Práctica 1

Ejercicios sobre Metodologías de Desarrollo en Cascada

1. Análisis de Requerimientos:

- **Ejercicio 1:** Un cliente te solicita una aplicación web para gestionar su inventario. Define los requisitos funcionales y no funcionales del sistema.

  Requisitos Funcionales:

  Gestión de Productos:

  - Crear, leer, actualizar y eliminar productos.
  - Visualización del stock actual de cada producto.
  - Gestión del precio de cada producto.

  Gestión de Proveedores:

  - Crear, leer, actualizar y eliminar proveedores.
  - Registrar y actualizar la información de contacto de los proveedores (nombre, teléfono, correo electrónico, dirección).

  Asignación de Productos a Proveedores:

  - Asociar productos con proveedores.
  - Registrar el precio del producto para cada proveedor

  Gestión de Pedidos:

- Crear, leer, actualizar y eliminar pedidos.
- Registrar los detalles del pedido, incluyendo el producto, proveedor, cantidad y precio.

Gestión de Usuarios:

- Crear, leer, actualizar y eliminar usuarios.
- Autenticación de usuarios (inicio de sesión) con nombre de usuario y contraseña.
- Recuperación de contraseña mediante correo electrónico.

Requisitos No Funcionales:

Seguridad:

- Protección de Datos: Asegurar la protección de datos mediante encriptación y otras técnicas de seguridad.
- Control de Acceso: Implementar un sistema robusto de control de acceso y permisos basado en roles.
- Auditoría: Registrar todas las acciones de los usuarios en un log de auditoría para seguimiento y análisis de seguridad.

Rendimiento:

- Escalabilidad: El sistema debe ser escalable para manejar un aumento en el número de usuarios y transacciones sin degradación del rendimiento.
- Tiempo de Respuesta: Las consultas y transacciones deben tener tiempos de respuesta rápidos, idealmente menos de 2 segundos para operaciones comunes.

Usabilidad:

- Interfaz Intuitiva: La interfaz de usuario debe ser intuitiva y fácil de usar, con navegación clara y accesible.
- Accesibilidad: El sistema debe cumplir con las normas de accesibilidad web para asegurar que pueda ser utilizado por personas con discapacidades.

Mantenibilidad:

- Documentación: El código debe estar bien documentado para facilitar el mantenimiento y las actualizaciones.

- Modularidad: La arquitectura del sistema debe ser modular para permitir la fácil integración de nuevas funcionalidades y la actualización de las existentes.

Compatibilidad:

- Multiplataforma: La aplicación debe ser compatible con los principales navegadores web (Chrome, Firefox, Safari, Edge) y dispositivos (escritorio, tablet, móvil).
- Integración: La aplicación debe permitir la integración con otros sistemas empresariales mediante APIs o servicios web.

Disponibilidad:

- Tiempo de Operación: La aplicación debe estar disponible al menos el 99.9% del tiempo, exceptuando periodos programados de mantenimiento.
- Recuperación de Desastres: Implementar un plan de recuperación ante desastres para minimizar el tiempo de inactividad en caso de fallos críticos.

- **Ejercicio 2:** Redacta un caso de uso para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

Caso de Uso: Agregar un nuevo producto

Actor Principal: Usuario (Administrador o Empleado)

Precondiciones:

1. El usuario ha iniciado sesión en la aplicación.
2. El usuario tiene los permisos necesarios para agregar un nuevo producto.

Flujo Principal:

1. El usuario selecciona la opción “Agregar Producto” en la interfaz de gestión de productos.
2. La aplicación muestra un formulario para ingresar los detalles del nuevo producto, incluyendo nombre, descripción, categoría, precio y cantidad en stock.

3. El usuario ingresa los detalles del nuevo producto en el formulario.
4. El usuario selecciona la opción “Guardar” para agregar el nuevo producto.
5. La aplicación valida los datos ingresados. Si los datos son válidos, la aplicación agrega el nuevo producto al inventario y muestra un mensaje de confirmación. Si los datos no son válidos, la aplicación muestra un mensaje de error y solicita al usuario que corrija los datos.

