DOCUMENTAÇÃO TÉCNICA DO PROJETO

– API DE GERENCIAMENTO DE CARROS

Título do Projeto: Cadastro de Carros

Grupo:

Guilherme Pontes Mendonça

Lucas Antonio Salomão Jarek

Vinicius da Costa Silva

Wallacy Gabriel Alves Bandeira

1. **Objetivo do Projeto**

O objetivo deste projeto é desenvolver uma API voltada à gestão de um acervo de carros, utilizando tecnologias atuais de desenvolvimento web. A aplicação foi concebida para o funcionamento de um sistema de gerenciamento de veículos, possibilitando operações como consulta, cadastro e exclusão de carros de forma estruturada e eficiente. Esse modelo é aplicável em contextos com grande volume de veículos, nos quais é fundamental garantir agilidade na localização, registro e manutenção de informações por parte dos usuários responsáveis pelo controle do acervo.

- **Estrutura da Solução**

2.1 Modelagem de Dados

A modelagem da aplicação foi centrada na entidade "Carro", definida com base nos requisitos de um sistema de gerenciamento de veículos básico. Os atributos considerados (como Id, Modelo, Marca, Ano e Placa) foram selecionados para representar informações essenciais para a identificação e classificação de cada item no acervo. Essa modelagem foi transformada em uma classe na camada de domínio (`Carro.cs`), servindo como base para a construção do banco de dados (via `AppDbContext.cs`) e definição das regras de negócio.

2.2 Integração da API com a Solução

A API foi estruturada para atuar como a interface de comunicação entre o sistema e os dados armazenados. Cada rota foi implementada para representar uma funcionalidade específica da aplicação, como consulta geral, busca por ID, cadastro e exclusão de carros. Através dessa estrutura, a solução permite que qualquer cliente (como uma aplicação front-end ou ferramenta de testes) possa interagir com a base de dados de forma segura, rápida e consistente.

3. **Endpoints da API**

A aplicação foi estruturada com rotas separadas para cada tipo de operação. Abaixo estão as rotas implementadas, conforme definido nos arquivos de endpoints da API:

Método	Rota	Descrição
GET	`/api/carros`	Retorna todos os carros.
GET	`/api/carros/{id}`	Retorna um carro por ID.
POST	`/api/carros`	Adiciona um novo carro.
DELETE	`/api/carros/{id}`	Remove um carro por ID.

4. Organização do Código

O projeto foi dividido em componentes com responsabilidades bem definidas, promovendo clareza e modularidade:

- `Models/Carro.cs` – Define a estrutura da entidade principal (o modelo de dados para um carro), incluindo suas propriedades como `Id`, `Modelo`, `Marca, Ano` e `Placa`.
- `Data/AppDbContext.cs` – Responsável pela configuração da base de dados (utilizando Entity Framework Core) e pela definição do `DbSet` para a entidade `Carro`, permitindo a interação com o banco de dados.
- `Endpoints/GetAllCarros.cs` – Contém o método de extensão que mapeia a rota `GET /api/carros` para retornar todos os carros do banco de dados.
- `Endpoints/GetCarroById.cs` – Contém o método de extensão que mapeia a rota `GET /api/carros/{id}` para retornar um carro específico por seu `Id`.
- `Endpoints/DeleteCarro.cs` – Contém o método de extensão que mapeia a rota `DELETE /api/carros/{id}` para remover um carro do banco de dados com base em seu `Id`.
- `Program.cs` – O arquivo principal de inicialização da aplicação. Ele configura os serviços necessários (como o `DbContext`), constrói o `WebApplication` e mapeia os endpoints da API, utilizando os métodos de extensão definidos nos arquivos da pasta `Endpoints`.
- `appsettings.json` – Arquivo de configuração base da aplicação, utilizado para definir configurações como níveis de log e hosts permitidos.
- `appsettings.Development.json` – Arquivo de configuração específico para o ambiente de desenvolvimento, que pode sobrescrever as configurações de `appsettings.json` para fins de depuração e desenvolvimento.
- `CadastroCarrosAPI.csproj.nuget.dgspec.json` – Arquivo de metadados do projeto que descreve as dependências do pacote NuGet, as versões do framework e outras configurações de build.
- `Repositories/CarroRepository.cs` – Uma implementação de repositório em memória para a entidade `Carro`. Embora presente, a estrutura com `AppDbContext` sugere que a persistência de dados pode estar migrando para o Entity Framework Core.
- `codigos.js` – Um arquivo JavaScript que demonstra a interação de um cliente (frontend) com a API, realizando operações como carregar, adicionar e deletar carros. É importante notar que as

rotas utilizadas neste arquivo (`/carros`) podem precisar de alinhamento com as rotas definidas na API (`/api/carros`).

5. **Justificativa Técnica**

A modelagem da entidade principal — o carro — foi feita com base nos atributos essenciais para identificação e catalogação, garantindo coerência com o contexto funcional de um sistema de gerenciamento de veículos. Essa modelagem orientou toda a arquitetura da aplicação, incluindo a configuração da base de dados, a definição das rotas e a estrutura do código.

A separação entre os arquivos responsáveis pelas rotas, o modelo de dados e o contexto do banco de dados adequa clareza na manutenção e favorece a evolução futura da aplicação, caso novas funcionalidades precisem ser incorporadas, como atualização de registros ou autenticação de usuários.