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:

1. *Registro de productos*

Los usuarios deben poder ingresar nuevos productos al sistema, proporcionando información como nombre, descripción, categoría, cantidad inicial.

2. *Búsqueda y filtrado de productos*

Los usuarios deben poder buscar y filtrar productos por nombre o categoría para localizar rápidamente elementos específicos en el inventario.

3. *Actualización de inventarios*

El sistema debe permitir a los usuarios actualizar las cantidades disponibles de productos en el inventario, ya sea agregando, eliminando o ajustando existencias.

4. *Gestión de proveedores*

Debe ser posible registrar y gestionar información de proveedores, asociando productos a proveedores específicos y manteniendo datos de contacto relevantes.

5. *Generación de Reportes*

Los usuarios deberían poder generar informes sobre el estado actual del inventario, incluyendo detalles como existencias disponibles, los tres productos con mayor y menor stock.

Requisitos no funcionales:

1. *Usabilidad*

La interfaz de usuario debe ser intuitiva y fácil de usar, considerando usuarios no técnicos que gestionarán el inventario.

2. *Rendimiento*

El sistema debe ser eficiente y responder rápidamente, incluso con grandes volúmenes de datos.

3. *Escalabilidad*

Debe ser capaz de manejar un crecimiento futuro en términos de cantidad de productos y usuarios concurrentes.

4. *Seguridad*

Deben implementarse medidas de seguridad robustas para proteger la integridad y confidencialidad de los datos del inventario.

5. *Disponibilidad*

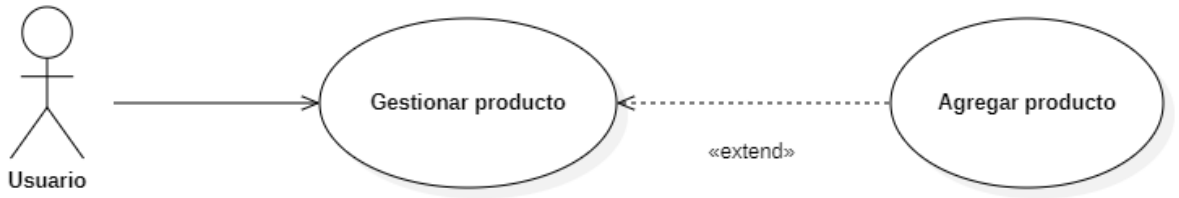
El sistema debe estar disponible en todo momento, minimizando tiempos de inactividad no planificados.

6. *Compatibilidad*

Debe ser compatible con diferentes navegadores web y dispositivos para garantizar una experiencia consistente para los usuarios.

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

Diagrama



Especificación

	Caso de uso de análisis	Fecha: 24/04/2024
Código	CUAN 01	
Nombre	Agregar producto	
Autor	Santiago Vittori	
Revisor	Alejandro Sartorio	
Versión	1.0	
Descripción	El usuario podrá agregar un nuevo producto	
Actores	Usuario	
Pre-condición	El usuario debe tener los permisos correspondientes para poder agregar un nuevo	

