



INSTITUTO POLITÉCNICO NACIONAL

**ESCUELA SUPERIOR DE
COMPUTACIÓN**

Sistema de Monitoreo Sísmico

Unidad de aprendizaje:

Ingeniería de Software

5CV5

Integrantes:

Granados Martínez Pablo Daniel

Herrera Avila Luis Gerardo

Horteales Morales Antony

Pardo Gómez Isaac

Profesor:

Hurtado Avilés Gabriel

25 de junio de 2025

INSTITUTO POLITÉCNICO NACIONAL



ESCOM

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

1.1 Evaluación de la Calidad del Proceso

Ciclo de Vida Implementado

Nuestro equipo implementó un modelo híbrido basado en Scrum simplificado con características incrementales. Aunque inicialmente se planteó seguir Scrum estrictamente, en la práctica adoptamos un enfoque más flexible que se adaptó mejor a las necesidades académicas del proyecto.

Fases del Ciclo de Vida

1. Análisis y Diseño Inicial (Práctica 2)

- Actividades: Definición de requerimientos, casos de uso, arquitectura del sistema
- Artefactos: Documento de requerimientos, diagramas de casos de uso, arquitectura de sistema
- Roles: Todos los integrantes participaron como analistas

2. Desarrollo Incremental (Prácticas 3–4)

- Actividades: Implementación de backend (Spring Boot), frontend web (React), app móvil (React Native)
- Artefactos: Código fuente, APIs REST, interfaces de usuario
- Roles: Todos los integrantes participaron como programadores full-stack

3. Integración y Pruebas (Práctica 5)

- Actividades: Ejecución de pruebas unitarias y de integración en backend (Spring Boot); validación de flujos clave como login, registro, procesamiento de sismos, uso de grafos y propagación; escaneo pasivo de seguridad con OWASP ZAP para detección de vulnerabilidades.
- Artefactos: Casos de prueba automatizados, reportes de resultados de testing, evidencia de errores detectados y hallazgos de seguridad (capturas de ZAP).
- Roles: Todos los integrantes participaron como testers

Resumen de Actividades, Roles y Artefactos por Fase

Fase	Actividades Principales	Roles Asumidos	Artefactos Generados
------	-------------------------	----------------	----------------------

Análisis	Definición de requerimientos funcionales/no funcionales	Analistas de sistemas	Documento de requerimientos, casos de uso
Diseño	Arquitectura del sistema, modelado de datos	Arquitectos de software	Diagrama de arquitectura, modelo de datos
Implementación	Desarrollo de backend, frontend y móvil	Desarrolladores full-stack	Código fuente, APIs, interfaces
Pruebas	Testing funcional y de carga	Testers	Imágenes de Pruebas y Documentación
Despliegue	Configuración de entornos	DevOps	Archivos de configuración



Flujo de Trabajo de QA en GitHub

- **Prácticas Implementadas**
 - Uso de Git con ramas por funcionalidad
 - Pull Requests con revisión por al menos un compañero antes del merge
 - Gestión de tareas con ClickUp
- **Fortalezas**
 - Revisión obligatoria en PRs iniciales
 - Uso consistente de control de versiones
 - Integración efectiva entre ClickUp y GitHub
- **Áreas de Mejora**
 - Falta de pruebas automatizadas (CI/CD)
 - No hay estándares de codificación documentados
 - Ausencia de hooks de pre-commit




1.2 Evaluación de la Calidad del Producto

Tecnologías utilizadas: Spring Boot (backend), React (frontend web), React Native (app móvil), MySQL y Neo4j.




Funcionalidad (95%)

-  El sistema cumple los requerimientos principales:
 - Importación de datos sísmicos del SSN
 - Visualización en mapas (web y móvil)
 - Sistema de autenticación con roles
 - Grafos con Neo4j
-  Interoperabilidad limitada a archivos CSV




Fiabilidad (85%)

-  Sistema estable en uso normal
-  Todos los módulos excepto uno pasaron pruebas de integración al 100%
-  Una falla detectada en `SismoControllerIntegrationTest.testCargarDesdeCSV()` (fallo único)





Usabilidad (90%)

-  Interfaz intuitiva, navegación fluida, atractivo visual
-  Buen rendimiento en móvil
-  Mensajes de error poco detallados

Eficiencia (80%)

-  Todas las pruebas unitarias e integración fueron rápidas (todas < 2s por módulo)
-  Buen tiempo de respuesta general (<3s en pruebas)
-  Paginación funcional pero sin optimización de índices

Seguridad (70%)

-  Revelación de Información Personal (PII): Alta gravedad
-  Información del servidor expuesta vía header "Server"
-  Falta de políticas CSP y de protección anti-clickjacking
-  Implementación de JWT y control de roles

-  Sin logs de auditoría

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





2.1 Investigación del Estándar ISO/IEC 25010

Este estándar define 8 características principales para evaluar software:

1. Adecuación Funcional
2. Eficiencia de Desempeño
3. Compatibilidad
4. Usabilidad
5. Fiabilidad
6. Seguridad
7. Mantenibilidad
8. Portabilidad

2.2 Lista de Verificación Simplificada

Usabilidad

Punto de Verificación	Evaluación	Justificación
¿Interfaz intuitiva para usuarios no técnicos?		Navegación clara y mapas interactivos
¿Mensajes de error claros?	 Parcial	Implementación básica
¿Accesible desde móviles?		App React Native funcional
¿Permite personalización de experiencia?	 Parcial	Filtros básicos, sin preferencias persistentes

