Ejercicio Técnico - Desarrollador Full-Stack

Objetivo

Implementar un sistema web para gestionar facturas, integrando datos desde un archivo JSON, almacenándolos en una base de datos y permitiendo su administración mediante una interfaz intuitiva.

Requisitos Claves

- **Integración desde JSON**
    - Cargar facturas desde un archivo json, tomando como base "**bd_exam.json**" en tiempo de ejecución.
    - "***invoice_number***" único.
    - Coherencia entre suma de subtotales de productos y "total_amount". Si no coincide, marcar como inconsistente y excluir del sistema activo (pero mantener en base para reporte).
    - Calcular de forma automática el estado de facturas:
        - "**Issued**" sin notas de crédito.
        - **"Cancelled"** suma montos NC igual al monto total de la factura.
        - **"Partial"** suma de montos NC es menor al monto total.
    - Calcular de forma automática el estado de pago de la factura según:
        - "**Pending**" pago pendiente dentro del plazo.
        - **"Overdue"** si a la fecha se encuentra vencida (payment_due_date).
        - **"Paid"** pago registrado.

Funcionalidades Principales

- **Búsqueda**
    - Permitir buscar facturas por número y estado de factura y estado de pago (utilizar vistas).
- **Gestión de notas de crédito (NC)**
    - Agregar NC a una factura con fecha de creación automática.
    - Validar que el monto de NC no supere el saldo pendiente de la factura. No se puede superar el monto pendiente de la factura.

- **Reportes**
    - Facturas consistentes, con más de 30 días vencidas sin pago o nota de crédito.
    - Resumen total y porcentaje de facturas por estado de pago (Paid, Pending, Overdue).
    - Facturas inconsistentes (total $ declarado no coincide con suma $ de productos).

Tecnologías requeridas

- **Backend:**
    - Implementado en **.NET 6 u 8**.
    - Exponer mediante API REST
    - Uso de **Swagger** para documentación.
    - Usar códigos HTTP de respuesta de los endpoints.
    - De forma opcional, agregar autenticación a la API.
- **Base de Datos:**
    - Utilizar **SQLite** (referencia) + Entity Framework Core (u otro ROM).
- **Frontend:**
    - React con interfaz intuitiva y validaciones en formularios.

Buenas prácticas de desarrollo

- Se debe implementar con **algún patrón de diseño** a elección del desarrollador y mencionarlo en el **README.md**.

Entrega Final

- Enviar al correo el repositorio git del código fuente del front, back y Scripts de base de datos (si aplica).
- Agregar en el repositorio, un archivo readme.md con instrucciones claras para instalación y ejecución, si hay autenticación incluir credenciales.
- La evaluación considerará la implementación hasta el último commit dentro del plazo. Se recomienda entregar la mejor versión posible (estable), priorizando la calidad del código y la organización del proyecto.