- | --- | --- |
| **Fase** | **Tipo de prueba** | **Ejemplo práctico** |
| Análisis | Casos de prueba | Basados en requerimientos funcionales y no funcionales |
| Diseño | Matriz de trazabilidad | Casos vs. requerimientos |
| Desarrollo | Pruebas unitarias | Validar la lógica de actualización de stock |
| Integración | Pruebas de API | Asegurar que POST/PUT/GET funcionen correctamente |
| Sistema | Pruebas E2E | Flujo completo desde alta de producto hasta movimiento |
| Aceptación | Pruebas basadas en roles | Asegurar que un auditor no pueda editar productos |
| Automatización | UI Testing o API Testing | Usar Cypress, Postman, o Playwright |

**⚙️ ¿Tecnologías sugeridas?**

Si vas a construirlo tú, te recomiendo algo así (aunque lo podemos adaptar a tus gustos o experiencia):

* **Frontend:** React o Vue (opcionalmente con TypeScript)
* **Backend:** Node.js con Express o Spring Boot
* **Base de datos:** PostgreSQL o MongoDB
* **Pruebas:**
  + Unitarias: Jest, Mocha
  + API: Postman + Newman o Supertest
  + E2E: Cypress o Playwright
  + Trazabilidad: Xray o Zephyr (si usas Jira)

**Artefactos de Pruebas**

***1. Casos de pruebas***

A.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Listo. Ya tienes el archivo con un conjunto inicial de **casos de prueba** para el sistema de gestión de inventario. Incluye pruebas funcionales, de validación y de roles, con diferentes prioridades y escenarios (incluyendo negativos).

Si deseas más casos para otros módulos (como usuarios, reportes avanzados, exportación de datos o notificaciones), o si prefieres una versión automatizada con herramientas específicas, ¡te los puedo generar también! ​

B.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Aquí tienes el documento con los **casos de prueba adicionales**, cubriendo los módulos de **reportes, usuarios y roles, y dashboard**. Incluyen validaciones funcionales, de seguridad y control de acceso, lo cual es ideal para diseñar pruebas completas y robustas.

Si quieres, podemos también armar una **matriz de trazabilidad**, un plan de pruebas o incluso **automatizar** algunos de estos casos. ¿Te gustaría avanzar con eso?

***2. Seguimiento de errores***

Aquí tienes una plantilla de **seguimiento de errores** en formato Excel, lista para que documentes los bugs encontrados durante la ejecución de los casos de prueba. Puedes usar el campo "Caso de Prueba Relacionado" para vincular directamente con los ID como TC001, TC005, etc.

Ya está listo el documento con ejemplos de errores agregados:

Incluye errores reales que podrías encontrar durante pruebas funcionales, de validación y de seguridad. Si necesitas generar reportes automáticos a partir de este documento o integrarlo con herramientas como Jira o TestRail, también puedo ayudarte. ¿Quieres dar ese siguiente paso?

**Requerimientos**

***1. Sistema de gestión de inventario***

***2. Chat GPT***

***3. Requerimientos funcionales y no funcionales***

**📘 Requerimientos Funcionales (RF)**

|  |  |
| --- | --- |
| **ID** | **Requerimiento Funcional** |
| RF01 | El sistema debe permitir a los usuarios con rol "admin" agregar nuevos productos indicando nombre, categoría, precio y stock inicial. |
| RF02 | El sistema debe validar que todos los campos obligatorios estén completos al registrar o editar un producto. |
| RF03 | El sistema debe permitir la modificación de productos existentes. |
| RF04 | El sistema debe permitir la eliminación de productos que no tengan movimientos registrados. |
| RF05 | El sistema debe registrar entradas de inventario y actualizar automáticamente el stock del producto correspondiente. |
| RF06 | El sistema debe registrar salidas de inventario y disminuir el stock del producto. |
| RF07 | El sistema debe evitar salidas de productos si la cantidad supera el stock disponible. |
| RF08 | El sistema debe generar reportes de inventario con filtros por categoría, fecha y stock mínimo. |
| RF09 | El sistema debe permitir exportar los reportes en formatos PDF y Excel. |
| RF10 | El sistema debe gestionar diferentes tipos de usuario (admin, vendedor, auditor) con permisos diferenciados. |
| RF11 | El sistema debe bloquear el acceso a funciones administrativas a usuarios no autorizados. |
| RF12 | El sistema debe mostrar un dashboard con indicadores clave: stock total, productos bajo mínimo, movimientos recientes. |
| RF13 | El sistema debe actualizar los KPIs del dashboard automáticamente tras un movimiento de inventario. |
| RF14 | El sistema debe permitir inhabilitar temporalmente a usuarios desde la administración. |
| RF15 | El sistema debe registrar un historial de movimientos de inventario por producto y por usuario. |