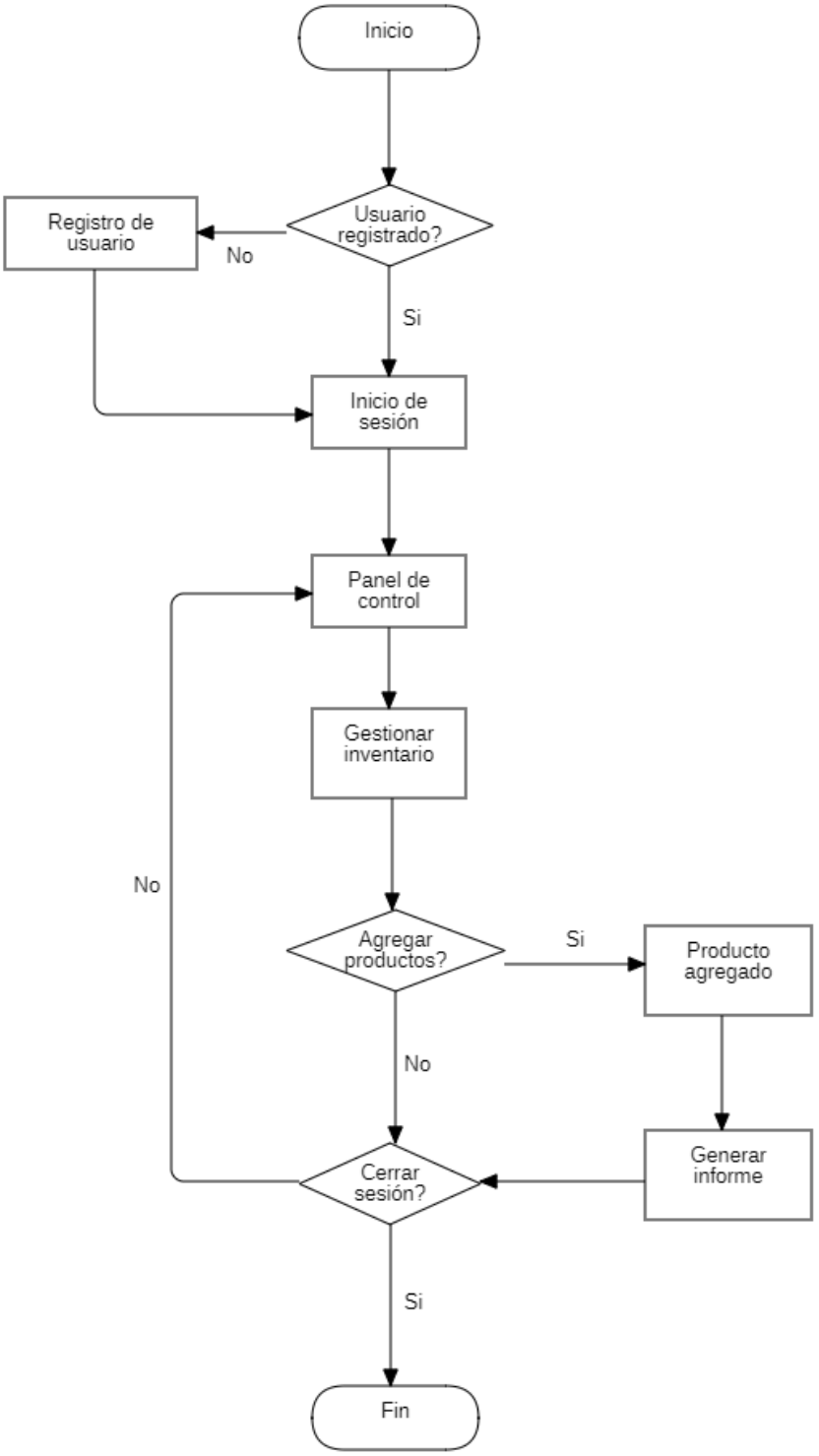
	producto
Pos-condición	El nuevo producto se agrega correctamente al inventario del sistema

	Caso de uso de diseño	Fecha: 24/04/2024
ID y Nombre	CUD 01 - Agregar producto	
Requerimiento	R1	
Estado	Pendiente	
Descripción	El usuario podrá agregar un nuevo producto	
Actor principal	Usuario	
Actor secundario	-	
Pre-condiciones	El usuario debe tener los permisos correspondientes para poder agregar un nuevo producto	
Puntos de extensión	-	
Condición	-	
Escenario principal	1) El usuario accede a la función "Agregar producto". 2) El sistema muestra un formulario de ingreso de datos para el nuevo producto. 3) El usuario completa el formulario proporcionando la siguiente información del producto: a) Nombre del producto (campo obligatorio). b) Descripción del producto (campo opcional). c) Categoría del producto (campo obligatorio). d) Cantidad inicial en inventario (campo obligatorio, número entero positivo). 4) El usuario confirma la acción de agregar el producto. 5) El sistema valida los datos ingresados por el usuario. 6) El sistema almacena el nuevo producto en la base de datos del inventario.	
Flujos alternativos	1) Cancelar la operación a) El usuario decide cancelar la operación antes de confirmar el formulario. b) El sistema redirige al usuario de vuelta a la página principal o a la lista de productos sin realizar ninguna acción. 2) Validación de datos fallida a) El usuario intenta confirmar la acción de agregar el producto. b) El sistema valida los datos ingresados por el usuario y detecta algún error (por ejemplo, campos obligatorios no completados o datos inválidos). c) El sistema muestra mensajes de error junto a los campos correspondientes para indicar los problemas. d) El usuario corrige los datos según las indicaciones proporcionadas. 3) Error en el procesamiento a) El usuario confirma la acción de agregar el producto. b) El sistema procesa la solicitud pero encuentra un error durante el proceso (por ejemplo, error de conexión a la base de datos).	