Postcondiciones:

1. El nuevo producto se ha agregado al inventario.
2. El usuario puede ver el nuevo producto en la lista de productos.

Flujo Alternativo:

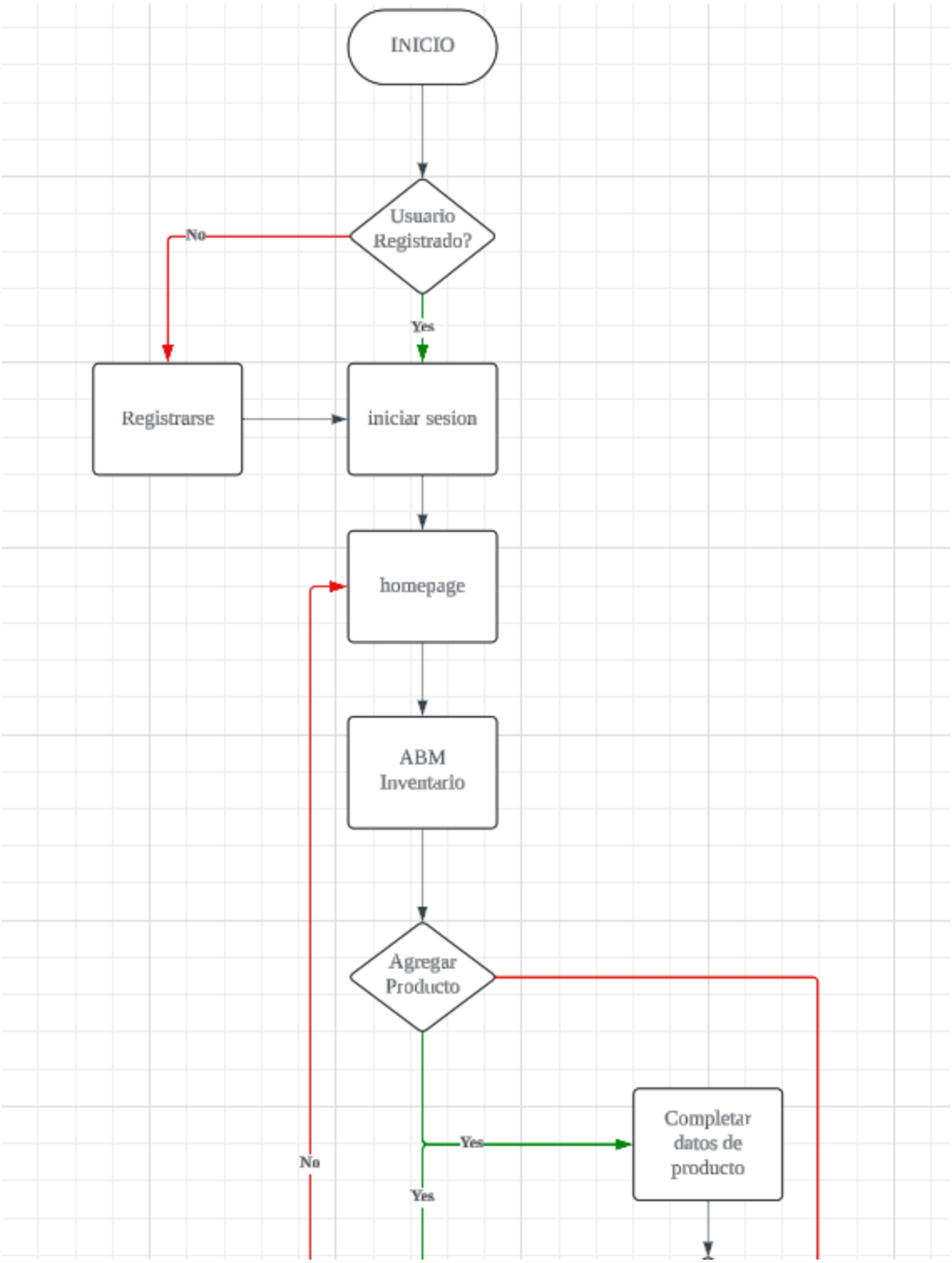
1. En el paso 4, el usuario puede seleccionar la opción “Cancelar” para abandonar la operación de agregar un nuevo producto. En este caso, la aplicación no agrega ningún producto y vuelve a la pantalla de gestión de productos.

