Residência TCE/RN: Desenvolvimento de soluções Web Aula 04

Professor: José Alex

Pesquisador e Gerente de Projetos
do Smart Metropolis

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD INSTITUTO METRÓPOLE DIGITAL

Agenda

- 05/07/2019
  - HTML 5, CSS e JavaScript
- 12/07/2019
  - Ajax, Jquery e Angular
- 19/07/2019
  - Arquitetura de serviços (SOA) e Web Services SOAP, XML e XSLT
- 26/07/2019
  - Serviços REST, C#, ASP.NET Core (API Web) e Swagger

Serviços REST

Serviços REST

- REpresentation State Transfer, definido por Roy T. Fielding em sua tese de PhD;

- Padrão RESTful:
  - Recursos com Identificador (*endpoints*);
  - Requisições/verbos HTTP GET/POST/PUT/DELETE;
  - Tudo que é feito pela interface gráfica, deve ser feito também pela API;
  - Servidor não guarda o estado da comunicação.

- Utiliza uma notação comum para transferência de dados, normalmente JSON;

JSON

- JavaScript Object Notation é basicamente um formato leve de troca de informações/dados entre sistemas.

- Principais vantagens:
  - Leitura mais simples;
  - Parsing mais fácil;
  - Velocidade maior na execução e transporte de dados;
  - Arquivo com tamanho reduzido.

- Exemplo:

Dúvidas? Sugestões?
Comentários...

C#

C#

- O C# (C-Sharp) é uma linguagem de programação orientada a objeto e fortemente tipada criada pela Microsoft como parte do Framework .NET;

- Código armazenado em um arquivo com extensão .cs;

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace ExemploWebApi
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        1 reference
        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

C#

- Modificadores de acesso: public, protected, internal e private
- Tipos de dados: byte, int, short, long, float, double, char, object, bool, string, decimal...

```
//Tipo da Exceção
string tipo = pExcecao.GetType().Name;

//Exceção Propagada
string mensagemOriginal = pExcecao.Message;

//Número de métodos por onde a exceção passou
int tamanhoPilha = this.CountStackTrace(pExcecao.StackTrace);
```

- Operadores: x = y; x +=y ou x = x + y; x %=y ou x % y; &&; | |; ! ; == ; != ; >; >=; <; <=; ++; --; condicao ? X : y
- Condicionais: if () {}; if () {} else {}; if () {} else if () {} else {}; switch (a) { case ‘condicao’: break default: }
- Laços: for ([inicio]); [condicao]; [incremento]){}; do {} while (condicao); while (condicao) {}; foreach(){}

Dúvidas? Sugestões?
Comentários...

ASP.NET Core

ASP.NET Core

- **ASP.NET** é um framework para o desenvolvimento web baseado em HTML, CSS, JavaScript e Server Scripting. Suporta três modelos de desenvolvimento: Web Pages (SPA), MVC (Model View Controller), e Web Forms;

- Evolução:
  - 1996 (ASP);
  - 2002 ASP.NET;
  - 2008 ASP.NET MVC;
  - 2012 ASP.NET WebAPI;
  - 2016 ASP.NET Core.

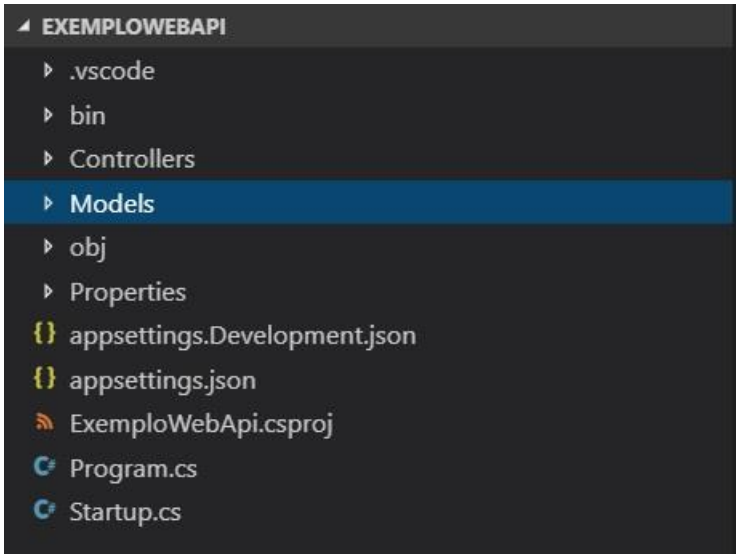
- ASP.NET Core é um novo framework open-source e multi-plataforma para criação de aplicativos modernos, tais como aplicações web e apps móveis;