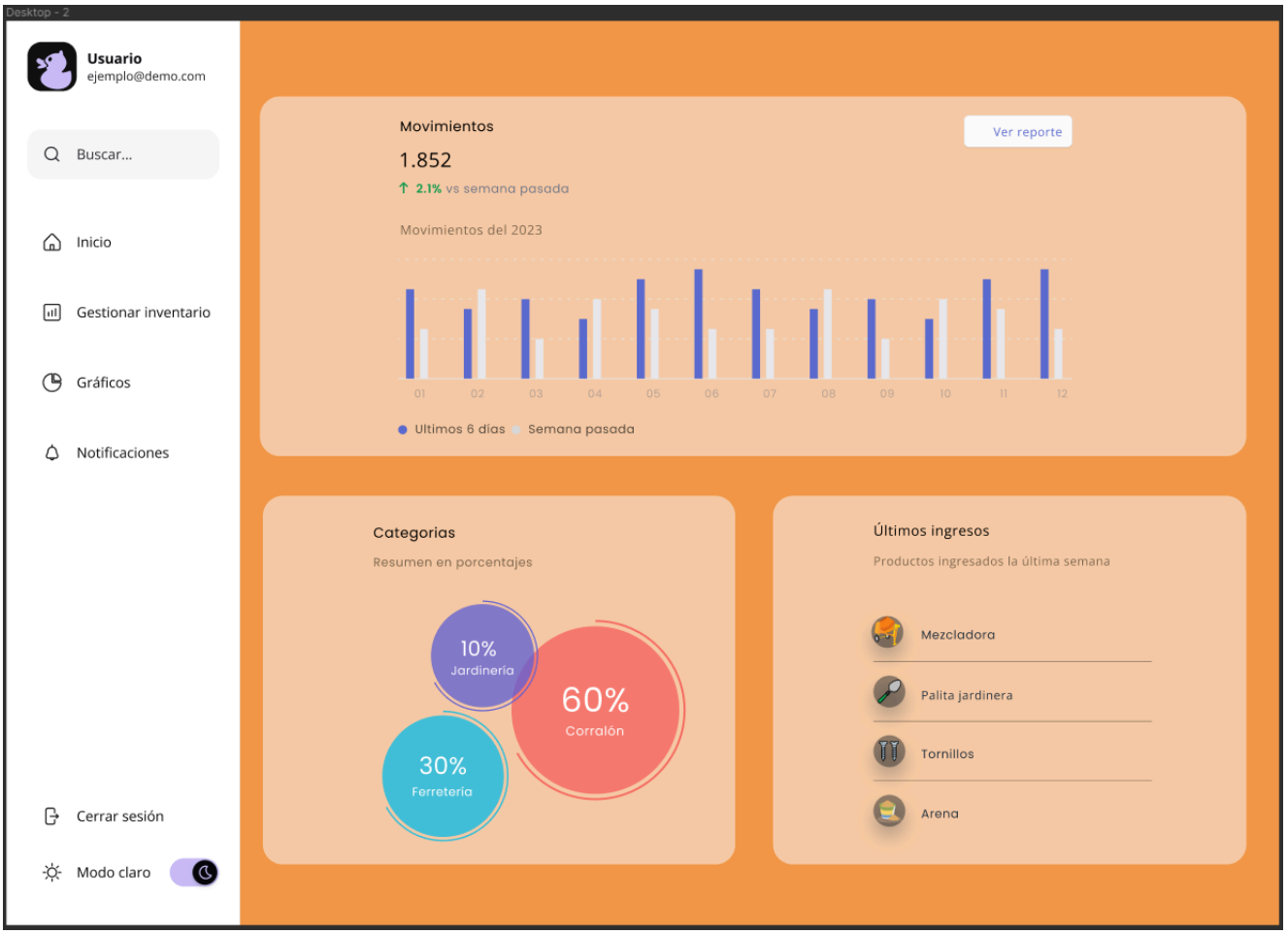
	c) El sistema muestra un mensaje de error genérico indicando que la operación no pudo completarse. d) El usuario puede intentar nuevamente más tarde o ponerse en contacto con el soporte técnico para resolver el problema.
Post-condiciones	El nuevo producto se agrega correctamente al inventario del sistema

