TOPICOS ESPECIAIS DE SISTEMA – ADS OSÓRIO NOITE (Sexta-Feira)

PROJETO: API RESTFUL

RGM 37230255 – Daniel Henrique Rocha

RGM 37298135 - João Paulo Bertagia

RGM 38988879 - Matheus Gustavo Saldanha Folle

CURITIBA

DOCUMENTAÇÃO TÉCNICA DO PROJETO
API DE VEÍCULOS FURTADOS

1. Objetivo

Desenvolver uma API completa para cadastro de veículos furtados, com rotas para listagem, busca por ID, inserção e exclusão.

2. Estrutura da Solução

- Entidade principal: Veiculo
- Campos: Id, Modelo, Marca, Cor, Placa, Chassis, Status
- Banco: SQLite (configurado com EF Core)
- Organização de código em arquivos separados (GET, POST, DELETE)
- Dados de seed: 10 registros iniciais configurados no OnModelCreating

3. Endpoints Implementados

GET	/api/veículos	→ Lista todos os veículos
GET	/api/veiculos/{id}	→ Busca veículo por ID
POST	/api/veículos	→ Cadastra novo veículo
DELETE	/api/veiculos/{id}	→ Remove veículo do sistema

4. Organização do Código

A API foi desenvolvida seguindo boas práticas de estruturação de projetos, com foco em legibilidade, separação de responsabilidades e manutenção facilitada.

Models/Veiculos.cs: Representa a entidade principal da aplicação com todas as suas propriedades.

Data/VeiucloContext: Responsável por configurar a conexão com o banco de dados SQLite.

Rotas/ROTA_GET.cs: Rotas para listagem de todos os veículos e busca por ID.

Rotas/ROTA_POST.cs: Rota para inserção de novo veículo.

Rotas/ROTA_DELETE.cs: Rota para exclusão de veículo.

Program.cs: Responsável por iniciar a aplicação, configurar o serviço de banco de dados e mapear todas as rotas, incluindo uma rota raiz com link direto para a listagem de veículos.

5. Justificativa Técnica

A modelagem da entidade principal — o veículo — foi feita com base nos atributos essenciais para identificação e rastreamento de ocorrências de furto, garantindo coerência com o contexto real de sistemas de segurança pública e monitoramento. Essa modelagem orientou toda a estrutura da aplicação, desde a definição do banco de dados até o mapeamento das rotas.

A separação entre os arquivos responsáveis pelas rotas (GET, POST, DELETE), o modelo de dados (Veiculo.cs) e o contexto do banco (VeiculoContext.cs) proporciona clareza na manutenção do sistema e favorece a escalabilidade futura da aplicação, como a inclusão de novos endpoints, status adicionais ou autenticação de acesso.