ASP.NET Core – API Web

- VS Code;
- SDK 2.2 ou posterior do .NET Core (https://dotnet.microsoft.com/download/dotnet-core);
- Extensão C#;
- No prompt:
  - dotnet new webapi –o nomedoprojeto
  - code –r nomedoprojeto
- Pressione “sim” nas caixas de diálogos para adicionar as dependências necessárias ao projeto;
- CTRL + F5 para executar o projeto e digite no browser https://localhost:5001/api/values para verificar se o projeto está funcionando.

ASP.NET Core – API Web

- Adicione uma pasta *Models*;



- Adicione uma classe dentro da pasta Models;

```
namespace ExemploWebApi.Models
{
    7 references
    public class ExemploWebApiItem
    {
        2 references
        public long Id { get; set; }
        1 reference
        public string Name { get; set; }
        0 references
        public bool IsComplete { get; set; }
    }
}
```

ASP.NET Core – API Web

- Adicione uma classe de contexto para o banco de dados (EntityFramework), na pasta Models, e os registre no arquivo *Startup.cs*;

```
using Microsoft.EntityFrameworkCore;

namespace ExemploWebApi.Models
{
    4 references
    public class ExemploWebApiContext : DbContext
    {
        0 references
        public ExemploWebApiContext(DbContextOptions<ExemploWebApiContext> options)
            : base(options)
        {
        }

        7 references
        public DbSet<ExemploWebApiItem> ExemploWebApiItems { get; set; }
    }
}
```

ASP.NET Core – API Web

- No *Startup.cs*:

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using ExemploWebApi.Models;
using Microsoft.OpenApi.Models;

namespace ExemploWebApi
{
    1 reference
    public class Startup
    {
        0 references
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        1 reference
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<ExemploWebApiContext>(opt =>
                opt.UseInMemoryDatabase("ExemploWebApiList"));
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
```

ASP.NET Core – API Web

- Adicione uma classe para o controlador na pasta *Controllers*;

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ExemploWebApi.Models;

namespace ExemploWebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    0 references
    public class ExemploWebApiController : ControllerBase
    {
        13 references
        private readonly ExemploWebApiContext _context;

        0 references
        public ExemploWebApiController(ExemploWebApiContext context)
        {
            _context = context;

            if (_context.ExemploWebApiItems.Count() == 0)
            {
                _context.ExemploWebApiItems.Add(new ExemploWebApiItem { Name = "Item1" });
                _context.SaveChanges();
            }
        }
```

ASP.NET Core – API Web

- Adicionando o GET:

```
// GET: api/ExemploWebApi
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<ExemploWebApiItem>>> GetItems()
{
    return await _context.ExemploWebApiItems.ToListAsync();
}

// GET: api/ExemploWebApi/1
[HttpGet("{id}")]
1 reference
public async Task<ActionResult<ExemploWebApiItem>> GetItem(long id)
{
    var Item = await _context.ExemploWebApiItems.FindAsync(id);

    if (Item == null)
    {
        return NotFound();
    }

    return Item;
}
```

- Roteamento para o método: [HttpGet(“/nomedarota”)]
- As rotas não são case-sensitive;

ASP.NET Core – API Web

- Adicionando o POST e o PUT:

```
// POST: api/ExemploWebApi
[HttpPost]
0 references
public async Task<ActionResult<ExemploWebApiItem>> PostExemploWebApiItem(ExemploWebApiItem item)
{
    _context.ExemploWebApiItems.Add(item);
    await _context.SaveChangesAsync();

    return CreatedAtAction(nameof(GetItem), new { id = item.Id }, item);
}

// PUT: api/ExemploWebApi/1
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutItem(long id, ExemploWebApiItem item)
{
    if (id != item.Id)
    {
        return BadRequest();
    }

    _context.Entry(item).State = EntityState.Modified;
    await _context.SaveChangesAsync();

    return NoContent();
}
```

