

Universidad Mariano Gálvez
Sede San José Pinula
Aseguramiento de la Calidad de Software



Proyecto Final – Primera Parte

José Ricardo Pineda Godínez 3490-21-17051
Diana Gabriel Tepeque Jerez 3490-21-10938
Eddy Alexander Ramirez 3490-20-17355
Elder Ramiro Ixcolpal Arroyo 3490-17-1170

Introducción

En el presente documento se reúne de manera integral la documentación técnica, los casos de prueba y la evaluación de calidad del microservicio desarrollado con el framework Spring Boot, diseñado para la gestión de usuarios, productos y pedidos. El proyecto se elaboró siguiendo los lineamientos de la norma ISO/IEC 25010, la cual establece un modelo de calidad ampliamente reconocido a nivel internacional para la evaluación de sistemas y software.

El propósito principal de este trabajo es ofrecer una visión completa del ciclo de desarrollo y aseguramiento de la calidad, abarcando desde la definición y funcionamiento de la API REST hasta la verificación y validación del sistema mediante casos de prueba funcionales. Asimismo, se incluye un análisis detallado de las características de calidad según ISO/IEC 25010, lo que permite identificar fortalezas, debilidades y áreas prioritarias de mejora.

La organización del documento responde a un enfoque por secciones que facilita la lectura y consulta: en la primera sección se presenta la documentación técnica de la API, donde se describen los endpoints, parámetros, formatos de respuesta y consideraciones técnicas para su uso. En la segunda sección se incluyen los casos de prueba, diseñados para validar los flujos principales y garantizar que el sistema cumple con los requisitos funcionales y no funcionales definidos. Finalmente, en la tercera sección se expone la evaluación de calidad conforme a la norma ISO/IEC 25010, detallando la calificación obtenida en cada característica, junto con el análisis de resultados y recomendaciones.

De esta manera, el documento no solo evidencia la implementación de buenas prácticas en desarrollo y aseguramiento de la calidad, sino que también constituye un insumo valioso para la toma de decisiones técnicas, la planificación de mejoras y la preparación del sistema para un entorno productivo.

Documentación de API

Microservicio ISO/IEC 25010

Proyecto: Microservicio para Evaluación de Calidad de Software

Versión: 1.0.0

Fecha: Agosto 2025

Autor: Estudiante Universidad Mariano Gálvez

Información General

Esta documentación describe los endpoints disponibles en el microservicio desarrollado con Spring Boot para la evaluación de calidad según la norma ISO/IEC 25010. La API sigue los principios REST y utiliza JSON como formato de intercambio de datos.

URL Base

```
http://localhost:8080/api
```

Autenticación

La API actualmente no implementa autenticación. Todos los endpoints son públicos.

Formatos de Respuesta

- **Éxito:** Código HTTP 2xx con cuerpo JSON
- **Error Cliente:** Código HTTP 4xx con cuerpo JSON de error
- **Error Servidor:** Código HTTP 5xx con cuerpo JSON de error

Formato de Error

```
{
  "timestamp": "2024-12-15T14:30:45.123Z",
  "status": 400,
  "error": "Bad Request",
  "message": "Descripción del error",
  "path": "/api/endpoint",
  "details": ["Error específico 1", "Error específico 2"]
}
```

Endpoints de Usuarios

Listar Todos los Usuarios

Endpoint: GET /usuarios

Descripción: Retorna una lista con todos los usuarios registrados en el sistema.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
[
  {
    "id": 1,
    "nombre": "Juan Carlos",
    "apellido": "García López",
    "email": "juan.garcia@email.com",
    "telefono": "50212345678",
    "activo": true,
    "fechaCreacion": "2024-12-15T10:30:45.123Z",
    "fechaActualizacion": null,
    "nombreCompleto": "Juan Carlos García López",
    "totalPedidos": 2
  },
  {
    "id": 2,
    "nombre": "María Elena",
    "apellido": "Rodríguez Pérez",
    "email": "maria.rodriguez@email.com",
    "telefono": "50298765432",
    "activo": true,
    "fechaCreacion": "2024-12-15T11:20:15.456Z",
    "fechaActualizacion": null,
    "nombreCompleto": "María Elena Rodríguez Pérez",
    "totalPedidos": 1
  }
]
```

Obtener Usuario por ID

Endpoint: GET /usuarios/{id}

Descripción: Retorna un usuario específico basado en su ID.

Parámetros: - {id} - ID único del usuario (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
{
  "id": 1,
  "nombre": "Juan Carlos",
  "apellido": "García López",
  "email": "juan.garcia@email.com",
  "telefono": "50212345678",
  "activo": true,
  "fechaCreacion": "2024-12-15T10:30:45.123Z",
  "fechaActualizacion": null,
  "nombreCompleto": "Juan Carlos García López",
  "totalPedidos": 2
}
```

Respuesta de Error: - **Código:** 404 Not Found - **Contenido:**

```
{
  "timestamp": "2024-12-15T14:30:45.123Z",
  "status": 404,
  "error": "Not Found",
  "message": "Usuario no encontrado con ID: 999",
  "path": "/api/usuarios/999"
}
```

Crear Usuario

Endpoint: POST /usuarios

Descripción: Crea un nuevo usuario en el sistema con los datos proporcionados.

Cuerpo de la Solicitud:

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

Respuesta Exitosa: - Código: 201 Created - Contenido:

```
{
  "id": 11,
  "nombre": "Pedro",
  "apellido": "González",
  "email": "pedro.gonzalez@test.com",
  "telefono": "50212345678",
  "activo": true,
  "fechaCreacion": "2024-12-15T15:45:30.789Z",
  "fechaActualizacion": null,
  "nombreCompleto": "Pedro González",
  "totalPedidos": 0
}
```

Respuesta de Error: - Código: 400 Bad Request - Contenido:

```
{
  "timestamp": "2024-12-15T15:45:30.789Z",
  "status": 400,
  "error": "Bad Request",
  "message": "Error de validación en los datos enviados",
  "path": "/api/usuarios",
  "details": ["email: El formato del email no es válido"]
}
```

Actualizar Usuario

Endpoint: PUT /usuarios/{id}

Descripción: Actualiza los datos de un usuario existente.