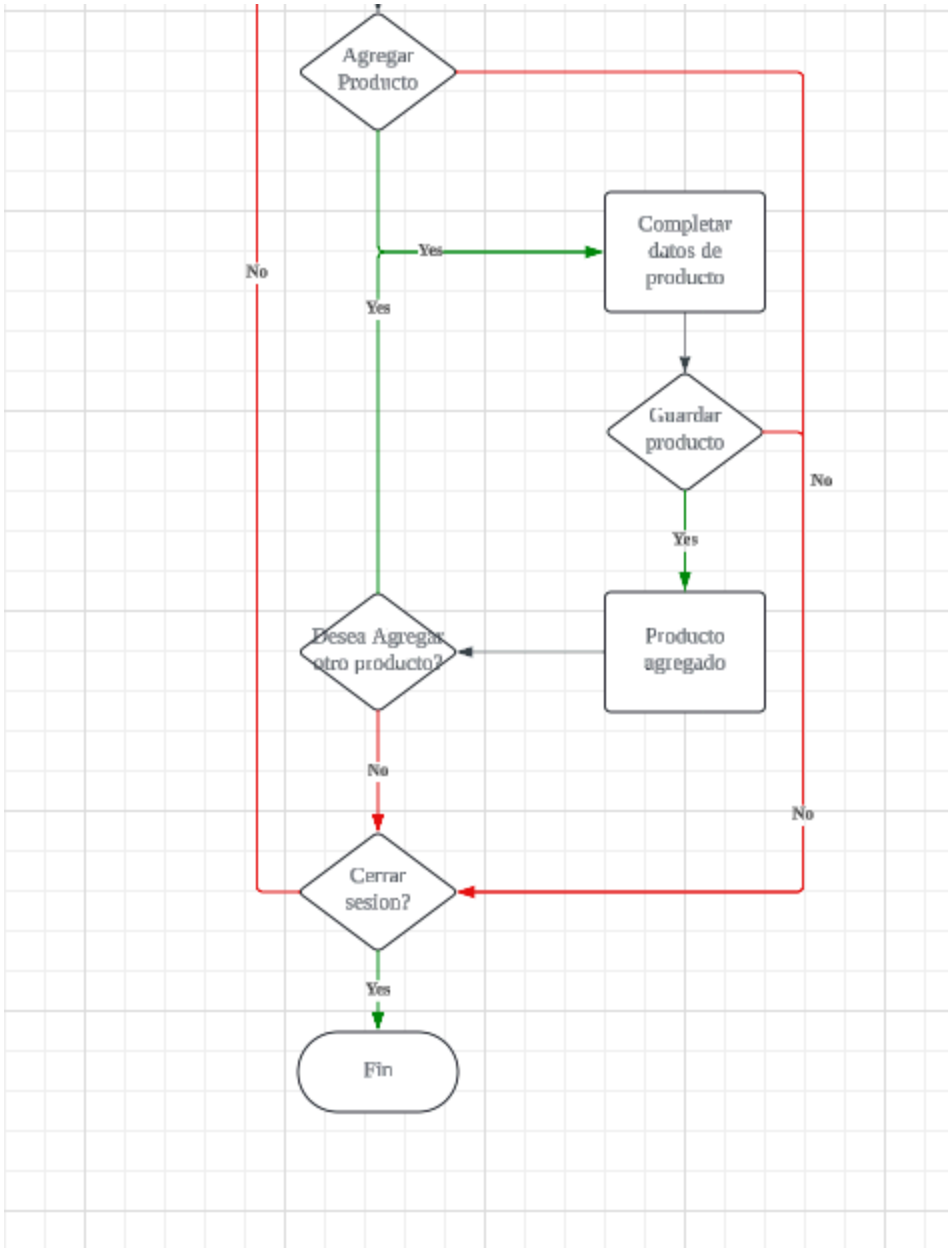
Excepciones:

1. Si el sistema no puede agregar el nuevo producto debido a un error del sistema, la aplicación muestra un mensaje de error y registra los detalles del error en el log de auditoría.

