Angular

Angular

- Angular é uma plataforma e framework para construção da interface de aplicações usando HTML, CSS e, principalmente, JavaScript, criada pelos desenvolvedores da Google.

- Possui alguns elementos básicos que tornam essa construção interessante.

- Dentre os principais, podemos destacar os componentes, templates, diretivas, roteamento, módulos, serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas, como a de executar os testes unitários de uma aplicação.

Angular

- Alguns outros pontos dessa plataforma que merecem destaque são o fato de que ela é open source, possui uma grande comunidade, existem várias empresas utilizando e tem muito material de estudo para quem deseja se aperfeiçoar.

Angular CLI

- Angular CLI é a ferramenta oficial para inicializar e trabalhar com projetos em Angular.

- Depois de instalar a Angular CLI, você precisa executar um comando para gerar um projeto e outro para servi-lo usando um servidor de desenvolvimento local para rodar sua aplicação.

- Como ocorre com a maioria das ferramentas de front-end nos dias de hoje, a Angular CLI foi criada com base no Node.js.

Node.js

- O Node.js é uma tecnologia de servidor que permite que você execute o JavaScript no servidor e crie aplicações da web no lado do servidor.
- No entanto, o Angular é uma tecnologia de front-end.
- Portanto, mesmo que você precise instalar o Node.js em sua máquina de desenvolvimento, é apenas para a execução da CLI.

Instalando Angular

- Primeiro, você precisa ter o Node e o npm instalados na sua máquina de desenvolvimento. Existem muitas maneiras de se fazer isso, tais como:
  - usar o NVM (Node Version Manager) para a instalação e trabalho com muitas versões do Node em seu sistema
  - usar o gerenciador de pacotes oficial do seu sistema operacional
  - instalar a partir do site oficial da web.
  - https://nodejs.org/en/download/

Instalando Angular

- Certifique-se de que o Node esteja instalado em seu sistema executando o comando abaixo em um prompt de comando, que deverá exibir a versão instalada do Node:
  - $ node –v
- Depois, execute o seguinte comando para instalar a Angular CLI:
  - $ npm install @angular/cli

AngularCLI

- Depois de instalar a Angular CLI, você pode executar diversos comandos. Comecemos conferindo a versão instalada da CLI:
  - $ ng version
- Um segundo comando que talvez você queira executar é o comando help, que ajudará a obter uma lista de comandos a serem usados:
  - $ ng help

AngularCLI

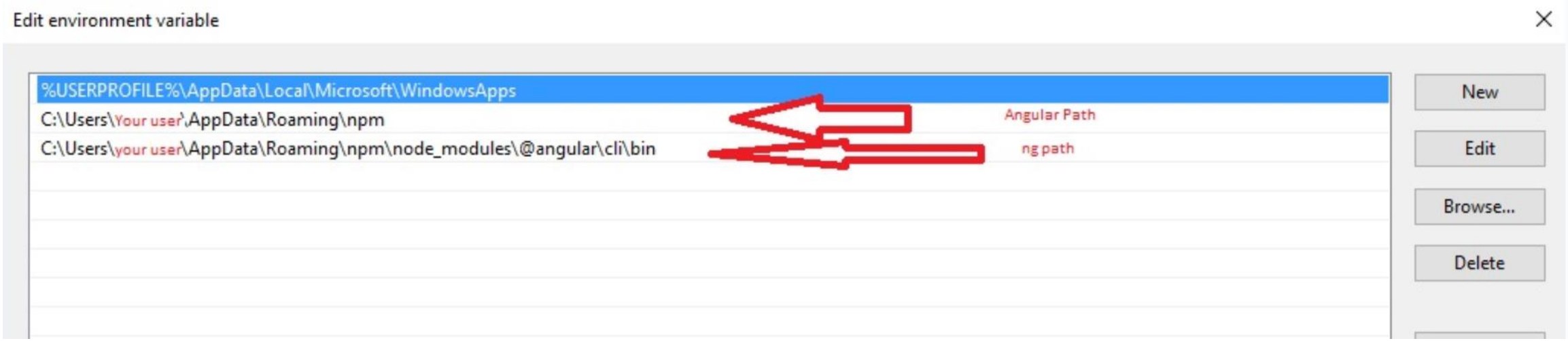
- Se der erro, siga os seguintes passos:
  - npm uninstall -g angular-cli
  - npm uninstall --save-dev angular-cli
  - npm install -g @angular/cli

AngularCLI

- Se der erro, siga os seguintes passos:
  - npm uninstall -g angular-cli
  - npm uninstall --save-dev angular-cli
  - npm install -g @angular/cli
    - Se der erro no install, tente:
      - npm install -g @angular/cli --force

AngularCLI

- Se der erro, siga os seguintes passos:
  - Depois de instalar você entra em sistema, configuração avançadas do sistema, Variáveis de Ambiente, Path, Editar



A CLI oferece os comandos a seguir:

- add: adiciona suporte a uma biblioteca externa em seu projeto.
- build (b): compila uma aplicação do Angular em seu diretório de saída, chamado dist/ no caminho de saída fornecido. Este comando deve ser executado dentro de um diretório de espaço de trabalho.
- config: obtém ou configura os valores de configuração do Angular.
- doc (d): abre a documentação oficial do Angular (angular.io) em um navegador, buscando uma palavra-chave determinada.
- e2e (e): faz o build e serve uma aplicação do Angular, depois executa testes de ponta a ponta usando o Protractor.

