

# Angular



The modern web developer's platform

1



# Contato

**Prof. Dr. Razer A N R Montaño**

razer@ufpr.br

SEPT/UFPR

2



## Conteúdo

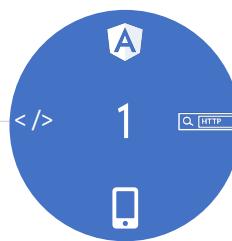
1. Introdução
2. Instalação do Ambiente
3. Hello World!
4. Estrutura do Projeto Hello World
5. Typescript
6. HTML
7. Projeto Soma
8. Elementos de *Templates* HTML
9. Bootstrap
10. Material Design
11. CRUD
12. Validação e Formatação de Campos
13. Modal para Mostrar Pessoa
14. Menu para Formulários
15. Outros Elementos de Formulários
16. Programação Reativa
17. Login
18. Web Services

Prof. Dr. Razer A N R Montaño

Angular

3

3



## Introdução

Prof. Dr. Razer A N R Montaño

Angular

4

4



## Introdução



### Objetivos

- ✓ Apresentar os conceitos básicos
- ✓ Apresentar a arquitetura de uma aplicação



### Referências

- <https://angular.io/docs>
- [https://en.wikipedia.org/wiki/Angular\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Angular_(web_framework))
- <https://blog.angular.io/angular-v13-is-now-available-cce66f7bc296>
- <https://www.typescriptlang.org/docs/>

Prof. Dr. Razer A N R Montaño

Angular

5

5



## Conteúdo

1. Tecnologias e Histórico
2. Organização de uma Aplicação

Prof. Dr. Razer A N R Montaño

Angular

6

6

1

## Tecnologias e Histórico

Prof. Dr. Razer A N R Montaño

Angular

7

7



## Angular

- *Framework* para desenvolvimento de Aplicações Web baseado em Componentes
- *Open-source*
- Baseado em TypeScript
- Criado pelo Google
  - Para resolver um problema de desenvolvimento interno
  - Mantido pelo time Angular na Google, mais indivíduos e corporações
  - É uma reescrita completa do AngularJS
- Usado para criar aplicações SPA (*Single Page Applications*)
  - Aplicações que rodam em uma única página
  - HTML, CSS, JS
  - Todo o conteúdo é carregado de uma vez ou carregado dinamicamente via AJAX
  - Evitam o *refresh* da página

Prof. Dr. Razer A N R Montaño

Angular

8

8



## Angular

- Aplicações SPA (*Single Page Applications*)
  - Melhora a experiência do usuário
  - Desempenho
  - Cliente (*client-side*) tem mais lógica
  - Como separam-se o *front* do *back* (em geral API REST) tem-se uma boa separação de responsabilidades e, manutenção

Prof. Dr. Razer A N R Montaño

Angular

9

9



## Angular.

- AngularJS != Angular
  - AngularJS é a versão 1:
    - Simples
    - baixo desempenho
    - código desorganizado
    - ES5
  - Angular é a versão 2+:
    - Usa padrões de projetos
    - Componentização
    - ES6
- É escrito em TypeScript
  - Pode ser usado com TS, ES5, ES6, Dart

Prof. Dr. Razer A N R Montaño

Angular

10

10



## Histórico

- 2009 – AngularJS : Misko Hevery e Adams Abrons. Framework para front-end SPA
- 14/09/2016 – Angular 2: Reescrita do AngularJS, muitas mudanças
- 13/12/2017 – Angular 4: Pulada a v3, melhorias diversas principalmente AOT
- 01/11/2017 – Angular 5: Melhorias diversas, suporte a *progressive web apps*
- 04/05/2018 – Angular 6: Melhorias voltadas às ferramentas
- 18/10/2018 – Angular 7: Melhorias diversas, suporte a Node 10
- 28/05/2019 – Angular 8: Melhorias diversas
- 06/02/2020 – Angular 9: Uso do TS 3.6 e 3.7, correções de bugs, compilador Ivy
- 24/06/2020 – Angular 10: Melhorias diversas

<continua>



## Histórico

- 11/Nov/2020 – Angular 11: Melhorias diversas. Migração para ESLint (verificação de código).
  - LTS terminou 11/Mai/2022.
  - <https://www.angularminds.com/blog/article/top-features-of-angular-11.html>
- **Da versão 2 até a 11 – Sem mais suporte**
- 12/Mai/2021 – Angular 12: Sintaxe de verificação de valores nulos (??). Uso do Ivy (Render Engine). Etc...
  - LTS termina 12/Nov/2022
  - <https://www.turing.com/blog/angular-12-new-features/>
- 04/Nov/2021 – Angular 13: Suporte TS 4.4. Uso somente do Ivy. Etc...
  - LTS termina em 04/Mai/2023
  - <https://www.mindinventory.com/blog/whats-new-in-angular-13/>
- 02/06/2022 – Angular 14: Standalone Components (não precisa de @NgModule). Formulários reativos tipados. Injeção de dependências fora dos construtores.
  - <https://blog.angular.io/angular-v14-is-now-available-391a6db736af>
  - LTS termina em 02/Dez/2023