**⚙️ Requerimientos No Funcionales (RNF)**

|  |  |
| --- | --- |
| **ID** | **Requerimiento No Funcional** |
| RNF01 | El sistema debe estar disponible al menos el 99% del tiempo durante el horario laboral. |
| RNF02 | El tiempo de respuesta de cualquier operación (guardar, consultar, generar reporte) no debe superar los 2 segundos. |
| RNF03 | El sistema debe proteger los datos de los usuarios mediante autenticación segura y cifrado de contraseñas. |
| RNF04 | Las interfaces del sistema deben ser intuitivas y accesibles desde navegadores modernos (Chrome, Firefox, Edge). |
| RNF05 | El sistema debe ser responsivo y funcionar correctamente en dispositivos móviles y tablets. |
| RNF06 | Los errores del sistema deben registrarse en un log accesible solo para administradores del sistema. |
| RNF07 | El sistema debe permitir escalar fácilmente en caso de que aumente el número de productos o usuarios. |
| RNF08 | El sistema debe integrarse fácilmente con herramientas de prueba automatizada y/o APIs externas. |
| RNF09 | El sistema debe permitir restaurar datos desde una copia de seguridad en caso de fallo. |
| RNF10 | El diseño debe cumplir con criterios básicos de accesibilidad (contrastes, navegación con teclado, etiquetas claras). |

**3. Casos de prueba de aceptación**

Aquí tienes el archivo con los **casos de prueba de aceptación**, alineados con los requerimientos funcionales y no funcionales, y vinculados (cuando corresponde) con los casos de prueba previamente definidos.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Pruebas de aceptación**

***1. Chat GPT***

***2. Invalidación de requerimientos:***

**Requerimientos Funcionales (RF)**

|  |  |
| --- | --- |
| **ID** | **Requerimiento Funcional** |
| RF01 | El sistema debe permitir a los usuarios con rol "admin" agregar nuevos productos indicando nombre, categoría, precio y stock inicial. |
| RF02 | El sistema debe validar que todos los campos obligatorios estén completos al registrar o editar un producto. |
| RF02.1 | El sistema debe permitir el registro o actualización de un producto, aun y cuando sus campos obligatorios no estén completos. |
| RF03 | El sistema debe permitir la modificación de productos existentes. |
| RF04 | El sistema debe permitir la eliminación de productos que no tengan movimientos registrados. |
| RF04.1 | El sistema no debe permitir la eliminación de ningún producto. |
| RF05 | El sistema debe registrar entradas de inventario y actualizar automáticamente el stock del producto correspondiente. |
| RF06 | El sistema debe registrar salidas de inventario y disminuir el stock del producto. |
| RF07 | El sistema debe evitar salidas de productos si la cantidad supera el stock disponible. |
| RF07.1 | El sistema debe permitir salidas de productos aun y cuando la cantidad supere el stock disponible. |
| RF08 | El sistema debe generar reportes de inventario con filtros por categoría, fecha y stock mínimo. |
| RF09 | El sistema debe permitir exportar los reportes en formatos PDF y Excel. |
| RF10 | El sistema debe gestionar diferentes tipos de usuario (admin, vendedor, auditor) con permisos diferenciados. |
| RF11 | El sistema debe bloquear el acceso a funciones administrativas a usuarios no autorizados. |
| RF11.1 | El sistema debe permitir el acceso a todas las funciones administrativas, independientemente de si el usuario está autorizado o no. |
| RF12 | El sistema debe mostrar un dashboard con indicadores clave: stock total, productos bajo mínimo, movimientos recientes. |
| RF13 | El sistema debe actualizar los KPIs del dashboard automáticamente tras un movimiento de inventario. |
| RF14 | El sistema debe permitir inhabilitar temporalmente a usuarios desde la administración. |
| RF15 | El sistema debe registrar un historial de movimientos de inventario por producto y por usuario. |

