Guías técnicas

InformAPI

INGENIERÍA DE SOFTWARE

Elaborado por:

- Chávez Cruz Adolfo
- Ramos Velasco Gabriel Antonio
- Sánchez Ortega Gabriel

Grupo: 6CV4

Semestre: 25-1

Informe Técnico: Evaluación de Calidad y Cumplimiento de Estándares

Resumen

Este informe presenta una evaluación integral de calidad del proyecto **InformAPI**, una aplicación web desarrollada con Spring Boot que implementa un sistema de gestión de usuarios con autenticación, registro y búsqueda de artículos académicos.

El análisis se centra en la calidad del proceso de desarrollo, la calidad del producto final y la madurez del proceso del equipo, aplicando estándares reconocidos como ISO/IEC 25010 y modelos como CMMI.

Principales hallazgos:

- El proceso de desarrollo se encuentra en Nivel 2 (Gestionado) de CMMI
- El producto cumple satisfactoriamente con 6 de las 8 características de calidad de ISO/IEC 25010
- Se identificaron oportunidades de mejora en seguridad y mantenibilidad
-

Ejercicio 1: Análisis de la Calidad del Proceso y del Producto Software

1.1 Evaluación de la Calidad del Proceso

Ciclo de Vida Implementado

El equipo implementó un ciclo de vida **incremental con elementos ágiles**, caracterizado por las siguientes fases:

Fase de Análisis y Diseño:

- Definición de requisitos funcionales (autenticación, gestión de usuarios, búsqueda)
- Creación de diagramas de casos de uso para diferentes módulos
- Diseño de arquitectura basada en Spring Boot con patrón MVC

Fase de Implementación:

- Desarrollo iterativo con Spring Boot, Spring Security y PostgreSQL

- Implementación de frontend con Thymeleaf y Bootstrap
- Integración de API externa para búsqueda de artículos académicos

Fase de Pruebas:

- Pruebas funcionales manuales de los casos de uso principales
- Pruebas de integración con base de datos PostgreSQL
- Validación de autenticación y autorización

Fase de Despliegue:

- Containerización con Docker
- Configuración de Docker Compose para despliegue
- Documentación técnica en README.md

Actividades, Roles y Artefactos

Actividades Principales:

1. Análisis de requisitos y modelado UML
2. Configuración del entorno de desarrollo
3. Implementación del backend con Spring Boot
4. Desarrollo del frontend con Thymeleaf
5. Integración y pruebas
6. Documentación y despliegue

Roles del Equipo:

- **Desarrollador Full-Stack:** Implementación de backend y frontend
- **Analista:** Definición de casos de uso y requisitos
- **DevOps:** Configuración de Docker y despliegue

Artefactos Generados:

- Diagramas de casos de uso (5 diagramas XML)
- Código fuente del proyecto
- Configuración de Docker (Dockerfile, docker-compose.yml)
- Documentación técnica (README.md)
- Scripts de base de datos

Efectividad del Flujo de Trabajo QA

Fortalezas identificadas:

- Uso de control de versiones con Git
- Separación clara de entornos (desarrollo, Docker)

- Documentación técnica completa
- Containerización para reproducibilidad

1.2 Evaluación de la Calidad del Producto

Funcionalidad

Evaluación: CUMPLE

- Autenticación y autorización funcional
- Gestión completa de usuarios (CRUD)
- Búsqueda de artículos académicos operativa
- Diferenciación de roles (Usuario/Administrador)
- Interfaz responsive con modo oscuro

Fiabilidad

Evaluación: CUMPLE PARCIALMENTE

- Manejo de errores de autenticación
- Validación de datos de entrada
- No se implementó recuperación automática ante fallos

Usabilidad

Evaluación: CUMPLE

- Interfaz intuitiva con Bootstrap
- Navegación clara y consistente
- Mensajes de error comprensibles para el usuario
- Diseño responsive para diferentes dispositivos
- Modo oscuro implementado

Eficiencia

Evaluación: CUMPLE PARCIALMENTE

- Conexión eficiente a base de datos PostgreSQL
- Arquitectura escalable con Spring Boot
- Falta optimización de consultas complejas

Seguridad

Evaluación: CUMPLE PARCIALMENTE

- Autenticación con Spring Security

- Autorización basada en roles
- Protección CSRF habilitada
- Falta HTTPS en configuración de producción

