

ANGULAR

DESENVOLVIMENTO DE SOFTWARE VISUAL

Prof. Evandro Zatti

ANGULAR



- O Angular é um *framework* de design de aplicativos e plataforma de desenvolvimento, construída em TypeScript, para a criação de aplicativos de página única. Ele inclui:
 - ✓ Uma estrutura baseada em componentes para construir aplicações web escaláveis
 - ✓ Uma coleção de bibliotecas integradas com vários recursos (roteamento, gerenciamento de formulários, comunicação cliente-servidor *etc.*);
 - ✓ Um conjunto de ferramentas de desenvolvedor para desenvolver, construir, testar e atualizar o código

ANGULAR CLI E NODE .JS



- O Angular CLI (*Command Line Interface – Interface de Linha de Comando*) é uma ferramenta de linha de comando usada para criar, gerenciar e construir projetos Angular;
- Essa ferramenta depende do Node.js para executar, pois utiliza o npm (Node Package Manager) para instalar e gerenciar as dependências do projeto:
 - ✓ <https://nodejs.org/en>

ATUALIZAÇÃO/REINSTALAÇÃO DO NODE NOS LABS

- Desinstalar o node.js
- Excluir as pastas:
 - ✓ C:\users\Default\AppData\Roaming\npm
 - ✓ C:\users\Aluno\AppData\Roaming\npm
- Instalar o node.js novamente
 - ✓ <https://nodejs.org/en>

CRIANDO A APLICAÇÃO ANGULAR (FRONT)

Atualização do pacote npm
neste caso para a versão 10.2.0

```
npm install -g npm@10.2.0
```

```
npm install -g @angular/cli
```

```
ng new AppEstacionamento
```

```
cd .\AppEstacionamento\
```

```
npm install ngx-bootstrap
```

Instalação da CLI

caso não funcione, desinstalar, limpar cache e reinstalar:

```
npm uninstall -g angular-cli  
npm uninstall --save-dev angular-cli  
npm cache clean
```

verificar a variável **path** das variáveis de ambiente:

```
C:\Users\usuário\AppData\Roaming\npm  
C:\Users\usuário\AppData\npm\node_modules\@angular\cli\bin
```

é necessário que o PowerShell tenha acesso de execução irrestrita de scripts; então sugere-se que a instalação seja feita direto de uma janela PowerShell do Windows em modo administrador.

Criação da aplicação

Instalação do bootstrap no diretório da aplicação

Se quiser autorizar a execução de janelas em modo normal (de dentro do VS Code), abra antes uma janela do PowerShell do Windows no modo administrador e execute o comando:
`Set-ExecutionPolicy Unrestricted`

PREPARAÇÃO DA API (BACK)

Program.cs

```
using API_Estacionamento.Data;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddDbContext<EstacionamentoDbContext>();
builder.Services.AddCors(); ----->
var app = builder.Build();
if (app.Environment.IsDevelopment()) {...}
app.UseHttpsRedirection();
app.UseCors(options => options.AllowAnyOrigin().AllowAnyHeader());
app.UseAuthorization();
app.MapControllers();
app.Run();
```

adicionar o serviço
Cors para possibilitar
a conversa entre
front e back

PREPARAÇÃO DA API (BACK)

appsettings.json

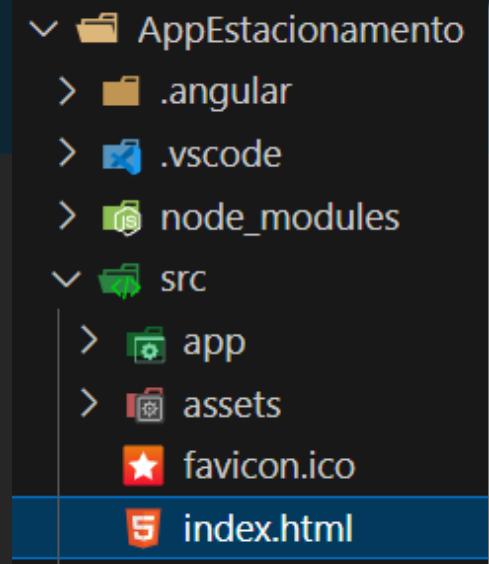
```
{  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft.AspNetCore": "Warning"  
    }  
  },  
  "AllowedHosts": "*",  
  "Kestrel":{  
    "Endpoints": {  
      "Http" : {  
        "Url" : "http://localhost:5000"  
      }  
    }  
  }  
}
```