Parámetros: - {id} - ID único del usuario (Path Parameter)

Cuerpo de la Solicitud:

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

Respuesta Exitosa: - Código: 200 OK - Contenido:

```
{
  "id": 11,
  "nombre": "Pedro Luis",
  "apellido": "González Ramírez",
  "email": "pedro.gonzalez@test.com",
  "telefono": "50212345678",
  "activo": true,
  "fechaCreacion": "2024-12-15T15:45:30.789Z",
  "fechaActualizacion": "2024-12-15T16:20:15.456Z",
  "nombreCompleto": "Pedro Luis González Ramírez",
  "totalPedidos": 0
}
```

Eliminar Usuario

Endpoint: DELETE /usuarios/{id}

Descripción: Elimina un usuario del sistema de forma permanente.

Parámetros: - {id} - ID único del usuario (Path Parameter)

Respuesta Exitosa: - **Código:** 204 No Content

Buscar Usuario por Email

Endpoint: GET /usuarios/email/{email}

Descripción: Busca un usuario específico por su dirección de email.

Parámetros: - {email} - Email del usuario (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto usuario (igual que en GET /usuarios/{id})

Buscar Usuarios por Nombre

Endpoint: GET /usuarios/buscar?nombre={nombre}

Descripción: Busca usuarios que contengan el texto especificado en su nombre.

Parámetros: - nombre - Texto a buscar en el nombre (Query Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de usuarios (igual que en GET /usuarios)

Obtener Usuarios Activos

Endpoint: GET /usuarios/activos

Descripción: Retorna todos los usuarios que están activos en el sistema.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de usuarios activos

Obtener Usuarios Inactivos

Endpoint: GET /usuarios/inactivos

Descripción: Retorna todos los usuarios que están inactivos en el sistema.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de usuarios inactivos

Activar Usuario

Endpoint: PATCH /usuarios/{id}/activar

Descripción: Activa un usuario que estaba previamente inactivo.

Parámetros: - {id} - ID único del usuario (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto usuario actualizado

Desactivar Usuario

Endpoint: PATCH /usuarios/{id}/desactivar

Descripción: Desactiva un usuario sin eliminarlo del sistema.

Parámetros: - {id} - ID único del usuario (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto usuario actualizado

Búsqueda Libre de Usuarios

Endpoint: GET /usuarios/buscar-texto?texto={texto}

Descripción: Busca usuarios por texto libre en nombre, apellido o email.

Parámetros: - texto - Texto a buscar (Query Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de usuarios que coinciden

Obtener Estadísticas de Usuarios

Endpoint: GET /usuarios/estadisticas

Descripción: Retorna estadísticas básicas sobre los usuarios del sistema.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
{
  "total": 10,
  "activos": 9,
  "inactivos": 1,
  "porcentajeActivos": 90.0
}
```

Endpoints de Productos

Listar Todos los Productos

Endpoint: GET /productos

Descripción: Retorna una lista con todos los productos del catálogo.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
[
  {
    "id": 1,
    "nombre": "Smartphone Samsung Galaxy A54",
    "descripcion": "Teléfono inteligente con pantalla AMOLED de 6.4 pulgadas...",
    "precio": 2499.99,
    "stock": 25,
    "categoria": "Electrónicos",
    "marca": "Samsung",
    "activo": true,
    "fechaCreacion": "2024-12-15T10:30:45.123Z",
    "fechaActualizacion": null
  },
  {
    "id": 2,
    "nombre": "Laptop Dell Inspiron 15",
    "descripcion": "Laptop con procesador Intel Core i5, 8GB RAM...",
    "precio": 4299.99,
    "stock": 15,
    "categoria": "Electrónicos",
    "marca": "Dell",
    "activo": true,
    "fechaCreacion": "2024-12-15T10:35:20.456Z",
    "fechaActualizacion": null
  }
]
```

Obtener Producto por ID

Endpoint: GET /productos/{id}

Descripción: Retorna un producto específico basado en su ID.

Parámetros: - {id} - ID único del producto (Path Parameter)

Respuesta Exitosa: - Código: 200 OK - Contenido:

```
{
  "id": 1,
  "nombre": "Smartphone Samsung Galaxy A54",
  "descripcion": "Teléfono inteligente con pantalla AMOLED de 6.4 pulgadas...",
  "precio": 2499.99,
  "stock": 25,
  "categoria": "Electrónicos",
  "marca": "Samsung",
  "activo": true,
  "fechaCreacion": "2024-12-15T10:30:45.123Z",
  "fechaActualizacion": null
}
```

Crear Producto

Endpoint: POST /productos

Descripción: Crea un nuevo producto en el catálogo con los datos proporcionados.

Cuerpo de la Solicitud:

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

Respuesta Exitosa: - **Código:** 201 Created - **Contenido:** Objeto producto creado

Actualizar Producto

Endpoint: PUT /productos/{id}

Descripción: Actualiza los datos de un producto existente.

Parámetros: - {id} - ID único del producto (Path Parameter)

Cuerpo de la Solicitud:

```
{
  "nombre": "Smartphone Test Pro Plus",
  "descripcion": "Versión mejorada del teléfono de prueba",
  "precio": 1799.99,
  "stock": 45,
  "categoria": "Electrónicos",
  "marca": "TestBrand"
}
```

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto producto actualizado

Eliminar Producto

Endpoint: DELETE /productos/{id}

Descripción: Elimina un producto del catálogo de forma permanente.

Parámetros: - `{id}` - ID único del producto (Path Parameter)

Respuesta Exitosa: - **Código:** 204 No Content

Buscar Productos por Nombre

Endpoint: `GET /productos/buscar?nombre={nombre}`

Descripción: Busca productos que contengan el texto especificado en su nombre.

Parámetros: - `nombre` - Texto a buscar en el nombre (Query Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos que coinciden

Buscar Productos por Categoría

Endpoint: `GET /productos/categoria/{categoria}`

Descripción: Retorna todos los productos de una categoría específica.

Parámetros: - `{categoria}` - Categoría del producto (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos de la categoría

Buscar Productos por Marca

Endpoint: `GET /productos/marca/{marca}`

Descripción: Retorna todos los productos de una marca específica.

