UniFECAF

Aluno: Rodrigo Neves Bastos
Professor: Vitor de Jesus

RELATÓRIO DO PROJETO – Sistema Automotivo – Gestão de Estoque de Veículos

1. Introdução

O projeto *Sistema Automotivo – Gestão de Estoque de Veículos* tem como objetivo desenvolver uma aplicação orientada a objetos em Java capaz de gerenciar marcas, modelos e veículos, incluindo controle de estoque, relatórios e persistência de dados em banco SQL e arquivo JSON.

A aplicação foi construída com:

- **Java 17**
- **IntelliJ IDEA**
- **MySQL + JDBC**
- **Gson (JSON)**
- **Maven**
- Arquitetura MVC simplificada (View → Service → Repository → Database)

---

2. DESAFIO

O cenário atual mostra que muitas concessionárias, lojas e revendas de veículos ainda enfrentam dificuldades na gestão de seus estoques. Grande parte dessas empresas utiliza métodos manuais, anotações dispersas ou sistemas pouco integrados, o que causa perda de informações, lentidão nos processos e falta de precisão na comunicação entre equipe, vendedores e clientes. Dessa forma, torna-se essencial o desenvolvimento de uma solução tecnológica capaz de organizar, consultar e manipular dados com eficiência.

2.1. Principais causas do problema

a) Falta de organização:
Muitas empresas ainda trabalham com planilhas, cadernos ou sistemas desatualizados, o que gera informações duplicadas ou inconsistentes, dificultando o acesso aos dados de veículos, modelos e marcas.

2.2. Quem são os afetados e como são afetados

a) Concessionárias e vendedores:
Sofrem com informações desatualizadas, erros no estoque e dificuldade de localizar rapidamente um veículo específico para venda ou consulta.

b) Clientes:
Têm uma experiência de compra prejudicada quando não conseguem encontrar veículos por modelo, ano, faixa de preço ou status. Isso gera insegurança, perda de tempo e pode ocasionar desistência da compra.

c) Gerentes e administradores:
Sem um sistema centralizado, o controle de estoque se torna confuso. Os responsáveis pela gestão têm dificuldade em gerar relatórios, acompanhar desempenho, prever demanda e tomar decisões estratégicas.

2.3. Possíveis soluções e seus prós e contras

a) Criar solução com Spring Boot

- *Prós:* Robusto, moderno e escalável.
- *Contras:* Exige mais conhecimento técnico e configuração.

b) Criar comunicação com Banco de Dados

- *Prós:* Dados organizados, integridade e persistência.
- *Contras:* Necessita conhecimento em SQL e JDBC.

c) Integração com Front-End

- *Prós:* Interface gráfica amigável e acessível ao usuário.

- *Contras:* Demanda aprendizado adicional (HTML/CSS/JS).

2.4. Barreiras e desafios

a) Falta de familiaridade com ferramentas modernas
b) Pouco conhecimento inicial sobre bancos de dados
c) Necessidade de integrar diferentes tecnologias

2.5. Relação com o conteúdo estudado em aula

a) Programação Orientada a Objetos:
O projeto aplica diretamente conceitos como encapsulamento, composição, classes e métodos.

b) Estruturas de Dados e Algoritmos:
Listas, buscas, filtragens e validações são utilizados para manipular informações no sistema.

---

2. Objetivos do Sistema

✔ Cadastrar, listar, atualizar e remover **marcas**, **modelos** e **veículos**
✔ Gerenciar estoque de veículos com status
✔ Gerar relatórios automáticos
✔ Validar dados antes de salvar
✔ Persistir dados em **MySQL**
✔ Fazer backup e restauração usando **JSON**
✔ Evitar duplicações em importações
✔ Seguir práticas de Programação Orientada a Objetos (POO)