adicionar o atributo
Kestrel, forçando a
url e porta dos
endpoints

REFERENCIANDO O BOOTSTRAP NO FRONT

index.html

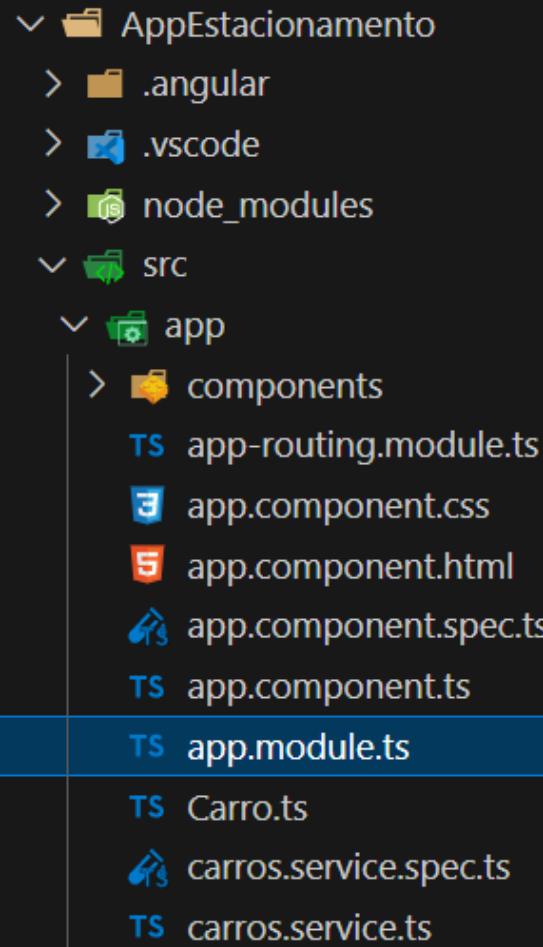
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>AppEstacionamento</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
        href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```



adicinar o link do
stylesheet

REGISTRO DOS MÓDULOS

app.module.ts



```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { CommonModule } from '@angular/common';
import { HttpClientModule } from '@angular/common/http';

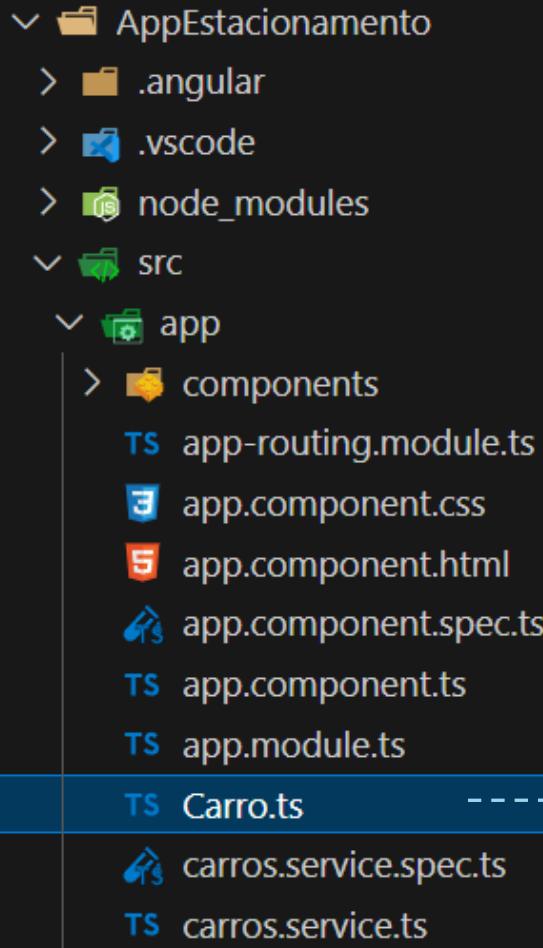
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { ReactiveFormsModule } from '@angular/forms';
import { ModalModule } from 'ngx-bootstrap/modal';

...
```

verificar quais módulos já existem e complementar

CRIAÇÃO DA MODEL NO FRONT



- É necessário criar a model no *frontend*, com a mesma estrutura do *backend*, utilizando a sintaxe *typescript* (.ts):