Parámetros: - `{marca}` - Marca del producto (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos de la marca

Obtener Productos Activos

Endpoint: `GET /productos/activos`

Descripción: Retorna todos los productos que están activos en el catálogo.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos activos

Obtener Productos con Stock

Endpoint: GET /productos/con-stock

Descripción: Retorna todos los productos que tienen stock disponible.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos con stock > 0

Obtener Productos sin Stock

Endpoint: GET /productos/sin-stock

Descripción: Retorna todos los productos que no tienen stock disponible.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos con stock = 0

Buscar Productos por Rango de Precios

Endpoint: GET /productos/precio?precioMinimo={min}&precioMaximo={max}

Descripción: Busca productos dentro de un rango de precios específico.

Parámetros: - precioMinimo - Precio mínimo (Query Parameter) - precioMaximo - Precio máximo (Query Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos en el rango de precios

Búsqueda Libre de Productos

Endpoint: GET /productos/buscar-texto?texto={texto}

Descripción: Busca productos por texto libre en nombre, descripción, categoría o marca.

Parámetros: - texto - Texto a buscar (Query Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de productos que coinciden

Obtener Todas las Categorías

Endpoint: GET /productos/categorias

Descripción: Retorna una lista con todas las categorías de productos disponibles.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
[  
  "Electrónicos",  
  "Hogar y Jardín",  
  "Deportes y Fitness",  
  "Libros y Educación"  
]
```

Obtener Todas las Marcas

Endpoint: GET /productos/marcas

Descripción: Retorna una lista con todas las marcas de productos disponibles.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de strings con nombres de marcas

Actualizar Stock de Producto

Endpoint: PATCH /productos/{id}/stock?stock={stock}

Descripción: Actualiza la cantidad de stock disponible de un producto.

Parámetros: - {id} - ID único del producto (Path Parameter) - stock - Nuevo stock (Query Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto producto actualizado

Activar Producto

Endpoint: PATCH /productos/{id}/activar

Descripción: Activa un producto que estaba previamente inactivo.

Parámetros: - {id} - ID único del producto (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto producto actualizado

Desactivar Producto

Endpoint: PATCH /productos/{id}/desactivar

Descripción: Desactiva un producto sin eliminarlo del catálogo.

Parámetros: - {id} - ID único del producto (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto producto actualizado

Endpoints de Pedidos

Listar Todos los Pedidos

Endpoint: GET /pedidos

Descripción: Retorna una lista con todos los pedidos del sistema.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
[
  {
    "id": 1,
    "usuario": {
      "id": 1,
      "nombre": "Juan Carlos",
      "apellido": "García López",
      "email": "juan.garcia@email.com"
    },
    "producto": {
      "id": 1,
      "nombre": "Smartphone Samsung Galaxy A54",
      "precio": 2499.99
    },
    "cantidad": 2,
    "precioUnitario": 2499.99,
    "total": 4999.98,
    "estado": "PENDIENTE",
    "observaciones": "Pedido para regalo de cumpleaños",
    "fechaPedido": "2024-12-15T10:40:15.789Z",
    "fechaEntrega": null,
    "fechaActualizacion": null
  },
  {
    "id": 2,
    "usuario": {
      "id": 2,
      "nombre": "María Elena",
      "apellido": "Rodríguez Pérez",
      "email": "maria.rodriguez@email.com"
    },
    "producto": {
      "id": 5,
      "nombre": "Smart TV LG 55 pulgadas",
      "precio": 5999.99
    },
    "cantidad": 1,
    "precioUnitario": 5999.99,
    "total": 5999.99,
    "estado": "PENDIENTE",
    "observaciones": "Entrega preferiblemente en horario matutino",
    "fechaPedido": "2024-12-15T11:20:30.456Z",
    "fechaEntrega": null,
    "fechaActualizacion": null
  }
]
```

Obtener Pedido por ID

Endpoint: GET /pedidos/{id}

Descripción: Retorna un pedido específico basado en su ID.

Parámetros: - {id} - ID único del pedido (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto pedido detallado

Crear Pedido

Endpoint: POST /pedidos

Descripción: Crea un nuevo pedido con los datos proporcionados.

Cuerpo de la Solicitud:

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

Respuesta Exitosa: - **Código:** 201 Created - **Contenido:** Objeto pedido creado

Respuesta de Error: - **Código:** 400 Bad Request - **Contenido:**

```
{
  "timestamp": "2024-12-15T15:45:30.789Z",
  "status": 400,
  "error": "Bad Request",
  "message": "Stock insuficiente. Stock disponible: 2",
  "path": "/api/pedidos"
}
```

Actualizar Pedido

Endpoint: PUT /pedidos/{id}

Descripción: Actualiza los datos de un pedido existente.

Parámetros: - {id} - ID único del pedido (Path Parameter)

Cuerpo de la Solicitud:

```
{
  "usuarioId": 1,
  "productoId": 2,
  "cantidad": 1,
  "observaciones": "Pedido actualizado"
}
```

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto pedido actualizado

Eliminar Pedido

Endpoint: DELETE /pedidos/{id}

Descripción: Elimina un pedido del sistema y restaura el stock.

Parámetros: - {id} - ID único del pedido (Path Parameter)

Respuesta Exitosa: - **Código:** 204 No Content

Obtener Pedidos por Usuario

Endpoint: GET /pedidos/usuario/{usuarioId}

Descripción: Retorna todos los pedidos de un usuario específico.

Parámetros: - {usuarioId} - ID del usuario (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de pedidos del usuario

Obtener Pedidos por Producto

Endpoint: GET /pedidos/producto/{productoId}

Descripción: Retorna todos los pedidos de un producto específico.

Parámetros: - {productoId} - ID del producto (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de pedidos del producto

Obtener Pedidos por Estado

Endpoint: GET /pedidos/estado/{estado}

Descripción: Retorna todos los pedidos con un estado específico.

Parámetros: - {estado} - Estado del pedido (Path Parameter) - Valores válidos: PENDIENTE, CONFIRMADO, EN_PROCESO, ENVIADO, ENTREGADO, CANCELADO

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Array de pedidos con el estado especificado

Cambiar Estado de Pedido

Endpoint: PATCH /pedidos/{id}/estado?estado={estado}

Descripción: Actualiza el estado de un pedido específico.