**Requerimientos No Funcionales (RNF)**

|  |  |
| --- | --- |
| **ID** | **Requerimiento No Funcional** |
| RNF01 | El sistema debe estar disponible al menos el 99% del tiempo durante el horario laboral. |
| RNF02 | El tiempo de respuesta de cualquier operación (guardar, consultar, generar reporte) no debe superar los 2 segundos. |
| RNF02.1 | El tiempo de respuesta de cualquier operación (guardar, consultar, generar reporte), no tiene ningún límite, el proceso puede esperar el tiempo que sea necesario. |
| RNF03 | El sistema debe proteger los datos de los usuarios mediante autenticación segura y cifrado de contraseñas. |
| RNF04 | Las interfaces del sistema deben ser intuitivas y accesibles desde navegadores modernos (Chrome, Firefox, Edge). |
| RNF05 | El sistema debe ser responsivo y funcionar correctamente en dispositivos móviles y tablets. |
| RNF06 | Los errores del sistema deben registrarse en un log accesible solo para administradores del sistema. |
| RNF06.1 | Los errores del sistema deben registrarse en un log accesible para todos los usuarios del sistema. |
| RNF07 | El sistema debe permitir escalar fácilmente en caso de que aumente el número de productos o usuarios. |
| RNF08 | El sistema debe integrarse fácilmente con herramientas de prueba automatizada y/o APIs externas. |
| RNF09 | El sistema debe permitir restaurar datos desde una copia de seguridad en caso de fallo. |
| RNF09.1 | El sistema no debe permitir ningún tipo de restauración de datos, con el fin de proteger la integridad. |
| RNF10 | El diseño debe cumplir con criterios básicos de accesibilidad (contrastes, navegación con teclado, etiquetas claras). |

***3. Errores en los casos de prueba***

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

***4. Seguimiento de errores***

Texto

El contenido generado por IA puede ser incorrecto.

**Diseño del sistema**

***1. Casos de uso***

A continuación, te presento una lista de casos de uso derivados directamente de los requerimientos funcionales y no funcionales del sistema de gestión de inventario. Incluyen el objetivo, actores involucrados y una breve descripción.