---

3. Estrutura da Aplicação

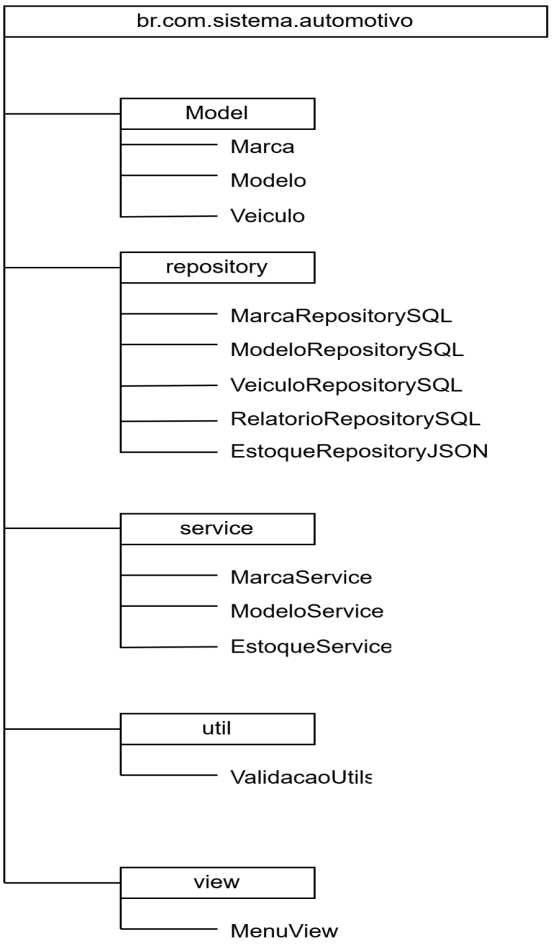
O sistema segue uma estrutura clara dividida em camadas:

3.1. Camadas

- **Model** → Classes de negócio (Marca, Modelo, Veiculo)

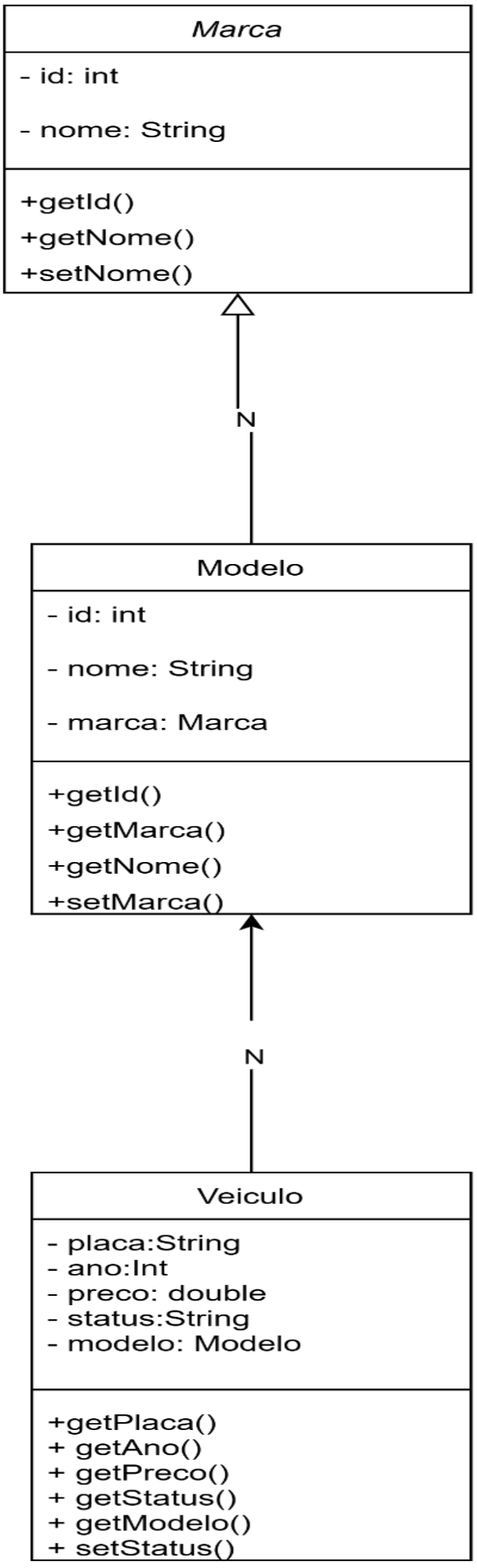
- **View** → Menu de interação com o usuário

- **Service** → Regras de negócio e validações
- **Repository** → Acesso ao MySQL + JSON
- **Util** → Validações de entrada
- **Database** → ConnectionFactory (conexão JDBC)



---

4. Modelos do Sistema


Marca
- id: int
- nome: String
+getId()
+getNome()
+setNome()
N
Modelo
- id: int
- nome: String
- marca: Marca
+getId()
+getMarca()
+getNome()
+setMarca()
N
Veiculo
- placa:String
- ano:Int
- preco: double
- status:String
- modelo: Modelo
+getPlaca()
+ getAno()
+ getPreco()
+ getStatus()
+ getModelo()
+ setStatus()

4.1. Marca

Representa o fabricante do veículo.
Atributos:

- id (int)
- nome (String)

4.2. Modelo

Representa um modelo dentro de uma marca.
Atributos:

- id (int)
- nome (String)
- marca (Marca)

4.3. Veículo

Representa um item no estoque.
Atributos:

- modelo (Modelo)
- ano (int)
- preco (double)
- placa (String)
- status (String)

---

5. Banco de Dados (MySQL)

5.1. Script utilizado

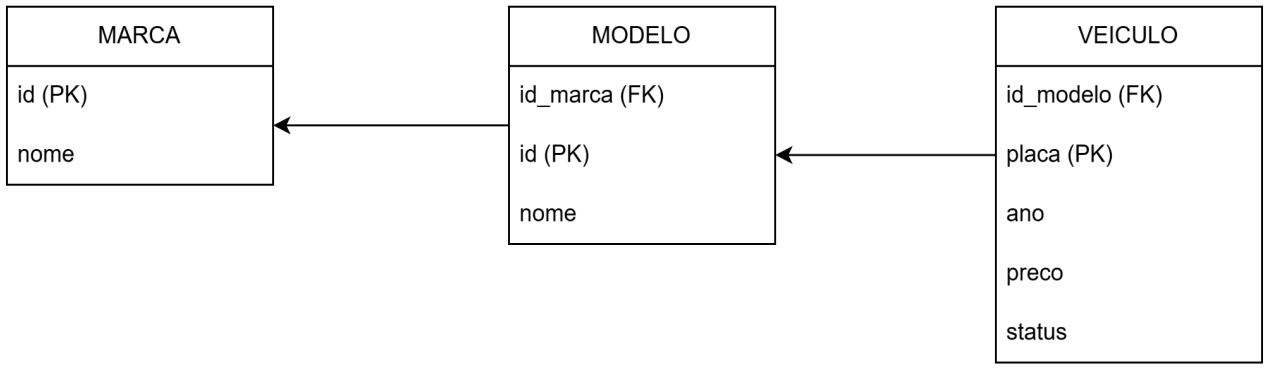
```
CREATE DATABASE sistema_automotivo;
USE sistema_automotivo;

CREATE TABLE marca (
```

```
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL
);

CREATE TABLE modelo (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    id_marca INT NOT NULL,
    FOREIGN KEY (id_marca) REFERENCES marca(id)
);

CREATE TABLE veiculo (
    placa VARCHAR(10) PRIMARY KEY,
    ano INT,
    preco DOUBLE,
    status VARCHAR(20),
    id_modelo INT NOT NULL,
    FOREIGN KEY (id_modelo) REFERENCES modelo(id)
);
```



---

6. Funcionalidades Implementadas

✔ Marcas

- Cadastrar
- Listar
- Buscar por ID

✔ Modelos

- Cadastrar
- Listar
- Buscar por ID
- Remover com validação (não pode remover se tiver veículos associados)

✔ Veículos

- Cadastrar
- Listar
- Buscar por placa
- Atualizar
- Remover
- Validação completa (placa duplicada, ano, preço, modelo etc.)

✔ Relatórios

- Total de veículos
- Total por marca
- Valor total do estoque
- Veículos por status (Disponíveis / Vendidos)

✔ JSON (Backup + Restauração)

- Exporta todos os dados do MySQL → `estoque.json`
- Importa e insere no MySQL sem gerar duplicações
- Recria dependências (marca → modelo → veículo)

---

7. Testes Executados

Foram realizados testes reais com:

▸ Cadastro de:

- Marcas (Toyota, Ford, Honda, Fiat, Volkswagem…)
- Modelos (Corolla, Civic, Ka, Fox…)
- Veículos com diferentes anos e status

▸ Atualização:

- Alteração de ano, preço e status

▸ Exclusão:

- Remoção de veículo
- Remoção de modelo (com validação)

▸ Relatórios:

- Contagem total
- Contagem por marca
- Valor total
- Disponíveis × Vendidos

▸ Backup e Restauração:

- Exportação do banco para JSON
- Limpeza do banco
- Reimportação bem-sucedida sem duplicatas

---

8. Conclusão

O sistema está **100% funcional** e atende todas as exigências do trabalho de Programação Orientada a Objetos:

✔ POO aplicada corretamente
✔ Encapsulamento
✔ Composição entre entidades
✔ Separação de responsabilidades
✔ Persistência SQL e JSON
✔ Validações robustas
✔ Relatórios completos

O projeto está pronto para ser entregue e avaliado.