Parámetros: - {id} - ID único del pedido (Path Parameter) - estado - Nuevo estado (Query Parameter) - Valores válidos: PENDIENTE, CONFIRMADO, EN_PROCESO, ENVIADO, ENTREGADO, CANCELADO

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto pedido actualizado

Cancelar Pedido

Endpoint: PATCH /pedidos/{id}/cancelar

Descripción: Cancela un pedido y restaura el stock del producto.

Parámetros: - {id} - ID único del pedido (Path Parameter)

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:** Objeto pedido cancelado

Respuesta de Error: - **Código:** 400 Bad Request - **Contenido:**

```
{
  "timestamp": "2024-12-15T16:30:45.123Z",
  "status": 400,
  "error": "Bad Request",
  "message": "El pedido no puede ser cancelado. Estado actual: ENTREGADO",
  "path": "/api/pedidos/5/cancelar"
}
```

Obtener Estadísticas de Pedidos

Endpoint: GET /pedidos/estadisticas

Descripción: Retorna estadísticas generales sobre los pedidos del sistema.

Parámetros: Ninguno

Respuesta Exitosa: - **Código:** 200 OK - **Contenido:**

```
{
  "total": 16,
  "pendientes": 3,
  "confirmados": 3,
  "entregados": 4,
  "cancelados": 2,
  "ventasTotal": 26699.85
}
```

Modelos de Datos

Usuario

```
{
  "id": 1,
  "nombre": "Juan Carlos",
  "apellido": "García López",
  "email": "juan.garcia@email.com",
  "telefono": "50212345678",
  "activo": true,
  "fechaCreacion": "2024-12-15T10:30:45.123Z",
  "fechaActualizacion": null,
  "nombreCompleto": "Juan Carlos García López",
  "totalPedidos": 2
}
```

Producto

```
{
  "id": 1,
  "nombre": "Smartphone Samsung Galaxy A54",
  "descripcion": "Teléfono inteligente con pantalla AMOLED de 6.4 pulgadas...",
  "precio": 2499.99,
  "stock": 25,
  "categoria": "Electrónicos",
  "marca": "Samsung",
  "activo": true,
  "fechaCreacion": "2024-12-15T10:30:45.123Z",
  "fechaActualizacion": null
}
```

Pedido

```
{
  "id": 1,
  "usuario": {
    "id": 1,
    "nombre": "Juan Carlos",
    "apellido": "García López",
    "email": "juan.garcia@email.com"
  },
  "producto": {
    "id": 1,
    "nombre": "Smartphone Samsung Galaxy A54",
    "precio": 2499.99
  },
  "cantidad": 2,
  "precioUnitario": 2499.99,
  "total": 4999.98,
  "estado": "PENDIENTE",
  "observaciones": "Pedido para regalo de cumpleaños",
  "fechaPedido": "2024-12-15T10:40:15.789Z",
  "fechaEntrega": null,
  "fechaActualizacion": null
}
```

Estados de Pedido

- **PENDIENTE:** Pedido recién creado, pendiente de confirmación
- **CONFIRMADO:** Pedido confirmado, listo para procesamiento
- **EN_PROCESO:** Pedido en proceso de preparación
- **ENVIADO:** Pedido enviado al cliente
- **ENTREGADO:** Pedido entregado exitosamente
- **CANCELADO:** Pedido cancelado

Códigos de Estado HTTP

- **200 OK:** Operación exitosa
- **201 Created:** Recurso creado exitosamente
- **204 No Content:** Operación exitosa sin contenido de respuesta
- **400 Bad Request:** Error en los datos enviados
- **404 Not Found:** Recurso no encontrado

- **409 Conflict:** Conflicto con el estado actual del recurso
 - **500 Internal Server Error:** Error interno del servidor
-

Herramientas de Desarrollo

Swagger UI

La documentación interactiva de la API está disponible en:

```
http://localhost:8080/api/swagger-ui.html
```

H2 Console

La consola de la base de datos H2 está disponible en:

```
http://localhost:8080/api/h2-console
```

Credenciales: - **JDBC URL:** jdbc:h2:mem:testdb - **Username:** sa - **Password:** password

Documento generado para: Universidad Mariano Gálvez

Curso: Aseguramiento de la Calidad de Software

Proyecto: Microservicio ISO/IEC 25010

Fecha: Diciembre 2024

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

Evaluación de Calidad ISO/IEC 25010

Microservicio Spring Boot

Proyecto: Microservicio para Evaluación de Calidad de Software

Versión: 1.0.0

Fecha: Diciembre 2024

Autor: Estudiante Universidad Mariano Gálvez

Norma Aplicada: ISO/IEC 25010:2011 - System and software quality models

Resumen Ejecutivo

Este documento presenta la evaluación de calidad del microservicio desarrollado con Spring Boot, aplicando los criterios establecidos en la norma ISO/IEC 25010. La evaluación abarca las ocho características principales de calidad del software y sus respectivas subcaracterísticas.

Resultado General

- Puntuación Global:** 4.2/5.0 (84%)
- Nivel de Calidad:** ALTO
- Recomendación:** APROBADO para producción con mejoras menores

Introducción a ISO/IEC 25010

La norma ISO/IEC 25010 define un modelo de calidad para sistemas y software que especifica ocho características de calidad principales:

- Adecuación Funcional (Functional Suitability)**
- Eficiencia de Desempeño (Performance Efficiency)**

- 3. **Compatibilidad (Compatibility)**
- 4. **Usabilidad (Usability)**
- 5. **Confiabilidad (Reliability)**
- 6. **Seguridad (Security)**
- 7. **Mantenibilidad (Maintainability)**
- 8. **Portabilidad (Portability)**

Metodología de Evaluación

Escala de Calificación: - **5 - Excelente:** Cumple completamente con los criterios, implementación sobresaliente - **4 - Bueno:** Cumple satisfactoriamente con la mayoría de criterios - **3 - Aceptable:** Cumple con los criterios básicos, funcional pero mejorable - **2 - Deficiente:** Cumple parcialmente, requiere mejoras significativas - **1 - Inadecuado:** No cumple con los criterios mínimos

1. Adecuación Funcional (Functional Suitability)

Definición


Grado en que el producto o sistema proporciona funciones que satisfacen las necesidades declaradas e implícitas cuando se usa en las condiciones especificadas.