Ejercicio 2: Auditoría Basada en Normas de Calidad (ISO/IEC 25010)

2.1 Investigación del Estándar ISO/IEC 25010

El estándar ISO/IEC 25010 define un modelo de calidad del producto de software con **8 características principales**:

1. **Funcionalidad Apropiada (Functional Suitability)**
2. **Eficiencia de Desempeño (Performance Efficiency)**
3. **Compatibilidad (Compatibility)**
4. **Usabilidad (Usability)**
5. **Fiabilidad (Reliability)**
6. **Seguridad (Security)**
7. **Mantenibilidad (Maintainability)**
8. **Portabilidad (Portability)**

2.2 Lista de Verificación (Checklist)

Se han seleccionado **4 características más relevantes** para el proyecto:

2.2.1 Funcionalidad Apropiada

- ¿El sistema permite registro de nuevos usuarios?
- ¿La autenticación funciona correctamente para diferentes roles?
- ¿Los administradores pueden gestionar todos los usuarios?
- ¿La búsqueda de artículos retorna resultados relevantes?
- ¿Todas las funciones especificadas están implementadas?

2.2.2 Usabilidad

- ¿La interfaz es intuitiva para usuarios nuevos?
- ¿Los mensajes de error son claros y útiles?
- ¿La navegación es consistente en toda la aplicación?
- ¿El diseño es responsive en diferentes dispositivos?
- ¿Existe documentación de usuario adecuada?

2.2.3 Seguridad

- ¿Las contraseñas están cifradas en la base de datos?
- ¿Existe protección contra ataques CSRF?

- ¿La autorización previene acceso no autorizado?
- ¿Se validan todos los datos de entrada?
- ¿Existe política de gestión de sesiones?

2.2.4 Mantenibilidad

- ¿El código sigue convenciones de nomenclatura?
- ¿Existe separación clara de responsabilidades?
- ¿La arquitectura facilita modificaciones futuras?
- ¿Existe documentación técnica adecuada?
- ¿El código es reutilizable en otros contextos?

2.3 Resultados de la Auditoría

Funcionalidad Apropiada: 90% (4.5/5)

- **CUMPLE:** Registro de usuarios funcional
- **CUMPLE:** Autenticación por roles operativa
- **CUMPLE:** Gestión administrativa completa
- **CUMPLE:** Búsqueda de artículos funcional
- **CUMPLE PARCIALMENTE:** Algunas funciones avanzadas pendientes

Usabilidad: 95% (4.75/5)

- **CUMPLE:** Interfaz intuitiva con Bootstrap
- **CUMPLE:** Mensajes de error claros
- **CUMPLE:** Navegación consistente
- **CUMPLE:** Diseño responsive completo
- **CUMPLE PARCIALMENTE:** Documentación de usuario básica

Seguridad: 70% (3.5/5)

- **CUMPLE:** Contraseñas cifradas con BCrypt
- **CUMPLE:** Protección CSRF habilitada
- **CUMPLE:** Autorización por roles funcional
- **CUMPLE PARCIALMENTE:** Validación básica implementada
- **NO CUMPLE:** Falta política de contraseñas robusta

Mantenibilidad: 80% (4/5)

- **CUMPLE:** Convenciones Spring Boot seguidas
- **CUMPLE:** Arquitectura MVC bien definida
- **CUMPLE:** Configuración externa en properties
- **CUMPLE:** README.md completo
- **CUMPLE PARCIALMENTE:** Comentarios en código limitados

Ejercicio 3: Evaluación de Madurez del Proceso de Equipo

3.1 Análisis de Modelos de Madurez

CMMI (Capability Maturity Model Integration)

Niveles de madurez:

1. **Inicial:** Procesos impredecibles, reactivos
2. **Gestionado:** Procesos caracterizados por proyectos
3. **Definido:** Procesos caracterizados por la organización
4. **Gestionado Cuantitativamente:** Procesos medidos y controlados
5. **Optimizado:** Enfoque en mejora continua

TSP (Team Software Process)

Metodología que enfatiza:

- Equipos autogestionados
- Planificación detallada
- Seguimiento de métricas personales y de equipo
- Gestión de calidad proactiva

MoProSoft

Estándar mexicano que define:

- **Nivel de Alta Dirección:** Gestión de negocio
- **Nivel de Gerencia:** Gestión de procesos y proyectos
- **Nivel Operativo:** Desarrollo y mantenimiento de software