2. Diseño del sistema

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

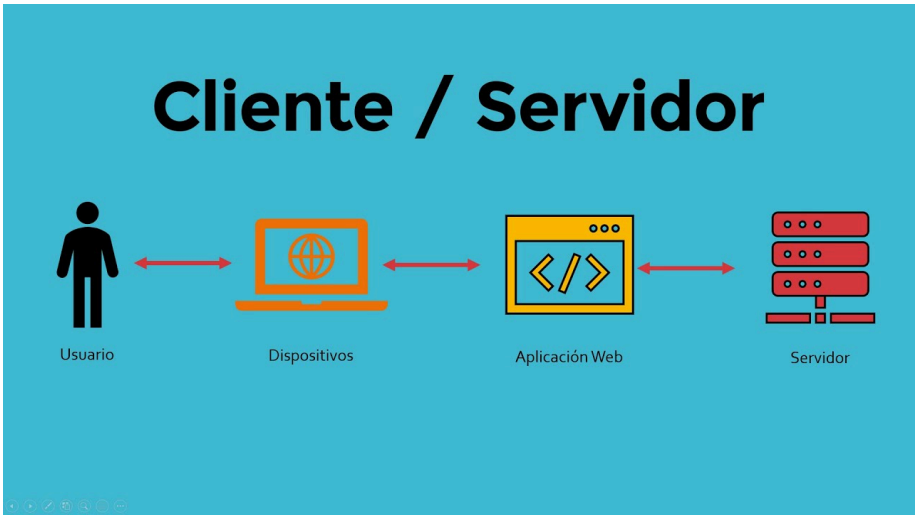


Ejercicio 4: Diseña la interfaz de usuario para la pantalla de "Inicio" de la aplicación web del ejercicio 1.



3. Diseño del programa

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



Arquitectura cliente-servidor

Justificación

Separación de responsabilidades

La arquitectura de tres capas permite separar claramente las responsabilidades de presentación, lógica de negocio y almacenamiento de datos. Esto facilita el mantenimiento y la escalabilidad de la aplicación, ya que cada capa puede ser modificada de manera independiente sin afectar a las demás.

Escalabilidad

Con esta arquitectura, es posible escalar cada capa de forma independiente según sea necesario.

Reutilización de componentes

Al separar la lógica de negocio y el almacenamiento de datos en capas independientes, es más fácil reutilizar componentes en diferentes partes de la aplicación o en otras aplicaciones relacionadas.