A CLI oferece os comandos a seguir:

- generate (g): gera e/ou modifica arquivos com base em um esquema.
- help: lista os comandos disponíveis e suas descrições breves.
- lint (l): executa as ferramentas de linting no código da aplicação em Angular em uma determinada pasta de projeto.
- new (n): cria um espaço de trabalho e uma aplicação inicial em Angular.
- run: executa um destino personalizado definido em seu projeto.

A CLI oferece os comandos a seguir:

- serve (s): faz o build e serve sua aplicação, fazendo um novo build a cada alteração de arquivos.
- test (t): executa os testes unitários em um projeto.
- update: atualiza sua aplicação e suas dependências. Consulte https://update.angular.io/ (em inglês)
- version (v): mostra a versão da Angular CLI.
- xi18n: extrai as mensagens i18n do código-fonte.

Criando um projeto novo

- Você pode usar a Angular CLI para gerar rapidamente seu projeto em Angular executando o seguinte comando em sua interface de linha de comando:
  - $ ng new OiMundo
- A CLI perguntará Would you like to add Angular routing? ("Quer adicionar o roteamento do Angular?"). Você pode responder com y (Sim) ou n (Não). Não, neste caso, é a opção padrão. **Responda Sim.**
- Também será perguntado sobre o formato da folha de estilos (stylesheet) que você quer usar (por exemplo, o CSS). Escolha CSS e pressione Enter para continuar.

Criando um projeto novo

- Depois disso, seu projeto estará criado com uma estrutura de diretórios e alguns arquivos com configurações e código.

- A maioria estará nos formatos TypeScript e JSON.

Criando um novo projeto

- Vejamos a função de cada diretório/arquivo:
  - /e2e/: contém os testes end-to-end (simulação do comportamento do usuário) do site
  - /node_modules/: todas as bibliotecas de terceiros são instaladas nesta pasta usando npm install
  - /src/: contém o código-fonte da aplicação. A maior parte do trabalho será feita aqui
  - /app/: contém módulos e componentes
  - /assets/: contém os ativos estáticos, como imagens, ícones e estilos

Criando um novo projeto

- Vejamos a função de cada diretório/arquivo:
  - /environments/: contém arquivos de configuração específicos do ambiente (produção e desenvolvimento)
  - browserslist: necessário para o autoprefixador para suporte ao CSS
  - favicon.ico: o favicon
  - index.html: o arquivo HTML principal
  - karma.conf.js: o arquivo de configuração para o Karma (uma ferramenta de testes)
  - main.ts: o arquivo inicial principal, a partir de onde o AppModule é iniciado

Criando um novo projeto

- Vejamos a função de cada diretório/arquivo:
  - polyfills.ts: polyfills necessários ao Angular
  - styles.css: o arquivo de folha de estilos (stylesheet) global do projeto
  - test.ts: um arquivo de configuração para o Karma
  - tsconfig.*.json: os arquivos de configuração para o TypeScript
  - angular.json: contém as configurações para a CLI

Criando um novo projeto

- Vejamos a função de cada diretório/arquivo:
  - package.json: contém as informações básicas do projeto (nome, descrição e dependências)
  - README.md: um arquivo em markdown que contém a descrição do projeto
  - tsconfig.json: o arquivo de configuração para o TypeScript
  - tslint.json: o arquivo de configuração para o TSlint (uma ferramenta de análise estática)

Como servir o projeto?

- A Angular CLI fornece um conjunto de ferramentas completo para desenvolver aplicações de front-end em sua máquina local.

- Assim, não é preciso instalar um servidor para servir o projeto—você pode, simplesmente, usar o comando ng serve a partir do terminal para servir seu projeto localmente.