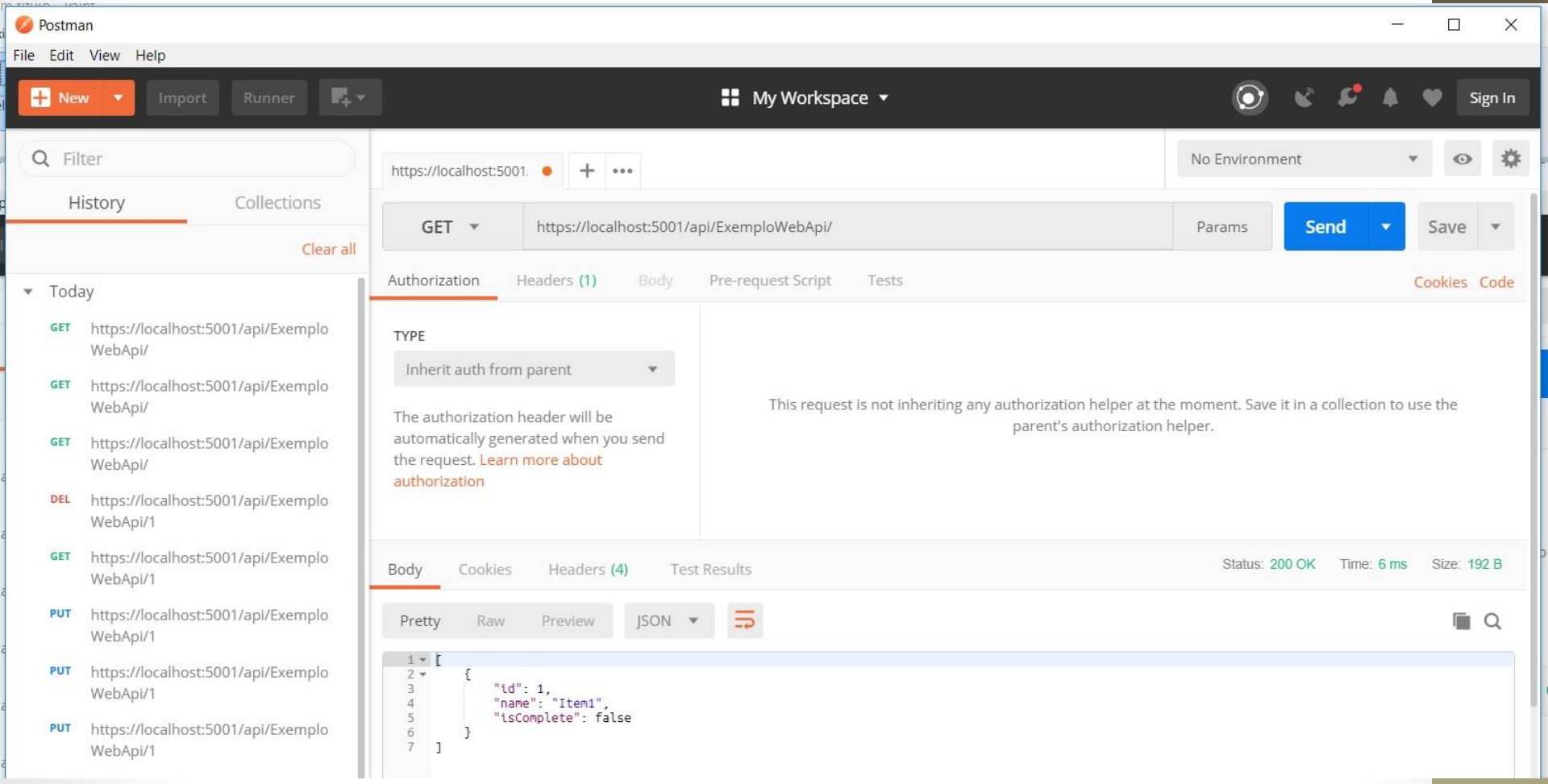
ASP.NET Core – API Web

- Adicionando o DELETE:

```
// DELETE: api/ExemploWebApi/1
[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteItem(long id)
{
    var ExemploWebApiItem = await _context.ExemploWebApiItems.FindAsync(id);

    if (ExemploWebApiItem == null)
    {
        return NotFound();
    }

    _context.ExemploWebApiItems.Remove(ExemploWebApiItem);
    await _context.SaveChangesAsync();

    return NoContent();
}
```

ASP.NET Core – API Web

- Postman:



Dúvidas? Sugestões?
Comentários...

