🏆 Desafio Técnico - CRUD Veículo

🎯 Objetivo

Criar uma **API RESTful** para gerenciar **veículos** 🚗🏍️🚛, aplicando conceitos de:

- **Orientação a Objetos**
- **Boas práticas no Spring Boot** (Controller, Service, Repository)
- **Lombok**

📌 Especificação do Problema

Você precisa implementar um sistema que gerencie **veículos**. O sistema deve permitir:

✅ Cadastrar diferentes tipos de veículos
✅ Listar todos os veículos cadastrados
✅ Buscar um veículo por ID
✅ Atualizar os dados de um veículo
✅ Remover um veículo
✅ Tratar exceções de forma adequada

🚗 Tipos de veículos

O sistema deve ter três tipos de veículos:

1. **Carro**
2. **Moto**
3. **Caminhão**

Cada veículo deve possuir **atributos gerais** e **atributos específicos**.

🧰 Requisitos Técnicos

✅ 1. Atributos Gerais (comuns a todos os veículos)

Esses atributos estarão na **classe abstrata** `Veiculo`:

- `Long id` → Identificador único
- `String tipo` → Ex: Carro, Moto, Caminhão
- `String marca` → Ex: Toyota, Honda
- `String modelo` → Ex: Corolla, CB 500
- `Integer anoFabricacao` → Ex: 2022

✅ 2. Atributos Específicos (por tipo de veículo)

Cada subclasse terá **atributos próprios**:

Tipo	Atributo	Descrição	Tipo
Carro	`numeroPortas`	Número de portas do carro	`Integer`
Moto	`temPartidaEletrica`	Se tem ou não partida elétrica	`Boolean`
Caminhão	`capacidadeCarga`	Capacidade máxima de carga em toneladas	`Integer`

✅ 3. Armazenar os veículos em um repositório JPA

Configurar um banco de dados em memória (H2).

🛠️ Estrutura do Projeto

📂 `src/main/br/tec/db/veiculos/`
├── 📂 **model** → Classes das entidades
├── 📂 **repository** → Interface JPA
├── 📂 **service** → Regras de negócio
├── 📂 **controller** → Endpoints REST
├── 📂 **dto** → Classes DTO
└── 📂 **exception** → Tratamento de exceções

🚀 O que será avaliado?

✅ Uso correto de **polimorfismo** (herança e sobrescrita de métodos)
✅ **Uso do Lombok**
✅ **Boas práticas de código** (organização, separação por camadas)
✅ **Tratamento de exceções**
✅ Uso correto do **Spring Data JPA** para persistência.
✅ **CRUD funcionando corretamente**
✅ **Testes unitários**

🔥 Bônus (Opcional)

- Criar **DTOs** para entrada e saída de dados
- Implementar **validações** com anotações Lombok
- Criar **tratamento de exceções personalizado**
- Implementar **Swagger**
- Implementar **testes de integração**

🎉 **Dica:** Teste os endpoints no **Postman** ou **Swagger** para garantir que tudo está funcionando!