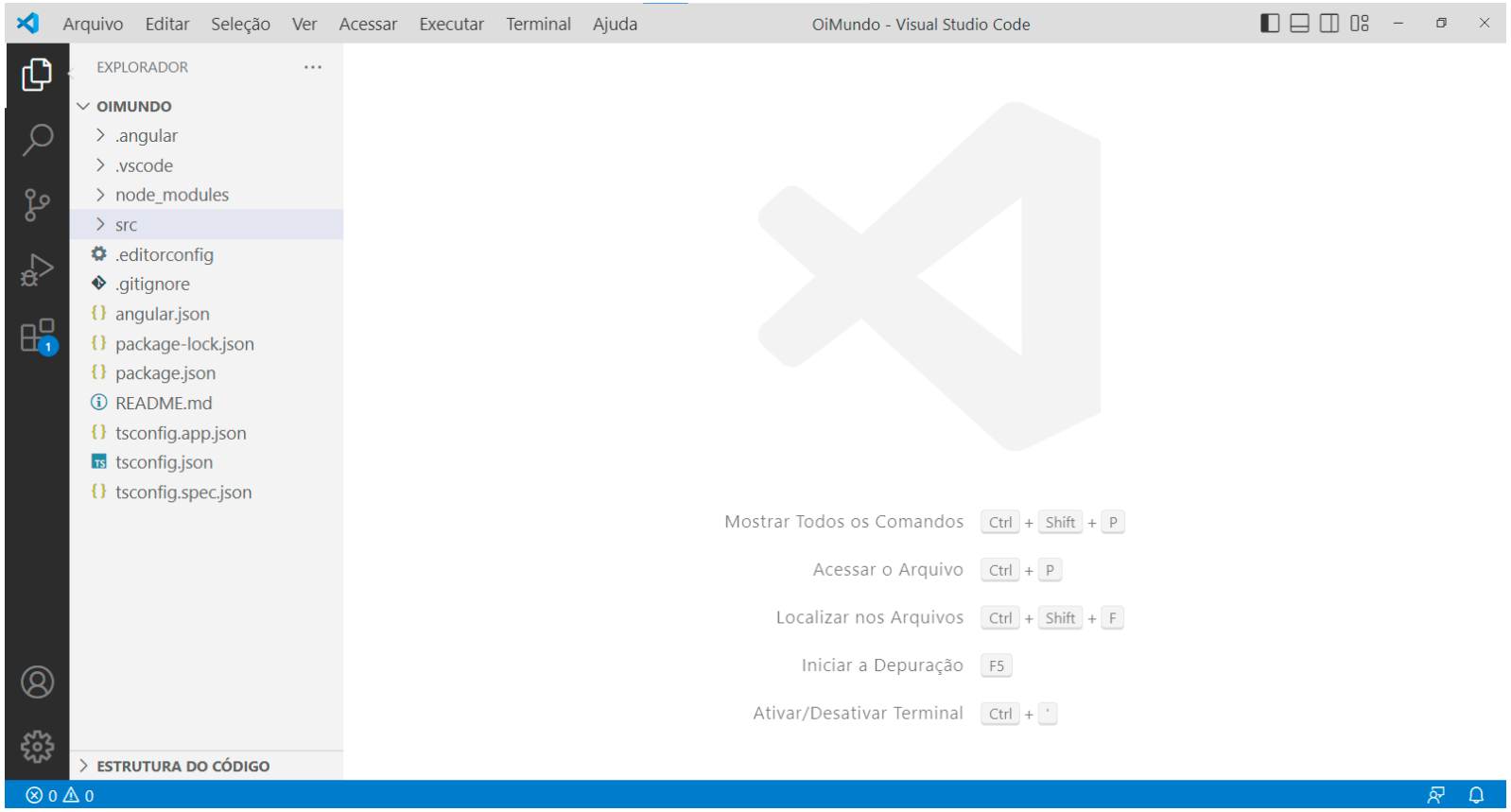
  - $ cd OiMundo

  - $ ng serve

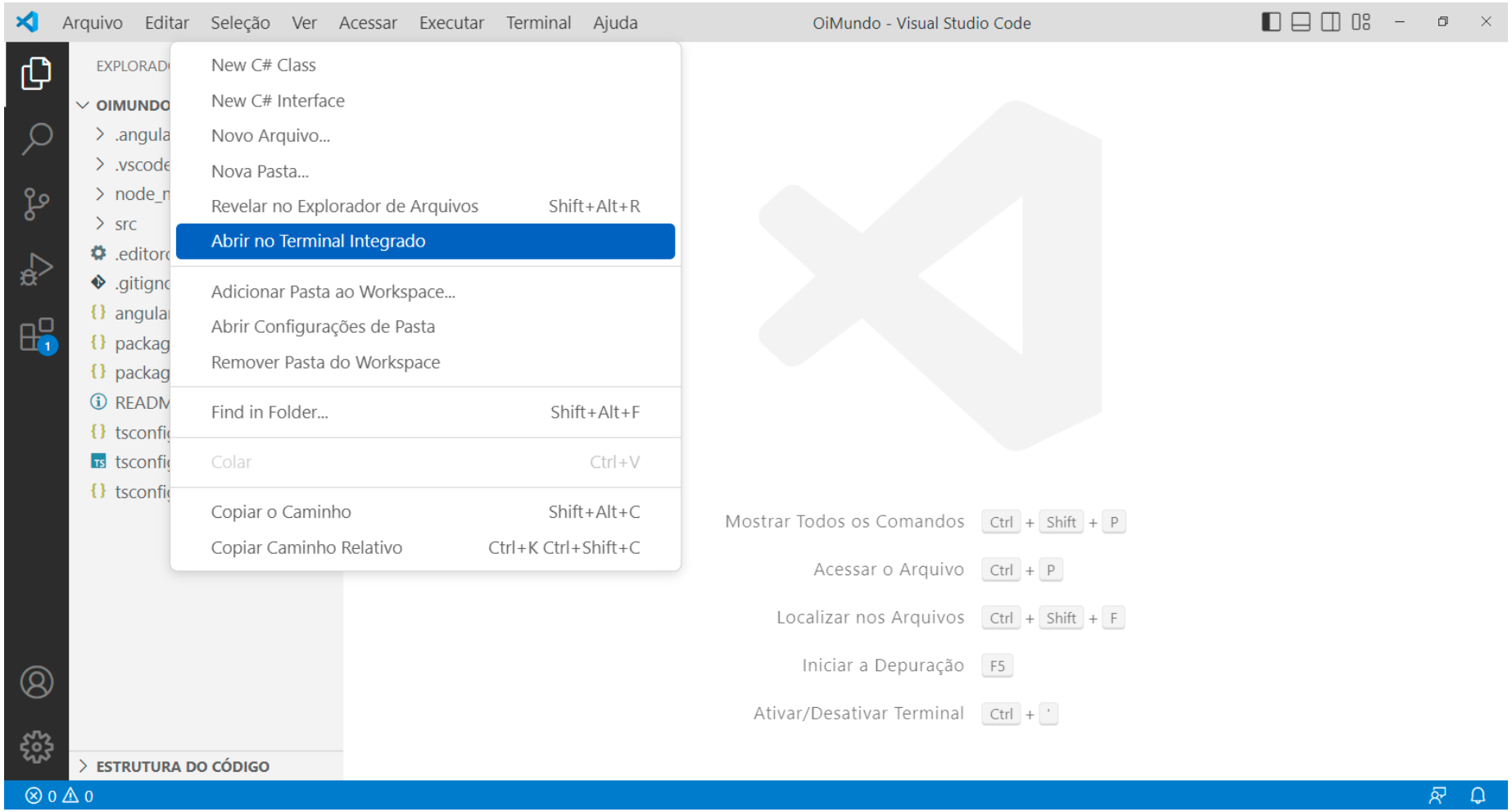
Como servir o projeto?

