

Ejercicio 8 – Zari Ezequiel

Ejercicio 1: Requisitos Funcionales y No Funcionales

Requisitos Funcionales:

1. Gestión de Inventario:

- El sistema debe permitir a los usuarios agregar, editar y eliminar productos del inventario.
- Los usuarios deben poder ver una lista de todos los productos en el inventario.
- El sistema debe permitir la búsqueda de productos por nombre, categoría o código de producto.

2. Control de Stock:

- El sistema debe alertar a los usuarios cuando el stock de un producto esté por debajo de un umbral definido.
- Debe permitir la actualización del stock al recibir nuevos productos.

3. Reportes:

- Generar reportes sobre el estado del inventario, incluyendo productos más vendidos, menos vendidos y productos fuera de stock.
- Los reportes deben ser exportables a formatos como PDF y Excel.

4. Gestión de Usuarios:

- El sistema debe permitir la creación y gestión de cuentas de usuario con diferentes roles (administrador, empleado).
- Los administradores deben poder asignar y revocar permisos a otros usuarios.

Requisitos No Funcionales:

1. Rendimiento:

- El sistema debe poder procesar las solicitudes de los usuarios en menos de 2 segundos.
- Debe soportar hasta 1000 usuarios concurrentes sin degradación del rendimiento.

2. Seguridad:

- Debe implementar autenticación y autorización seguras.

- Los datos sensibles deben ser cifrados tanto en tránsito como en reposo.

3. Usabilidad:

- La interfaz de usuario debe ser intuitiva y fácil de usar.
- Debe ser accesible desde dispositivos móviles y de escritorio.

4. Escalabilidad:

- El sistema debe ser escalable para manejar un crecimiento futuro en el número de usuarios y volumen de datos.

5. Mantenimiento:

- El código debe estar bien documentado para facilitar el mantenimiento y actualización.
- Debe permitir la fácil integración de nuevas funcionalidades en el futuro.

Ejercicio 2: Requisitos Funcionales y No Funcionales

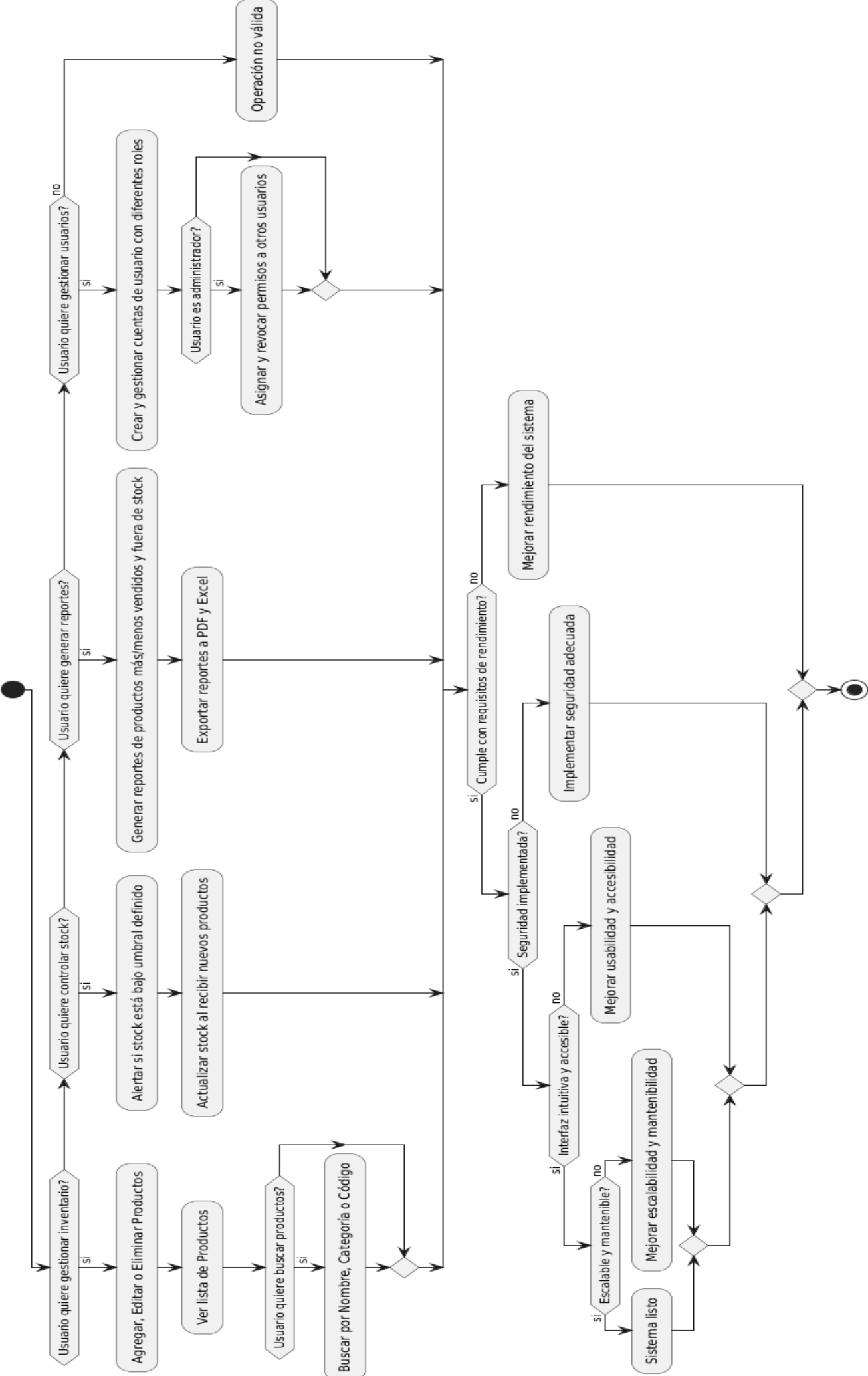
	CASO DE USO	Fecha: 16/06/2024
Código	CUAP 01	
Nombre	Agregar un Nuevo producto	
Autor	Ezequiel Zari	
Versión	1.0	
Descripción	El administrador o empleado agrega un nuevo producto al inventario de la aplicación web.	
Actores	Administrador, empleado	
pre-condición	El usuario debe haber iniciado sesión en el sistema. El usuario debe tener permisos para agregar productos.	