Carro.ts

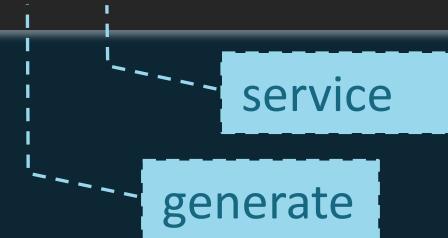
```
export class Carro {  
  placa: string = '';  
  descricao: string = '';  
}
```

adicionar um novo arquivo **Carro.ts** dentro da pasta **app**

CRIAÇÃO DO SERVIÇO

- Para dar funcionalidade à entidade, deve-se criar um serviço a partir da model:

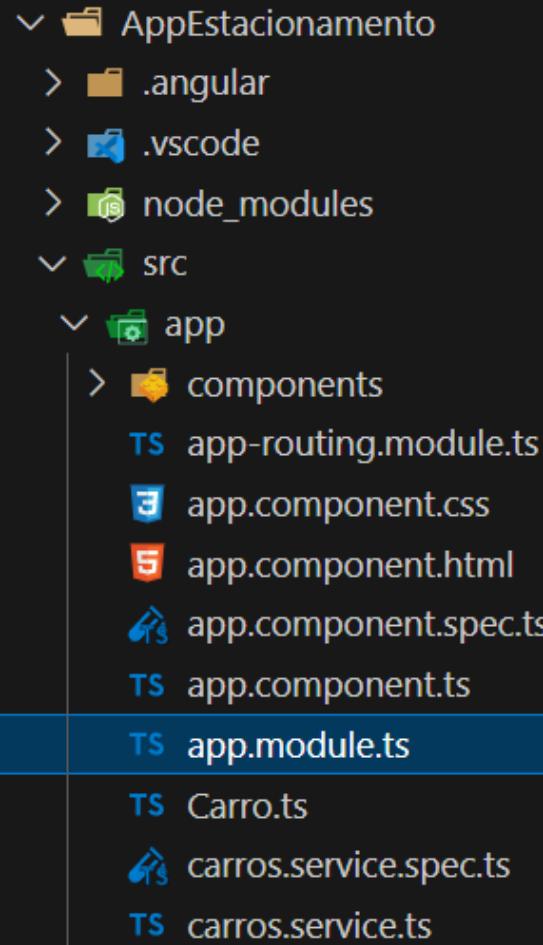
```
cd .\AppEstacionamento  
cd .\src\app  
ng g s Carros
```



o serviço é um recurso análogo à *controller* do *backend*: ele será responsável por conter as funcionalidades da entidade