- Agora, você pode navegar até o endereço http://localhost:4200/ para começar a mexer com sua aplicação de front-end.
- A página recarregará automaticamente caso você mude qualquer arquivo do código-fonte.
- Para abrir o projeto no Visual code digite:
  - $ code .

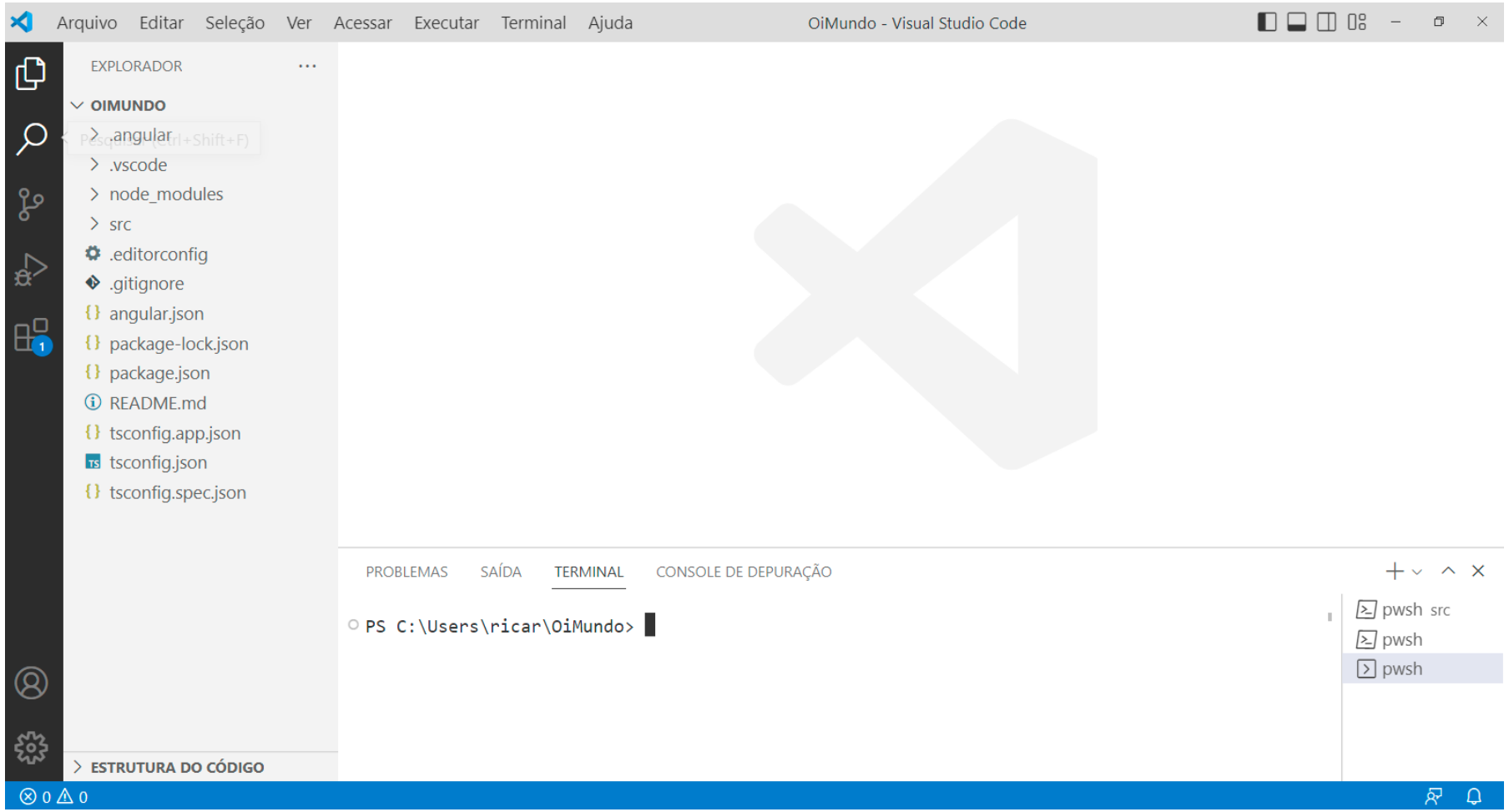
Visual code


Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda
OiMundo - Visual Studio Code
EXPLORADOR
OIMUNDO
.angular
.vscode
node_modules
src
.editorconfig
.gitignore
angular.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
ESTRUTURA DO CÓDIGO
Mostrar Todos os Comandos Ctrl + Shift + P
Acessar o Arquivo Ctrl + P
Localizar nos Arquivos Ctrl + Shift + F
Iniciar a Depuração F5
Ativar/Desativar Terminal Ctrl + '
0 0

Visual code


Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda
OiMundo - Visual Studio Code
EXPLORAD
OIMUNDO
.angula
.vscode
node_m
src
.editorc
.gitignc
angular
packag
packag
READM
tsconfi
tsconfi
tsconfi
New C# Class
New C# Interface
Novo Arquivo...
Nova Pasta...
Revelar no Explorador de Arquivos Shift+Alt+R
Abrir no Terminal Integrado
Adicionar Pasta ao Workspace...
Abrir Configurações de Pasta
Remover Pasta do Workspace
Find in Folder... Shift+Alt+F
Colar Ctrl+V
Copiar o Caminho Shift+Alt+C
Copiar Caminho Relativo Ctrl+K Ctrl+Shift+C
Mostrar Todos os Comandos Ctrl + Shift + P
Acessar o Arquivo Ctrl + P
Localizar nos Arquivos Ctrl + Shift + F
Iniciar a Depuração F5
Ativar/Desativar Terminal Ctrl + '
ESTRUTURA DO CÓDIGO
0 0