post-condición	El nuevo producto se agrega exitosamente al inventario. El sistema confirma la adición del producto.
----------------	---------------------------------------------------------------------------------------------------------

ID y nombre: CUAP 01 Agregar un nuevo producto
Referencias: R1
Estado: Pendiente
Descripción: El administrador o empleado puede agregar un nuevo producto al inventario de la aplicación web
Actor Principal: Administrador, empleado
Actor Secundario: -
Precondición: El usuario debe haber iniciado sesión en el sistema y tener permisos para agregar productos.
Puntos de Extensión: -
Condición: -
Escenario Principal: <ol style="list-style-type: none"> 1. El usuario selecciona la opción "Agregar nuevo producto" en el menú de navegación del sistema. 2. Se renderiza el formulario de registro de productos en la pantalla con cada campo vacío. 3. El usuario ingresa la información requerida para el registro: nombre del producto, código del producto, categoría, cantidad inicial en stock, precio y descripción (opcional). 4. El sistema valida el tipo de datos ingresados por el usuario y que el producto no exista previamente. 5. El sistema crea el nuevo producto en la base de datos. 6. Se despliega un mensaje informativo indicando la creación exitosa del producto.
Flujos Alternativos: <ol style="list-style-type: none"> 2.1. Se despliega un mensaje informando el dato inválido y se hace foco en dicho campo del formulario. 4.1. Se despliega un mensaje informativo indicando que el producto ya existe. 4.2. Fin CU.
Postcondiciones: Se crea exitosamente el nuevo producto en el inventario de la aplicación web.

2 .Diseño de Sistema

Ejercicio 3: Elabora el diagrama de flujo de datos para la aplicación 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.

Para la aplicación web descrita en el ejercicio 1, que requiere gestionar inventario, control de stock, reportes, gestión de usuarios, y cumplir con requisitos no funcionales como rendimiento, seguridad, usabilidad, escalabilidad y mantenimiento, una arquitectura adecuada sería la arquitectura de **microservicios**. A continuación, justifico esta elección:

Justificación de la Arquitectura de Microservicios:

1. Desacoplamiento y Modularidad:

- Los microservicios permiten dividir la aplicación en servicios independientes, como gestión de inventario, control de stock, reportes y gestión de usuarios. Esto simplifica el desarrollo, prueba y despliegue de cada servicio por separado.

2. Escalabilidad:

- Cada microservicio puede escalar de manera independiente según las necesidades específicas de carga de trabajo. Por ejemplo, el servicio de gestión de inventario puede escalar diferente al servicio de reportes, optimizando recursos y cumpliendo con requisitos de escalabilidad.

3. Facilidad de Mantenimiento y Actualización:

- Al estar separados, los microservicios facilitan el mantenimiento y la actualización. Cambios en un servicio no afectan a los demás, reduciendo errores y permitiendo una evolución ágil de la aplicación.