REGISTRO DO SERVIÇO NOS MÓDULOS

app.module.ts



```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { CommonModule } from '@angular/common';
import { HttpClientModule } from '@angular/common/http';

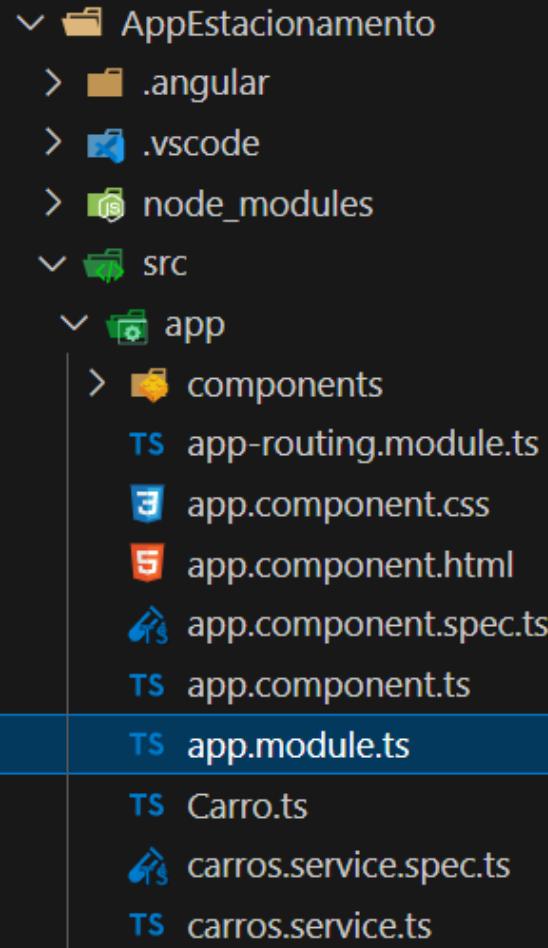
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { ReactiveFormsModule } from '@angular/forms';
import { ModalModule } from 'ngx-bootstrap/modal';

import { CarrosService } from './carros.service';
```

REGISTRO DOS MODULOS

app.module.ts



```
...
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    CommonModule,
    HttpClientModule,
    ReactiveFormsModule,
    ModalModule.forRoot()
  ],
  providers: [HttpClientModule, CarrosService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

CONFIGURAÇÃO DO SERVIÇO

carros.service.ts

```
✓ └─ AppEstacionamento
  >   └─ .angular
  >   └─ .vscode
  >   └─ node_modules
  ✓ └─ src
    └─ app
      >   └─ components
        TS app-routing.module.ts
        IC app.component.css
        S app.component.html
        IC app.component.spec.ts
        TS app.component.ts
        TS app.module.ts
        TS Carro.ts
        IC carros.service.spec.ts
        TS carros.service.ts
```

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Carro } from './Carro';
const httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
}
@Injectable({
  providedIn: 'root'
})
export class CarrosService {
  apiUrl = 'http://localhost:5000/Carro';
  constructor(private http: HttpClient) { }
}
```

atente para os imports!

definir o tipo de cabeçalho

definir a rota do serviço na API

CONFIGURAÇÃO DO SERVIÇO

```
✓ AppEstacionamento
  > .angular
  > .vscode
  > node_modules
  ✓ src
    > app
      > components
        TS app-routing.module.ts
        SCSS app.component.css
        HTML app.component.html
        TS app.component.spec.ts
        TS app.component.ts
        TS app.module.ts
        TS Carro.ts
        TS carros.service.spec.ts
    TS carros.service.ts
```

implementar os verbos http

carros.service.ts

```
export class CarrosService {
  apiUrl = 'http://localhost:5000/Carro';
  constructor(private http: HttpClient) { }
  listar(): Observable<Carro[]> {
    const url = `${this.apiUrl}/listar`;
    return this.http.get<Carro[]>(url);
  }
  buscar(placa: string): Observable<Carro> {
    const url = `${this.apiUrl}/buscar/${placa}`;
    return this.http.get<Carro>(url);
  }
  cadastrar(carro: Carro): Observable<any> {
    const url = `${this.apiUrl}/cadastrar`;
    return this.http.post<Carro>(url, carro, httpOptions);
  }
  atualizar(carro: Carro): Observable<any> {
    const url = `${this.apiUrl}/atualizar`;
    return this.http.put<Carro>(url, carro, httpOptions);
  }
  excluir(placa: string): Observable<any> {
    const url = `${this.apiUrl}/buscar/${placa}`;
    return this.http.delete<string>(url, httpOptions);
  }
}
```

COMPONENTES

- Componentes são os blocos de construção que compõem um aplicativo.
- Um componente inclui uma classe TypeScript com um decorador `@Component()`, um modelo HTML e estilos.
- O decorador `@Component()` especifica as seguintes informações específicas do Angular:

COMPONENTES

- ✓ Um seletor CSS que define como o componente é usado em um modelo;
- ✓ Um modelo HTML que instrui o Angular como renderizar o componente;
- ✓ Um conjunto opcional de estilos CSS que definem a aparência dos elementos HTML do modelo.