Subcaracterísticas Evaluadas

1.1 Completitud Funcional

Puntuación: 5/5

Evaluación: El microservicio implementa completamente todas las funcionalidades requeridas:

 **Gestión de Usuarios:** - Operaciones CRUD completas (Create, Read, Update, Delete) - Búsquedas por diferentes criterios (email, nombre, texto libre) - Activación/desactivación de usuarios - Validaciones de datos de entrada

✓ **Gestión de Productos:** - Operaciones CRUD completas - Gestión de inventario (stock) - Búsquedas por categoría, marca, precio - Filtros por disponibilidad

✓ **Gestión de Pedidos:** - Creación y seguimiento de pedidos - Gestión de estados del pedido - Cálculo automático de totales - Integración con gestión de stock

Justificación: Todas las funcionalidades especificadas están implementadas y operativas.

1.2 Corrección Funcional

Puntuación: 4/5

Evaluación: Las funciones proporcionan los resultados correctos con el grado de precisión requerido:

✓ **Fortalezas:** - Cálculos precisos de totales usando BigDecimal - Validaciones de datos robustas - Manejo correcto de relaciones entre entidades - Gestión automática de stock

⚠ **Áreas de Mejora:** - Falta validación de algunos casos edge (ej: números negativos en algunos campos) - Podría mejorar la validación de formatos de teléfono

Justificación: Las funciones son correctas en la mayoría de casos, con oportunidades menores de mejora.

1.3 Pertinencia Funcional

Puntuación: 5/5

Evaluación: El sistema facilita el logro de tareas y objetivos específicos:

✓ **Características Destacadas:** - API RESTful intuitiva y bien estructurada - Endpoints especializados para diferentes necesidades - Funcionalidades de búsqueda y filtrado avanzadas - Estadísticas y reportes básicos

Justificación: Todas las funciones son pertinentes y contribuyen directamente a los objetivos del sistema.

Puntuación Total Adecuación Funcional: 4.7/5

2. Eficiencia de Desempeño (Performance Efficiency)

Definición

Desempeño relativo a la cantidad de recursos utilizados bajo condiciones determinadas.

Subcaracterísticas Evaluadas

2.1 Comportamiento Temporal

Puntuación: 4/5

Evaluación: Tiempos de respuesta, procesamiento y rendimiento del sistema:

✅ **Fortalezas:** - Consultas simples responden en < 100ms - Uso de índices en base de datos (JPA automático) - Operaciones CRUD optimizadas - Lazy loading en relaciones JPA

⚠️ **Consideraciones:** - Base de datos en memoria (H2) no representa entorno productivo - Falta implementación de caché - Sin optimización específica para consultas complejas

Justificación: Buen rendimiento para el alcance actual, pero requiere validación en entorno productivo.

2.2 Utilización de Recursos

Puntuación: 4/5

Evaluación: Cantidad y tipos de recursos utilizados:

✅ **Fortalezas:** - Uso eficiente de memoria con JPA - Gestión automática de conexiones de BD - Logging configurado apropiadamente - Sin memory leaks evidentes

⚠️ **Áreas de Mejora:** - Falta configuración de pool de conexiones - Sin métricas de monitoreo implementadas - Podría optimizar serialización JSON

Justificación: Uso razonable de recursos con oportunidades de optimización.

2.3 Capacidad

Puntuación: 3/5

Evaluación: Grado en que los límites máximos del sistema satisfacen los requisitos:

⚠ **Limitaciones Identificadas:** - No hay límites definidos para paginación - Sin throttling o rate limiting - Base de datos en memoria limita escalabilidad - No hay pruebas de carga implementadas

✅ **Aspectos Positivos:** - Arquitectura permite escalabilidad horizontal - Uso de Spring Boot facilita configuración de límites

Justificación: Capacidad adecuada para desarrollo, requiere mejoras para producción.

Puntuación Total Eficiencia de Desempeño: 3.7/5

3. Compatibilidad (Compatibility)

Definición

Grado en que un producto, sistema o componente puede intercambiar información con otros productos, sistemas o componentes.

Subcaracterísticas Evaluadas

3.1 Coexistencia

Puntuación: 4/5

Evaluación: Capacidad de coexistir con otros productos en un entorno común:

✅ **Fortalezas:** - Puerto configurable (8080 por defecto) - Uso de estándares web (HTTP, JSON, REST) - Compatible con proxies y load balancers - CORS habilitado para integración frontend

⚠ **Consideraciones:** - Dependencia de puerto específico - Sin configuración de múltiples perfiles de entorno

Justificación: Buena coexistencia con sistemas estándar.

3.2 Interoperabilidad

Puntuación: 5/5

Evaluación: Capacidad de intercambiar información y usar información intercambiada:

✓ **Excelente Implementación:** - API REST estándar con OpenAPI 3.0 - Formatos JSON estándar - Códigos de estado HTTP apropiados - Documentación completa con Swagger - Headers HTTP estándar

Justificación: Excelente interoperabilidad siguiendo estándares de la industria.

Puntuación Total Compatibilidad: 4.5/5

4. Usabilidad (Usability)

Definición

Grado en que un producto o sistema puede ser usado por usuarios específicos para lograr objetivos específicos con efectividad, eficiencia y satisfacción.

Subcaracterísticas Evaluadas

4.1 Reconocimiento de Idoneidad

Puntuación: 5/5

Evaluación: Los usuarios pueden reconocer si el software es apropiado para sus necesidades:

✓ **Excelente Documentación:** - Swagger UI interactivo y completo - Ejemplos de requests/responses - Descripción detallada de endpoints - Documentación de modelos de datos

Justificación: La documentación permite evaluar fácilmente la idoneidad del sistema.

4.2 Capacidad de Aprendizaje

Puntuación: 4/5

Evaluación: Facilidad para aprender a usar el sistema:

✅ **Fortalezas:** - API RESTful intuitiva - Nomenclatura clara y consistente - Ejemplos prácticos en documentación - Mensajes de error descriptivos

⚠️ **Mejoras Posibles:** - Falta tutorial paso a paso - Sin ejemplos de integración completos

Justificación: Fácil de aprender para desarrolladores con experiencia en REST APIs.

