

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