Visual code


Arquivo
Editar
Seleção
Ver
Acessar
Executar
Terminal
Ajuda
OiMundo - Visual Studio Code
EXPLORADOR
OIMUNDO
.angular
.vscode
node_modules
src
.editorconfig
.gitignore
angular.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
ESTRUTURA DO CÓDIGO
PROBLEMAS
SAÍDA
TERMINAL
CONSOLE DE DEPURAÇÃO
PS C:\Users\ricar\OiMundo>
pwsh src
pwsh
pwsh
0 0

Geração de artefatos do Angular

- A Angular CLI fornece um comando ng generate que ajuda os desenvolvedores a gerar artefatos básicos do Angular, como módulos, componentes, diretivas, pipes e serviços:
  - $ ng generate component meu-componente
- meu-componente é o nome do componente. A Angular CLI automaticamente adicionará uma referência a components, directives e pipes no arquivo src/app.module.ts.

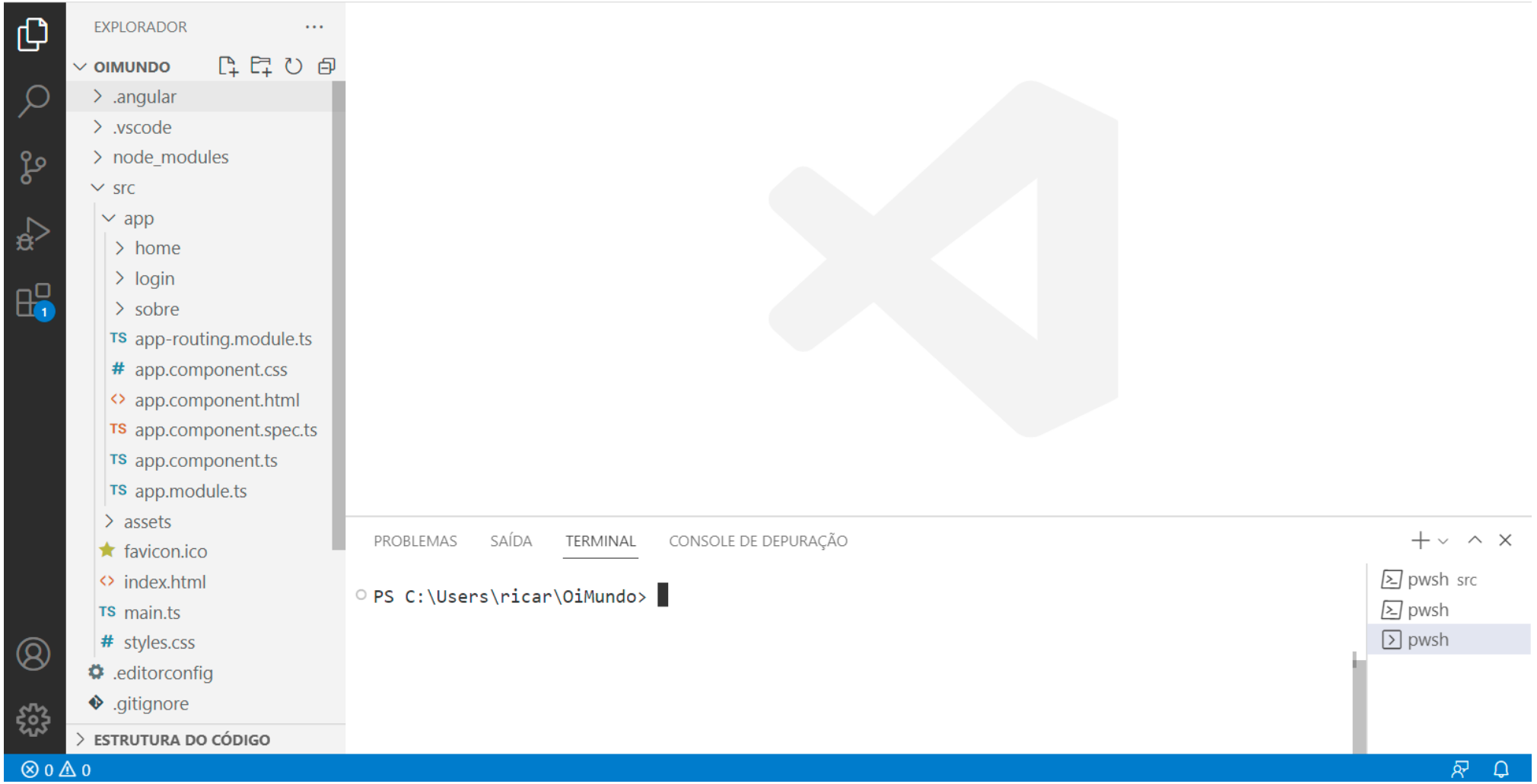
Geração de artefatos do Angular

- Se quiser adicionar seu componente, diretiva ou pipe para outro módulo (que não seja o módulo principal da aplicação, app.module.ts), você pode simplesmente prefixar o nome do componente com o nome do módulo e uma barra assim:
  - $ ng g component meu-modulo/meu-componente
- meu-modulo é o nome de um módulo existente.