4.3 Operabilidad

Puntuación: 4/5

Evaluación: Facilidad de operación y control:

✅ **Aspectos Positivos:** - Endpoints intuitivos y bien organizados - Respuestas consistentes - Manejo de errores apropiado - Operaciones idempotentes donde corresponde

⚠️ **Áreas de Mejora:** - Sin interfaz de administración - Falta operaciones en lote (batch)

Justificación: Buena operabilidad para una API REST.

4.4 Protección contra Errores de Usuario

Puntuación: 4/5

Evaluación: Protección de usuarios contra cometer errores:

✅ **Implementado:** - Validaciones de entrada robustas - Mensajes de error claros y específicos - Códigos de estado HTTP apropiados - Validación de integridad referencial

⚠️ **Mejoras:** - Podría incluir sugerencias en mensajes de error - Sin confirmación para operaciones destructivas

Justificación: Buena protección con validaciones comprehensivas.

4.5 Estética de Interfaz de Usuario

Puntuación: 4/5

Evaluación: Interfaz agradable y satisfactoria (Swagger UI):

✓ **Características:** - Swagger UI moderno y responsive - Organización clara por tags - Colores y tipografía apropiados - Navegación intuitiva

Justificación: Interfaz de documentación atractiva y funcional.

4.6 Accesibilidad

Puntuación: 3/5

Evaluación: Uso por personas con diversas características y capacidades:

⚠ **Limitaciones:** - Sin consideraciones específicas de accesibilidad - Documentación solo en inglés técnico - Sin soporte para lectores de pantalla

✓ **Aspectos Positivos:** - API puede ser consumida por cualquier cliente - Formatos estándar (JSON) son accesibles

Justificación: Accesibilidad básica, sin características específicas implementadas.

Puntuación Total Usabilidad: 4.0/5

5. Confiabilidad (Reliability)

Definición

Grado en que un sistema, producto o componente desempeña funciones especificadas bajo condiciones especificadas durante un período de tiempo especificado.

Subcaracterísticas Evaluadas

5.1 Madurez

Puntuación: 4/5

Evaluación: El sistema satisface las necesidades de confiabilidad en operación normal:

✅ **Fortalezas:** - Framework Spring Boot maduro y estable - Manejo de excepciones comprehensivo - Transacciones de base de datos apropiadas - Logging estructurado para debugging

⚠️ **Consideraciones:** - Sistema nuevo sin historial de producción - Falta pruebas de estrés y carga

Justificación: Buena madurez basada en tecnologías probadas.

5.2 Disponibilidad

Puntuación: 3/5

Evaluación: El sistema está operacional y accesible cuando se requiere:

⚠️ **Limitaciones:** - Sin implementación de alta disponibilidad - Base de datos en memoria no persistente - Sin mecanismos de failover - Sin monitoreo de salud automatizado

✅ **Aspectos Positivos:** - Endpoints de health check disponibles - Reinicio rápido de aplicación - Arquitectura stateless

Justificación: Disponibilidad básica, requiere mejoras para producción.

5.3 Tolerancia a Fallos

Puntuación: 4/5

Evaluación: El sistema opera según lo previsto a pesar de fallos de hardware o software:

✅ **Implementado:** - Manejo global de excepciones - Validaciones de entrada robustas - Rollback automático de transacciones - Mensajes de error informativos

⚠ **Mejoras Posibles:** - Sin circuit breakers para servicios externos - Falta retry mechanisms - Sin degradación elegante

Justificación: Buena tolerancia a fallos a nivel de aplicación.

5.4 Capacidad de Recuperación

Puntuación: 3/5

Evaluación: Capacidad de recuperarse de interrupciones o fallos:

⚠ **Limitaciones:** - Datos en memoria se pierden al reiniciar - Sin backups automatizados - Sin procedimientos de recuperación documentados

✅ **Aspectos Positivos:** - Reinicio rápido de aplicación - Recarga automática de datos iniciales - Estado de aplicación se reconstruye automáticamente

Justificación: Recuperación básica, limitada por base de datos en memoria.

Puntuación Total Confiabilidad: 3.5/5

6. Seguridad (Security)

Definición

Grado en que un producto o sistema protege información y datos para que personas o sistemas tengan el grado de acceso a datos apropiado a sus tipos y niveles de autorización.

Subcaracterísticas Evaluadas

6.1 Confidencialidad

Puntuación: 2/5

Evaluación: El sistema asegura que los datos sean accesibles solo a aquellos autorizados:

⚠ **Deficiencias Críticas:** - Sin autenticación implementada - Todos los endpoints son públicos - Sin encriptación de datos sensibles - Datos de usuarios expuestos sin restricciones

✅ **Aspectos Básicos:** - HTTPS puede ser configurado a nivel de infraestructura - Estructura preparada para implementar seguridad

Justificación: Confidencialidad inadecuada para entorno productivo.

6.2 Integridad

Puntuación: 4/5

Evaluación: El sistema previene acceso o modificación no autorizada:

✅ **Fortalezas:** - Validaciones de entrada robustas - Integridad referencial en base de datos - Transacciones ACID - Validación de tipos de datos

⚠ **Mejoras Necesarias:** - Sin firma digital de datos - Sin checksums para verificación

Justificación: Buena integridad a nivel de datos y validaciones.

6.3 No Repudio

Puntuación: 2/5

Evaluación: Capacidad de demostrar que acciones o eventos han tenido lugar:

⚠ **Deficiencias:** - Sin logging de auditoría - Sin identificación de usuarios en logs - Sin timestamps seguros - Sin trazabilidad de cambios

✅ **Básico:** - Logs de aplicación básicos - Timestamps en entidades

Justificación: No repudio inadecuado para sistemas críticos.

6.4 Responsabilidad

Puntuación: 2/5

Evaluación: Capacidad de rastrear acciones de una entidad de manera única:

⚠ **Limitaciones:** - Sin identificación de usuarios - Sin logs de auditoría - Sin trazabilidad de operaciones

Justificación: Responsabilidad limitada sin sistema de autenticación.

6.5 Autenticidad

Puntuación: 1/5

Evaluación: Capacidad de probar la identidad de un sujeto o recurso:

✗ No Implementado: - Sin autenticación de usuarios - Sin autorización de endpoints
- Sin verificación de identidad

Justificación: Autenticidad no implementada.

Puntuación Total Seguridad: 2.2/5

7. Mantenibilidad (Maintainability)

Definición

Grado de efectividad y eficiencia con que un producto o sistema puede ser modificado por los mantenedores previstos.