Swagger

Swagger

- Solução para geração de páginas de ajuda e de documentação úteis para APIs Web;
- É uma especificação independente de linguagem para descrever API RESTs;
- Interface Web;
- No .NET é implementada pelo projeto **Swashbuckle.AspNetCore;**

Swagger

- Instalação (no prompt):
  - dotnet add seuprojeto.csproj package Swashbuckle.AspNetCore –v 5.0.0-rc2

- Na classe *Startup.cs:*

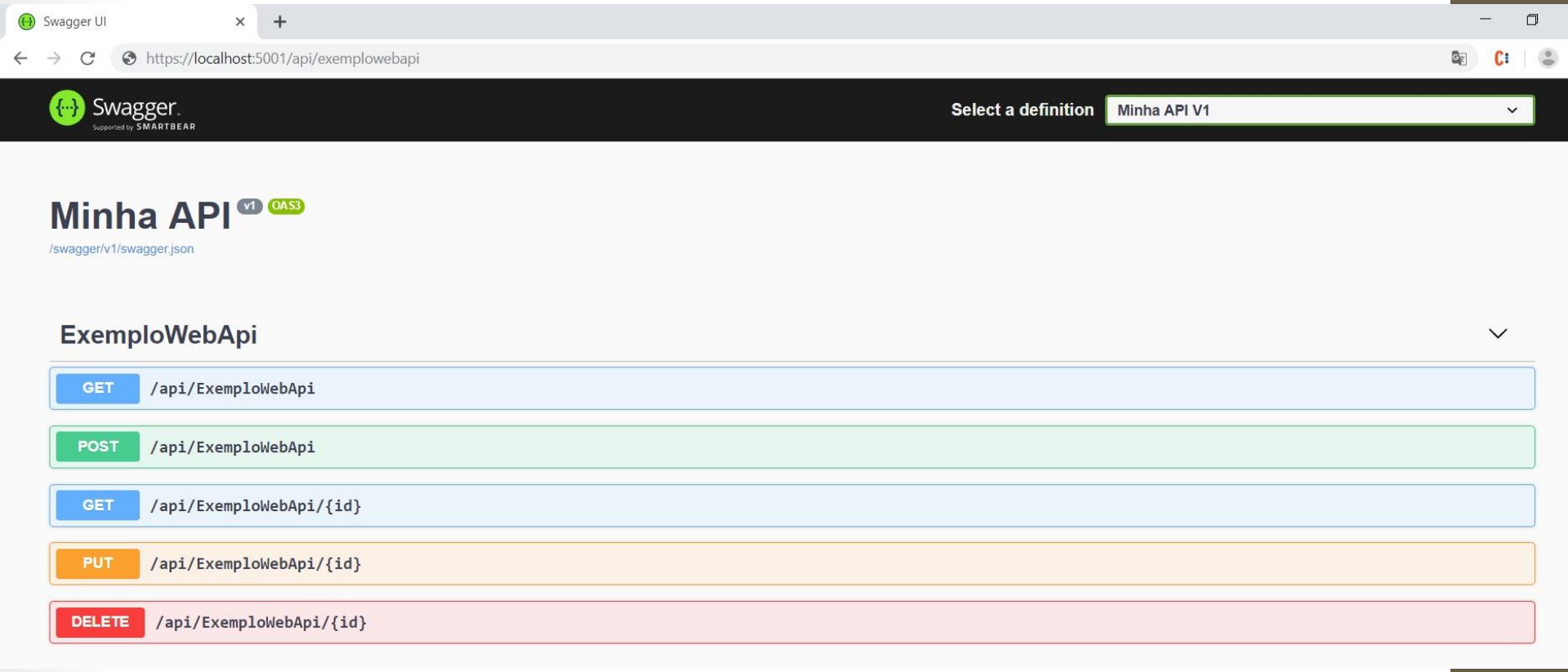
```
using Microsoft.Extensions.Options;
using ExemploWebApi.Models;
using Microsoft.OpenApi.Models;

namespace ExemploWebApi
{
    1 reference
    public class Startup
    {
        0 references
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        1 reference
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<ExemploWebApiContext>(opt =>
                opt.UseInMemoryDatabase("ExemploWebApiList"));
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);

            // Register the Swagger generator, defining 1 or more Swagger documents
            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new OpenApiInfo { Title = "Minha API", Version = "v1" });
            });
        }
```