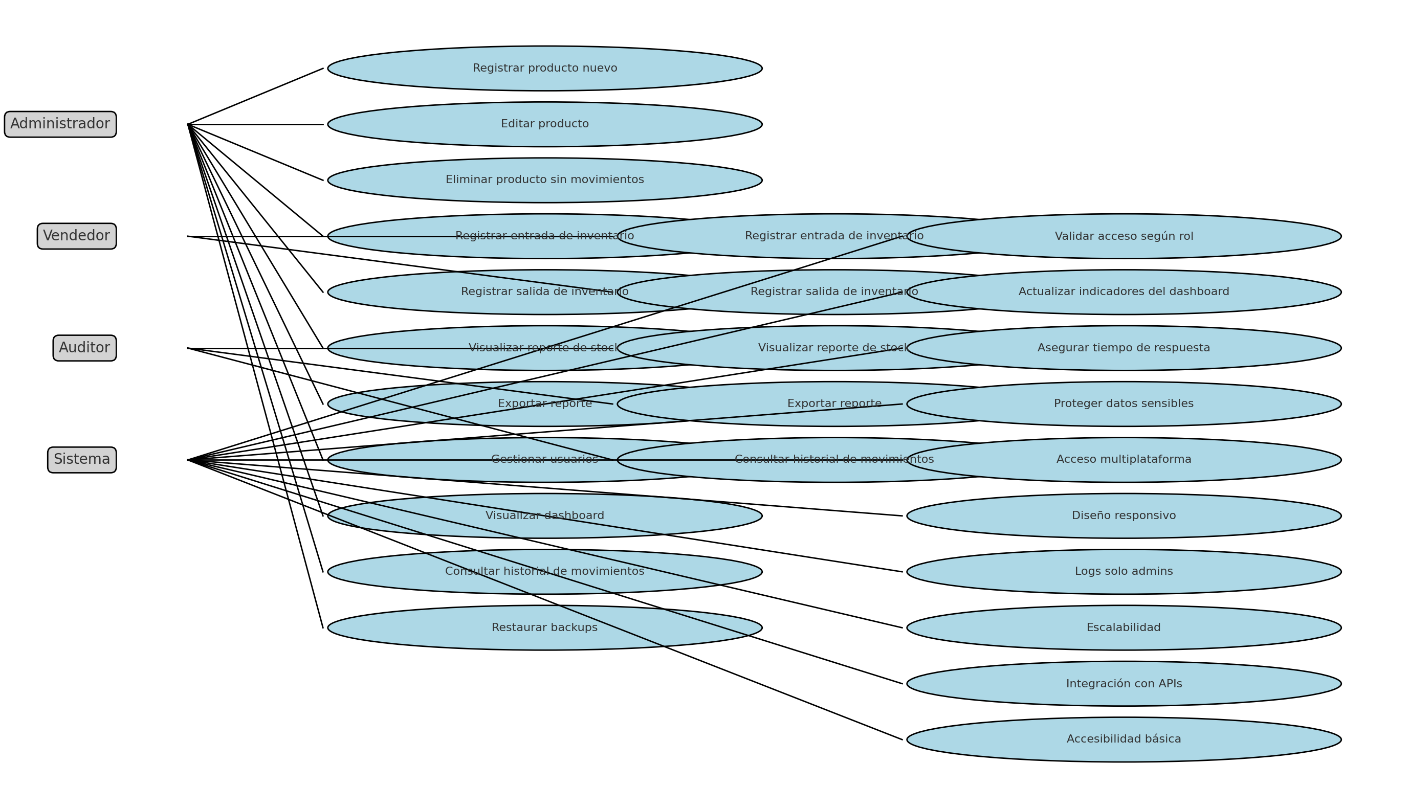
**📘 Casos de Uso del Sistema de Inventario**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Nombre del Caso de Uso** | **Actor Principal** | **Objetivo** | **Requerimientos Relacionados** |
| CU01 | Registrar producto nuevo | Administrador | Agregar un producto al inventario con sus datos completos. | RF01, RF02 |
| CU02 | Editar producto existente | Administrador | Modificar los datos de un producto previamente creado. | RF03 |
| CU03 | Eliminar producto sin movimientos | Administrador | Eliminar un producto que no tenga historial de movimientos. | RF04 |
| CU04 | Registrar entrada de inventario | Administrador / Vendedor | Aumentar el stock de un producto registrando una entrada. | RF05 |
| CU05 | Registrar salida de inventario | Administrador / Vendedor | Disminuir el stock de un producto registrando una salida. | RF06, RF07 |
| CU06 | Visualizar reporte de stock | Administrador / Auditor | Generar y consultar reportes del inventario filtrados por categoría, fecha o stock. | RF08 |
| CU07 | Exportar reporte | Administrador / Auditor | Exportar el reporte generado en formatos PDF o Excel. | RF09 |
| CU08 | Gestionar usuarios del sistema | Administrador | Crear, editar, inhabilitar usuarios y asignar roles. | RF10, RF14 |
| CU09 | Validar acceso según rol | Sistema | Controlar que los usuarios accedan solo a las funciones correspondientes a su rol. | RF11 |
| CU10 | Visualizar dashboard | Administrador | Consultar métricas clave del inventario en una vista general. | RF12 |
| CU11 | Actualizar indicadores del dashboard | Sistema | Refrescar automáticamente los KPIs tras registrar entradas o salidas. | RF13 |
| CU12 | Consultar historial de movimientos | Administrador / Auditor | Ver los registros de entradas y salidas por producto y por usuario. | RF15 |

**⚙️ Casos de Uso Técnicos / No Funcionales**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Nombre del Caso de Uso** | **Actor Principal** | **Objetivo** | **Requerimientos Relacionados** |
| CU13 | Asegurar tiempo de respuesta | Sistema | Responder a las acciones del usuario en menos de 2 segundos. | RNF02 |
| CU14 | Proteger datos sensibles | Sistema | Cifrar contraseñas y proteger datos de los usuarios. | RNF03 |
| CU15 | Permitir acceso multiplataforma | Usuario | Acceder al sistema desde navegadores modernos. | RNF04 |
| CU16 | Visualizar correctamente en móviles | Usuario | Ver el sistema de forma adaptada en dispositivos móviles. | RNF05 |
| CU17 | Registrar errores en logs privados | Sistema | Registrar errores del sistema en logs accesibles solo a administradores. | RNF06 |
| CU18 | Escalar sistema | Administrador | Asegurar que el sistema soporte crecimiento de usuarios/productos. | RNF07 |
| CU19 | Integrar con APIs externas | Administrador / DevOps | Conectar el sistema con herramientas externas. | RNF08 |
| CU20 | Restaurar información desde respaldo | Administrador | Recuperar datos ante un fallo o pérdida. | RNF09 |
| CU21 | Garantizar accesibilidad básica | Usuario | Usar el sistema con ayudas visuales o teclado. | RNF10 |

