CONTROL: 8.27 PRINCIPIOS DE CODIFICACIÓN SEGURA

Proyecto: Aplicación Móvil "Alerta Mujer" **Fecha:** 2 de octubre de 2025 **Estado del Control en el Proyecto:** Faltante (F)

1. Propósito

El control 8.27 requiere que se establezcan y apliquen principios, reglas y guías para escribir código y configurar ajustes de manera segura. El objetivo es **prevenir la introducción de vulnerabilidades de seguridad conocidas** (como las catalogadas por el OWASP Top 10) desde la fase inicial de desarrollo.

Para una aplicación crítica como "Alerta Mujer", la codificación segura es esencial para:

- **Integridad:** Evitar la manipulación de la evidencia multimedia o el historial de alertas (RF9.1, 5.25).
- **Confidencialidad:** Garantizar el correcto uso de las funciones de cifrado y el manejo de PII (RNF4.1, 5.31).
- **Disponibilidad:** Prevenir errores de *software* que puedan causar fallos en la activación de la alerta (RNF6, 5.26).
-

2. Procedimiento de Aplicación del Control 8.27

La implementación de este control se estructura en dos pilares: la **Guía Formal** y la **Aplicación con Herramientas**.

2.1. Adopción de la Guía de Codificación Segura

- **Estándar de Referencia:** El equipo de desarrollo adoptará formalmente el **OWASP Top 10** como el conjunto mínimo de riesgos a mitigar en el código. Para la aplicación móvil, se priorizará la guía **OWASP Mobile Application Security (MAS)**.
- **Guía Interna:** Se creará un documento de **"Principios de Codificación Segura para Alerta Mujer"** que incluirá:
    - **Validación de Entradas:** Reglas estrictas para validar y sanear **todas las entradas de usuario** (ej. nombres, contraseñas, mensajes) para prevenir ataques de inyección (SQL, Comandos).

- **Manejo de Errores y Excepciones:** Políticas para que los mensajes de error no revelen información sensible (ej. detalles del sistema, rutas de archivos).
- **Uso de Criptografía:** Reglas para la correcta implementación del cifrado (RF1.3, RNF4.1), incluyendo la gestión segura de las claves y el uso de algoritmos fuertes (5.32).
- **Manejo de Sesiones:** Asegurar la invalidación de sesiones al cerrar la sesión o por inactividad.

2.2. Integración de Herramientas de Análisis de Código Estático (SAST)

Para hacer cumplir la guía anterior de forma sistemática y automatizada:

- **Selección de Herramienta:** Se seleccionará e implementará una herramienta de **SAST** (ej. SonarQube, Bandit, u otra adecuada para los lenguajes de programación utilizados, ej. Java/Kotlin para Android, Python para *backend*).
- **Integración en CI/CD:** La herramienta SAST se integrará en la pipeline de Integración Continua / Despliegue Continuo (CI/CD).
- **Criterio de Bloqueo:** El sistema SAST se configurará para que **bloquee automáticamente** la fusión (*Merge*) de código si detecta vulnerabilidades de nivel **Crítico o Alto** o si el código incumple reglas de seguridad predefinidas.
- **Capacitación:** El equipo de desarrollo recibirá formación obligatoria sobre cómo utilizar los resultados del SAST para corregir vulnerabilidades (control 6.3 faltante)

LIDER DEL PROYECTO. EQUIPO DE TRABAJO. EQUIPO DE TRABAJO.