Teste ASP.NET Core

Wagner de Almeida Santos

São Paulo, 24 de Dezembro de 2020

Sumário

Introdução

A aplicação de teste tem um backend desenvolvido usando .NET Core. Na parte de ORM usando EntityFramework Core, Sql Server e Migrations. O banco de dados é criado automaticamente utilizando o método Migrate, e a opção por esse método ao invés do EnsuredCreated, é devido ao Migrate olhar para as Migrations. As novas Migrations adicionadas depois da criação do banco serão aplicadas, e além de aplicar as Migrations, o método Migrate cria o banco de dados caso ainda não exista.

Execução

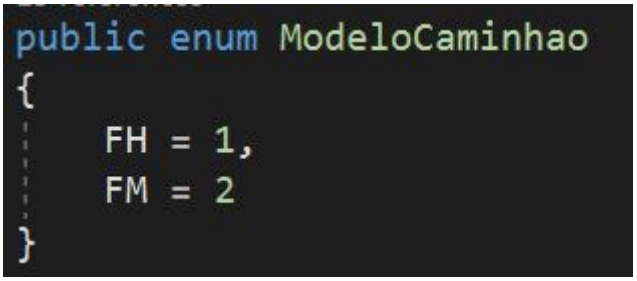
Para executar a aplicação com sucesso, é necessário realizar o build para carregar as dependências do projeto. Feito isso, basta executar a aplicação. Com o programa em execução, é possível visualizar a documentação da API utilizando Swagger, como seus respectivos endpoints, modelos, enumerados, etc. Na parte superior do Swagger, temos as informações de contato, onde as informações sobre a API podem ser acessadas, além de poder contatar o responsável pelo desenvolvimento.

Performance

Além de contar com a documentação de API, o sistema foi projetado para ter boa performance. Os dados são paginados, ou seja, quem consome o serviço da API define a quantidade de dados que quer receber, e caso nenhum valor seja informado, por padrão o sistema retorna 15 valores por página.

Parâmetros API

Para realizar o registro de um novo caminhão existem alguns parâmetros para serem passados, como: Nome, Ano Modelo e Modelo, sendo o modelo um enumerado, onde o número 1 diz respeito a FH e 2 FM. Caso seja passado algum valor diferente desse, o sistema realiza a validação retornando a mensagem de erro.
Modelo do Enumerado:



Validações

Na parte de validações, o sistema conta com o suporte do FluentValidation, uma biblioteca para validações. Todos os erros são armazenados em uma coleção e retornados em um objeto Errors que contém o array com as mensagens de erro.

Boas Práticas

Utilizando boas práticas e conceitos do SOLID, toda a parte de acesso a métodos é feita por interfaces, onde injetamos via construtor, realizando a injeção de dependência utilizando o container do ASP.NET core.