

Angular

Consumindo uma API com Angular

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 FormsModule, 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';
11
```

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 > ...  
1  export interface Pessoa {  
2      id?: number;  
3      nome: string;  
4  }
```

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:

```
TS pessoa.service.ts •
src > app > services > TS pessoa.service.ts > PessoaService
1  import { Injectable } from '@angular/core';
2  import { HttpClient, HttpResponse, 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     }
27 }
```

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 // atualiza 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: HttpResponse) {
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

```
1 ~ import { Component } from '@angular/core';
2 import { PessoaService } from '../services/pessoa.service';
3 import { Pessoa } from '../models/pessoa';
4 import { NgForm } from '@angular/forms';
5 ~ @Component({
6     selector: 'app-root',
7     templateUrl: '../app.component.html',
8     styleUrls: ['../app.component.css']
9 })
10 ~ export class AppComponent {
11     title = 'OiMundo';
12
13     pessoa = {} as Pessoa;
14     pessoas: Pessoa[] = [];
15
16     constructor(private pessoaService: PessoaService) {}
17
18 ~     ngOnInit() {
19         this.getPessoas();
20     }
21
```

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:

```
1 <head>
2   <meta charset="utf-8">
3   <title>AngularHttp</title>
4   <base href="/">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <link rel="icon" type="image/x-icon" href="favicon.ico">
7   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
8 </head>
9 <h1>{{title}}</h1>
10
11 <nav aria-label="breadcrumb">
12   <ol class="breadcrumb">
13     <li class="breadcrumb-item active" aria-current="page">Pessoas</li>
14   </ol>
15 </nav>
```

Melhorando a interface

Modifique o 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             <td>
32               <button type="button" class="btn btn-warning btn-sm" (click)="editPessoa(pessoa)">Editar</button>
33               <button type="button" class="btn btn-danger btn-sm ml-1" (click)="deletePessoa(pessoa)">Deletar</button>
34             </td>
35           </tr>
36         </tbody>
37       </table>
38     </div>
39   </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	Fabício	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

[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