## 

## PLANIFICA VZLA, C.A.

## Pre-Informe de Auditoría de Código Fuente

**Empresa Auditora:** PLANIFICA VZLA, C.A. **Empresa Auditada:** DataShield Systems S.L. **Sistema Auditado:** SISTEMA DE GESTIÓN DE INFORMACIÓN Y MANTENIMIENTO EN LA PLANTA BUS YARACUY (REGIM) **Fecha del Pre-Informe:** 15 de octubre de 2025

### Resumen Ejecutivo de Hallazgos

El análisis estático y manual del código fuente de **REGIM** revela la presencia de vulnerabilidades de **alto riesgo** que comprometen la **confidencialidad** (manejo de secretos) y la **integridad** (riesgo de inyección) de los datos de mantenimiento y flotas. Los fallos principales se centran en la configuración del entorno y la interacción con la capa de base de datos.

### Matriz de Hallazgos Evidenciados

| ID | Hallazgo Evidenciado | Instrumentos / Evidencia Recopilada | Nivel de Riesgo |
| --- | --- | --- | --- |
| **H-1** | **Exposición de Claves Privadas en Frontend (Next.js):** Se identificaron *tokens* de API y claves de servicio privados expuestos en variables de entorno con prefijo NEXT\_PUBLIC\_, haciéndolos visibles al navegador del usuario. | Checklist (CH-5), Evidencia E-6 (Configuración de Entorno). | **CRÍTICO** |
| **H-2** | **Riesgo de *SQL Injection* por Prisma.raw():** El módulo de Órdenes de Mantenimiento utiliza la función Prisma.raw() para ejecutar sentencias SQL. No se encontró la parametrización adecuada de las variables de entrada de usuario. | Checklist (CH-1), Evidencia E-1 (Código Fuente). | **ALTO** |
| **H-3** | **Almacenamiento Débil de Contraseñas:** Las contraseñas se *hashean* usando una implementación sencilla y no especializada en TypeScript (ej. módulo Usuarios con MD5), lo que las hace vulnerables a ataques de fuerza bruta. | Checklist (CH-2), Evidencia E-1 (Código Fuente). | **ALTO** |

### Conclusiones Preliminares

1. **Vulnerabilidad Crítica de Configuración:** La arquitectura de la aplicación en Next.js no implementa el principio de separación de secretos (H-1). Esto representa el riesgo más alto, ya que expone credenciales que podrían ser utilizadas por un atacante para comprometer la integridad de la base de datos o de otros sistemas externos.
2. **Riesgo de Integridad de Datos:** La detección de Inyección SQL (H-2) debido al uso incorrecto de Prisma.raw() demuestra una deficiencia en la validación de datos. Esto es un incumplimiento de las prácticas de codificación segura y podría permitir la alteración o la extracción masiva de información de mantenimiento.
3. **Falta de Proceso de Seguridad:** La auditoría de cumplimiento (Cuestionario P-1) confirmó que la ausencia de una política formal de *Code Review* centrada en seguridad contribuye directamente a que estos fallos de alto riesgo lleguen al entorno de pruebas o producción.

### Recomendaciones Inmediatas

1. **Prioridad (CRÍTICO - H-1):** Mover inmediatamente todas las claves privadas a un **Gestor de Secretos** (o a archivos de configuración internos de Next.js sin el prefijo NEXT\_PUBLIC\_), asegurando que solo sean accesibles desde las API Routes (Backend).
2. **Reingeniería de Queries (H-2):** Sustituir el uso de Prisma.raw() por las **funciones ORM seguras de Prisma** o, si es estrictamente necesario, garantizar la correcta parametrización de las variables en las consultas SQL *raw*.
3. **Actualización de Hashing (H-3):** Migrar el algoritmo de *hashing* de contraseñas a una función robusta y moderna (ej. **bcrypt** o **Argon2**).