Angular

- Digite:
  - $ ng g c home
  - $ ng g c login
  - $ ng g c sobre

Angular


EXPLORADOR
OIMUNDO
.angular
.vscode
node_modules
src
app
home
login
sobre
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
assets
favicon.ico
index.html
main.ts
styles.css
.editorconfig
.gitignore
ESTRUTURA DO CÓDIGO
PROBLEMAS
SAÍDA
TERMINAL
CONSOLE DE DEPURAÇÃO
PS C:\Users\ricar\OiMundo>
pwsh src
pwsh
pwsh
0 0

Angular

app-routing.module.ts

src > app > app-routing.module.ts > ...

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './login/login.component';
import { SobreComponent } from './sobre/sobre.component';
import { HomeComponent } from './home/home.component';

const routes: Routes = [
  {path: 'login', component: LoginComponent},
  {path: 'sobre', component: SobreComponent},
  {path: 'home', component: HomeComponent}
];

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

Angular

index.html

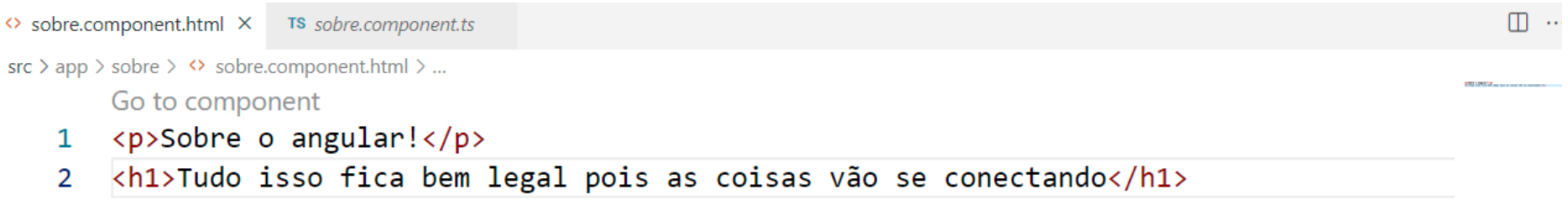
src > index.html > html > body > search

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Olá mundo</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <menu></menu>
  <search></search>
  <app-root></app-root>
</body>
</html>

```

Angular

- Coloque texto no html de cada componente criado



Angular

- O app.component é onde é iniciado todo o projeto.

app.component.html

src > app > app.component.html > router-outlet

Go to component

```
1  <h1>{{title}}</h1>
2  <nav>
3    <a routerLink="/">Index</a> <br>
4    <a routerLink="/home">Home </a><br>
5    <a routerLink="/login">Login </a><br>
6    <a routerLink="/sobre">Sobre</a><br>
7  </nav>
8  <router-outlet></router-outlet>
```

Atividade:

- Assista os vídeos:
  - https://www.youtube.com/watch?v=Yf0rC7dERjg
- Assista, pelo menos, os 8 primeiros vídeos desta playlist
  - https://www.youtube.com/watch?v=vJt_K1bFUeA&list=PLnDvRpP8Bnex2GQEN0768_AxZg_RaIGmw

Atividade

- Farei uma prova em cima do que foi abordado nos vídeos

Atividade

- Farei uma prova em cima do que foi abordado nos vídeos
  - BRINCADEIRA!

Atividade

- Farei uma prova em cima do que foi abordado nos vídeos
  - BRINCADEIRA!
- Mas seria bom vocês assistirem... hehe

Consumindo uma API com Angular

- Para usar o HttpClient, precisamos adicionar o modulo HttpClientModule no arquivo app.module.ts.
- Para fazer isso, vamos abrir o arquivo app.module.ts
- Dentro de imports do decorator @NgModule, adicione o modulo HttpClientModule
    - import { HttpClientModule } from '@angular/common/http';

Consumindo uma API com Angular
No app.module.ts:

- Vamos aproveitar e adicionar o modulo FormsMoule, esse modulo nos ajudará em nosso formulário para nossa tela de exemplo, porém não é necessário para o uso do HttpClient.

src > app > TS app.module.ts > AppModule

```
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { LoginComponent } from './login/login.component';
7  import { SobreComponent } from './sobre/sobre.component';
8  import { HomeComponent } from './home/home.component';
9  import { HttpClientModule } from '@angular/common/http';
10 import { FormsModule } from '@angular/forms';
```

Consumindo uma API com Angular
No app.module.ts:

```
13  @NgModule({
14    declarations: [
15      AppComponent,
16      LoginComponent,
17      SobreComponent,
18      HomeComponent
19    ],
20    imports: [
21      BrowserModule,
22      AppRoutingModule,
23      HttpClientModule,
24      FormsModule
25    ],
26    providers: [HttpClientModule, FormsModule],
27    bootstrap: [AppComponent]
28  })
```

Model

- Vamos criar uma interface de modelo para os dados.
- Na raiz do projeto vamos executando o seguinte comando:
  - ng g interface models/pessoa

Model

- Agora vamos adicionar o seguinte conteúdo dentro de nosso model pessoa.ts

src > app > models > TS pessoa.ts > ...