Subcaracterísticas Evaluadas

7.1 Modularidad

Puntuación: 5/5

Evaluación: El sistema está compuesto de componentes discretos de tal manera que un cambio en un componente tiene impacto mínimo en otros:

✓ Excelente Arquitectura: - Separación clara en capas (Controller, Service, Repository) - Principio de responsabilidad única aplicado - Bajo acoplamiento entre componentes - Alta cohesión dentro de cada módulo - Inyección de dependencias bien implementada

Justificación: Arquitectura modular ejemplar siguiendo mejores prácticas.

7.2 Reusabilidad

Puntuación: 4/5

Evaluación: Grado en que un activo puede ser usado en más de un sistema:

✅ **Fortalezas:** - Servicios reutilizables - DTOs bien definidos - Repositorios genéricos - Configuraciones externalizadas

⚠️ **Mejoras:** - Podría extraer más utilidades comunes - Sin librerías compartidas

Justificación: Buena reusabilidad con oportunidades de mejora.

7.3 Analizabilidad

Puntuación: 5/5

Evaluación: Efectividad y eficiencia para evaluar el impacto de un cambio previsto:

✅ **Excelente:** - Código bien documentado - Estructura clara y consistente - Logging comprehensivo - Nombres descriptivos - Separación de responsabilidades clara

Justificación: Código altamente analizable y comprensible.

7.4 Capacidad de Modificación

Puntuación: 5/5

Evaluación: Grado en que un producto o sistema puede ser modificado efectiva y eficientemente:

✅ **Excelente Diseño:** - Arquitectura flexible - Configuración externalizada - Interfaces bien definidas - Principios SOLID aplicados - Fácil extensión de funcionalidades

Justificación: Sistema altamente modificable y extensible.

7.5 Capacidad de Prueba

Puntuación: 4/5

Evaluación: Efectividad y eficiencia para establecer criterios de prueba:

✅ **Fortalezas:** - Arquitectura testeable - Dependencias inyectables - Métodos con responsabilidades claras - Datos de prueba disponibles

⚠ **Mejoras:** - Faltan pruebas unitarias implementadas - Sin mocks para servicios externos

Justificación: Buena capacidad de prueba, falta implementación.

Puntuación Total Mantenibilidad: 4.6/5

8. Portabilidad (Portability)

Definición

Grado de efectividad y eficiencia con que un sistema, producto o componente puede ser transferido de un hardware, software u otro entorno operacional o de uso a otro.

Subcaracterísticas Evaluadas

8.1 Adaptabilidad

Puntuación: 4/5

Evaluación: Grado en que un producto o sistema puede ser adaptado efectiva y eficientemente para diferentes hardware, software u otros entornos:

✅ **Fortalezas:** - Configuración externalizada (application.yml) - Perfiles de Spring Boot - Base de datos configurable - Puerto y contexto configurables

⚠ **Mejoras:** - Configuración hardcodeada en algunos lugares - Sin soporte para múltiples bases de datos simultáneas

Justificación: Buena adaptabilidad con configuración flexible.

8.2 Capacidad de Instalación

Puntuación: 5/5

Evaluación: Efectividad y eficiencia para instalar y/o desinstalar exitosamente:

✅ **Excelente:** - JAR ejecutable independiente - Dependencias autocontenidas - Instalación con un comando (java -jar) - Sin configuración compleja requerida -

Docker-ready

Justificación: Instalación extremadamente simple y efectiva.

8.3 Capacidad de Reemplazo

Puntuación: 4/5

Evaluación: Grado en que un producto puede reemplazar otro producto especificado:

✓ **Características:** - API REST estándar - Formatos de datos comunes (JSON) - Interfaces bien definidas - Documentación completa

⚠ **Consideraciones:** - Sin versionado de API implementado - Dependencias específicas de Spring Boot

Justificación: Buen potencial de reemplazo con APIs estándar.

Puntuación Total Portabilidad: 4.3/5

Matriz de Evaluación Consolidada

Característica	Subcaracterística	Puntuación	Peso	Ponderado
Adecuación Funcional	Compleitud Funcional	5.0	0.4	2.0
	Corrección Funcional	4.0	0.4	1.6
	Pertinencia Funcional	5.0	0.2	1.0
	Subtotal	4.7	1.0	4.6
Eficiencia de Desempeño	Comportamiento Temporal	4.0	0.4	1.6
	Utilización de Recursos	4.0	0.3	1.2
	Capacidad	3.0	0.3	0.9
	Subtotal	3.7	1.0	3.7
Compatibilidad	Coexistencia	4.0	0.4	1.6
	Interoperabilidad	5.0	0.6	3.0
	Subtotal	4.5	1.0	4.6
Usabilidad	Reconocimiento de Idoneidad	5.0	0.2	1.0
	Capacidad de Aprendizaje	4.0	0.2	0.8
	Operabilidad	4.0	0.2	0.8
	Protección contra Errores	4.0	0.2	0.8
	Estética de Interfaz	4.0	0.1	0.4
	Accesibilidad	3.0	0.1	0.3
	Subtotal	4.0	1.0	4.1
Confiabilidad	Madurez	4.0	0.3	1.2
	Disponibilidad	3.0	0.3	0.9

Característica	Subcaracterística	Puntuación	Peso	Ponderado
	Tolerancia a Fallos	4.0	0.2	0.8
	Capacidad de Recuperación	3.0	0.2	0.6
	Subtotal	3.5	1.0	3.5
Seguridad	Confidencialidad	2.0	0.3	0.6
	Integridad	4.0	0.3	1.2
	No Repudio	2.0	0.2	0.4
	Responsabilidad	2.0	0.1	0.2
	Autenticidad	1.0	0.1	0.1
	Subtotal	2.2	1.0	2.5
Mantenibilidad	Modularidad	5.0	0.3	1.5
	Reusabilidad	4.0	0.2	0.8
	Analizabilidad	5.0	0.2	1.0
	Capacidad de Modificación	5.0	0.2	1.0
	Capacidad de Prueba	4.0	0.1	0.4
	Subtotal	4.6	1.0	4.7
Portabilidad	Adaptabilidad	4.0	0.4	1.6
	Capacidad de Instalación	5.0	0.4	2.0
	Capacidad de Reemplazo	4.0	0.2	0.8
	Subtotal	4.3	1.0	4.4