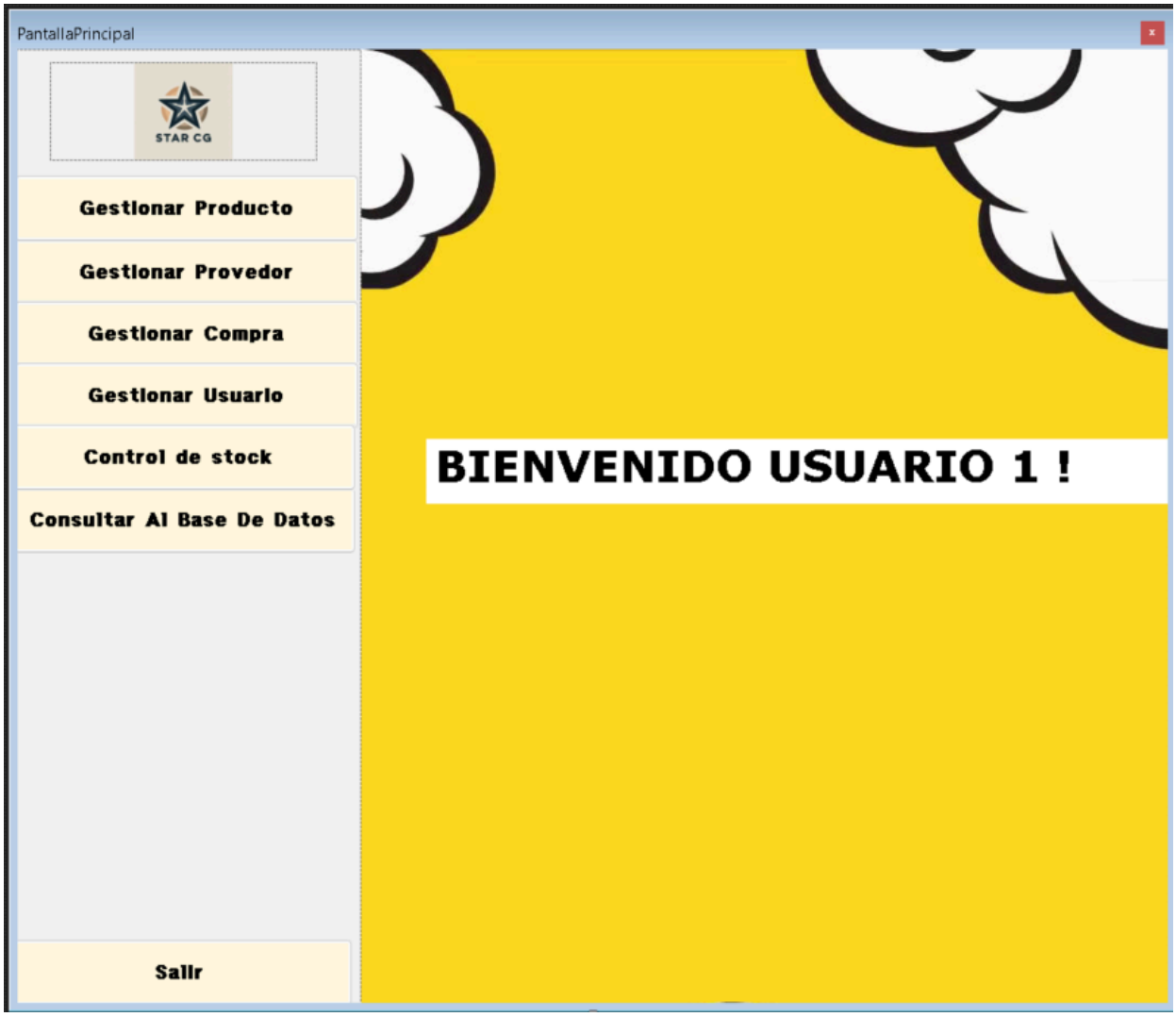
2. Diseño del Sistema:

- **Ejercicio 3:** Elabora un diagrama de flujo de datos para la aplicación web del ejercicio 1.


INICIO
Usuario Registrado?
No
Yes
Registrarse
iniciar sesion
homepage
ABM Inventario
Agregar Producto
Yes
Completar datos de producto
No
Yes


Agregar Producto
Yes
No
Completar datos de producto
Yes
Guardar producto
No
Yes
Producto agregado
Desea Agregar otro producto?
No
No
Cerrar sesion?
Yes
Fin

- **Ejercicio 4:** Diseña la interfaz de usuario para la pantalla de "Inicio" de la aplicación web del ejercicio 1.



3. Diseño del Programa:

- **Ejercicio 5:** Elige una arquitectura adecuada para la aplicación web del ejercicio 1 y justifica tu elección.

Para la aplicación web de gestión de inventario, he elegido la Arquitectura de

Microservicios.

  - Escalabilidad:

Si necesitamos más capacidad para una parte específica, como la gestión de productos, podemos aumentar solo los recursos de ese microservicio sin tocar los demás. Esto es súper eficiente.

- Despliegue Independiente:

  Podemos actualizar o desplegar cada microservicio por separado. Esto significa que si necesitamos arreglar un bug o agregar una nueva característica en la gestión de usuarios, no tenemos que rediseñar ni apagar toda la aplicación.

