Tarea 2: Desarrollo Front-end / Back-end con Integración DevSecOps

Repositorio base obligatorio: https://github.com/sancano22/practica2

Objetivo de la tarea

A partir del repositorio Práctica 2, el/la estudiante deberá diseñar, completar e integrar un pipeline CI/CD con enfoque DevSecOps**,** incorporando herramientas de calidad, seguridad y automatización**, y** justificando técnicamente cada decisión tomada.

Esta tarea simula un escenario real de desarrollo profesional, donde el código solo puede avanzar si cumple criterios funcionales y de seguridad**.**

Contexto

El repositorio Práctica 2 ya provee:

- Separación frontend / backend
- Uso de contenedores
- Estructura para CI/CD en GitHub Actions
- Base para despliegue y seguridad

1. Analizar el repositorio base
    a. Clonar el repositorio **Práctica 2**.
    b. Comprender la arquitectura general del sistema:
        i. Frontend
        ii. Backend
        iii. Contenedores
    c. Verificar que la aplicación es funcional antes de integrar el pipeline.
2. Diseñar e implementar el pipeline CI/CD (archivo .yml)El/la estudiante deberá crear o completar el archivo de pipeline ubicado en **.github/workflows/**
   El pipeline debe incluir, **como mínimo**, las siguientes etapas:
    a. Instalación reproducible
        - Instalación automática de dependencias.
        - Uso de comandos reproducibles (npm ci, pip install).
        - El pipeline debe fallar si existen inconsistencias.
    b. Análisis de calidad de código
        - Integración de una herramienta de análisis de estilo (ESLint / Pylint).
        - Detección automática de errores comunes y malas prácticas.
    c. Testing automático
        - Ejecución de pruebas unitarias y/o de integración.
        - El pipeline debe detenerse si los tests fallan
    d. Seguridad del código (SAST)
        - Integración de una herramienta de análisis estático de seguridad (Semgrep o SonarQube).
        - Detección de vulnerabilidades en el código fuente antes del despliegue.
    e. Seguridad de dependencias (SCA)
        - Análisis automático de librerías externas.

- Detección de vulnerabilidades conocidas (CVEs).
- El pipeline debe fallar ante riesgos críticos.

f. Build de contenedores

- Construcción de la imagen Docker del sistema.
- Versionado del artefacto generado.

g. Seguridad de contenedores

- Escaneo de la imagen Docker (Trivy o Grype).
- Detección de vulnerabilidades del sistema base y librerías internas.

3. Justificar las decisiones técnicas (obligatorio)

El/la estudiante deberá elaborar un **documento de justificación técnica** donde explique:

- Qué herramienta se usa en cada etapa del pipeline.
- En qué fase de DevSecOps actúa.
- Qué tipo de riesgo mitiga.
- Por qué es necesaria incluso cuando el sistema ya es funcional.

Este documento puede entregarse como:

- README.md en el repositorio **o**
- Documento PDF breve (1–2 páginas).

4. Verificar y evidenciar la ejecución del pipeline

- Ejecutar el pipeline CI/CD.
- Verificar que todas las etapas se ejecutan correctamente.
- Adjuntar evidencia:
  - Logs del pipeline
  - Capturas de pantalla
  - Enlaces a ejecuciones exitosas o fallidas justificadas.

Entregables:

Repositorio Git (fork o copia de **Práctica 2**) que incluya:

- Pipeline CI/CD completo (.yml)
- Código frontend y backend sin modificaciones funcionales mayores
- Documento de justificación técnica
- Evidencia de ejecución del pipeline

Rúbricas

Criterio de evaluación	Excelente	Adecuado	Básico	Insuficiente	Puntaje máx.
Diseño global del pipeline CI/CD con enfoque DevSecOps	Pipeline claramente estructurado, con secuencia lógica completa (Code, Build, Test, Security, Deploy) y enfoque DevSecOps explícito.	Pipeline funcional, con secuencia mayormente correcta pero con debilidades menores.	Pipeline incompleto o con orden poco claro.	Pipeline inexistente o incoherente.	**30**
Selección y ubicación de herramientas	Herramientas correctamente seleccionadas y ubicadas en la fase DevSecOps correspondiente (calidad, testing, SAST, SCA, contenedores).	Mayoría de herramientas bien ubicadas, con errores menores.	Uso de herramientas sin relación clara con la fase correspondiente.	Herramientas mal ubicadas o sin sentido técnico.	**25**
Justificación técnica de decisiones	Justificación clara, profunda y técnica; explica riesgos mitigados y consecuencias de no usar cada herramienta.	Justificación correcta pero superficial o incompleta.	Justificación descriptiva, sin análisis técnico.	No existe justificación técnica.	**25**
Automatización y gates de seguridad	El pipeline actúa como sistema de control automático; el código solo avanza si cumple criterios funcionales y de seguridad.	Existen controles, pero no todos actúan como gates efectivos.	Controles solo informativos, sin bloqueo real.	No existen controles ni gates.	**20**
Total					**100 pts**