Justificación Técnica del Pipeline DevSecOps CI/CD

Este documento describe las decisiones técnicas adoptadas en la construcción del pipeline CI/CD con enfoque DevSecOps. El objetivo es garantizar que el software solo avance en el flujo de integración y despliegue si cumple con criterios de calidad, seguridad y funcionamiento.

1. Instalación reproducible (npm ci)

Herramienta: npm ci
Fase DevSecOps: Build
Riesgo mitigado: inconsistencias de dependencias y builds no deterministas.
Justificación: asegura que todas las dependencias coincidan exactamente con el lockfile, evitando errores entre entornos.

2. Calidad de código (ESLint)

Herramienta: ESLint
Fase DevSecOps: Code Quality
Riesgo mitigado: errores de estilo, malas prácticas y posibles defectos lógicos.
Justificación: mejora la mantenibilidad del código y previene fallos antes de ejecutar el sistema.

3. Testing automático

Herramienta: npm test
Fase DevSecOps: Testing
Riesgo mitigado: regresiones funcionales y errores en lógica de negocio.
Justificación: garantiza que las funcionalidades existentes no se rompan con nuevos cambios.

4. SAST (Semgrep)

Herramienta: Semgrep
Fase DevSecOps: Security (código)
Riesgo mitigado: vulnerabilidades en el código fuente como inyecciones o uso inseguro de funciones.
Justificación: permite detectar problemas de seguridad antes del despliegue.

5. SCA (npm audit)

Herramienta: npm audit
Fase DevSecOps: Security (dependencias)
Riesgo mitigado: vulnerabilidades en librerías externas.
Justificación: incluso si el código propio es seguro, las dependencias pueden introducir riesgos.

6. Build de contenedores (Docker)

Herramienta: Docker Compose
Fase DevSecOps: Build

Riesgo mitigado: diferencias entre entornos y despliegues inconsistentes.
Justificación: permite empaquetar la aplicación de forma estandarizada.

7. Seguridad de contenedores (Trivy)

Herramienta: Trivy
Fase DevSecOps: Security (runtime)
Riesgo mitigado: vulnerabilidades en la imagen de contenedor.
Justificación: protege el entorno de ejecución final del sistema.

8. Smoke Test y Healthcheck

Herramienta: curl /health
Fase DevSecOps: Deploy/Verification
Riesgo mitigado: servicios que no inician correctamente.
Justificación: verifica que el sistema está operativo tras el despliegue.

El pipeline implementado actúa como un mecanismo de control automático que bloquea la integración y despliegue cuando se detectan errores de calidad o seguridad. Esto garantiza que únicamente versiones seguras, funcionales y verificadas lleguen a producción.