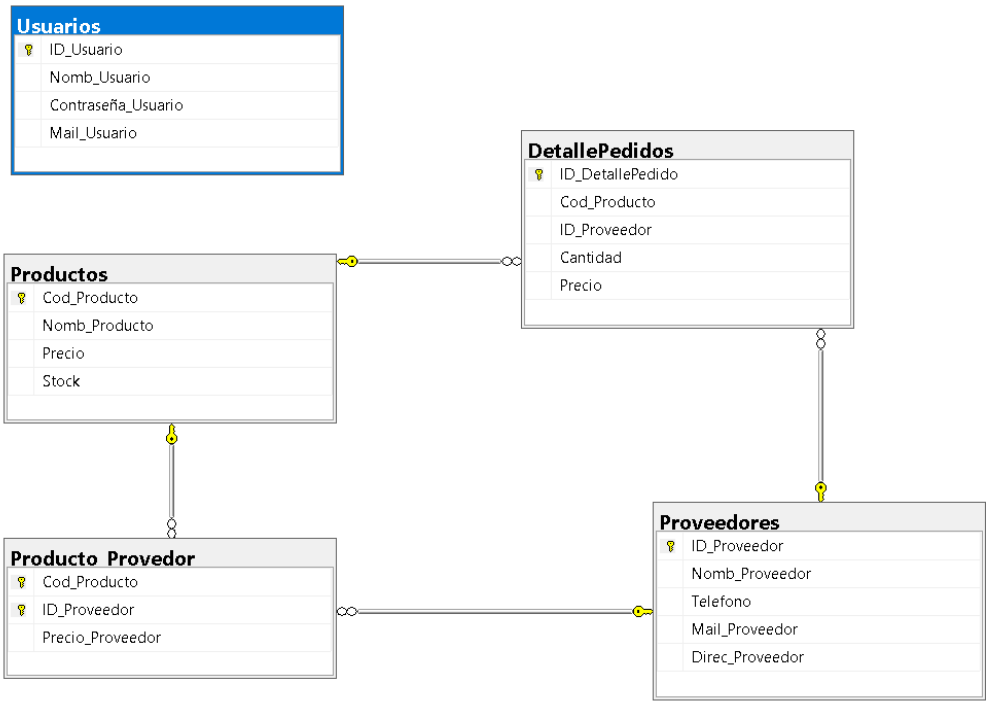
- Resiliencia:

  Si un microservicio falla, el resto de la aplicación puede seguir funcionando. Por ejemplo, si hay un problema con el microservicio que genera informes, la gestión de productos seguirá funcionando sin problemas.

- Tecnología Mixta:

  Podemos usar diferentes tecnologías para diferentes microservicios según lo que mejor se adapte a sus necesidades. Por ejemplo, podemos usar Node.js para el backend del microservicio de productos y Python para los informes, si así lo preferimos.

- **Ejercicio 6:** Diseña la base de datos para la aplicación web del ejercicio 1.



4. Diseño:

Utilizando los siguientes diagrama resuelva los casos de usos de los ejercicios 7 y 8:

1. **Diagrama de Dominio**: Identifica las entidades, atributos y relaciones del sistema.
2. **Diagrama de Robustez**: Analiza cómo el sistema responde a diferentes escenarios de uso.
3. **Prototipo**: Crea una versión simplificada del sistema para probar la usabilidad y funcionalidad.
4. **Diagrama de Secuencia**: Describe la interacción entre los diferentes objetos del sistema.
5. **Diagrama de Clases**: Define las clases, sus atributos, métodos y relaciones

- **Ejercicio 7:** Implementa la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1 utilizando el lenguaje de programación de tu preferencia.

- Diagrama de Dominio:

  Entidades: Productos, Proveedores.

  Atributos: Cod_Producto, Nomb_Producto, Precio, Stock.

  Relaciones: Un producto puede estar relacionado con varios proveedores.

- Diagrama de Clases:

  Clase Producto: con atributos Cod_Producto, Nomb_Producto, Precio, Stock.

  Clase Proveedor: con atributos ID_Proveedor, Nomb_Proveedor, etc.

- Diagrama de Secuencia para "Agregar un Nuevo Producto":
  1. Usuario llena el formulario de "Agregar Producto".
  2. El sistema valida los datos.
  3. El sistema crea un nuevo objeto Producto.
  4. El sistema guarda el objeto Producto en la base de datos.
  5. El sistema confirma la creación al usuario.
  6.

