Proyecto 3 – Diseño y Arquitectura de Software

Entrega 3 – Construcción de Software con Pruebas, CI/CD y DevSecOps

Curso: Diseño y Arquitectura de Software
Profesor: César Augusto Vega Fernández

Instrucciones para la entrega

La entrega de soluciones es en grupos de 3 estudiantes. Por favor escriba claramente su nombre, código de estudiante y sección en cada hoja entregada y en cada archivo de código (a modo de comentario). Adicionalmente, cite cualquier fuente de información que utilizó.

Enunciado Fase 3: Construcción de Software (Entrega 3)

En esta tercera y última entrega, su equipo debe enfocarse en la construcción del software siguiendo los lineamientos de la arquitectura seleccionada en entregas anteriores. La entrega debe demostrar la capacidad del equipo para implementar un sistema robusto, modular, seguro y escalable, aplicando buenas prácticas de pruebas, TDD, automatización de pipelines CI/CD y principios de DevSecOps.

Objetivo

El objetivo de la entrega final del proyecto es que los estudiantes desarrollen e implementen un sistema o aplicación que:
1. Cumpla con los requisitos de la arquitectura seleccionada.
2. Integre pruebas automatizadas: unitarias, autónomas, de interfaz gráfica, de carga y de APIs.
3. Aplique TDD para guiar el diseño del sistema y aumentar la calidad del código.
4. Implemente un pipeline de CI/CD que ejecute pruebas, despliegue y monitoreo continuo.
5. Incorpore prácticas de DevSecOps: análisis estático, dinámico, escaneo de dependencias y contenedores.
6. Documente y presente los resultados, desafíos y aprendizajes con soporte gráfico (modelos 4+1 y C4).

Tareas

Desarrollo del Software

- Construir la aplicación o sistema según la arquitectura seleccionada (Microservicios, Monolítica, Event-Driven, etc.).
- Seguir principios de diseño (SOLID, Arquitectura Limpia) y patrones arquitectónicos definidos.
- Implementar seguridad básica (validaciones, cifrado, autenticación/autorización).

Pruebas

Realizar pruebas exhaustivas utilizando herramientas adecuadas, integradas al pipeline:
- Pruebas unitarias (JUnit, PyTest, PHPUnit, Mocha).
- Pruebas autónomas (Mockito, WireMock).
- Pruebas de carga (JMeter, Gatling, k6).
- Pruebas de GUI (Selenium, Cypress).
- Pruebas de API (Postman + Newman).

Integración Continua y Despliegue Continuo (CI/CD)

- Configurar un pipeline de CI que ejecute automáticamente todas las pruebas en cada commit.
- Configurar un pipeline de CD para automatizar el despliegue en entornos de pruebas/producción.
- Usar contenerización (Docker) y orquestación (Kubernetes) para asegurar portabilidad y escalabilidad.
- Monitorear despliegues con Prometheus, Grafana y centralización de logs (ELK/EFK).

DevSecOps

- SAST (SonarQube, Semgrep).
- DAST (OWASP ZAP).
- Dependency Scanning (Snyk, OWASP Dependency-Check).
- Secrets Scanning (Gitleaks, TruffleHog).
- Escaneo de imágenes de contenedores (Trivy).
- Aplicar principios de seguridad en Kubernetes: RBAC, Secrets, Network Policies.

Análisis de Calidad de Código

- Integrar SonarQube para análisis de calidad, cobertura y vulnerabilidades.
- Establecer umbrales de cobertura (mínimo 80%).

Reportes de Pruebas

- Generar informes visuales (Allure Report o equivalentes).
- Incluir evidencias de ejecución: logs, capturas de pantalla, métricas de monitoreo.

Documentación

- Documentar el proceso de pruebas y CI/CD.
- Usar modelos arquitectónicos 4+1 y C4.
- Registrar decisiones de seguridad y escalabilidad.
- Incluir retos técnicos enfrentados y soluciones implementadas.

Evaluación para la Entrega 3

- Subir el código fuente en un repositorio (GitHub/GitLab).
- Entregar informe técnico en PDF con documentación.
- Presentar resultados en un máximo de 20 minutos mostrando:
  - Ejecución de pruebas.
  - Pipeline CI/CD en funcionamiento.
  - Evidencias de seguridad y escalabilidad.

Criterios clave: claridad, justificación de decisiones arquitectónicas, integración de DevSecOps, uso de TDD y documentación completa.