

PRD – Sistema de Rondines Integrado (Opción 1)

1. Resumen Ejecutivo

El proyecto consiste en reemplazar completamente el sistema obsoleto de descarga, interpretación y reporte de rondines de vigilancia, manteniendo los lectores físicos actuales conectados por puerto COM. El nuevo sistema será una solución moderna, integrada y centralizada, basada en React, Supabase y Railway, con un agente local para la descarga directa del hardware.

2. Objetivo del Producto

- Reemplazar el software legacy de descarga.
 - Automatizar la interpretación de rondines.
 - Detectar rondines completos, incompletos, inválidos y no realizados.
 - Generar reportes automáticos en segundos.
 - Centralizar la información en la nube.
 - Reducir tiempos operativos y errores humanos.
-

3. Alcance

Incluye

- Descarga directa del lector vía COM.
- Parser de datos crudos.
- Motor de rondas cíclicas cerradas.
- Gestión de rutas, estaciones y tolerancias.
- Gestión de turnos 24x48.
- Reportes web y exportables.

No incluye (fase inicial)

- App móvil.
 - Evidencia fotográfica o comentarios.
 - GPS o geolocalización.
-

4. Contexto Operativo

- Vigilantes con rol **24x48**.
 - Rondines cíclicos cada **2 o 3 horas**.
 - Una misma ruta se repite durante el turno.
 - Cada estación tiene un **TAG fijo**.
 - El rondín **inicia y termina obligatoriamente en la Estación 1**.
 - Validación únicamente por TAG.
-

5. Reglas de Negocio

5.1 Ronda válida

- Secuencia: $E1 \rightarrow E2 \rightarrow \dots \rightarrow E_n \rightarrow E1$.
- Cumple orden y tiempos dentro de tolerancia.

5.2 Ronda incompleta

- Falta una o más estaciones.
- Existe al menos un retardo fuera de tolerancia.

5.3 Ronda inválida

- No inicia en Estación 1.
- No finaliza en Estación 1.

5.4 Ronda no realizada

- No existe inicio (E1) dentro de la ventana de tiempo esperada.

6. Ventanas de Tiempo

- Las rondas se esperan según la frecuencia de la ruta (120 o 180 minutos).
- Cada ventana genera una ronda esperada.
- Si no hay inicio en E1 dentro de la ventana, se marca como **NO REALIZADA**.

7. Arquitectura

```
[Lector COM]
  ↓
[Agente Local Node.js]
  ↓ HTTPS
[Backend API - Railway]
  ↓
[Supabase - DB + Auth]
  ↓
[React Web]
```

8. Componentes del Sistema

8.1 Agente Local de Descarga

- Corre en Windows.
- Detecta puerto COM.

- Lee datos del lector.
- Envía datos crudos al backend.
- Reemplaza totalmente el software anterior.

8.2 Backend (Railway)

- API REST.
- Parseo de datos.
- Motor de rondas.
- Persistencia en Supabase.
- Control de duplicados y auditoría.

8.3 Base de Datos (Supabase)

- PostgreSQL.
- Autenticación y roles.
- Políticas RLS.

8.4 Frontend (React)

- Dashboard de rondines.
- Gestión de rutas y estaciones.
- Reportes por turno, fecha y vigilante.

9. Modelo de Datos (Resumen)

vigilantes

- id
- nombre
- activo

turnos

- id
- vigilante_id
- inicio
- fin

rutas

- id
- nombre
- frecuencia_min

estaciones

- id
- ruta_id
- nombre
- tag

- orden
- tiempo_esperado_seg
- tolerancia_seg

eventos

- id
- tag
- fecha_hora
- procesado

rondas

- id
- ruta_id
- turno_id
- inicio
- fin
- estatus

ronda_detalle

- ronda_id
 - estacion_id
 - orden
 - fecha_hora
 - diferencia_seg
 - estatus
-

10. Estatus de Ronda

- COMPLETA
 - INCOMPLETA
 - INVALIDA
 - NO_REALIZADA
-

11. Reportes

- Rondas por vigilante.
 - Rondas no realizadas.
 - Cumplimiento por turno.
 - Retardos por estación.
 - Histórico por fecha.
-

12. Seguridad

- Agente local autenticado por token.

- API protegida.
 - Supabase RLS.
 - Logs de descarga y procesamiento.
-

13. Roadmap

Fase 1

- Agente local + backend + reportes básicos.

Fase 2

- Dashboards avanzados.
- Alertas automáticas.

Fase 3

- Migración opcional a app móvil.
-

14. Tiempo Estimado

- Desarrollo total: **3-4 semanas**.
-

15. Criterios de Éxito

- Descarga sin software legacy.
- Procesamiento automático.
- Reportes en segundos.
- Reducción significativa de tiempo operativo.