Código en Flask para "Agregar un Nuevo Producto":

```
from flask import Flask, request, render_template, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///inventario.db'
db = SQLAlchemy(app)

class Producto(db.Model):
    Cod_Producto = db.Column(db.Integer, primary_key=True)
    Nomb_Producto = db.Column(db.String(100), nullable=False)
    Precio = db.Column(db.Float, nullable=False)
    Stock = db.Column(db.Integer, nullable=False)
```

```python
@app.route('/productos/nuevo', methods=['GET', 'POST'])
def agregar_producto():
    if request.method == 'POST':
        nombre = request.form['nombre']
        precio = request.form['precio']
        stock = request.form['stock']
        nuevo_producto = Producto(Nomb_Producto=nombre,
Precio=float(precio), Stock=int(stock))

        try:
            db.session.add(nuevo_producto)
            db.session.commit()
            return redirect('/productos')
        except:
            return "Hubo un problema al agregar el producto"
    else:
        return render_template('agregar_producto.html')

if __name__ == "__main__":
    app.run(debug=True)
```

Plantilla HTML (agregar_producto.html):

```html
<!DOCTYPE html>
<html>
<head>
    <title>Agregar Producto</title>
</head>
<body>
    <h1>Agregar Nuevo Producto</h1>
    <form method="POST">
        <label for="nombre">Nombre del Producto:</label><br>
        <input type="text" id="nombre" name="nombre" required><br>
        <label for="precio">Precio:</label><br>
        <input type="text" id="precio" name="precio" required><br>
```

```
        <label for="stock">Stock:</label><br>
        <input type="number" id="stock" name="stock"
required><br><br>
        <input type="submit" value="Agregar Producto">
    </form>
</body>
</html>
```

- **Ejercicio 8:** Implementa la lógica de negocio para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

- Inicio de sesión:
    - Creamos una ruta en Flask para el inicio de sesión donde el usuario pueda ingresar su nombre de usuario y contraseña.
    - Utilizamos sesiones para mantener al usuario autenticado durante su sesión.

- Gestión de inventario:
    - Después de iniciar sesión, el usuario verá un dashboard donde podrá gestionar el inventario.
    - Agregar producto:
    -
    - Al hacer clic en "Agregar producto", se mostrará un formulario HTML para ingresar los detalles del nuevo producto.

- Validación de datos:
    - Utilizamos JavaScript para realizar una validación previa en el lado del cliente para asegurarnos de que todos los campos obligatorios estén completos y que los formatos de los datos sean correctos.
    - En el backend, también realizamos una validación adicional para garantizar la integridad de los datos.

- Inserción en la base de datos:
    - Si los datos son válidos, insertamos el nuevo producto en la base de datos utilizando SQL Alchemy o alguna otra biblioteca ORM.
    -

- Mensaje de éxito:
    - Mostramos un mensaje de éxito al usuario después de agregar exitosamente el producto.

- Generación de reportes:
    - Después de agregar el producto, la aplicación puede generar un reporte detallado, que puede estar disponible para descargar desde el dashboard.

5. Pruebas:

- **Ejercicio 9:** Define un conjunto de pruebas unitarias para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

- Prueba de inserción exitosa:
    - Escenario: El usuario completa todos los campos obligatorios del formulario de agregar producto y hace clic en "Enviar".
    - Acción: Se envía la solicitud al backend de la aplicación.
    - Resultado esperado: Se agrega correctamente el nuevo producto a la base de datos y se devuelve un mensaje de éxito al usuario.

- Prueba de validación de datos:
    - Escenario: El usuario intenta agregar un producto con campos vacíos o datos inválidos.
    - Acción: Se envía la solicitud al backend de la aplicación.
    - Resultado esperado: Se muestra un mensaje de error indicando los campos que deben completarse correctamente antes de agregar el producto.

- Prueba de seguridad:
    - Escenario: Un usuario no autenticado intenta acceder directamente a la funcionalidad de agregar un nuevo producto.
    - Acción: Se intenta acceder a la URL de agregar producto sin iniciar sesión.
    - Resultado esperado: Se redirige al usuario a la pantalla de inicio de sesión y se muestra un mensaje indicando que se requiere autenticación.

- Prueba de interfaz de usuario:
    - Escenario: El usuario interactúa con el formulario de agregar producto y completa todos los campos correctamente.
    - Acción: Se completa el formulario y se envía la solicitud.
    - Resultado esperado: Se verifica que todos los campos se completen correctamente y que la solicitud se envíe correctamente al backend.

