Prova técnica da Localiza

Por: Gustavo Henrique Alves do Carmo

1) Conceitos iniciais para o desenvolvimento

Para este desafio foi escolhido apenas o desenvolvimento para a aplicação Back-end. A API foi especificada para ser em .NET 5.0.

A arquitetura utilizada foi a Clean Architecture.

Banco de dados MySQL.

2) Início do desenvolvimento

A API foi iniciada com a criação da solução e do conjunto de projetos que fazem parte do sistema. Os projetos existentes são:

- LocacaoVeiculo.Domain:

com a definição do modelo dos domínios e interfaces.

- LocacaoVeiculo.Infrastructure:

com a definição da lógica de acesso a dados, contexto, configurações e ORM.

- LocacaoVeiculo.Application:

com a definição das regras de aplicação, serviços, mapeamentos e DTOs.

- LocacaoVeiculo.CrossCutting:

projeto compartilhado onde estão definidos a injeção de dependência e o registro dos serviços e recursos usados pelos demais projetos.

- LocacaoVeiculo.API: controladores

com a definição dos controladores.

Nesta solução os projetos ficaram estruturados com as seguintes dependências:

LocacaoVeiculo.Domain:

-nenhum

LocacaoVeiculo.Infrastructure:

- LocacaoVeiculo.Domain

LocacaoVeiculo.Application:

- LocacaoVeiculo.Domain

LocacaoVeiculo.CrossCutting:

- LocacaoVeiculo.Domain

- LocacaoVeiculo.Infrastructure

- LocacaoVeiculo.Application

LocacaoVeiculo.API:

- LocacaoVeiculo.CrossCutting

3) Desenvolvimento

3.1) Cadastro, autenticação e autorização de usuário, operador e cliente

Para a validação do usuário, foi levantado que haveria as tabelas de Usuarios, Clientes e Operadores. Como um login cujo valor corresponda a um CPF válido seria considerado um cliente, foi adicionado um método de extensão para a classe String para facilitar nesta identificação. Sendo identificado como CPF, o usuário seria adicionado na tabela Usuarios com o campo chave formatado com 11 algarismos, mesmo que o valor inicial não estivesse neste formato (CPFs que começam com zero ou com pontuação).

Para completar o cadastro do cliente ou operador, o usuário precisa ainda cadastrar seus dados pessoais conforme seu perfil. Um usuário com chave igual a CPF, só tem acesso para a API de cadastro de Clientes e um usuário com chave diferente de CPF, só tem acesso ao cadastro de Operadores.

Obs.: As tabelas desta aplicação foram adicionadas via Entity Framework no formato code-first.

3.2) Inclusão do token e perfis de utilização

Para a utilização de autenticação e autorização da API foi adicionado o recurso do Bearer Token. No momento da inclusão dos usuários, conforme o valor da chave (CPF ou outro), o token era gerado com um dos perfis: Cliente ou Operador. Estes perfis foram configurados para permitir os seguintes acessos:

Cliente: CRUD de clientes

Operador: CRUD de operadores

3.3) Inclusão de consulta por usuário

A API possui acesso aos dados dos clientes e operadores por suas respectivas rotas de clientes e operadores. Para buscar os dados cadastrais, independente do perfil, foi adicionada uma consulta que retorna esta informação, de acordo com o login realizado.

3.4) Inclusão dos domínios

Para esta aplicação foram adicionados os seguintes domínios:

-Marcas

-Modelos

-Veiculos

Para cada um deles foram criados os controladores com as funções de um CRUD. Cada etapa desta implementação foi colocada em seu respectivo projeto da solução.

3.5) Serviços de simulação de locação, agendamento e checklist

Para a execução destes serviços, foram adicionadas aplicações que através dos dados dos veículos e de outros valores externos (Ex: total de horas), seriam retornadas sem o arquivamento em banco.

3.6) Modelo de contrato de locação via PDF

Para este item, foi disponibilizado apenas um retorno pela API de um arquivo em PDF modelo para locação. A geração do arquivo via código não foi contemplada nesta aplicação.

4) Testes Unitários

Um projeto para testes unitários foi criado após a criação da API. Ele foi criado como um projeto xUnit.

5) Rotas da API LOCACAO.VEICULO

**Atividades (Roles = "Operador, Cliente")**

POST ​/api​/Atividades​/simulacaolocacao

POST ​/api​/Atividades​/agendamento

POST​ /api​/Atividades​/ckecklist

GET ​ /api​/Atividades​/modelocontrato

**Authorize**

GET (Roles = "Operador, Cliente") ​/api​/Authorize

POST ​/api​/Authorize​/register

POST ​/api​/Authorize​/login

**Clientes (Roles = "Cliente")**

GET ​ /api​/Clientes

POST ​ /api​/Clientes

GET ​ /api​/Clientes​/{id}

PUT ​ /api​/Clientes​/{id}

DELETE ​/api​/Clientes​/{id}

GET ​/api​/Clientes​/CPF​/{CPF}

**Marcas**

GET (Roles = "Operador, Cliente") ​/api​/Marcas

POST (Roles = "Operador") ​/api​/Marcas

GET (Roles = "Operador, Cliente")​ /api​/Marcas​/{id}

PUT (Roles = "Operador") ​/api​/Marcas​/{id}

DELETE (Roles = "Operador") ​/api​/Marcas​/{id}

**Modelos**

GET (Roles = "Operador, Cliente") ​/api​/Modelos

POST (Roles = "Operador") ​ /api​/Modelos

GET (Roles = "Operador, Cliente") ​/api​/Modelos​/{id}

PUT (Roles = "Operador") ​/api​/Modelos​/{id}

DELETE (Roles = "Operador") ​/api​/Modelos​/{id}

**Operadores (Roles = "Operador")**

GET ​ /api​/Operadores

POST ​ /api​/Operadores

GET ​ /api​/Operadores​/{matricula}

PUT ​ /api​/Operadores​/{matricula}

DELETE ​/api​/Operadores​/{matricula}

**Veiculos**

GET (Roles = "Operador, Cliente") ​/api​/Veiculos

POST (Roles = "Operador") ​/api​/Veiculos

GET (Roles = "Operador, Cliente")​ /api​/Veiculos​/{id}

PUT (Roles = "Operador")​ /api​/Veiculos​/{id}

DELETE (Roles = "Operador") ​ /api​/Veiculos​/{id}