Informe Técnico – Gestión de Incidentes

1. Decisiones técnicas tomadas

Para el desarrollo de la API se utilizó ASP.NET Core 8 debido a su alto rendimiento, arquitectura modular y soporte nativo para APIs REST. Se implementó Entity Framework Core con SQLite como base de datos por su simplicidad durante la prueba; sin embargo, la arquitectura queda preparada para migrar fácilmente a SQL Server o PostgreSQL en ambientes productivos. Se incorporó un middleware de manejo global de excepciones para asegurar respuestas consistentes y mejorar el monitoreo de errores. Se utilizó ILogger para registrar eventos relevantes como creación, actualización y errores en los endpoints. Para garantizar calidad, se implementaron pruebas unitarias con xUnit usando EF Core InMemory, permitiendo validar reglas de negocio sin depender de la base de datos real. El diseño se estructuró en capas claras (Controllers, Services, Data, Models, DTOs), lo que mejora la mantenibilidad y escalabilidad del sistema.

2. Organización del equipo en un sprint

En un sprint de dos semanas, el equipo se organizaría en tres roles principales: Backend Developer(s), DevOps/Infra y QA. El flujo de trabajo seguiría la metodología Scrum, con reuniones diarias cortas y revisiones de sprint. Las tareas se asignarían por prioridad técnica y de negocio, asegurando entregables incrementales.

3. Riesgos técnicos y mitigación

• Alta carga (1000 incidentes/min): Escalamiento horizontal, indexación, optimización SQL, NoTracking y posibilidad de colas (RabbitMQ/Azure Service Bus).
• Inconsistencias por concurrencia: Uso de transacciones y concurrencia optimista.
• Falta de trazabilidad: Middleware global + logging estructurado.
• Caídas del servicio: Health checks, retry policies (Polly) y despliegue en contenedores.

4. Historias de usuario para un sprint de dos semanas

Prioridad Alta (Día 1–6):
- Crear incidente.
- Consultar incidentes.
- Consultar incidente por ID.
- Actualizar incidente.

Prioridad Media (Día 6–10):
- Registrar comentarios.
- Obtener comentarios.

Prioridad Baja (Día 10–14):
- Middleware + logging.
- Pruebas unitarias.
- Documentación.

Justificación del orden: Se priorizan primero las funcionalidades esenciales del flujo CRUD, seguidas de trazabilidad y calidad técnica.