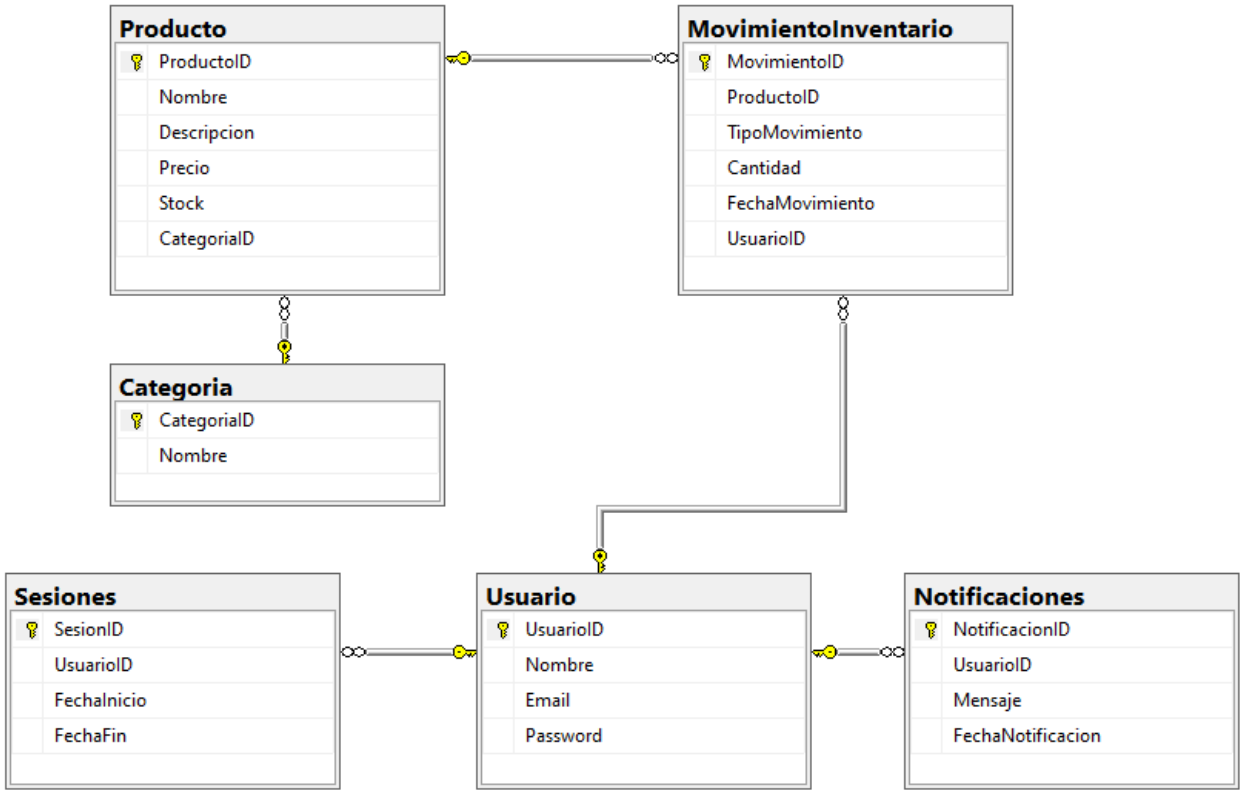
Facilidad de mantenimiento

La clara separación de responsabilidades facilita la identificación y corrección de problemas en la aplicación. Los cambios en una capa no afectarán directamente a las demás, lo que simplifica el mantenimiento y reduce el riesgo de introducir nuevos errores.

Interoperabilidad

La arquitectura de tres capas es compatible con estándares y protocolos de comunicación ampliamente utilizados, lo que facilita la integración con otros sistemas y servicios externos.

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



4. Diseño

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.

Voy a explicar de forma resumida cuales son los pasos que yo haría y cómo quedaría el código resumido para dicha implementación.

Primero, debemos tener Node.js instalado. Luego, instalaremos Express y sqlite3 utilizando npm:

```
mkdir gestion_inventario
cd gestion_inventario
npm init -y
npm install express sqlite3
```

Luego en una carpeta public, crear el index.html.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Gestión de Inventario</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Agregar Producto</h1>
    <form id="agregarProductoForm">
        <label for="nombre">Nombre:</label><br>
        <input type="text" id="nombre" name="nombre"><br>
        <label for="descripcion">Descripción:</label><br>
        <textarea id="descripcion" name="descripcion"></textarea><br>
        <label for="precio">Precio:</label><br>
        <input type="number" id="precio" name="precio"><br>
        <label for="stock">Stock:</label><br>
        <input type="number" id="stock" name="stock"><br>
        <button type="submit">Agregar Producto</button>
    </form>
    <script src="script.js"></script>
</body>
</html>
```

Dentro de public, crear script.js.

