🛠️ Ejercicio Práctico

📌 **Título:** Identificación de Buenas Prácticas en un Entorno de Desarrollo Seguro

---

🎯 Objetivo:

Analizar un flujo de desarrollo simple e identificar en qué puntos se pueden aplicar **técnicas básicas de seguridad y automatización**, según las buenas prácticas de **DevSecOps**.

🧱 Escenario:

Eres parte de un equipo de desarrollo que trabaja en una aplicación web. Actualmente, el flujo de trabajo de tu equipo es el siguiente:

1. Los desarrolladores suben código directamente al repositorio `main`, sin revisiones.
2. No se utiliza ninguna herramienta para analizar la calidad ni seguridad del código.
3. Las dependencias del proyecto se instalan sin verificación de versiones ni auditorías.
4. La aplicación se despliega manualmente a producción sin pasar por pruebas automáticas.

📋 Actividades:

1. **Identifica al menos 3 riesgos o malas prácticas en este flujo de trabajo.**
2. **Propón una herramienta o técnica específica para mitigar cada uno de esos riesgos.**
3. **Explica brevemente por qué tu propuesta mejora la seguridad o calidad del desarrollo.**

📌 Formato sugerido para la respuesta:

Riesgo Detectado	Técnica o Herramienta Propuesta	Justificación

---

💡 Recomendaciones:

- Puedes usar herramientas como **SonarQube, Snyk, GitHub Actions, ESLint, análisis SAST/DAST**, etc.
- Sé breve pero claro en tus explicaciones. Este ejercicio busca que apliques conceptos clave de **automatización y seguridad en el desarrollo**.

✅ Solución Modelo – Ejercicio Práctico

📌 **Escenario:** Proyecto de aplicación web sin controles de seguridad ni automatización implementados.

📋 Tabla de solución:

Riesgo Detectado	Técnica o Herramienta Propuesta	Justificación
Código subido directamente a `main` sin revisiones	Uso de **pull requests** y revisiones de código ✅	Permite detectar errores y vulnerabilidades antes de llegar a producción.
Falta de análisis de seguridad del código fuente	Implementar **análisis estático (SAST)** con **SonarQube** ✅	Detecta vulnerabilidades en el código antes de la ejecución del sistema.
Uso de dependencias sin verificar versiones ni parches	Integración de **Snyk** para auditoría de dependencias ✅	Identifica y alerta sobre librerías vulnerables o desactualizadas.
Despliegue manual sin pruebas automatizadas	Configuración de **GitHub Actions** para pruebas CI ✅	Automatiza pruebas de calidad y seguridad antes del despliegue.

💡 Justificación general:

Estas acciones incorporan principios clave de **DevSecOps**, permitiendo que la seguridad sea parte del ciclo de desarrollo desde el inicio. Al combinar revisión de código, análisis automatizado y control de dependencias, el equipo podrá entregar software **más seguro, confiable y sostenible** en el tiempo.