```ts
export interface Pessoa {
    id?: number;
    nome: string;
}
```

Service

- Vamos criar os métodos responsáveis pelas requisições http que faremos usando o HttpClient, mas antes devemos criar um arquivo service, no angular é recomendado criar services para os métodos que faz chamadas http.
- Para criar o serviço, digite o seguinte comando:
  - ng g service services/pessoa
- Será criado um arquivo com o nome pessoa.services.ts dentro da pasta services.
- Após criar nosso service, já podemos começar a utilizar o HttpClient

HttpClient

- Dentro do nosso service pessoa.services.ts, vamos criar alguns métodos http utilizando o HttpClient.

- Abra o arquivo pessoa.services.ts dentro da pasta services e inclua o seguinte código:

pessoa.service.ts

src > app > services > pessoa.service.ts > PessoaService

```
1  import { Injectable } from '@angular/core';
2  import { HttpClient, HttpErrorResponse, HttpHeaders } from '@angular/common/http';
3  import { Observable, throwError } from 'rxjs';
4  import { retry, catchError } from 'rxjs/operators';
5  import { Pessoa } from '../models/pessoa';
6
7  @Injectable({
8    providedIn: 'root'
9  })
```

HttpClient

- Continuando:

```
10  export class PessoaService {
11    url = 'https://localhost:7051/api/pessoas';
12
13    // injetando o HttpClient
14
15    constructor(private httpClient: HttpClient) { }
16    // Headers
17    httpOptions = {
18      headers: new HttpHeaders({ 'Content-Type': 'application/json' })
19    }
20    // Obtem todas pessoas
21    getPessoas(): Observable<Pessoa[]> {
22      return this.httpClient.get<Pessoa[]>(this.url)
23        .pipe(
24          retry(2),
25          catchError(this.handleError))
26    }
```

HttpClient

- Continuando:

```
28    // Obtem uma pessoa pelo id
29    getPessoaById(id: number): Observable<Pessoa> {
30     return this.httpClient.get<Pessoa>(this.url + '/' + id)
31       .pipe(
32         retry(2),
33         catchError(this.handleError))
34    }
35
36    // salva uma pessoa
37    savePessoa(pessoa: Pessoa) : Observable<Pessoa>{
38      return this.httpClient.post<Pessoa>(this.url, JSON.stringify(pessoa), this.httpOptions)
39        .pipe(
40          retry(2),
41          catchError(this.handleError)
42        )
43    }
44
```

HttpClient

- Continuando:

```
45    // utualiza uma pessoa
46    updatePessoa(pessoa: Pessoa): Observable<Pessoa> {
47      return this.httpClient.put<Pessoa>(this.url + '/' + pessoa.id, JSON.stringify(pessoa),
48      this.httpOptions)
49        .pipe(
50          retry(1),
51          catchError(this.handleError)
52        )
53    }
54
55    // deleta uma pessoa
56    deletePessoa(pessoa: Pessoa) {
57      return this.httpClient.delete<Pessoa>(this.url + '/' + pessoa.id)
58        .pipe(
59          retry(1),
60          catchError(this.handleError)
61        )
62    }
```

HttpClient

- Continuando:

```
64   addPessoa(pessoa: Pessoa){
65     console.log(this.url, JSON.stringify(pessoa))
66     return this.httpClient.post<Pessoa>(this.url, JSON.stringify(pessoa))
67     .pipe(
68       retry(2),
69       catchError(this.handleError)
70     )
71   }
72     // Manipulação de erros
73     handleError(error: HttpErrorResponse) {
74       let errorMessage = '';
75       if (error.error instanceof ErrorEvent) {
76         // Erro ocorreu no lado do client
77         errorMessage = error.error.message;
78       } else {
79         // Erro ocorreu no lado do servidor
80         errorMessage = `Código do erro: ${error.status}, ` +
81         `mensagem: ${error.message}`;
82       }
83       console.log(errorMessage);
84       return throwError(errorMessage);
85     };
```

No app.componente.ts

```ts
import { Component } from '@angular/core';
import { PessoaService } from './services/pessoa.service';
import { Pessoa } from './models/pessoa';
import { NgForm } from '@angular/forms';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'OiMundo';

  pessoa = {} as Pessoa;
  pessoas: Pessoa[] = [];

  constructor(private pessoaService: PessoaService) {}

  ngOnInit() {
    this.getPessoas();
  }
```

No app.component.ts

```
22  addPessoa(){
23    this.pessoa.nome = "Astolfo";
24    this.pessoaService.savePessoa(this.pessoa);
25
26  }
27
28  savePessoa(form: NgForm) {
29    if (this.pessoa.id !== undefined) {
30      this.pessoaService.updatePessoa(this.pessoa).subscribe(() => {
31        this.cleanForm(form);
32      });
33    } else {
34      this.pessoaService.savePessoa(this.pessoa).subscribe(() => {
35        this.cleanForm(form);
36      });
37    }
38  }
39
40  SavePessoaNome(n: string){
41    this.pessoa.nome = n;
42    this.pessoaService.savePessoa(this.pessoa);
43  }
```

No app.component.ts

```
45    getPessoas() {
46      this.pessoaService.getPessoas().subscribe((pessoas: Pessoa[]) => {
47        this.pessoas = pessoas;
48      });
49    }
50
51    deletePessoa(pessoa: Pessoa) {
52      this.pessoaService.deletePessoa(pessoa).subscribe(() => {
53        this.getPessoas();
54      });
55    }
56
57    editPessoa(pessoa: Pessoa) {
58      this.pessoa = { ...pessoa };
59    }
60
61    cleanForm(form: NgForm) {
62      this.getPessoas();
63      form.resetForm();
64      this.pessoa = {} as Pessoa;
65    }
66  }
```

Entendendo nosso serviço

- Injetamos o HttpClient e atribuímos a uma variável chamada httpClient, observe como é simples usar os métodos get, post, put e delete do nosso httpClient:
  - this.httpClient.get<Pessoa[]>(this.url)
  - this.httpClient.post< Pessoa >(this.url, JSON.stringify(pessoa), this.httpOptions)
  - this.httpClient.put< Pessoa >(this.url + '/' + pessoa.id, JSON.stringify(pessoa), this.httpOptions)
  - this.httpClient.delete< Pessoa >(this.url + '/' + pessoa.id, this.httpOptions)

Entendendo nosso serviço

- Note que passamos uma variável chamada “url” essa conterá o endereço http://localhost:5258/api/ disponibilizada pela API REST que temos desenvolvido no Visual Studio ASP.NET Web API .

- Adicionamos em nosso HttpClient um objeto do tipo HttpOptions, contendo nosso header através da classe HttpHeaders.

No app.component.html

