Trabalho solid:

Controllers

SRP

- Antes, as manipulações de requisições HTTP e a regra de negócio estavam nessas classes, agora:

A ClasseController tem responsabilidades relacionadas à manipulação de requisições HTTP para operações CRUD de veículos.
Ele delega a lógica de negócios real para um serviço (ClasseService), o que é uma boa prática, mantendo a responsabilidade única do controlador para lidar com as requisições HTTP.

OCP

- A classe ClasseController esta fechada para modificação, mas aberta para extensão, pois novos métodos podem ser adicionados para suportar novos recursos sem modificar a implementação existente.

LSP

- Não possui herança

ISP

- Não tem interfaces nela, não fez-se necessário

DIP

- O código usa injeção de dependência (@Autowired) para injetar a dependência ClasseService. Isso segue o princípio de depender de abstrações e não de implementações concretas, promovendo a inversão de controle.

Em geral, as classes ClasseController demonstram aderência a alguns dos princípios SOLID, principalmente o SRP e o DIP. Contudo, a aplicação desses princípios pode variar dependendo do contexto e da arquitetura geral do sistema.

Services

SRP

- A classe ClasseService parece ter uma única responsabilidade: gerenciar operações de negócios relacionadas a sua entidade, delegando as operações de persistência a ClasseRepository. Essa abordagem está alinhada com o SRP.

OCP

- A classe ClasseService está aberta para extensão, pois novos métodos relacionados a operações de veículos podem ser adicionados sem modificar a implementação existente. No entanto, essa extensão está mais relacionada ao crescimento natural da classe do que a uma extensão explícita para suportar novos requisitos.

LSP

- A classe ClasseService não apresenta herança direta ou polimorfismo, mas a interação com ClasseRepository sugere que ela segue o LSP, pois utiliza métodos padrão da interface do Spring Data JPA sem problemas.

ISP

- A classe ClasseService não implementa interfaces diretamente, mas depende da interface ClasseRepository. No entanto, ela não está envolvida na criação ou implementação de interfaces no contexto da classe.

DIP

- A classe ClasseService depende da abstração ClasseRepository (por meio da injeção de dependência @Autowired). Essa dependência é de uma abstração, seguindo o DIP. Isso permite que diferentes implementações da ClasseRepository sejam injetadas, promovendo a flexibilidade.

Interfaces(Repositorues) : Segue os 5 princípios do solid.

SRP

- ela seguirá o SRP, pois sua responsabilidade principal é fornecer métodos para realizar operações no banco de dados relacionadas a entidades de veículo.

OCP

- É aberta para extensão, permitindo que novos métodos ou funcionalidades específicas do domínio sejam adicionados sem modificar a implementação existente. Ou seja, sempre que precisarmos de um novo método, basta apenas criar dentro da mesma.

LSP

- segue o LSP, permitindo que implementações específicas do JPA (como Hibernate) substituam a interface sem impactar o código que a utiliza.

ISP

- Interfaces são específicas para o domínio da entidade que estão manipulando. Cada interface fornece um conjunto específico de métodos relacionados a operações CRUD para a entidade associada. Isso ajuda a evitar interfaces abrangentes que são implementadas por todas as classes, seguindo o ISP.

DIP

- classes, como ClasseService ou ClasseController, dependem da interface VeiculoRepository. Isso segue o DIP, pois a dependência é de uma abstração (a interface) em vez de uma implementação concreta.