

Casos de Prueba Funcionales

Microservicio ISO/IEC 25010

Proyecto: Microservicio para Evaluación de Calidad de Software

Versión: 1.0.0

Fecha: Diciembre 2024

Autor: Estudiante Universidad Mariano Gálvez

Introducción

Este documento contiene los casos de prueba funcionales diseñados para validar el correcto funcionamiento del microservicio desarrollado con Spring Boot. Los casos de prueba cubren las operaciones CRUD principales y las funcionalidades de negocio más críticas del sistema.

Objetivo

Verificar que todas las funcionalidades del microservicio cumplan con los requisitos especificados y mantengan la integridad de los datos durante las operaciones.

Alcance

Los casos de prueba incluyen: - Gestión de usuarios (CRUD) - Gestión de productos (CRUD) - Gestión de pedidos (CRUD) - Validaciones de negocio - Manejo de errores

Entorno de Pruebas

- **URL Base:** http://localhost:8080/api
- **Base de Datos:** H2 en memoria
- **Herramientas:** Postman, Swagger UI, cURL

- **Datos:** Conjunto de datos de prueba predefinidos
-

Caso de Prueba CP001: Crear Usuario Válido

Información General

- **ID:** CP001
- **Nombre:** Crear Usuario con Datos Válidos
- **Módulo:** Gestión de Usuarios
- **Prioridad:** Alta
- **Tipo:** Funcional Positiva

Descripción

Verificar que el sistema permita crear un nuevo usuario cuando se proporcionan todos los datos obligatorios con formato válido.

Precondiciones

- El microservicio está ejecutándose
- La base de datos está inicializada
- El email a utilizar no existe en el sistema

Datos de Entrada

```
{  
  "nombre": "Pedro",  
  "apellido": "González",  
  "email": "pedro.gonzalez@test.com",  
  "telefono": "50212345678"  
}
```

Pasos de Ejecución

1. Enviar request POST a `/api/usuarios`

2. Incluir los datos de entrada en el body como JSON
3. Establecer header `Content-Type: application/json`
4. Ejecutar la petición

Resultado Esperado

- **Código de Respuesta:** 201 Created
- **Body de Respuesta:** `json { "id": [número generado], "nombre": "Pedro", "apellido": "González", "email": "pedro.gonzalez@test.com", "telefono": "50212345678", "activo": true, "fechaCreacion": "[timestamp]", "fechaActualizacion": null, "nombreCompleto": "Pedro González", "totalPedidos": 0 }`
- **Validaciones:**
 - El usuario se crea con ID único
 - El campo `activo` se establece en `true` por defecto
 - La `fechaCreacion` se establece automáticamente
 - El `nombreCompleto` se genera correctamente

Criterios de Aceptación

- ☒ El usuario se crea exitosamente
- ☒ Se retorna código HTTP 201
- ☒ Los datos se almacenan correctamente en la base de datos
- ☒ Se pueden consultar los datos del usuario creado

Estado

- ☐ Pendiente
 - ☐ En Ejecución
 - ☐ Ejecutado
 - ☐ Aprobado
 - ☐ Rechazado
-

Caso de Prueba CP002: Crear Producto y Validar Stock

Información General

- **ID:** CP002
- **Nombre:** Crear Producto con Stock y Validar Disponibilidad
- **Módulo:** Gestión de Productos
- **Prioridad:** Alta
- **Tipo:** Funcional Positiva

Descripción

Verificar que el sistema permita crear un producto con stock inicial y que las consultas de disponibilidad funcionen correctamente.

Precondiciones

- El microservicio está ejecutándose
- Se tiene acceso a los endpoints de productos

Datos de Entrada

```
{
  "nombre": "Smartphone Test Pro",
  "descripcion": "Teléfono de prueba con características avanzadas",
  "precio": 1599.99,
  "stock": 50,
  "categoria": "Electrónicos",
  "marca": "TestBrand"
}
```

Pasos de Ejecución

1. **Crear Producto:**
2. Enviar POST a `/api/productos` con los datos de entrada
3. Verificar respuesta exitosa

4. Validar Creación:

5. Enviar GET a `/api/productos/{id}` con el ID retornado
6. Verificar que los datos coincidan

7. Consultar Productos con Stock:

8. Enviar GET a `/api/productos/con-stock`
9. Verificar que el producto aparezca en la lista

10. Validar Stock Específico:

11. Verificar que el campo `stock` sea 50
12. Confirmar que el producto está activo

Resultado Esperado

- **Creación:** Código 201 con datos del producto
- **Consulta Individual:** Código 200 con datos completos
- **Lista con Stock:** El producto aparece en la respuesta
- **Validaciones de Stock:** Stock = 50, activo = true

Criterios de Aceptación

- ☒ El producto se crea con todos los campos correctos
- ☒ El stock inicial se establece correctamente
- ☒ El producto aparece en consultas de productos con stock
- ☒ Los cálculos de precio son precisos (BigDecimal)

Estado

- ☐ Pendiente
- ☐ En Ejecución
- ☐ Ejecutado
- ☐ Aprobado

- [] Rechazado

Caso de Prueba CP003: Crear Pedido y Gestionar Stock

Información General

- **ID:** CP003
- **Nombre:** Crear Pedido y Validar Reducción de Stock
- **Módulo:** Gestión de Pedidos
- **Prioridad:** Crítica
- **Tipo:** Funcional de Integración

Descripción

Verificar que al crear un pedido, el sistema reduzca automáticamente el stock del producto y calcule correctamente el total del pedido.