- **Ejercicio 10:** Ejecuta pruebas de integración para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

- **Configurar un entorno de prueba:** Prepara un entorno de prueba que sea similar al entorno de producción, pero que no afecte los datos reales de la aplicación. Esto puede implicar configurar una base de datos de prueba y un servidor de prueba si es necesario.

- **Escribir las pruebas de integración:** Define las pruebas que simularán el flujo completo de agregar un nuevo producto en la aplicación. Estas pruebas deben interactuar con la aplicación a nivel de HTTP, enviando solicitudes al servidor y recibiendo respuestas.

- **Configurar herramientas de prueba**: Utiliza herramientas como pytest, Selenium o Flask-Testing para escribir y ejecutar las pruebas de integración. Configura el entorno de prueba y las dependencias necesarias.

- **Ejecutar las pruebas:** Ejecuta las pruebas de integración y observa si todas pasan con éxito. Si alguna prueba falla, identifica el motivo y corrige el problema en el código de la aplicación.

- **Analizar los resultados:** Después de ejecutar las pruebas, revisa los resultados para asegurarte de que la funcionalidad de "Agregar un nuevo producto" funcione correctamente en el entorno de prueba.

6. Despliegue del Programa:

Ejercicio 11: Definir un plan de despliegue para la aplicación web del ejercicio 1.

Pasos del Despliegue:

1. Preparación del Entorno de Producción:
    - Seleccionar un servidor de producción adecuado para alojar la aplicación web.
    - Instalar y configurar el software necesario en el servidor, incluyendo el servidor web (por ejemplo, Apache, Nginx), el servidor de aplicaciones (Node.js) y el sistema de gestión de bases de datos (SQLite).

2. Empaquetado de la Aplicación:
    - Compilar la aplicación web en un paquete que pueda ser desplegado en el servidor de producción.
    - Incluir todos los archivos necesarios, como HTML, CSS, JavaScript, imágenes y archivos de configuración.

3. Pruebas en Entorno de Preproducción:
    - Configurar un entorno de preproducción que sea una réplica del entorno de producción.
    - Realizar pruebas exhaustivas en el entorno de preproducción para garantizar que la aplicación funcione correctamente y sin errores antes de desplegarla en producción.

4. Despliegue en Producción:
    - Coordinar con el equipo de operaciones para programar el despliegue en un momento que minimice el impacto en los usuarios.
    - Transferir el paquete de la aplicación al servidor de producción y descomprimirlo en el directorio de la aplicación.
    - Configurar el servidor web y el servidor de aplicaciones para asegurar el correcto funcionamiento de la aplicación.

5. Verificación del Despliegue:
    - Realizar pruebas de verificación en el entorno de producción para asegurarse de que la aplicación se desplegó correctamente y está funcionando según lo esperado.

- Verificar que todas las funcionalidades principales de la aplicación, incluida la funcionalidad de agregar un nuevo producto, estén disponibles y funcionando sin problemas.

6. Aprobación y Comunicación:
    - Obtener la aprobación del equipo de calidad y del cliente de que el despliegue en producción ha sido exitoso.
    - Comunicar a los usuarios finales y al equipo de soporte que la nueva versión de la aplicación está disponible y proporcionar cualquier instrucción adicional necesaria.

Ejercicio 12: Despliega la aplicación web del ejercicio 1 en un servidor de producción.

1. Preparación del Entorno de Producción:
    - Designa un servidor de producción para alojar la aplicación web.
    - Instala y configura el software necesario en el servidor, como el servidor web (por ejemplo, Apache, Nginx), el servidor de aplicaciones (Node.js) y el sistema de gestión de bases de datos (SQLite).

2. Empaquetado de la Aplicación:
    - Compila la aplicación web en un paquete que pueda ser desplegado en el servidor de producción.
    - Asegúrate de incluir todos los archivos necesarios, como HTML, CSS, JavaScript, imágenes y archivos de configuración.

3. Despliegue en Producción:
    - Coordina con el equipo de operaciones para programar el despliegue en un momento que minimice el impacto en los usuarios.
    - Transfiere el paquete de la aplicación al servidor de producción y descomprímelo en el directorio de la aplicación.
    - Configura el servidor web y el servidor de aplicaciones para asegurar el correcto funcionamiento de la aplicación.
4. Verificación del Despliegue:
    - Realiza pruebas de verificación en el entorno de producción para asegurarte de que la aplicación se desplegó correctamente y está funcionando según lo esperado.