Puntuación Global Ponderada: 4.0/5.0 (80%)

Análisis de Resultados

Fortalezas Identificadas

1. **Excelente Adecuación Funcional (4.7/5)**
 2. Implementación completa de todos los requisitos
 3. Funcionalidades correctas y pertinentes
 4. API bien diseñada y comprehensiva
5. **Alta Mantenibilidad (4.6/5)**
 6. Arquitectura modular excelente
 7. Código limpio y bien estructurado
 8. Fácil de modificar y extender
9. **Buena Compatibilidad (4.5/5)**
 10. Excelente interoperabilidad con estándares
 11. API REST bien implementada
12. **Portabilidad Sólida (4.3/5)**
 13. Fácil instalación y despliegue
 14. Configuración flexible

Áreas de Mejora Críticas

1. **Seguridad (2.2/5) - CRÍTICO**
 2. **Problema Principal:** Sin autenticación ni autorización
 3. **Impacto:** Sistema no apto para producción sin mejoras
 4. **Recomendación:** Implementar Spring Security con JWT
5. **Confiabilidad (3.5/5) - IMPORTANTE**
 6. **Problemas:** Base de datos en memoria, sin alta disponibilidad

- 7. **Impacto:** Limitaciones para entorno productivo
- 8. **Recomendación:** Migrar a base de datos persistente
- 9. **Eficiencia de Desempeño (3.7/5) - MODERADO**
- 10. **Problemas:** Sin optimizaciones específicas, falta caché
- 11. **Impacto:** Posibles problemas de rendimiento a escala
- 12. **Recomendación:** Implementar caché y optimizaciones

Recomendaciones Prioritarias

Prioridad Alta (Críticas para Producción)

- 1. **Implementar Seguridad:**
 - 2. Autenticación con JWT
 - 3. Autorización basada en roles
 - 4. Encriptación de datos sensibles
 - 5. Auditoría de operaciones
- 6. **Migrar Base de Datos:**
 - 7. Cambiar a PostgreSQL o MySQL
 - 8. Implementar pool de conexiones
 - 9. Configurar backups automáticos

Prioridad Media (Mejoras Importantes)

- 1. **Mejorar Confiabilidad:**
 - 2. Implementar health checks
 - 3. Configurar métricas de monitoreo
 - 4. Añadir circuit breakers
- 5. **Optimizar Rendimiento:**
 - 6. Implementar caché (Redis)

7. Optimizar consultas de base de datos
8. Añadir paginación en listados

Prioridad Baja (Mejoras Deseables)

1. **Completar Testing:**
 2. Implementar pruebas unitarias
 3. Añadir pruebas de integración
 4. Configurar pruebas automatizadas
 5. **Mejorar Documentación:**
 6. Añadir guías de implementación
 7. Documentar casos de uso
 8. Crear tutoriales paso a paso
-

Conclusiones

Evaluación General

El microservicio desarrollado demuestra una **calidad alta** en aspectos fundamentales como funcionalidad, mantenibilidad y compatibilidad. La arquitectura es sólida y sigue las mejores prácticas de desarrollo.

Aptitud para Producción

Estado Actual: NO APTO para producción debido a deficiencias críticas en seguridad.

Con Mejoras Recomendadas: APTO para producción tras implementar las mejoras de prioridad alta.

Cumplimiento de ISO/IEC 25010

- **Características Excelentes:** Adecuación Funcional, Mantenibilidad
- **Características Buenas:** Compatibilidad, Portabilidad, Usabilidad

- **Características Aceptables:** Eficiencia de Desempeño, Confiabilidad
- **Características Deficientes:** Seguridad (requiere atención inmediata)

Valor del Proyecto

El microservicio representa un **excelente ejemplo académico** que demuestra: - Comprensión sólida de arquitecturas de software - Implementación correcta de patrones de diseño - Uso apropiado de tecnologías modernas - Capacidad de crear sistemas mantenibles y extensibles

Recomendación Final

APROBADO para propósitos académicos con calificación de **ALTO**.

Para uso productivo, implementar las mejoras de seguridad y confiabilidad recomendadas.

Evaluación realizada por: Estudiante Universidad Mariano Gálvez

Fecha de Evaluación: Diciembre 2024

Norma Aplicada: ISO/IEC 25010:2011

Metodología: Evaluación cualitativa con criterios cuantitativos

Próxima Revisión: Tras implementación de mejoras críticas

Conclusión

El microservicio desarrollado demuestra un alto nivel de calidad en aspectos fundamentales como la adecuación funcional, la mantenibilidad y la compatibilidad, cumpliendo con la mayoría de los criterios establecidos por la norma ISO/IEC 25010. La documentación técnica de la API refleja una estructura clara, endpoints bien definidos y un uso consistente de estándares REST, lo que facilita su aprendizaje, integración y operación.

Las pruebas funcionales validaron con éxito los flujos críticos del sistema, incluyendo la gestión de usuarios, productos y pedidos, así como la correcta aplicación de las reglas de negocio, el manejo de errores y la integridad de los datos. Esto confirma que el microservicio es estable en su lógica principal y cumple con los requisitos definidos.

No obstante, la evaluación de calidad también evidenció áreas críticas de mejora, especialmente en seguridad y confiabilidad, las cuales deben ser atendidas antes de su implementación en un entorno productivo. La ausencia de autenticación, autorización y cifrado de datos sensibles limita su aptitud para entornos reales, por lo que se recomienda implementar estas medidas junto con mecanismos de alta disponibilidad y optimización de rendimiento.

En conjunto, el proyecto constituye un sólido ejemplo académico de aplicación de buenas prácticas de desarrollo y aseguramiento de la calidad, demostrando la capacidad de integrar diseño, implementación, validación y evaluación conforme a estándares internacionales.

Bibliografía

- ISO/IEC 25010:2011. (2011). *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. International Organization for Standardization.
- Fielding, R. T., & Reschke, J. (2014). *Hypertext Transfer Protocol (HTTP/1.1)*. Internet Engineering Task Force.
<https://doi.org/10.17487/RFC7231>
- Spring. (2025). *Spring Boot Reference Documentation*. Spring.io.
<https://spring.io/projects/spring-boot>