Prueba técnica para backend

Contexto: En un parqueadero puede:

1. Inicia el parqueadero (El parqueadero abre sus puertas).
2. Ingresar un vehículo (es cuando ingresan al parqueadero).
3. Facturar un vehículo (Cuando van al punto de pago).
4. Retirar un vehículo (Al momento de salir del parqueadero).
5. Cierre del parqueadero (Acaba el parqueadero)

Desarrollo:

1. Debes crear una base de datos llamada **parqueadero** con las siguientes tablas para cada uno de los puntos anteriores **(Requerido)**

Tablas:

1. **inicio**
   id_inicio (auto incremental), fecha_inicio (datetime)
2. **ingresos**
   id_ingreso (auto incremental), placa (varchar 5), id_inicio, fecha_factura (datetime)
3. **facturas**
   id_factura (auto incremental), id_ingreso, valor_factura (double), fecha_factura (datetime)
4. **salidas**
   id_salida (auto incremental), id_factura, fecha_salida (datetime)
5. **cierres**
   id_cierre (auto incremental), id_inicio, total_facturas (double), fecha_cierre (datetime)

Las tablas deben estar creadas en MySQL manteniendo la estructura y relaciones según lo descrito anteriormente.

2. Debes crear un API desarrollado en su totalidad con C# Net Core 6 o superior y esta debe contar con 5 servicios: **(Requerido)**

1. **Guardado de inicio del parqueadero**: En este se debe guardar los datos de la tabla **inicio** y este servicio debe devolver el id_inicio

2. **Guardado de ingresos:** Este debe guardar los datos de la tabla **ingresos** y debe recibir el id_inicio en caso no exista en la tabla **inicio** se debe devolver un mensaje de error especificando el problema. En caso contrario que todo esté correcto debe devolver el id_ingreso

3. **Guardado de facturas:** Este debe guardar los datos de la tabla **facturas** y debe recibir el id_ingreso como en el punto anterior si no existe el id_ingreso en la tabla **ingresos** debe devolver un error en caso contrario debe devolver el id_factura

4. **Guardado de salidas:** Este debe guardar los datos de la tabla **salidas** y debe recibir el id_factura y la misma lógica de los puntos anteriores.

5. **Guardado de cierres:** Este debe guardar los datos de la tabla **cierres** y debe recibir el id_inicio y la misma lógica de los puntos anteriores.

Todos los servicios deben ser tipo POST el request debe ser un json. En caso falte algún campo de cada servicio se debe devolver un mensaje de error diciendo que campo hace falta.

3. Debe contar con una documentación en swagger. **(Requerido)**
4. El código debe estar en github. **(Requerido)**
5. Adjuntar al repositorio las indicaciones para correr y probar su proyecto. **(Requerido)**
6. Test unitarios **(Opcional).**
7. Base de datos PostgreSQL. **(Opcional)**
8. El proyecto Dockerizado **(Opcional)**