4. Rendimiento y Tiempo de Respuesta:

- Cada microservicio puede optimizarse individualmente para cumplir con requisitos como procesamiento rápido de solicitudes (menos de 2 segundos). Esto evita que problemas en un servicio afecten negativamente al rendimiento global.

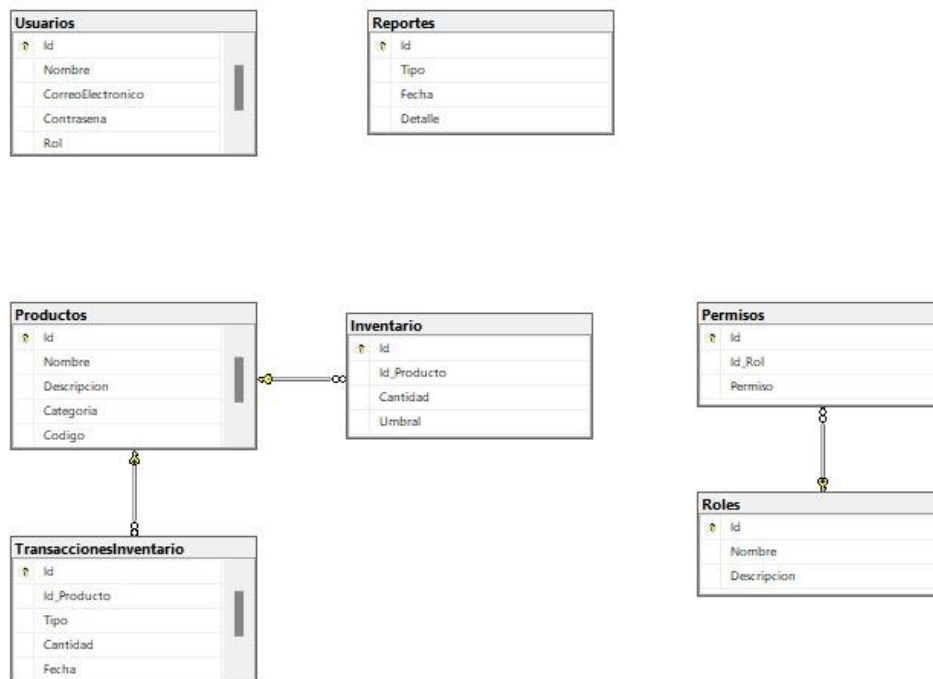
5. Seguridad:

- Los microservicios pueden implementar capas de seguridad específicas, como autenticación y autorización, de manera granular y robusta. Esto asegura el manejo preciso y seguro de datos sensibles.

6. Usabilidad y Adaptabilidad:

- La arquitectura de microservicios facilita la creación de interfaces de usuario intuitivas y accesibles. Se pueden adaptar fácilmente para funcionar en dispositivos móviles y de escritorio, cumpliendo con requisitos de usabilidad.

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



- **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:

El diagrama de dominio identifica las entidades principales, sus atributos y las relaciones entre ellas en el sistema. Para el ejercicio 1, las entidades clave incluyen Usuarios, Productos, Inventario, TransaccionesInventario, Reportes, Roles, y Permisos.

Diagrama de Clases:


El diagrama de clases define las clases, sus atributos, métodos y relaciones. Aquí se incluirían clases como Usuario, Producto, Inventario, TransaccionInventario, Reporte, Rol, Permiso, y posiblemente clases adicionales para servicios y controladores.

Ruta (routes/productos.js):

```
javascript Copiar código  
  
const express = require('express');  
const router = express.Router();  
const productosController = require('../controllers/productosController');  
  
router.post('/agregar', productosController.agregarProducto);  
  
module.exports = router;
```


Controlador (controllers/productosController.js):

javascript

 Copiar código

```
const Producto = require('../models/Producto');

async function agregarProducto(req, res) {
  try {
    const { nombre, descripcion, categoria, codigo } = req.body;
    const nuevoProducto = new Producto({
      nombre,
      descripcion,
      categoria,
      codigo
    });
    await nuevoProducto.save();
    res.status(201).json({ mensaje: 'Producto agregado correctamente' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ mensaje: 'Error al agregar el producto' });
  }
}
```

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

```
// controllers/productosController.js

const Producto = require('../models/producto');

async function agregarProducto(req, res) {
  try {
    const { nombre, descripcion, categoria, codigo } = req.body;

    // Validación de datos
    if (!nombre || !codigo) {
      return res.status(400).json({ mensaje: 'El nombre y el código del producto son obligatorios' });
    }

    // Verificar si el código del producto ya existe
    const existeProducto = await Producto.findOne({ codigo });
    if (existeProducto) {
      return res.status(400).json({ mensaje: 'Ya existe un producto con este código' });
    }
  }
}
```

```
// Crear nuevo producto
const nuevoProducto = new Producto({
  nombre,
  descripcion,
  categoria,
  codigo
});

// Guardar producto en la base de datos
await nuevoProducto.save();
res.status(201).json({ mensaje: 'Producto agregado correctamente', producto: nuevoProducto });
} catch (error) {
  console.error('Error al agregar el producto:', error);
  res.status(500).json({ mensaje: 'Error al agregar el producto' });
}
}

module.exports = {
  agregarProducto
};
```

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.

