Documento ApiCars

Este projeto foi construído usando java 17 com spring boot versão 3.1.1 com postgreSQL e estruturado seguindo o padrão arquitetural MVC (Model-View-Controller), onde cada camada desempenha um papel na organização e na funcionalidade da aplicação que foi Divido em Entitys, Repository, Services, DTO (data transfer object) e Controller para expor endpoints rest.

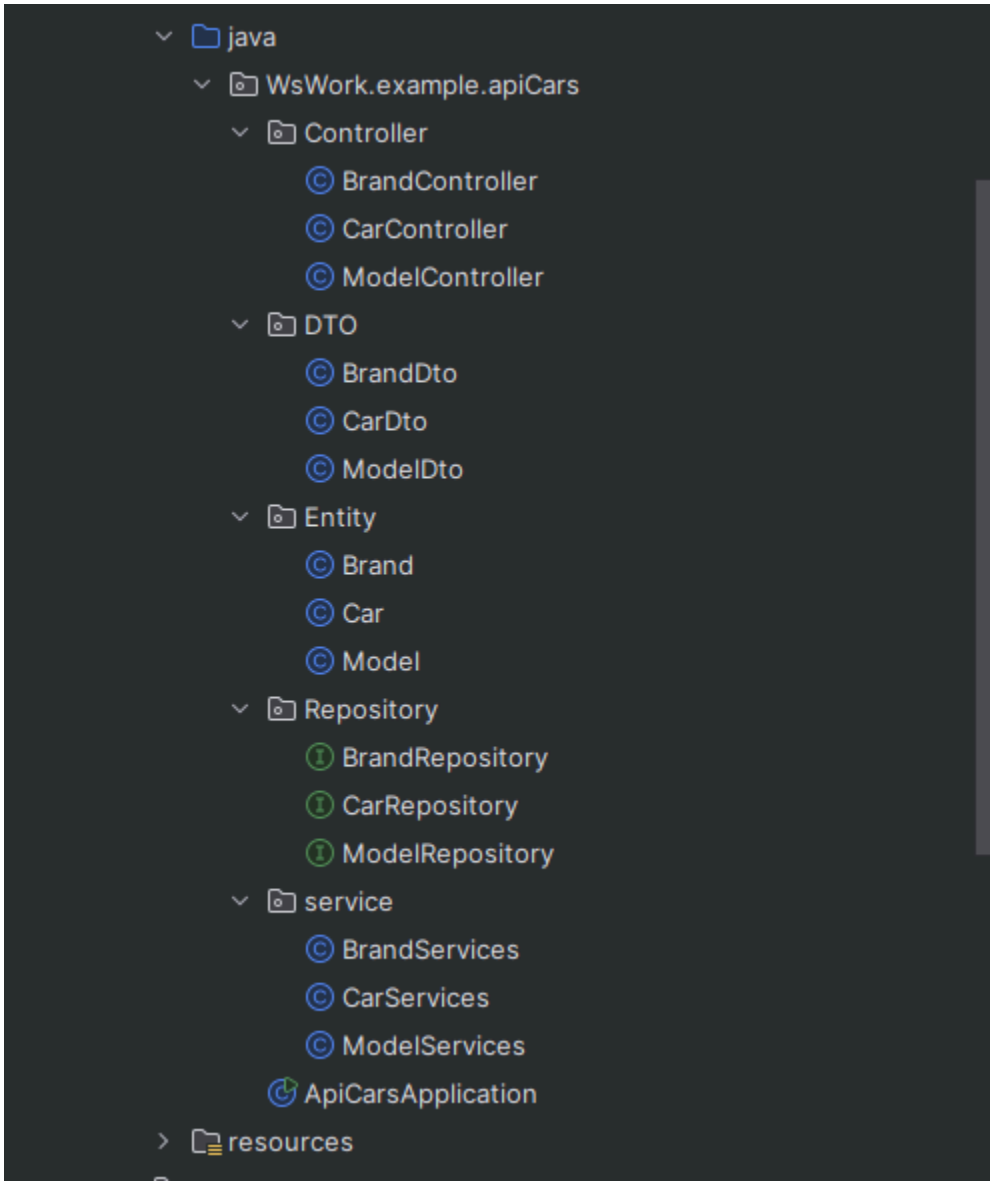
Entity: As (Entitys) representam as tabelas no banco de dados. Elas são anotadas com JPA (@Entity) para mapeamento objeto-relacional, permitindo a persistência dos dados.

Repository: Os Repositorys proporcionam uma abstração sobre o acesso aos dados. Eles permitem que os serviços e lógicas de negócio interajam com as entidades para realizar operações. Foram utilizados para transformar e modelar dados para os endpoints de **GET**.

Service: A camada de services contém a lógica da aplicação de CRUD e modelação de dados por DTO.

Controller: Os Controllers expõem endpoints REST para interação com a aplicação. Eles recebem requisições HTTP, e baseado na opção HTTP as operações para os serviços apropriados e retornam as respostas correspondentes, muitas vezes usando objetos DTO (Data Transfer Objects) para estruturar os dados de saída.
Este padrão proporciona uma separação clara de responsabilidades entre os diferentes componentes da aplicação, facilitando a manutenção, escalabilidade e testabilidade do sistema. A utilização de DTOs ajuda na manipulação e na formatação dos dados para as respostas dos endpoints REST, promovendo uma interface consistente e eficiente para os clientes da API.

ERD de estrutura do projeto:




Baseado na hipótese de que o frontend seja hospedado em outro domínio, foi utilizado a anotação @CrossOrigin(origins = "*") em TODOS controllers para que seja liberada para qualquer aplicação fazer requisições, podendo no futuro ou em ambiente de deploy ser alterado para o domínio específico de aplicação a qual a API servirá.

Baseado no exemplo de formato de dados que foi disponibilizado no documento de apresentação do Desafio, foi utilizada DTO apenas para os métodos de listar dados da API podendo no futuro ser aplicado em outros métodos como de Create e Update.

Abaixo segue como Enviar requisições  via Post para todas as Entitys:

http://localhost:8080/cars.json

```
{
  "register_date": "2024-07-02T10:30:00",
  "model": {
    "id": 1
  },
  "year": 2023,
  "gas_type": "Gasolina",
  "num_Doors": 4,
  "color": "Preto"
}
```

http://localhost:8080/models.json

```
{
  "brand": {
    "id": 3
  },
  "name": "corvette 2024",
  "fipe_value": 700
}
```

http://localhost:8080/brands.json

```
{
    "name_brand": "honda"
}
```

Dependências utilizadas:

Spring Data JPA
Spring Web
Spring-devtools
postgreSQL

Repositório no GitHub: https://github.com/1LUCASPEDROSO/WsWork-api