**Diagramas de casos de uso**:



***2. Interfaces de usuario:***

Basándonos en los **módulos del sistema**, así como en los **requerimientos funcionales y no funcionales**, aquí tienes un listado de las **interfaces de usuario (pantallas)** recomendadas para el sistema de gestión de inventario. Cada una está pensada para cumplir con la lógica funcional, la experiencia del usuario y los roles definidos (Administrador, Vendedor, Auditor).

**🧩 Módulos del Sistema**

1. **Gestión de productos**
2. **Movimientos de inventario (entradas y salidas)**
3. **Reportes**
4. **Dashboard (KPIs)**
5. **Gestión de usuarios y roles**
6. **Historial de movimientos**
7. **Seguridad, autenticación y accesibilidad**
8. **Administración técnica (logs, backups)**

**🖥️ Interfaces de Usuario por Módulo**

**1. Pantalla de Login / Autenticación**

* Campos: usuario, contraseña
* Funcionalidad: inicio de sesión, recuperación de contraseña
* Requerimientos: RNF03 (seguridad), RNF10 (accesibilidad)

**2. Pantalla Principal / Home (según rol)**

* Redirección basada en el tipo de usuario
* Módulos visibles según permisos (RF10, RF11)

**3. Gestión de Productos**

* Lista de productos (tabla con filtros y paginación)
* Botón "Agregar producto"
* Formulario de edición
* Validaciones de campos obligatorios
* Acciones: Ver, Editar, Eliminar (RF01–RF04)

**4. Registro de Movimiento de Inventario**

* Pantalla dividida en:
  + Entradas
  + Salidas
* Selección de producto, cantidad, tipo de movimiento
* Control de stock negativo (RF05–RF07)

**5. Dashboard (KPIs)**

* Indicadores:
  + Stock total
  + Productos bajo mínimo
  + Últimos movimientos
* Actualización automática o manual (RF12, RF13)

**6. Reportes**

* Filtros: categoría, rango de fechas, stock mínimo
* Tabla de resultados
* Botones de exportación: PDF y Excel (RF08, RF09)

**7. Historial de movimientos**

* Búsqueda por producto o usuario
* Filtros por fecha
* Tabla detallada con tipo de movimiento, cantidad, responsable (RF15)

**8. Gestión de Usuarios**

* Lista de usuarios y roles
* Formulario de creación/edición
* Activar/desactivar usuario (RF14)

**9. Administración Técnica**

* Panel solo visible para administradores
* Acceso a logs del sistema (RNF06)
* Opción para restaurar respaldos (RNF09)

**10. Interfaz de Configuración / Preferencias**

* Cambiar contraseña
* Preferencias de notificaciones
* Modo oscuro/claro (UX)

**11. Diseño Responsivo y Accesible**

* Navegación con teclado
* Etiquetas claras
* Contrastes adecuados (RNF05, RNF10)

**📌 Extras:**

* **Barra lateral de navegación con íconos**
* **Notificaciones (éxito, error, advertencias)**
* **Modal de confirmación para acciones sensibles (eliminar, restaurar)**

***3. Casos de prueba de sistema****:*

Aquí tienes el archivo con los **casos de prueba de sistema** organizados por módulo, objetivos, requerimientos relacionados y tipo de usuario:

***4. Documentación en Excel:***

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Pruebas del sistema**

**1. Altera el documento de casos de uso para producir errores:**

Una captura de pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.

**2. Chat GPT.**

**3. Aplicación de casos de prueba de sistema:**

Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.

**4. Registro de los errores:**

Tabla

El contenido generado por IA puede ser incorrecto.

**Diseño de arquitectura**

**1.1 Diagrama de infraestructura**

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

**1.2 Diagrama de componentes**

Interfaz de usuario gráfica, Diagrama, Aplicación

El contenido generado por IA puede ser incorrecto.