COMPONENTES (EXEMPLO GENÉRICO)

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>Meu primeiro componente!</p>
  `
})
export class HelloWorldComponent {
  // Este trecho de código define
  // o comportamento do componente.
}
```

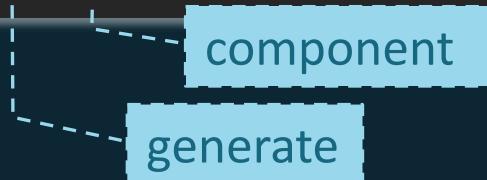
uso:

```
<hello-world></hello-world>
```

CRIAÇÃO DO COMPONENTE

- Agora é preciso criar o componente visual da entidade, composto de arquivos html, css, ts...

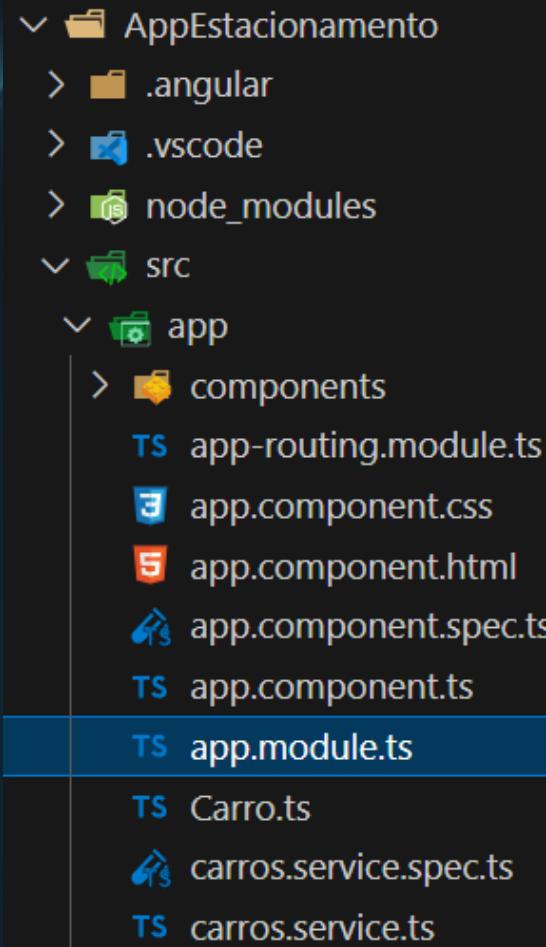
```
cd .\AppEstacionamento
cd .\src\app
mkdir components
cd components
ng g c Carros
```



cria-se uma pasta específica para conter os componentes; dentro dela o *framework* irá criar uma subpasta para conter os arquivos de componente de cada entidade

REGISTRO DO COMPONENTE NOS MÓDULOS

app.module.ts



```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { CommonModule } from '@angular/common';
import { HttpClientModule } from '@angular/common/http';

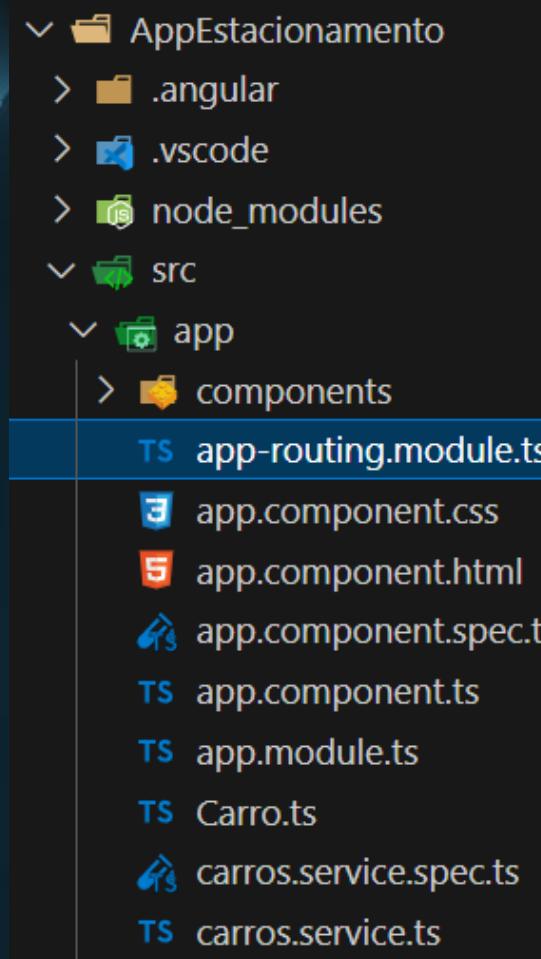
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { ReactiveFormsModule } from '@angular/forms';
import { ModalModule } from 'ngx-bootstrap/modal';