```
document.getElementById('agregarProductoForm').addEventListener('submit'
, async function(event) {
    event.preventDefault();

    const formData = new FormData(this);
    const nombre = formData.get('nombre');
    const descripcion = formData.get('descripcion');
    const precio = formData.get('precio');
    const stock = formData.get('stock');

    try {
        const response = await fetch('/productos', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ nombre, descripcion, precio, stock })
        });

        if (!response.ok) {
            throw new Error('Error al agregar producto');
        }

        alert('Producto agregado correctamente');
        this.reset();
    } catch (error) {
        console.error(error);
        alert('Hubo un error al agregar el producto');
    }
});
```

Por último, creamos server.js para el backend.

```
const express = require('express');
const sqlite3 = require('sqlite3');

const app = express();
const port = 3000;

// Middleware para manejar solicitudes JSON
app.use(express.json());

// Conexión a la base de datos SQLite
const db = new sqlite3.Database('inventario.db');

// Crear tabla de productos si no existe
db.serialize(() => {
    db.run(`
        CREATE TABLE IF NOT EXISTS Producto (
```

```
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nombre TEXT,
            descripcion TEXT,
            precio REAL,
            stock INTEGER
        )
    `);
});

// Ruta para agregar un nuevo producto
app.post('/productos', (req, res) => {
    const { nombre, descripcion, precio, stock } = req.body;

    db.run(`
        INSERT INTO Producto (nombre, descripcion, precio, stock)
        VALUES (?, ?, ?, ?)
    `, [nombre, descripcion, precio, stock], (err) => {
        if (err) {
            console.error(err);
            res.status(500).send('Error al agregar el producto');
        } else {
            res.status(200).send('Producto agregado correctamente');
        }
    });
});

app.use(express.static('public'));

app.listen(port, () => {
    console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

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

Esta es la lógica de negocio paso a paso utilizando un enfoque con proceso unificado.

Análisis de requisitos
Revisar detalladamente los requisitos funcionales relacionados con la adición de un nuevo producto, incluyendo los campos necesarios, restricciones de datos y validaciones requeridas.

Diseño preliminar
Diseñar la estructura de datos en la capa de datos para almacenar la información del nuevo producto.

Implementación
Desarrollar la lógica de negocio en la capa de lógica de aplicación para procesar la solicitud de agregar un nuevo producto.

Pruebas unitarias
Desarrollar pruebas unitarias específicas para validar la lógica de negocio relacionada con la adición de productos.

Pruebas de integración
Integrar la funcionalidad de agregar un nuevo producto en el contexto de la aplicación web más amplia.

Validación del cliente
Demostrar la funcionalidad de agregar un nuevo producto al cliente para obtener su validación y feedback.

Despliegue en producción
Preparar la aplicación para su despliegue en el entorno de producción, incluyendo la configuración del servidor y la instalación de la aplicación.

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 unitaria 1: Agregar un nuevo producto con datos válidos
Descripción:
Se prueba que la función de agregar un nuevo producto funcione correctamente cuando se proporcionan datos válidos para el nuevo producto.
Pasos:
Crear un nuevo producto con datos válidos, como un nombre válido, una descripción válida, un precio positivo, categoría válida y un stock positivo.
Llamar a la función de agregar un nuevo producto con los datos del nuevo producto.
Verificar que la función devuelve un mensaje indicando que el producto ha sido agregado correctamente.
Resultado esperado:
La función debería devolver un mensaje de éxito indicando que el producto ha sido agregado correctamente.