Precondiciones

- Existe un usuario con ID válido
- Existe un producto con stock disponible (mínimo 5 unidades)
- El microservicio está funcionando correctamente

Datos de Entrada

```
{
  "usuarioId": 1,
  "productoId": 1,
  "cantidad": 3,
  "observaciones": "Pedido de prueba para validación de stock"
}
```

Pasos de Ejecución

1. Consultar Stock Inicial:

2. GET `/api/productos/1`
3. Anotar el stock actual
4. **Crear Pedido:**
5. POST `/api/pedidos` con los datos de entrada
6. Verificar respuesta exitosa
7. **Validar Pedido Creado:**
8. Verificar que el total se calculó correctamente
9. Confirmar que el estado inicial es "PENDIENTE"
10. **Verificar Reducción de Stock:**
11. GET `/api/productos/1`
12. Confirmar que el stock se redujo en 3 unidades
13. **Consultar Pedido:**
14. GET `/api/pedidos/{id}`
15. Verificar todos los campos del pedido

Resultado Esperado

- **Stock Inicial:** Ejemplo: 25 unidades
- **Pedido Creado:**

```
json { "id": [generado], "usuario": { "id": 1, ... },  
"producto": { "id": 1, ... }, "cantidad": 3, "precioUnitario":  
[precio del producto], "total": [precioUnitario * 3], "estado":  
"PENDIENTE", "observaciones": "Pedido de prueba para validación de  
stock", "fechaPedido": "[timestamp]" }
```
- **Stock Final:** 22 unidades (25 - 3)

Criterios de Aceptación

-  El pedido se crea exitosamente

- ☒ El stock se reduce automáticamente
- ☒ El total se calcula correctamente
- ☒ Las relaciones entre entidades funcionan
- ☒ La fecha de pedido se establece automáticamente

Estado

- ☐ Pendiente
 - ☐ En Ejecución
 - ☐ Ejecutado
 - ☐ Aprobado
 - ☐ Rechazado
-

Caso de Prueba CP004: Validar Error de Stock Insuficiente

Información General

- **ID:** CP004
- **Nombre:** Validar Error al Crear Pedido con Stock Insuficiente
- **Módulo:** Gestión de Pedidos
- **Prioridad:** Alta
- **Tipo:** Funcional Negativa

Descripción

Verificar que el sistema rechace la creación de un pedido cuando la cantidad solicitada excede el stock disponible del producto.

Precondiciones

- Existe un producto con stock limitado (ejemplo: 2 unidades)

- Existe un usuario válido
- El sistema de validaciones está activo

Datos de Entrada

```
{
  "usuarioId": 1,
  "productoId": [ID de producto con stock = 2],
  "cantidad": 5,
  "observaciones": "Pedido que debe fallar por stock insuficiente"
}
```

Pasos de Ejecución

1. **Identificar Producto con Stock Bajo:**
2. GET `/api/productos/sin-stock` o consultar productos
3. Seleccionar un producto con stock menor a 5
4. **Intentar Crear Pedido:**
5. POST `/api/pedidos` con cantidad mayor al stock
6. Capturar la respuesta de error
7. **Validar Mensaje de Error:**
8. Verificar código de respuesta HTTP
9. Validar estructura del mensaje de error
10. **Confirmar Stock No Modificado:**
11. GET `/api/productos/{id}`
12. Verificar que el stock no cambió

Resultado Esperado

- **Código de Respuesta:** 400 Bad Request
- **Body de Error:** json { "timestamp": "[timestamp]", "status": 400, "error": "Bad Request", "message": "Stock insuficiente. Stock

```
disponible: 2", "path": "/api/pedidos" }
```

- **Stock del Producto:** Sin cambios

Criterios de Aceptación

- ☒ El pedido NO se crea
- ☒ Se retorna código HTTP 400
- ☒ El mensaje de error es claro y específico
- ☒ El stock del producto permanece inalterado
- ☒ No se crean registros huérfanos en la base de datos

Estado

- ☐ Pendiente
 - ☐ En Ejecución
 - ☐ Ejecutado
 - ☐ Aprobado
 - ☐ Rechazado
-

Caso de Prueba CP005: Cambiar Estado de Pedido y Validar Flujo

Información General

- **ID:** CP005
- **Nombre:** Cambiar Estado de Pedido a través del Flujo Completo
- **Módulo:** Gestión de Pedidos
- **Prioridad:** Alta
- **Tipo:** Funcional de Flujo de Trabajo

Descripción

Verificar que el sistema permita cambiar el estado de un pedido siguiendo el flujo de negocio correcto desde PENDIENTE hasta ENTREGADO.