Las pruebas unitarias se enfocan en verificar el funcionamiento correcto de unidades individuales de código, como funciones o métodos, de manera aislada. Para la funcionalidad de "Agregar un nuevo producto", podrías considerar los siguientes puntos para tus pruebas unitarias:

1. **Validación de Datos:** Asegúrate de tener pruebas que verifiquen que la función de agregar producto valide correctamente los datos de entrada, como el nombre, la descripción, la categoría y el código del producto.
2. **Manejo de Errores:** Incluye pruebas para verificar que la función maneje adecuadamente errores esperados, como datos faltantes o intentos de agregar productos con códigos que ya existen.
3. **Persistencia en la Base de Datos:** Verifica que la función guarde el producto correctamente en la base de datos y que los datos almacenados sean consistentes con los proporcionados.

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

Las pruebas de integración se centran en verificar que diferentes partes del sistema interactúen correctamente entre sí. Para la funcionalidad de "Agregar un nuevo producto", aquí hay aspectos clave a considerar para tus pruebas de integración:

1. **Interacción con la API:** Utiliza herramientas como Postman o supertest para simular solicitudes HTTP a tu API REST que maneja la creación de productos. Verifica que la API responda correctamente según los diferentes casos de uso, como agregar un producto nuevo con datos válidos o datos inválidos.
2. **Integración con la Base de Datos:** Asegúrate de que la información enviada a través de la API se refleje correctamente en la base de datos. Puedes verificar esto consultando la base de datos directamente después de realizar una solicitud de agregar producto.
3. **Escenarios Complejos:** Prueba casos complejos como la concurrencia (varios usuarios intentando agregar productos al mismo tiempo), manejo de transacciones y cualquier otro escenario específico de tu aplicación.

Metodología y Herramientas

- **Frameworks de Pruebas:** Utiliza frameworks como Mocha, Jest, o Jasmine para organizar y ejecutar tus pruebas.
- **Assertions:** Emplea bibliotecas de aserciones como Chai o Jest para verificar los resultados esperados de manera clara y concisa.
- **Mocking y Stubs:** En algunos casos, es útil utilizar técnicas de mock o stub para simular dependencias externas como bases de datos o servicios web durante las pruebas unitarias.
- **Automatización:** Integra tus pruebas en tu proceso de integración continua (CI) para ejecutarlas automáticamente cada vez que se realice un cambio en el código.

Resultados Esperados

Al completar estos ejercicios, deberías tener un conjunto sólido de pruebas unitarias que verifiquen cada aspecto de la funcionalidad de agregar productos individualmente, así como pruebas de integración que aseguren que esta funcionalidad funcione correctamente dentro del contexto más amplio de tu aplicación.

7. Mantenimiento:

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

Monitoreo Continuo: Implementa herramientas para supervisar el rendimiento y detectar problemas.

Actualizaciones de Seguridad: Establece un calendario para aplicar parches y actualizaciones de seguridad.

Backup y Recuperación: Implementa un sistema de respaldo robusto y realiza pruebas periódicas de recuperación.

Optimización de Rendimiento: Evalúa y optimiza el rendimiento de la aplicación regularmente.

Documentación y Gestión de Cambios: Mantén documentación actualizada y gestiona cambios de manera estructurada.

Capacitación y Soporte: Proporciona capacitación continua y un sistema de soporte efectivo.

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

Identificación del Problema: Utiliza herramientas de monitoreo y retroalimentación para identificar el problema.

Reproducción del Error: Intenta reproducir el error en un entorno de desarrollo para entender su causa raíz.

Desarrollo de la Solución: Desarrolla una solución efectiva siguiendo buenas prácticas de desarrollo.

Pruebas de Verificación: Realiza pruebas exhaustivas para asegurar que la solución funcione correctamente.

Implementación en Producción: Despliega la corrección de errores minimizando el impacto en los usuarios.

Monitoreo Post-Implementación: Supervisa la aplicación para confirmar que la corrección sea efectiva y estable.