Swagger

- Ainda na classe *Startup.cs:*

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        // The default HSTS value is 30 days. You may want to change this for production scenarios,
        app.UseHsts();
    }

    app.UseHttpsRedirection();

    // Enable middleware to serve generated Swagger as a JSON endpoint.
    app.UseSwagger();

    // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
    // specifying the Swagger JSON endpoint.
    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Minha API V1");
    });

    app.UseMvc();
}
```

Swagger


Swagger UI
https://localhost:5001/api/exemplowebapi
Swagger
Supported by SMARTBEAR
Select a definition
Minha API V1
Minha API
v1
OAS3
/swagger/v1/swagger.json
ExemploWebApi
GET /api/ExemploWebApi
POST /api/ExemploWebApi
GET /api/ExemploWebApi/{id}
PUT /api/ExemploWebApi/{id}
DELETE /api/ExemploWebApi/{id}

Dúvidas? Sugestões?
Comentários...

Chamando a API pelo Angular

No Angular

- Na classe de serviço:

```
import { Injectable } from '@angular/core';
import { Pessoa } from './pessoa';
import { HttpClient } from '@angular/common/http';

@Injectable()
export class PessoaService {
  private pessoas: Pessoa[] = new Array();

  Url = 'https://localhost:5001/api/exemplowebapi';

  constructor(private http: HttpClient) { }

  obterPessoas(){
    return this.http.get<any[]>(`${this.Url}`);

    /*let pessoa = new Pessoa();
    pessoa.id = 1;
    pessoa.nome = "Alex";
    pessoa.sobrenome = "Medeiros";
    pessoa.idade = 39;
    this.pessoas.push(pessoa);

    pessoa = new Pessoa();
    pessoa.id = 2;
    pessoa.nome = "Jose";
    pessoa.sobrenome = "Lima";
    pessoa.idade = 40;
    this.pessoas.push(pessoa);
```

No Angular

- No app.module:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { RouterModule, Routes } from '@angular/router';
import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';
import { ListarPessoasComponent } from './listar/listar.component';
import { EditarPessoaComponent } from './editar/editar.component';
import { HttpClient } from 'selenium-webdriver/http';


const appRoutes: Routes = [
  { path: 'pessoas', component: ListarPessoasComponent },
  { path: 'pessoas/:id', component: EditarPessoaComponent }
];

@NgModule({
  declarations: [
    AppComponent,
    ListarPessoasComponent,
    EditarPessoaComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule,
    RouterModule.forRoot(appRoutes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

No Angular

- Na classe do componente:

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Pessoa } from '../pessoa';
import { PessoaService } from '../pessoa.service';

@Component({
  selector: 'app-listar',
  templateUrl: './listar.component.html',
  styleUrls: ['./listar.component.css'],
  providers: [PessoaService]
})
export class ListarPessoasComponent implements OnInit {
  //pessoas: Pessoa[];

  pessoas: Array<any>;

  constructor(
    private router: Router,
    private service: PessoaService) { }

  ngOnInit() {
    //this.pessoas = this.service.obterPessoas();
    this.service.obterPessoas().subscribe(dados => this.pessoas = dados);
  }

  editarPessoa(pessoa: Pessoa) {
    this.router.navigate(['/pessoas', pessoa.id]);
  }
```

Dúvidas? Sugestões?
Comentários...

Vamos praticar!!!