Precondiciones

- Existe un pedido en estado PENDIENTE
- El pedido tiene ID válido y accesible
- Todos los endpoints de cambio de estado están disponibles

Datos de Entrada

- **ID del Pedido:** [ID de pedido existente en estado PENDIENTE]
- **Estados a Probar:** CONFIRMADO → EN_PROCESO → ENVIADO → ENTREGADO

Pasos de Ejecución

1. **Consultar Estado Inicial:**
2. GET `/api/pedidos/{id}`
3. Verificar que el estado sea "PENDIENTE"
4. **Cambiar a CONFIRMADO:**
5. PATCH `/api/pedidos/{id}/estado?estado=CONFIRMADO`
6. Verificar respuesta exitosa
7. **Cambiar a EN_PROCESO:**
8. PATCH `/api/pedidos/{id}/estado?estado=EN_PROCESO`
9. Verificar transición correcta
10. **Cambiar a ENVIADO:**
11. PATCH `/api/pedidos/{id}/estado?estado=ENVIADO`
12. Validar cambio de estado

13. Cambiar a ENTREGADO:

14. PATCH `/api/pedidos/{id}/estado?estado=ENTREGADO`

15. Verificar estado final y fecha de entrega

16. Validar Estado Final:






17. GET `/api/pedidos/{id}`

18. Confirmar todos los campos actualizados

Resultado Esperado

- **Cada Cambio de Estado:**
- Código 200 OK
- Estado actualizado correctamente
- `fechaActualizacion` modificada
- **Estado Final ENTREGADO:** `json { "id": [ID del pedido], "estado": "ENTREGADO", "fechaEntrega": "[timestamp automático]", "fechaActualizacion": "[timestamp]", ... }`

Criterios de Aceptación

-  Todos los cambios de estado son exitosos
-  La `fechaActualizacion` se modifica en cada cambio
-  Al llegar a ENTREGADO, se establece `fechaEntrega`
-  El pedido mantiene integridad de datos
-  Las consultas posteriores reflejan el estado correcto

Estado

- `[]` Pendiente
- `[]` En Ejecución
- `[]` Ejecutado

- ☐ Aprobado
- ☐ Rechazado

Resumen de Casos de Prueba

ID	Nombre	Módulo	Tipo	Prioridad	Estado
CP001	Crear Usuario Válido	Usuarios	Positiva	Alta	Pendiente
CP002	Crear Producto y Validar Stock	Productos	Positiva	Alta	Pendiente
CP003	Crear Pedido y Gestionar Stock	Pedidos	Integración	Crítica	Pendiente
CP004	Validar Error Stock Insuficiente	Pedidos	Negativa	Alta	Pendiente
CP005	Cambiar Estado de Pedido	Pedidos	Flujo	Alta	Pendiente

Matriz de Cobertura

Funcionalidades Cubiertas

- ☒ Creación de entidades (Usuario, Producto, Pedido)
- ☒ Validaciones de datos de entrada
- ☒ Gestión automática de stock
- ☒ Cálculos de totales y precios
- ☒ Flujo de estados de pedidos
- ☒ Manejo de errores y excepciones
- ☒ Integridad referencial entre entidades

Tipos de Prueba

- **Funcionales Positivas:** 60% (3/5)
- **Funcionales Negativas:** 20% (1/5)
- **Pruebas de Integración:** 20% (1/5)

Módulos Cubiertos

- **Gestión de Usuarios:** 20% (1/5)
- **Gestión de Productos:** 20% (1/5)
- **Gestión de Pedidos:** 60% (3/5)

Criterios de Éxito del Proyecto

Para considerar exitosa la implementación del microservicio, todos los casos de prueba deben:

1. **Ejecutarse sin errores técnicos**
2. **Cumplir con los criterios de aceptación definidos**
3. **Mantener la integridad de los datos**
4. **Proporcionar respuestas consistentes y predecibles**
5. **Manejar errores de manera elegante y informativa**

Recomendaciones para Ejecución

Herramientas Recomendadas

- **Postman:** Para ejecución manual e interactiva
- **Swagger UI:** Para exploración y pruebas rápidas
- **cURL:** Para automatización y scripts
- **H2 Console:** Para verificación de datos

Orden de Ejecución Sugerido

1. CP001 - Crear Usuario (base para otros casos)
2. CP002 - Crear Producto (necesario para pedidos)
3. CP003 - Crear Pedido (funcionalidad principal)
4. CP004 - Validar Errores (casos negativos)

5. CP005 - Flujo de Estados (proceso completo)

Datos de Prueba

Utilizar los datos precargados en `data.sql` o crear datos específicos para cada caso según sea necesario.

Documento generado para: Universidad Mariano Gálvez

Curso: Aseguramiento de la Calidad de Software

Proyecto: Microservicio ISO/IEC 25010

Fecha: Diciembre 2024