import { CarrosService } from './carros.service'
import { CarrosComponent } from
  './components/carros/carros.component';

...
```

REGISTRO DO COMPONENTE NAS ROTAS



app-routing.module.ts

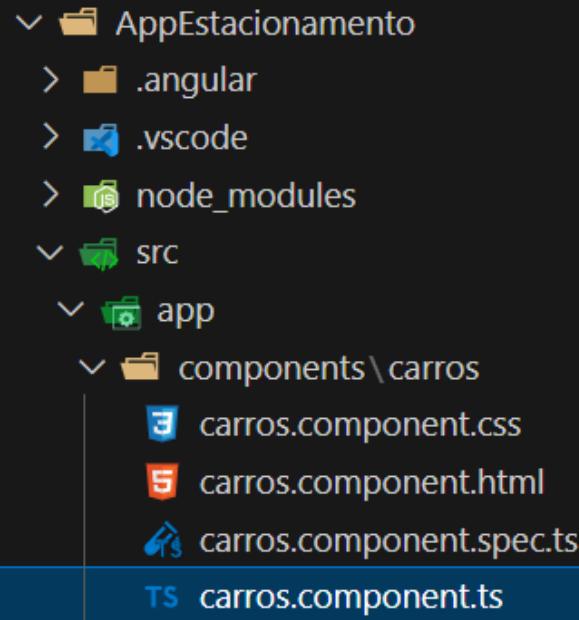
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CarrosComponent } from './components/carros/carros.component';

const routes: Routes = [
  { path: 'carros', component:CarrosComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

a rota é case-sensitive!

DEFINIÇÃO DO COMPONENTE



carros.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup } from '@angular/forms';
import { CarrosService } from 'src/app/carros.service';
import { Carro } from 'src/app/Carro';

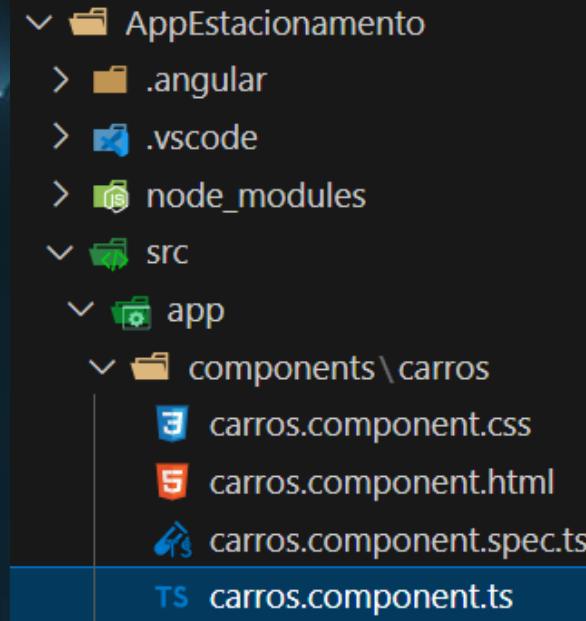
@Component({
  selector: 'app-carros',
  templateUrl: './carros.component.html',
  styleUrls: ['./carros.component.css']
})
```

atente para os imports!

```
export class CarrosComponent implements OnInit {
  ...
}
```