    - Verifica que todas las funcionalidades principales de la aplicación, incluida la funcionalidad de agregar un nuevo producto, estén disponibles y funcionando sin problemas.
5. Aprobación y Comunicación:
    - Obtiene la aprobación del equipo de calidad y del cliente de que el despliegue en producción ha sido exitoso.
    - Comunica a los usuarios finales y al equipo de soporte que la nueva versión de la aplicación está disponible y proporciona cualquier instrucción adicional necesaria.

7. Mantenimiento:

Ejercicio 13: Definir un plan de mantenimiento para la aplicación web del ejercicio 1.

1. Actualizaciones de Seguridad y Parches:
    - Realizar actualizaciones periódicas de todos los componentes de software para proteger la aplicación contra vulnerabilidades conocidas.
    - Aplicar parches de seguridad tan pronto como estén disponibles para mantener la integridad y seguridad de la aplicación.

2. Monitoreo y Alertas:
    - Configurar herramientas de monitoreo para supervisar el rendimiento y la disponibilidad de la aplicación.
    - Establecer alertas para detectar y responder rápidamente a cualquier problema que pueda surgir en la aplicación.

3. Respuesta a Incidentes:
    - Mantener un procedimiento documentado para responder a incidentes de seguridad y otros problemas críticos.
    - Establecer un equipo de respuesta a incidentes para abordar rápidamente cualquier amenaza o vulnerabilidad.

4. Respaldo y Recuperación de Datos:
    - Implementar un sistema de respaldo automatizado para realizar copias de seguridad regulares de la base de datos y los archivos de la aplicación.
    - Probar regularmente la capacidad de recuperación de datos para garantizar la rápida restauración en caso de pérdida de datos.

5. Comunicación con los Usuarios:
    - Mantener una comunicación transparente con los usuarios finales sobre cualquier mantenimiento programado, actualización o cambio en la aplicación.
    - Proporcionar canales de soporte adecuados para que los usuarios puedan informar sobre problemas y recibir ayuda en caso de necesidad

Ejercicio 14: Implementa una corrección de errores para un problema detectado en la aplicación web del ejercicio 1.

1. Identificar el Problema:Comprende el problema detectado en la aplicación.
2. Desarrollar una Solución:Corrige el error en el código.
3. Pruebas:Verifica la corrección con pruebas unitarias y de integración.
4. Despliegue:Implementa la solución en el entorno de producción.
5. Verificación Post-Despliegue:Confirma que la corrección se ha aplicado correctamente.
6. Comunicación:Informa a los usuarios sobre la corrección y cómo proceder en caso de problemas adicionales.

8. Nos preparamos para nuevos retos

- **Ejercicio 15:** Arme un equipo de trabajo y defina los roles para realizar los ejercicios anteriores para un futuro dominio de aplicación relacionado con inteligencia artificial generatica.

- Líder de Proyecto de IA Generativa:
    - Coordina el equipo y garantiza que el proyecto avance según lo planificado.

    - Define la visión del proyecto y establece los objetivos.
    - Interactúa con los stakeholders para comprender sus necesidades y expectativas.

- Desarrollador de IA:
    - Responsable de implementar y mantener los modelos de inteligencia artificial generativa.
    - Escribe código para entrenar y mejorar los modelos de IA.
    - Colabora con el científico de datos en la implementación de algoritmos y técnicas de IA.

- Científico de Datos Especializado en IA Generativa:
    - Desarrolla y mejora los modelos de inteligencia artificial generativa.
    - Investiga nuevas técnicas y algoritmos para mejorar el rendimiento de los modelos.
    - Trabaja en estrecha colaboración con el equipo de desarrollo para implementar soluciones basadas en IA.

- Ingeniero de Infraestructura de IA:
    - Configura y mantiene la infraestructura tecnológica necesaria para admitir modelos de inteligencia artificial generativa.
    - Optimiza el rendimiento y la escalabilidad de los sistemas de IA.
    - Implementa prácticas de seguridad para proteger los datos y los modelos.

- Analista de Calidad de IA:
    - Realiza pruebas exhaustivas para asegurar el correcto funcionamiento de los modelos de inteligencia artificial generativa.
    - Desarrolla casos de prueba y escenarios de prueba para validar la funcionalidad y la precisión de los modelos.
    - Identifica y reporta problemas o deficiencias en los modelos y colabora en su resolución.

- Stakeholders:
    - Proporcionan requisitos y feedback para el proyecto.
    - Participan en revisiones y demostraciones para validar la dirección y el progreso del proyecto.
    - Aportan conocimientos del dominio para guiar el desarrollo y la implementación de los modelos de IA generativa.