Criando um app Angular e consumindo uma API REST [2 de 3]

https://medium.com/@andrewchanm/criando-um-app-angular-7-e-consumindo-uma-api-rest-2-de-3-5747972ef56e

Olá pessoal, sou o Andrew desenvolvedor .net, sempre pensei em compartilhar meu conhecimento, esse é minha primeira série de posts e escolhi Angular 7, então podem fazer críticas e sugestões, será muito bom o feedback.

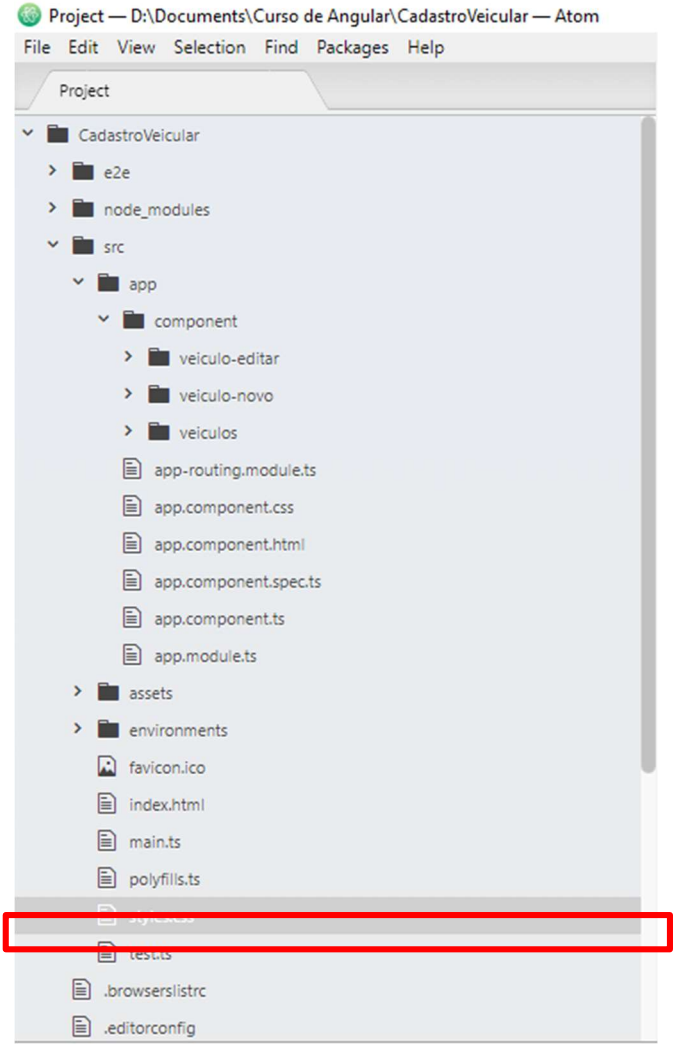
Para quem não viu a Parte 1 segue o **link**, vamos continuar, nessa parte vamos instalar o material design e criar a service para comunicar com nosso Api REST, bom chega de papo, vamos ao código!!!

---

1. Configurando o Layout

Baixe o Bootstrap 4 e descompacte

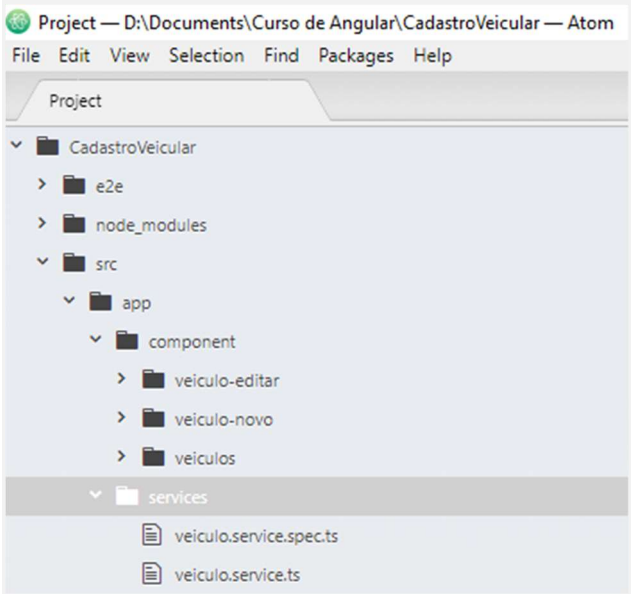
Copie o conteúdo do arquivo **bootstrap.min.css** e cole no arquivo style.css:



2. Criando a Service para acessar nossa Api REST

Nessa etapa, nós vamos criar um service com o nome “veiculo.service” via promp de comando da seguinte forma, ele vai gerar para nós uma pasta “src/service”:

```
ng g service api
```



Antes de editar nosso service, vamos criar uma pasta model e adicionar um arquivo na “src/model/veiculo.ts”

```
export class Veiculo
{
    id: number;
    chapa:  string;
    marca:  string;
    ano: number;
    descricao:  string;
    vendido:  boolean;
    created: Date;
    updated: Date;
}
```

Vamos abrir e editar a service “src/app/service/veiculo.service.ts”

Explicando um pouco o código:

Aqui nós temos os imports. O principal que vamos usar no decorrer do código é o HttpClient, a partir dele nós vamos fazer as chamadas com os verbos http (post, get, put, delete) integrando assim com a api, vamos colocar ele no construtor para ser injetado(lembrando que ele foi adicionado no módulo anteriormente), criamos uma constante httpOptions para identificar nosso headers(cabeçalho de requisição) no formato Json, e temos a constante apiUrl que é a nossa url da api.

```
import { Injectable } from '@angular/core';
import { Observable, of, throwError } from 'rxjs';
import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
import { catchError, tap, map } from 'rxjs/operators';
import { Veiculo } from 'src/model/veiculo';

const httpOptions = { headers: new HttpHeaders({'Content-Type':
'application/json'})};
const baseUrl = 'http://localhost:8080/veiculos';

@Injectable({providedIn: 'root'})
export class VeiculoService {

  constructor(private http: HttpClient) { }
```

Parte 1: veiculo.service.ts

Adicionando uma função de erro para interceptar nossos erros caso ocorra.

```
private handleError<T> (operation = 'operation', result?: T)
  {
    return (error: any): Observable<T> =>
    {
      console.error(error);
      return of(result as T);
    };
  }
```

Agora vamos criar todas as chamadas para a nossa Api (Get, Post, Put e Delete).

```
import { Injectable } from '@angular/core';
import { Observable, of, throwError } from 'rxjs';
import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
import { catchError, tap, map } from 'rxjs/operators';
import { Veiculo } from 'src/model/veiculo';

const httpOptions = { headers: new HttpHeaders({'Content-Type':
'application/json'})};
const baseUrl = 'http://localhost:8080/veiculos';

@Injectable({providedIn: 'root'})
export class VeiculoService {

  constructor(private http: HttpClient) { }

  getAll(): Observable<any> {
    return this.http.get(baseUrl);
  }

  get(id): Observable<any> {
    return this.http.get(`${baseUrl}/${id}`);
  }

  findByDescricao(descricao): Observable<any> {
    return this.http.get(`${baseUrl}/find?descricao=${descricao}`);
  }

  create(data): Observable<any> {
    return this.http.post<Veiculo>(baseUrl, data);
  }

  update(id, data): Observable<any> {
    return this.http.put(`${baseUrl}/${id}`, data);
  }

  sale(id, data): Observable<any> {
    return this.http.patch(`${baseUrl}/${id}`, data);
  }

  delete(id): Observable<any> {
    return this.http.delete(`${baseUrl}/${id}`);
  }

  deleteAll(): Observable<any> {
    return this.http.delete(baseUrl);
  }

  distribuicaoAno(): Observable<any> {
    return this.http.get(`${baseUrl}/distrib_ano`);
  }
```

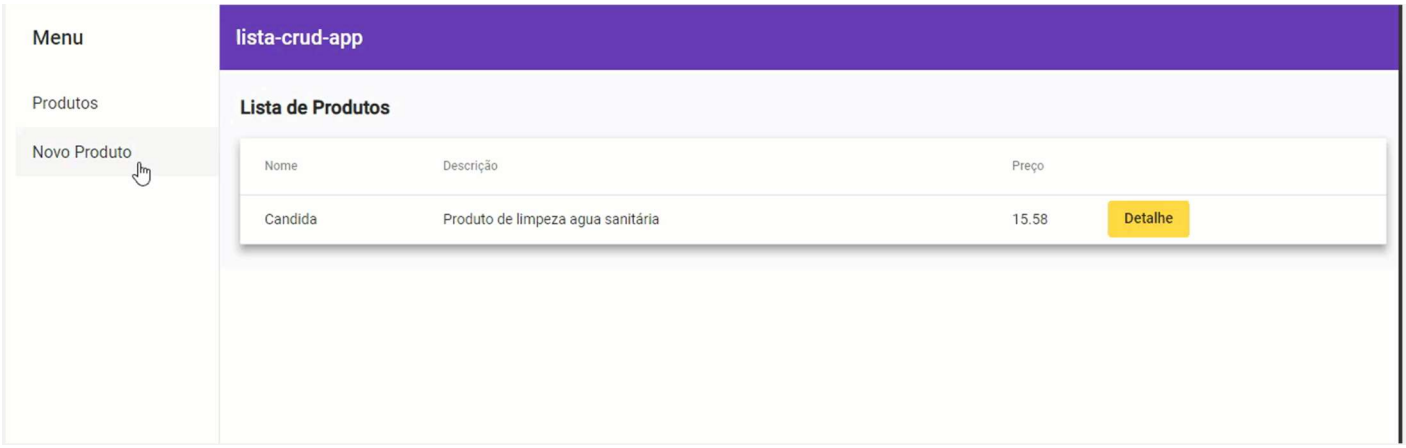
```
  distribuicaoMarca(): Observable<any> {
    return this.http.get(`${baseUrl}/distrib_marca`);
  }

  private handleError<T> (operation = 'operation', result?: T)
  {
    return (error: any): Observable<T> =>
    {
      console.error(error);
      return of(result as T);
    };
  }
}
```

3.Considerações Finais

Bom aqui nós conseguimos criar a nossa service que comunica com a nossa api.

No próximo artigo irei mostrar como criar e editar os componentes que criamos na primeira parte funcionando como um CRUD(CREATE, READ, UPDATE,DELETE).Olha o cheirinho de como vai ficar:



App Final

É triste mas voltarei em breve com a Parte 3, fico a disposição a perguntas e dúvidas, abraços

Referências

Documentação Oficial Angular
Schematics Material
Angular Service
O que é REST?