```
16  <div class="container">
17    <div class="card list-pessoa">
18      <h5 class="card-header">Lista de pessoa</h5>
19      <div class="card-body">
20        <table class="table">
21          <thead>
22            <tr>
23              <th scope="col">Id</th>
24              <th scope="col">Nome</th>
25            </tr>
26          </thead>
27          <tbody>
28            <tr *ngFor="let pessoa of pessoas">
29              <td>{{pessoa.id}}</td>
30              <td>{{pessoa.nome}}</td>
31            </tr>
32          </tbody>
33        </table>
34      </div>
35    </div>
36
```

No app.component.html

```
42  <button (click)="getPessoas()" >Obter Todos</button> <br>
43  <button (click)="addPessoa()" >Inserir o Astolfo</button>
```

Caso ocorra erro ao acessar a API

- Adicione no Program.cs da API

```
public static void Main(string[] args)
{
    var builder = WebApplication.CreateBuilder(args);
    var MyAllowSpecificOrigins = "_myAllowSpecificOrigins";
    // Add services to the container.

    builder.Services.AddCors(options =>
    {
        options.AddPolicy(MyAllowSpecificOrigins,
                          policy =>
                          {
                              policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();

                          });
    });

    builder.Services.AddControllers();
```

```
37          app.UseHttpsRedirection();
38
39          app.UseAuthorization();
40          app.UseCors(MyAllowSpecificOrigins);
41
42          app.MapControllers();
43
44          app.Run();
```

Melhorando a interface
Modifique o app.component.html:

```
<head>
  <meta charset="utf-8">
  <title>AngularHttp</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/4.3.1/css/bootstrap.min.css">
</head>
<h1>{{title}}</h1>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item active" aria-current="page">Pessoas</li>
  </ol>
</nav>
```

Melhorando a interface
Modifique o app.component.html:

```
<div class="container">
  <div class="card list-pessoa">
    <h5 class="card-header">Lista de pessoa</h5>
    <div class="card-body">
      <table class="table">
        <thead>
          <tr>
            <th scope="col">Id</th>
            <th scope="col">Nome</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let pessoa of pessoas">
            <td>{{pessoa.id}}</td>
            <td>{{pessoa.nome}}</td>
            <td>
              <button type="button" class="btn btn-warning btn-sm" (click)="editPessoa(pessoa)">Editar</button>
              <button type="button" class="btn btn-danger btn-sm ml-1" (click)="deletePessoa(pessoa)">Deletar</button>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
```

Melhorando a interface
Modifique o app.component.html:

```
<div class="card add-car">
  <h5 class="card-header">Adicionar/Atualizar pessoa</h5>
  <div class="card-body">
      <form id="add-form" (ngSubmit)="f.form.valid && savePessoa(f)" #f="ngForm" novalidate>
        <div class="form-group">
            <label for="model">Nome</label>
            <input type="text" [(ngModel)]="pessoa.nome" id="nome" name="nome" #model="ngModel" required class="form-control" [ngClass]="{
                'is-invalid': f.submitted && model.invalid }" placeholder="Digite o Nome">
            <div *ngIf="f.submitted && model.invalid" class="invalid-feedback">
                <div *ngIf="model.errors">Nome é obrigatório</div>
            </div>
        </div>
       <button type="submit" class="btn btn-primary btn-add-car">Salvar</button>
        <button type="reset" class="btn btn-secondary btn-add-car" (click)="cleanForm(f)">Cancelar</button>
      </form>
  </div>
</div>
```

Lista de pessoa

Id	Nome	
7	Fabrício	Editar Deletar
5	Ricardo Frohlich da Silva	Editar Deletar
19	Julia	Editar Deletar
9	Mateus	Editar Deletar
18	Eduardo	Editar Deletar
22	Gabriel Mello Pinto	Editar Deletar

Adicionar/Atualizar pessoa

Nome

Digite o Nome

Salvar Cancelar

Materiais de apoio
Dica: Acompanhem todos!

- https://www.freecodecamp.org/portuguese/news/como-instalar-o-angular-no-windows-um-guia-de-angular-cli-node-js-e-ferramentas-de-criacao/
- https://medium.com/nave-recife/entendendo-a-estrutura-de-um-projeto-angular-aa22833a7491
- https://vidafullstack.com.br/angular/o-que-e-um-e-como-criar-um-componente-com-angular/
- https://balta.io/blog/angular-rotas-guardas-navegacao
- https://medium.com/angularbr/angular-5-trabalhando-com-rotas-8335617fcdbc
- https://www.devmedia.com.br/angular-http-como-realizar-requisicoes-em-suas-aplicacoes/40642
- https://www.youtube.com/watch?v=6QXBTeVf0VY
- https://medium.com/@fernandoevangelista_28291/consumindo-api-rest-com-httpclient-no-angular-8-62c5d733ffb6
- https://learn.microsoft.com/pt-br/aspnet/core/security/cors?view=aspnetcore-7.0
- https://www.macoratti.net/18/05/aspcore_cors1.htm