Prueba unitaria 2: Intentar agregar un nuevo producto sin nombre
Descripción:
Se prueba que la función de agregar un nuevo producto maneje correctamente el caso en que se intenta agregar un producto sin especificar un nombre.
Pasos:
Crear un nuevo producto sin proporcionar un nombre, pero con otros datos válidos.
Llamar a la función de agregar un nuevo producto con los datos del nuevo producto sin nombre.
Verificar que la función devuelve un mensaje de error indicando que el nombre del producto es obligatorio.
Resultado esperado:
La función debería devolver un mensaje de error indicando que el nombre del producto es obligatorio.

Prueba unitaria 3: Intentar agregar un nuevo producto con precio negativo
Descripción:
Se prueba que la función de agregar un nuevo producto maneje correctamente el caso en que se intenta agregar un producto con un precio negativo.
Pasos:
Crear un nuevo producto con un precio negativo, pero con otros datos válidos como nombre, descripción y stock positivo.
Llamar a la función de agregar un nuevo producto con los datos del nuevo producto con precio negativo.
Verificar que la función devuelve un mensaje de error indicando que el precio del producto debe ser mayor o igual a cero.
Resultado esperado:
La función debería devolver un mensaje de error indicando que el precio del producto debe ser mayor o igual a cero.

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

Se utiliza el concepto de iteración, presente en el proceso unificado para realizar las pruebas de integración. Cada una itera sobre la anterior.

Prueba de integración 1: Verificar la inserción del producto en la base de datos
Descripción:
Se prueba que la funcionalidad de agregar un nuevo producto se integra correctamente con la base de datos y que el producto se inserta correctamente en la tabla de productos.
Pasos:
Ejecutar la función de agregar un nuevo producto con datos válidos.
Realizar una consulta a la base de datos para verificar si el nuevo producto ha sido insertado correctamente.
Verificar que los datos del producto insertado coinciden con los datos proporcionados en la función de agregar.
Resultado esperado:
Los datos del nuevo producto insertado en la base de datos deberían coincidir con los datos proporcionados al agregar el producto.

Prueba de integración 2: Verificar la respuesta del backend al agregar un nuevo producto
Descripción:
Se prueba que el backend responde correctamente al agregar un nuevo producto y que devuelve la respuesta esperada al cliente.
Pasos:
Enviar una solicitud HTTP POST a la ruta correspondiente para agregar un nuevo producto, con datos válidos en el cuerpo de la solicitud.
Capturar la respuesta del backend.
Verificar que la respuesta contiene un código de estado HTTP 200 (OK) y un mensaje indicando que el producto ha sido agregado correctamente.
Resultado esperado:

La respuesta del backend debería incluir un código de estado HTTP 200 y un mensaje indicando que el producto ha sido agregado correctamente.

Prueba de integración 3: Verificar la visualización del nuevo producto en la interfaz web

Descripción:

Se prueba que la interfaz web muestra correctamente el nuevo producto agregado después de realizar la acción de agregar.

Pasos:

Agregar un nuevo producto a través de la interfaz web.

Navegar a la página o sección donde se muestran todos los productos.

Verificar que el nuevo producto agregado aparece en la lista de productos mostrados en la interfaz web.

Resultado esperado:

El nuevo producto agregado debería ser visible en la interfaz web junto con los demás productos existentes.

6. Despliegue del programa

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

Pasos del despliegue adaptado a proceso unificado

Inicio: preparación y planificación

Definir los requisitos de despliegue y las necesidades del entorno de producción.

Identificar el servidor de producción y los recursos necesarios, como software y capacidades de hardware.

Establecer un equipo de despliegue y asignar responsabilidades.

Elaboración: preparación del entorno de producción

Designar y configurar el servidor de producción con el software necesario, incluyendo el servidor web y el servidor de aplicaciones.

Configurar la infraestructura de red para permitir el acceso a la aplicación desde los usuarios finales.

Realizar pruebas preliminares de conectividad y rendimiento en el entorno de producción.

Construcción: empaquetado y despliegue

Empaquetar la aplicación web en un paquete que pueda ser desplegado en el servidor de producción, incluyendo todos los archivos necesarios y las configuraciones.

