Sistema de Locadora de Veículos

Relatório Técnico

Pedro Henrique Rodrigues Evangelista

2025

Resumo

Este relatório apresenta o desenvolvimento completo de um sistema de gerenciamento para uma Locadora de Veículos, incluindo API em C#, banco de dados com Entity Framework Core, CRUDs, consultas complexas com relacionamentos, e front-end em HTML, CSS (Bootstrap) e JavaScript. São descritos o problema, a modelagem, os artefatos gerados e as funcionalidades implementadas.

1 Introdução

O objetivo deste projeto é desenvolver um sistema completo para auxiliar o gerenciamento de uma Locadora de Veículos. O sistema deve permitir o controle de clientes, fabricantes, veículos, aluguéis e pagamentos, fornecendo ainda consultas especializadas para apoiar decisões de negócio.

Para isso, foi implementada uma API REST em ASP.NET Core (C#), conectada a um banco de dados SQL Server via Entity Framework Core. O front-end foi desenvolvido em HTML com Bootstrap e consome a API via Fetch API.

O sistema também apresenta uma interface de consultas onde o usuário pode extrair relatórios em tempo real.

2 Descrição do Problema

A empresa de locação necessitava de um sistema que permitisse:

- Cadastrar, editar, listar e excluir clientes.
- Cadastrar fabricantes de veículos.
- Gerenciar veículos ligados aos fabricantes.

- Registrar aluguéis associando clientes e veículos.
- Registrar pagamentos vinculados aos aluguéis.
- Consultar dados consolidados com relacionamentos (joins).
- Ter uma interface simples que pudesse ser utilizada diretamente via navegador.

Antes do sistema, o processo era totalmente manual, dificultando análises e relatórios.

3 Modelagem Conceitual

A seguir apresentamos o modelo conceitual simplificado, utilizando notação textual:

```
Cliente (1) -------- (N) Aluguel (N) -------- (1) Veiculo (N) -------- (1) Fabricante

                         |
                         |
                         N
                         |
                         |
                    Pagamento
```

4 Modelos da Aplicação

Os principais modelos criados em C# foram:

- **Cliente**: Nome, CPF, Email, Telefone.
- **Fabricante**: Nome e País.
- **Veiculo**: Modelo, Ano, Cor, Placa e FK de Fabricante.
- **Aluguel**: Datas, KM, Cliente, Veículo.
- **Pagamento**: Data, Valor, Forma de Pagamento e FK de Aluguel.

Todos os modelos foram criados utilizando atributos de validação (DataAnnotations).

5 Banco de Dados e Entity Framework

A solução usa:

- **Code-First com Migrations**
- **Relacionamentos 1:N e N:1 mapeados com ForeignKey**
- **SQL Server como banco de dados**

O contexto foi definido na classe:

```
public class LocadoraBD : DbContext
{
    public DbSet<Cliente> Clientes { get; set; }
    public DbSet<Fabricante> Fabricantes { get; set; }
    public DbSet<Veiculo> Veiculos { get; set; }
    public DbSet<Aluguel> Alugueis { get; set; }
    public DbSet<Pagamento> Pagamentos { get; set; }
}
```

6 API REST em C#

Foram implementados controladores REST para todas as entidades:

- **/api/Clientes**
- **/api/Fabricantes**
- **/api/Veiculos**
- **/api/Alugueis**
- **/api/Pagamentos**
- **/api/Consultas**

Cada controlador implementa:

- POST (Criar)
- GET (Listar e ObterPorId)
- PUT (Atualizar)
- DELETE (Excluir)

Além disso, o controlador **ConsultasController** implementa relatórios com múltiplos joins.

7 Consultas Avançadas

As seguintes consultas foram programadas e expostas via API:

1. Veículos com respectivo Fabricante.
2. Aluguéis com Cliente e Veículo.
3. Pagamentos com nome do Cliente.
4. Clientes que alugaram veículos de um determinado fabricante.
5. Histórico completo de aluguéis de um cliente.

Exemplo de consulta (Veículos com Fabricante):

```
GET /api/consultas/veiculos-com-fabricante
```

Retorno:

```
[
    { "modelo": "Onix", "placa": "ABC1234", "fabricante": "Chevrolet" },
    { "modelo": "Fiesta", "placa": "XYZ9090", "fabricante": "Ford" }
]
```

8 Front-End em HTML + Bootstrap

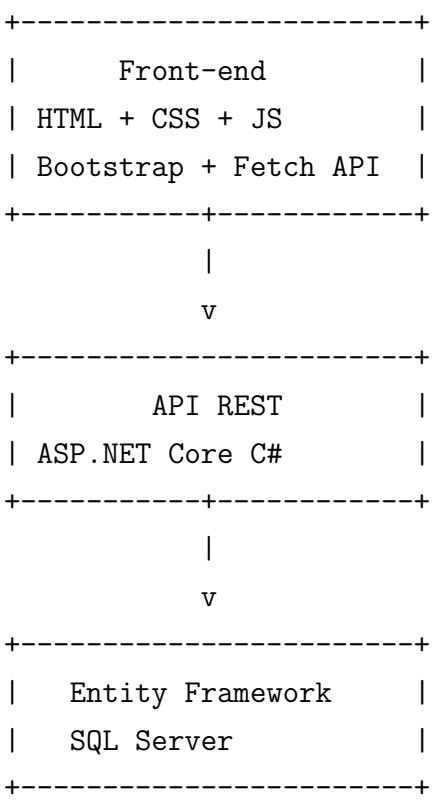
Foi criado um conjunto de páginas HTML:

- home.html (Menu principal)
- CRUDs de Cliente, Fabricante, Veículo, Aluguéis e Pagamento
- consultas.html (Página com todas as consultas)

As páginas consomem a API utilizando Fetch API:

```
fetch("/api/clientes")
    .then(resp => resp.json())
    .then(data => console.log(data));
```

9 Arquitetura da Solução



10 Conclusão

O sistema desenvolvido atende completamente às necessidades da locadora, permitindo controle total das entidades e consultas avançadas. A solução é modular, extensível e serviu de prática para:

- Modelagem de dados
- CRUDs completos
- Criação de API
- Relacionamentos com EF Core
- Consumo de API com JavaScript
- Desenvolvimento de interface responsiva

Como próximos passos, podem ser implementados:

- Autenticação de usuários
- Relatórios em PDF

- Dashboards com gráficos
- Tela administrativa para permissões