**1.3 Diagrama de base de datos**

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

**2. Casos de prueba de integración:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Módulos Involucrados | Descripción | Precondiciones | Pasos | Resultado Esperado |
| CPI-01 | Gestión de Productos ↔ Inventario | Verificar que al agregar un producto nuevo se actualice correctamente el stock inicial en el módulo de inventario. | Usuario admin autenticado | 1. Ingresar al módulo Gestión de Productos. 2. Registrar un producto nuevo con stock inicial. 3. Consultar stock en módulo de Inventario. | El producto debe aparecer en inventario con el stock correcto. |
| CPI-02 | Inventario ↔ Dashboard | Comprobar actualización automática de KPIs en Dashboard después de realizar entradas/salidas. | Existencia de productos registrados. | 1. Realizar entrada/salida en módulo Inventario. 2. Visualizar Dashboard. | Los KPIs del dashboard reflejan correctamente las entradas y salidas realizadas. |
| CPI-03 | Usuarios y Roles ↔ Gestión de Productos | Validar restricción de acceso a gestión de productos según roles. | Usuarios creados con diferentes roles (admin, vendedor, auditor). | 1. Intentar agregar un producto con usuario vendedor/auditor. 2. Intentar agregar producto con usuario admin. | Solo usuario admin puede agregar productos nuevos. |
| CPI-04 | Inventario ↔ Historial y Auditoría | Verificar registro automático en historial después de movimiento en inventario. | Existencia previa de productos. | 1. Realizar una salida o entrada de producto. 2. Revisar módulo Historial y Auditoría. | El movimiento realizado queda registrado con detalles (usuario, fecha, cantidad). |
| CPI-05 | Inventario ↔ Reportes | Comprobar generación correcta de reportes después de movimientos en inventario. | Productos con movimientos realizados. | 1. Realizar diversos movimientos en Inventario. 2. Generar reporte por categoría y fechas. 3. Exportar reporte en PDF y Excel. | Reporte generado coincide con movimientos registrados y exporta correctamente en formatos PDF y Excel. |
| CPI-06 | Gestión de Productos ↔ LogsErrores | Probar registro de errores en log al intentar eliminar un producto con movimientos. | Producto con movimientos previos registrados. | 1. Intentar eliminar producto con movimientos previos. 2. Verificar módulo LogsErrores. | Se impide eliminación y se registra error en logs adecuadamente. |
| CPI-07 | Usuarios ↔ Backups | Validar creación y restauración de backups realizados por usuarios administradores. | Existencia de datos en la base de datos. | 1. Crear backup del sistema. 2. Simular pérdida de datos. 3. Restaurar backup. | Backup se crea correctamente, y la restauración recupera todos los datos correctamente. |
| CPI-08 | Autenticación ↔ Usuarios | Verificar autenticación segura y manejo correcto de intentos fallidos. | Usuario existente en la base de datos. | 1. Intentar autenticación con contraseña incorrecta repetidamente. 2. Iniciar sesión con credenciales correctas. | Acceso bloqueado temporalmente tras varios intentos fallidos y posteriormente permitido con credenciales correctas. |

**Pruebas de integración**

**1. Casos erróneos de pruebas de integración:**

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**2. ChatGPT**

**3. Aplicación de casos de prueba a la arquitectura del sistema:**

He revisado la estructura del archivo que proporcionaste. Los casos están definidos en varias filas, algunas con información fragmentada o con valores faltantes.

Voy a realizar lo siguiente para analizar claramente los casos proporcionados:

1. Organizar y limpiar el contenido del archivo para reconstruir claramente cada caso de prueba.
2. Analizar y verificar cada caso contra los diagramas definidos (Infraestructura, Componentes, Base de datos).
3. Identificar errores, en caso de existir, con base en las incoherencias con los diagramas.
4. Recopilar errores en una tabla con formato solicitado.

**4. Registro de errores:**

Imagen que contiene Gráfico

El contenido generado por IA puede ser incorrecto.

**Diseño Modular**

**1. Flujos y Entidades principales.**

**Diagramas de clases:**

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Diagrama de flujo:**

Pantalla de un video juego

El contenido generado por IA puede ser incorrecto.

**2. Casos de pruebas sobre los diagramas:**

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**3. Registro de los casos de pruebas:**

Texto

El contenido generado por IA puede ser incorrecto.