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, ajustar la aplicación y el servidor para garantizar su correcto funcionamiento.

Transición: verificación y aprobación
Realizar pruebas exhaustivas en el entorno de preproducción para garantizar que la aplicación funcione correctamente y sin errores antes de desplegarla en producción.
Realizar pruebas de verificación en el entorno de producción para asegurarse de que la aplicación se desplegó correctamente.
Obtener la aprobación del equipo de calidad y del cliente de que el despliegue en producción ha sido exitoso.

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

En este caso, siendo más técnico, se relaciona mucho con el punto anterior, donde se definió el plan, y aquí se debería ejecutar.

Para agregar a los pasos que se deberían llevar a cabo, podría mencionar las personas involucradas en el proceso.

- Equipo de desarrollo: Encargado de compilar la aplicación y prepararla para el despliegue.
- Equipo de operaciones: Responsable de configurar el servidor de producción y coordinar el despliegue.
- Equipo de calidad o QA: Encargado de realizar pruebas de verificación para garantizar que la aplicación se despliegue correctamente.
- Cliente y usuarios finales: Responsables de aprobar el despliegue y utilizar la aplicación desplegada.

Para hacer efectivo el despliegue se deberá configurar correctamente el hosting con el respectivo nombre de dominio de la aplicación web. A partir de ello se podrá levantar la app y podría accederse desde internet, siendo útil para el cliente.

7. Mantenimiento

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

Plan de mantenimiento

1. *Monitoreo continuo*

Establecer sistemas de monitoreo automatizados para supervisar el rendimiento, la disponibilidad y la integridad de la aplicación web.
Realizar revisiones periódicas de los registros de errores y alertas.

2. *Actualizaciones de seguridad*

Mantener al día todas las bibliotecas, frameworks y dependencias utilizadas en la aplicación para mitigar las vulnerabilidades.
Aplicar parches de seguridad según sea necesario para proteger la aplicación contra posibles amenazas.

3. *Optimización del rendimiento*

Realizar pruebas de rendimiento periódicas para identificar y abordar cuellos de botella en el rendimiento de la aplicación.
Optimizar consultas de bases de datos, mejorar la velocidad de carga de páginas y reducir el uso de recursos para mejorar la experiencia del usuario.

4. Actualizaciones

Recopilar y evaluar solicitudes de nuevas características de los usuarios y las partes interesadas.

5. Copias de seguridad

Establecer un programa regular de copias de seguridad automatizadas para proteger los datos de la aplicación.

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

1. Identificación del problema: El equipo de soporte detecta un problema en la aplicación web.
2. Análisis del problema: El equipo de desarrollo examina el problema para comprender su causa.
3. Corrección del error: Se modifica el código de la aplicación para resolver el problema detectado.
4. Pruebas de verificación: Se realizan pruebas para asegurar que la corrección solucione el problema.
5. Despliegue en producción: La corrección de errores se despliega en el entorno de producción.
6. Verificación y aprobación: Se verifica que la corrección funcione correctamente y se obtiene la aprobación para el despliegue.
7. Comunicación y documentación: Se informa a los usuarios sobre la corrección y se documenta.

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 generativa.

Para mi equipo de trabajo, los roles serían los siguientes teniendo en cuenta el proceso unificado:

1. Gerente de proyecto:

Coordina y gestiona el equipo.
Define objetivos y asigna tareas.
Comunica con el cliente y gestiona riesgos.

2. Analista de requisitos:

Documenta requisitos del sistema.
Define casos de uso y escenarios.

3. Arquitecto de software:

Diseña la arquitectura.

Define comunicaciones entre módulos.

4. Desarrollador de software:

Implementa el código de la aplicación.
Integra modelos de IA.

5. Científico de datos:

Desarrolla y entrena modelos de IA.
Realiza análisis de datos.

6. Ingeniero de infraestructura:

Configura y mantiene la infraestructura.
Gestiona recursos y optimiza el rendimiento.

7. Analista de calidad o QA:

Diseña y ejecuta pruebas.
Reporta y gestiona problemas.