definir herança;
implementação no próximo slide

DEFINIÇÃO DO COMPONENTE



Esta função será chamada no click do botão enviar

carros.component.ts

```
...
export class CarrosComponent implements OnInit {
  formulario: any;
  tituloFormulario: string = '';
  constructor(private carrosService : CarrosService) { }
  ngOnInit(): void {
    this.tituloFormulario = 'Novo Carro';
    this.formulario = new FormGroup({
      placa: new FormControl(null),
      descricao: new FormControl(null)
    })
    enviarFormulario(): void {
      const carro : Carro = this.formulario.value;
      this.carrosService.cadastrar(carro).subscribe(result => {
        alert('Carro inserido com sucesso.');
      })
    }
}
```

formulario e título

grupo de controles do formulário

os controles devem ter o mesmo nome dos campos da model

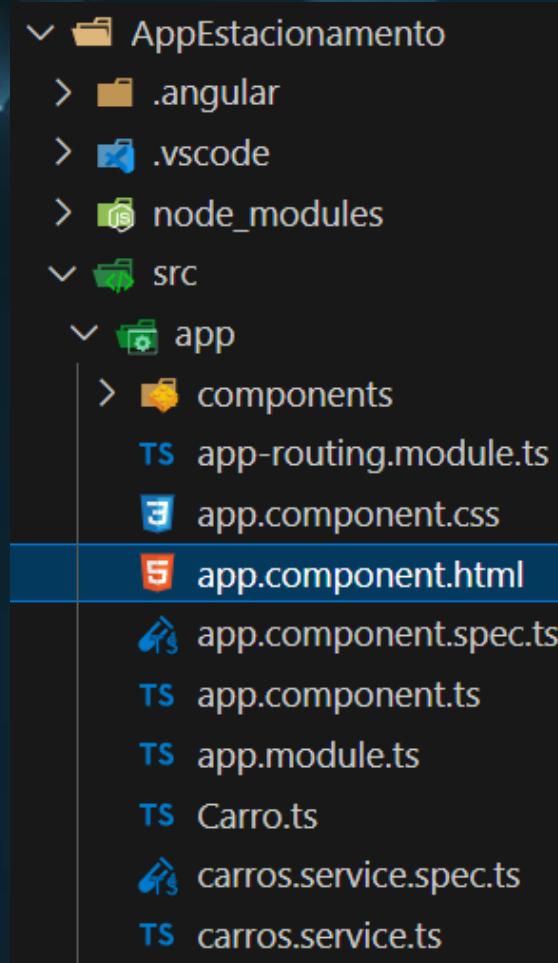
DEFINIÇÃO DO COMPONENTE (HTML)

carros.component.html

```
<div class="container p-5">
  <div class="row">
    <div class="col-6 border border-light rounded p-5 shadow" *ngIf="formulario">
      <h6>{{ tituloFormulario }}</h6>
      <form [formGroup]="formulario" (ngSubmit)="enviarFormulario()">
        <div class = "form-group">
          <label>Placa</label>
          <input type = "text" class="form-control form-control-sm" formControlName="placa"/>
        </div>
        <div class = "form-group">
          <label>Descrição</label>
          <input type = "text" class="form-control form-control-sm" formControlName="descricao"/>
        </div>
        <div class = "container">
          <div class="row">
            <div>
              <button type="submit" class="btn btn-sm btn-outline-secondary">Salvar</button>
            </div>
            <div class="pl-1">
              <button type="button" class="btn btn-sm btn-light">Voltar</button>
            </div>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
```

ALTERAÇÃO DO COMPONENTE INICIAL

app.component.html



excluir todo o conteúdo da página,
mantendo somente a última linha,
referente ao roteamento

COMPILAÇÃO E EXECUÇÃO DA APLICAÇÃO

```
cd .\AppEstacionamento  
ng serve
```

- A aplicação responde no endereço:
✓ <http://localhost:4200/>
- No navegador, complemente com o serviço para testar:
✓ <http://localhost:4200/carros> ----- a rota é *case sensitive!*