**Pruebas Unitarias.**

**1. ChatGPT**

**2. Código funcional:**

Ahora que sé que tienes **VS Code** y **Playwright**, y quieres algo **muy simple para probar localmente la clase Inventario**, te voy a ayudar paso a paso a lograrlo **sin complicaciones.**

**🎯 ¿Qué vamos a lograr?**

Crear un **archivo .py simple** que:

* Tenga la clase Inventario
* Te permita **agregar stock**, **reducir stock** y **ver resultados**
* Se pueda ejecutar con un clic en VS Code
* No necesita instalación extra (solo Python)

**✅ Paso 1: Crea tu archivo Python en VS Code**

1. En VS Code, crea un archivo nuevo llamado:

inventario.py

1. Copia y pega este código:

class Inventario:

def \_\_init\_\_(self, nombre\_producto, cantidad\_inicial):

self.nombre\_producto = nombre\_producto

self.cantidad = cantidad\_inicial

def mostrar\_stock(self):

print(f"📦 {self.nombre\_producto}: {self.cantidad} unidades en inventario.")

def agregar\_stock(self, cantidad):

self.cantidad += cantidad

print(f"✅ Se agregaron {cantidad} unidades.")

def reducir\_stock(self, cantidad):

if cantidad > self.cantidad:

print("❌ No hay suficiente stock.")

else:

self.cantidad -= cantidad

print(f"🗑️ Se retiraron {cantidad} unidades.")

# 🧪 Prueba básica

inventario = Inventario("Mouse inalámbrico", 10)

inventario.mostrar\_stock()

inventario.agregar\_stock(5)

inventario.mostrar\_stock()

inventario.reducir\_stock(3)

inventario.mostrar\_stock()

inventario.reducir\_stock(20) # prueba con más de lo disponible

inventario.mostrar\_stock()

**🧪 Paso 2: Ejecutar la prueba**

1. Abre la terminal en VS Code (en el menú: **Terminal > Nuevo terminal**)
2. Asegúrate de estar en la carpeta donde guardaste inventario.py
3. Ejecuta el script con:

python inventario.py

Texto

El contenido generado por IA puede ser incorrecto.

**3. Prueba con Playwright:**

**✅ PASO 1: Crea el archivo HTML (interfaz visual)**

1. En la misma carpeta del proyecto, crea un archivo nuevo llamado:

inventario.html

1. Pega este contenido básico:

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Inventario</title>

</head>

<body>

<h1>📦 Inventario: Mouse Inalámbrico</h1>

<p id="stock">Stock actual: <span id="cantidad">10</span> unidades</p>

<button onclick="agregarStock()">Agregar 5</button>

<button onclick="reducirStock()">Quitar 3</button>

<script>

let cantidad = 10;

function actualizarVista() {

document.getElementById("cantidad").textContent = cantidad;

}

function agregarStock() {

cantidad += 5;

actualizarVista();

}

function reducirStock() {

if (cantidad >= 3) {

cantidad -= 3;

} else {

alert("❌ No hay suficiente stock.");

}

actualizarVista();

}

</script>

</body>

</html>

**✅ PASO 2: Lanza un servidor local simple**

Playwright necesita una URL para abrir el HTML, así que haremos un servidorcito.

1. En la terminal de VS Code, escribe:

python -m http.server

Esto abre un servidor en:  
📍 http://localhost:8000

Abre eso en el navegador y verás tu interfaz de inventario ✨

**✅ PASO 3: Crear el test automatizado con Playwright**

1. En tu proyecto, crea un nuevo archivo:

test\_inventario.py

1. Pega este código Playwright:

from playwright.sync\_api import sync\_playwright

import time

def test\_inventario():

with sync\_playwright() as p:

navegador = p.chromium.launch(headless=False, slow\_mo=500) # <-- aquí está la magia

pagina = navegador.new\_page()

pagina.goto("http://localhost:8000/inventario.html")

print("📋 Stock inicial:", pagina.inner\_text("#cantidad"))

pagina.click("text=Agregar 5")

print("➕ Luego de agregar:", pagina.inner\_text("#cantidad"))

pagina.click("text=Quitar 3")

print("➖ Luego de quitar:", pagina.inner\_text("#cantidad"))

assert pagina.inner\_text("#cantidad") == "12", "❌ El stock final no es correcto"

time.sleep(3) # Espera antes de cerrar el navegador

navegador.close()

test\_inventario()