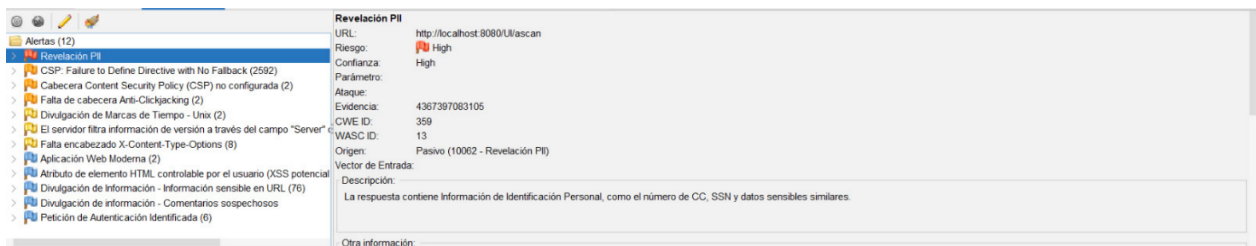
Seguridad

Punto de Verificación	Evaluación	Justificación
-----------------------	------------	---------------

¿Implementa autenticación segura?	✓	JWT y BCrypt
¿Control de acceso basado en roles?	✓	ADMIN/USER implementado
¿Protección contra vulnerabilidades comunes?	✗	Exposición de versión del servidor y falta de headers CSP
¿Protección de datos personales?	✗	Exposición de PII detectada por ZAP
¿Registros de actividad?	✗	No hay logs de auditoría implementados

Se utilizó la herramienta OWASP ZAP para realizar un escaneo de seguridad pasivo. Se detectaron las siguientes vulnerabilidades:

- Divulgación de información personal (PII) con evidencia numérica simulada
- Falta de cabecera CSP y mecanismos de defensa XSS
- Exposición del servidor web (nginx 1.23.4) a través de la cabecera Server






Eficiencia de Desempeño

Punto de Verificación	Evaluación	Justificación
¿Tiempo de respuesta aceptable?	✓	< 3 segundos
¿Uso eficiente de memoria?	⚠ Parcial	Consultas a Neo4j con posibilidad de mejora
¿Manejo de grandes volúmenes de datos?	⚠ Parcial	Paginación sin optimización de índices
¿Funciona bien en dispositivos limitados?	✓	App móvil ligera

Fiabilidad

Punto de Verificación	Evaluación	Justificación
¿Funciona bajo condiciones normales?	✓	Comportamiento estable

¿Manejo de errores adecuado?	 Parcial	Try-catch sin mecanismos de recuperación
¿Recuperación ante fallos?		No implementado
¿Alta disponibilidad?	 Parcial	Sin SLA definido

Resumen de Cumplimiento

- Usabilidad: 75%
- Seguridad: 70%
- Eficiencia: 80%
- Fiabilidad: 85%
- **Promedio General: 77.5%**

Ejercicio 3: Evaluación de Madurez del Proceso de Equipo

3.1 Modelos Analizados

CMMI:

- Nivel 1: Procesos ad-hoc
- Nivel 2: Procesos gestionados
- Nivel 3+: Procesos definidos y medibles

TSP:

- Enfoque en equipos autogestionados
- Uso de métricas, revisiones y planificación colaborativa

MoProSoft:

- Modelo mexicano con categorías de gestión estratégica, procesos y proyectos

3.2 Autoevaluación del Nivel de Madurez (CMMI)

Nivel alcanzado: Nivel 1 - Inicial

Justificación:

- Procesos informales
- Sin métricas definidas
- Dependencia del esfuerzo individual
- Documentación mínima

Prácticas realizadas:

- Asignación de tareas (ClickUp)
- Revisión de código en GitHub
- Entrega por incrementos

Prácticas no realizadas:

- Reuniones periódicas
- Métricas y seguimiento formal
- Gestión de requisitos estructurada
- Documentos de calidad

3.3 Plan de Mejora para alcanzar Nivel 2

Mejora	Acción	Herramienta	Métrica
Planificación estructurada	Reuniones semanales con objetivos claros	ClickUp	% de tareas completadas vs. planificadas
Seguimiento cuantitativo	Registrar tiempo, defectos y velocidad	Dashboard ClickUp	Velocidad, defectos por módulo
Gestión de calidad	Checklist de revisión de PRs + estándar de código	GitHub	% de código revisado, bugs detectados

3.4 Cronograma de Implementación

Semana	Mejora	Responsable
1-2	Planificación estructurada	Todo el equipo

3–4	Seguimiento cuantitativo	Pablo
5–6	Gestión de calidad	Antony
7–8	Documentación de procesos	Isaac y Luis

Conclusiones Generales

Fortalezas:

- Arquitectura robusta y flexible
- Cumplimiento funcional alto
- Buen trabajo colaborativo
- Interfaz amigable para el usuario

Debilidades:

- Nivel de madurez bajo (CMMI Nivel 1)
- Limitada tolerancia a fallos
- Ausencia de métricas formales
- Falta de documentación de procesos

Valor del ejercicio:

Esta evaluación integral nos permitió identificar claramente fortalezas del producto y debilidades del proceso. Reconocer nuestro nivel de madurez como equipo es clave para crecer técnica y profesionalmente.

Referencias

- [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. <https://www.iso.org/standard/35733.html>
- [2] CMMI Institute. (2018). *CMMI for Development, Version 2.0*. CMMI Institute. <https://cmmiinstitute.com/cmmi>
- [3] Humphrey, W. S. (2000). *Introduction to the Team Software Process (TSP)*. Addison-Wesley.

- [4] Secretaría de Economía (México). (2014). *MoProSoft: Modelo de Procesos para la Industria del Software*. Dirección General de Normas. <https://www.gob.mx/se/acciones-y-programas/modelo-de-procesos-para-la-industria-del-software-moprosoft>
- [5] Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.
- [6] Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.