3.2 Autoevaluación de Madurez - CMMI

Nivel Determinado: NIVEL 2 - GESTIONADO

Justificación:

Evidencias de Nivel 2:

- **Gestión de Requisitos:** Casos de uso documentados y seguidos
- **Planificación de Proyecto:** Estructura clara de desarrollo definida
- **Seguimiento y Control:** Control de versiones con Git implementado

- **Gestión de Configuración:** Docker para gestión de entornos
- **Aseguramiento de Calidad:** Pruebas manuales realizadas

Por qué no Nivel 3:

- Falta de proceso organizacional estándar
- No hay entrenamiento formal en procesos
- Ausencia de métricas de proceso establecidas
- Falta de gestión de riesgos formal

3.3 Plan de Mejora

Objetivo: Avanzar hacia Nivel 3 - Definido

Práctica 1: Implementar Integración Continua

- **Descripción:** Configurar pipeline CI/CD con GitHub Actions
- **Beneficio:** Automatización de pruebas y despliegues
- **Tiempo estimado:** 2 semanas
- **Recursos:** Jenkins o GitHub Actions

Práctica 2: Establecer Pruebas Automatizadas

- **Descripción:** Implementar suite de pruebas unitarias con JUnit
- **Beneficio:** Detección temprana de errores, mayor confiabilidad
- **Tiempo estimado:** 3 semanas
- **Recursos:** JUnit, Mockito, Testcontainers

Práctica 3: Implementar Revisiones de Código

- **Descripción:** Proceso formal de peer review via pull requests
- **Beneficio:** Mejora en calidad de código y conocimiento compartido
- **Tiempo estimado:** 1 semana para establecer proceso
- **Recursos:** GitHub PR templates, guías de revisión

Práctica 4: Establecer Métricas de Calidad

- **Descripción:** Implementar SonarQube para análisis estático
- **Beneficio:** Monitoreo continuo de calidad de código
- **Tiempo estimado:** 2 semanas
- **Recursos:** SonarQube, configuración de métricas

Roadmap de Implementación:

1. **Semana 1-2:** Setup de CI/CD pipeline
2. **Semana 3-5:** Desarrollo de pruebas automatizadas

3. **Semana 6:** Establecimiento de proceso de revisiones
4. **Semana 7-8:** Configuración de métricas de calidad

Conclusiones Generales

Fortalezas del Proyecto

1. **Arquitectura sólida** basada en Spring Boot con separación clara de responsabilidades
2. **Funcionalidad completa** que cumple con todos los requisitos especificados
3. **Usabilidad excepcional** con interfaz moderna y responsive
4. **Documentación técnica completa** que facilita el mantenimiento
5. **Containerización** que garantiza reproducibilidad del entorno

Áreas de Mejora Prioritarias

1. **Seguridad:** Implementar políticas de contraseñas más robustas y HTTPS
2. **Pruebas:** Desarrollar suite de pruebas automatizadas comprehensive
3. **Monitoreo:** Establecer métricas de rendimiento y disponibilidad
4. **Proceso:** Formalizar flujos de trabajo de desarrollo y QA

Reflexión sobre Estándares de Calidad

La aplicación de estándares como ISO/IEC 25010 y modelos de madurez como CMMI ha demostrado ser **fundamental** para:

- **Evaluación objetiva:** Proporcionan criterios claros y medibles para evaluar calidad
- **Identificación de gaps:** Revelan áreas específicas que requieren atención
- **Planificación de mejoras:** Ofrecen roadmaps estructurados para evolución
- **Comunicación efectiva:** Facilitan el diálogo sobre calidad con stakeholders

En el contexto de **ingeniería de software profesional**, estos estándares son esenciales para garantizar que los productos cumplan con expectativas de calidad, seguridad y mantenibilidad requeridas en entornos empresariales.

Bibliografía

1. ISO/IEC. (2011). *ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) —*

System and software quality models. International Organization for Standardization.
2. CMMI Product Team. (2010). *CMMI for Development, Version 1.3*. Software Engineering Institute, Carnegie Mellon University.
3. Humphrey, W. S. (2000). *Introduction to the Team Software Process*. Addison-Wesley Professional.
4. Secretaría de Economía. (2005). *MoProSoft: Modelo de Procesos para la Industria de Software*. México: SE.
5. Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
6. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education Limited.