**✅ PASO 4: Ejecutar la prueba**

Con el servidor local corriendo, abre otra terminal y ejecuta:

python test\_inventario.py

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**4. Alteraciones en el código funcional:**

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Después de esta prueba se volvió a dejar el código como estaba originalmente\*

**5. Registro de errores:**

Código final html y prueba:

***inventario.html***

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Inventario</title>

</head>

<body>

<h1>📦 Inventario: Mouse Inalámbrico</h1>

<p id="stock">Stock actual: <span id="cantidad">10</span> unidades</p>

<button onclick="agregarStock()">Agregar 1</button>

<button onclick="reducirStock()">Quitar 1</button>

<script>

let cantidad = 10;

function actualizarVista() {

document.getElementById("cantidad").textContent = cantidad;

}

function agregarStock() {

cantidad += 1;

actualizarVista();

}

function reducirStock() {

if (cantidad >= 1) {

cantidad -= 1;

} else {

alert("❌ No hay suficiente stock.");

}

actualizarVista();

}

</script>

</body>

</html>

***Test\_inventario.py***

from playwright.sync\_api import sync\_playwright

import time

from datetime import datetime

casos = [

(5, 3, 12),

(2, 4, 8),

(0, 12, 0),

(10, 10, 10),

(1, 15, "error"), # Error esperado por lógica

(4, 2, 99) # Error esperado por comparación

]

def log\_error(caso, error\_msg):

with open("errores\_playwright.txt", "a", encoding="utf-8") as f:

f.write(f"\n[{datetime.now()}] ERROR en caso {caso}: {error\_msg}\n")

def ejecutar\_prueba(agregar, quitar, esperado):

with sync\_playwright() as p:

navegador = p.chromium.launch(headless=False, slow\_mo=300)

pagina = navegador.new\_page()

pagina.goto("http://localhost:8000/inventario.html")

# Reiniciar stock a 10 desde JavaScript

pagina.evaluate("cantidad = 10; actualizarVista();")

# Agregar stock

for \_ in range(agregar):

pagina.click("text=Agregar 1")

try:

if esperado == "error":

# Esperamos una alerta al intentar quitar más stock del disponible

with pagina.expect\_event("dialog") as dialog\_info:

for \_ in range(quitar):

pagina.click("text=Quitar 1")

dialog = dialog\_info.value

dialog.dismiss()

stock\_actual = pagina.inner\_text("#cantidad")

print(f"🧪 Agregado: {agregar}, Quitado: {quitar} → Stock final: {stock\_actual} (alerta OK)")

else:

# Quitar stock

for \_ in range(quitar):

pagina.click("text=Quitar 1")

stock\_actual = pagina.inner\_text("#cantidad")

print(f"🧪 Agregado: {agregar}, Quitado: {quitar} → Stock final: {stock\_actual}")

assert int(stock\_actual) == esperado, f"Esperado {esperado}, obtenido {stock\_actual}"

except Exception as e:

log\_error((agregar, quitar, esperado), str(e))

time.sleep(1)

navegador.close()

# Ejecutar todas las pruebas

for caso in casos:

ejecutar\_prueba(\*caso)

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**Clasificación.**

**1. ChatGPT**

**2. Análisis y definición de tipos de errores:**

Perfecto, encontré un total de 27 errores, muchos de ellos relacionados con contradicciones en los requerimientos, errores de validación, funcionalidad rota, ambigüedad en casos de uso o fallas técnicas.

Voy a clasificar cada uno en estos tipos de error:

1. Validación de datos
2. Lógica de negocio
3. Funcionalidad inactiva
4. Control de acceso
5. Inconsistencia de requerimientos
6. Defecto en caso de uso
7. Ambigüedad o error en documentación
8. Falla técnica / Automatización

**3. Actualización del documento:**

Tabla

El contenido generado por IA puede ser incorrecto.

**Ejecución.**

**1. Pruebas de la aplicación web:**

***Boton Agregar***

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

El contenido generado por IA puede ser incorrecto.

***Boton Quitar***

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

El contenido generado por IA puede ser incorrecto.

**Pruebas Automatizadas.**

**1. Selenium IDE:**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**2. Generación de Código:**

Texto

El contenido generado por IA puede ser incorrecto.

**3. Migración del código a Playwright:**

Texto

El contenido generado por IA puede ser incorrecto.