Universidad Interamericana De Panamá

Facultad De Ingenieria, Arquitectura Y Diseño

Materia:
Calidad de Software

Estudiante - Cédula
Jeyson Rodriguez 8-951-480

Profesor Facilitador:
Marcel Castillo

Asignación
Tarea

Tema:
Introducción a la Calidad en el Desarrollo de Software: de la teoría a la práctica

Introducción a la Calidad en el Desarrollo de Software: de la teoría a la práctica

Fase 1 – Activación y sentido

¿Qué significa calidad para mí?

La calidad de software representa mucho más que lograr que un programa "funcione". Para mí, calidad implica que el software sea confiable, robusto, seguro, claro, mantenible y coherente, incluso cuando recibe entradas inesperadas o cuando el usuario comete errores. La calidad es aplicar buenas prácticas desde el inicio: validar datos, probar funcionalidades, documentar el sistema, manejar errores adecuadamente, registrar eventos y utilizar control de versiones.

¿Funciona vs. funciona con calidad?

Funciona:
- Ejecuta el caso básico.
- No garantiza estabilidad ante entradas inválidas.
- No maneja errores.
- No está probado exhaustivamente.
- No tiene trazabilidad ni documentación.

Funciona con calidad:
- Tiene pruebas realizadas.
- Valida entradas y fechas.
- Maneja excepciones de manera controlada.
- Utiliza persistencia confiable.
- Registra eventos en una bitácora (log).
- Está organizado modularmente.
- Usa control de versiones.

En términos de calidad:
Funciona = verificación mínima.
Funciona con calidad = verificación + validación + aseguramiento.

Fase 2 – introducción conceptual

Un software que falla en producción vs. un software con validación de calidad

Un software sin calidad puede fallar en producción por razones como:
- No validar las entradas del usuario.
- No manejar fechas incorrectas o valores fuera de rango.
- Ausencia de pruebas y casos límite.
- Errores silenciosos sin registro.
- No documentar ni aplicar control de versiones.

- Falta de trazabilidad durante el desarrollo.

Un software con calidad:
- Realiza pruebas antes de su liberación.
- Maneja errores y excepciones sin detener la aplicación.
- Valida formatos (fechas, tipos de datos).
- Deja bitácora de eventos que permiten rastrear errores.
- Usa un repositorio versionado.
- Tiene código modular, claro y mantenible.

Esta fase evidencia que la calidad se integra en todas las etapas del ciclo de vida del software y no como una actividad aislada al final.

Fase 3 – proyecto práctico

Mini–Proyecto: Agenda de Tareas en Consola (Python)

Desafío elegido

Construcción de un prototipo avanzado de consola que administre tareas, incorporando prácticas reales de calidad: validaciones, persistencia, manejo de errores, bitácora, modularidad y pruebas simples.

Este prototipo fue desarrollado en Python utilizando archivos JSON para persistencia y un archivo log.txt como bitácora.

Requisitos funcionales del sistema
- RF1: Permitir agregar una tarea.
- RF2: Mostrar la lista de tareas registradas.
- RF3: Marcar una tarea como completada.
- RF4: Eliminar una tarea.

Requisitos no funcionales (atributos de calidad)
- Fiabilidad: manejo de errores e índices inválidos.
- Usabilidad: mensajes claros de interacción en consola.
- Mantenibilidad: código organizado en funciones.
- Portabilidad: uso de Python, ejecutable en cualquier sistema.

Requisitos del sistema

Requisitos funcionales (RF)
- RF1: Agregar una tarea con descripción, prioridad y fecha límite.
- RF2: Listar todas las tareas.
- RF3: Listar tareas pendientes.
- RF4: Listar tareas completadas.

- RF5: Filtrar tareas pendientes de alta prioridad.
- RF6: Marcar una tarea como completada.
- RF7: Editar tarea existente.
- RF8: Eliminar tarea por ID.
- RF9: Listar tareas vencidas automáticamente.

Requisitos no funcionales (RNF)

- Fiabilidad: manejo robusto de errores, validación de tipos, fechas y campos vacíos.
- Usabilidad: menú simple, mensajes claros, flujos predecibles.
- Mantenibilidad: código modular dividido en funciones.
- Portabilidad: aplicación ejecutable en cualquier sistema con Python.
- Trazabilidad: registro detallado en bitácora log.txt.
- Persistencia: almacenamiento en tareas.json.

Casos de Prueba Diseñados

Caso	Entrada	Resultado Esperado
1	Descripción válida, prioridad media, fecha válida	Tarea agregada correctamente
2	Descripción vacía	Error: descripción no puede estar vacía
3	Descripción > 200 caracteres	Error: límite excedido
4	Fecha límite en formato inválido (2024/01/01)	Error y fecha no asignada
5	Fecha límite en el pasado	Error: fecha no puede ser anterior a hoy
6	ID inexistente al marcar completada	Mensaje de error + log
7	ID no numérico (“abc”)	Error de validación
8	Editar tarea con nueva descripción válida	Actualización exitosa
9	Listar tareas sin registros	Mostrar mensaje: “No hay tareas registradas”
10	Eliminar tarea existente	Eliminación correcta
11	Eliminar tarea inexistente	Error controlado
12	Mostrar tareas vencidas	Listar solo las que cumplen condición
13	Prioridad inválida (opción 5)	Asignar “baja” y registrar en log
14	Crear varias tareas y observar IDs	IDs incrementales y únicos

Checklist de Calidad Aplicada

Criterio Evaluado	Cumple	Evidencia / Observación
El programa se ejecuta sin errores	Sí	Menú completo
Validación de descripción	Sí	Evita vacío y límite de caracteres
Validación de prioridad	Sí	Opción 1–3, fallback a “baja”
Validación de fechas	Sí	Rechaza formatos incorrectos y fechas pasadas
Manejo de errores	Sí	IDs inválidos, entradas no numéricas
Persistencia en JSON	Sí	Guardado/lectura correcto
Bitácora de eventos	Sí	Archivo log.txt con timestamps
Funcionalidades completas	Sí	9 funciones implementadas
Código modular	Sí	Funciones separadas y ordenadas
Usabilidad	Sí	Mensajes claros en consola
Control de versiones	Sí	Git con commits (capturas)
Evidencias de pruebas	Sí	Capturas incluidas

Fase 4 – Socialización y Retroalimentación

Se presentó el funcionamiento del sistema y las prácticas de calidad aplicadas. Se discutió cómo la validación de entradas, el manejo de errores, el uso de JSON y la bitácora aportan trazabilidad y robustez al software. Se brindaron retroalimentación respecto a mejoras posibles, incluyendo añadir más filtros y validaciones adicionales.

La reflexión destacó que la calidad no es solo técnica: implica ética profesional, responsabilidad y compromiso con los usuarios.

Reflexión

La calidad en el desarrollo de software es un compromiso ético fundamental. Un sistema que no valida datos, no registra errores o no está probado adecuadamente puede fallar en momentos críticos y afectar directamente a usuarios, organizaciones e incluso la seguridad personal. En entornos como salud, banca o transporte, una falla mínima puede generar pérdidas significativas.

Aplicar calidad no es opcional; es parte de la responsabilidad profesional que un desarrollador debe asumir.

Repositorio

https://github.com/JeysonRodriguez/Calidad-de-Software