## Histórico.

- 18/Nov/2022 – Angular 15: API Standalone em produção. Diretiva de imagem ficou estável. Etc.
  - LTS termina 18/Mai/2024.
  - <https://www.bacancytechnology.com/blog/whats-new-in-angular-15>
- ??01/Mai/2023 – Angular 16: ???
  - LTS termina 12/Nov/2022

Prof. Dr. Razer A N R Montaño

Angular

13

13



## ECMAScript

- Nome oficial do JavaScript
- Padronizado pela Ecma International
  - Antiga ECMA (European Computer Manufacturers Association), surgiu em 1961
  - **Ecma:** É uma organização de padrões (como a ISO) para sistemas de informações e comunicações
- Documento ECMA-262
- É o padrão da linguagem Javascript (interoperabilidade entre navegadores)
- Usada principalmente do lado do cliente, em navegadores
- Mas cresce do lado do servidor, com Node.js

Prof. Dr. Razer A N R Montaño

Angular

14

14



## ECMAScript

- Surgiu em 1995 por Brendan Eich
- Se tornou um padrão Ecma em 1997
- ES5, ECMAScript 5, Lançado em 2009
  - Uma versão da linguagem ECMAScript (Javascript)
- ES6, ES2015, ECMAScript 2015
  - Melhorias na sintaxe
  - Adição de classes
  - Adição de importações de módulos
  - Etc
- ECMAScript 2016, 2017, 2018, 2019, 2020, 2021, 2022
- Está na 13<sup>a</sup> edição: ECMAScript 2022
  - <https://kangax.github.io/compat-table/es6/>
- Compatibilidades: <https://kangax.github.io/compat-table/es6/>

Prof. Dr. Razer A N R Montaño

Angular

15

15



## ECMAScript: Compatibilidades.

Scripting engine conformance

Scripting engine	Reference application(s)	Conformance <sup>[46]</sup>			
		ES5 <sup>[47]</sup>	ES6 (2015) <sup>[48]</sup>	ES7 (2016) <sup>[49]</sup>	Newer (2017+) <sup>[49][50]</sup>
SpiderMonkey	Firefox 94	100%	98%	100%	100%
V8	Google Chrome 95, Microsoft Edge 95, Opera 80	100%	98%	100%	100%
JavaScriptCore	Safari 15	100%	99%	100%	90%

FONTE: <https://en.wikipedia.org/wiki/ECMAScript>

Prof. Dr. Razer A N R Montaño

Angular

16

16



## Node.js

- Ambiente de execução Javascript (*runtime environment*)
- Criado por Ryan Dahl em 2009
- *Open-source*
- *Cross-platform*
- Permite execução de scripts do lado do servidor (*back-end*)
  - Criação de APIs
  - Produção de páginas dinâmicas
- Possui um gerenciador de pacotes: NPM
  - *Node Package Manager*

Prof. Dr. Razer A N R Montaño

Angular

17

17



## Node.js.

- Site oficial
  - <https://nodejs.org/en/>
- Versão recomendada: Versão 18.15.0
- Atualmente na versão 19.x.x
  - <https://nodejs.org/en/about/releases/>
- Instalação
  - <https://nodejs.org/en/download/>
- Via BREW
  - <https://formulae.brew.sh/formula/node>

Prof. Dr. Razer A N R Montaño

Angular

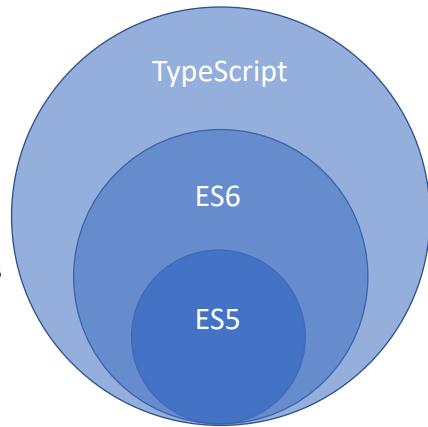
18

18



## TypeScript

- Criada pela Microsoft
- É um superconjunto do ES6
- Objetivos:
  - Melhorar a tipagem do Javascript
  - Checagem estática (antes da execução) de tipos de dados
  - Encapsulamento: uso de atributos privados
  - Herança
  - Abstração
  - Polimorfismo



Prof. Dr. Razer A N R Montaño

Angular

19

19



## TypeScript

- Out/2012 – TypeScript 0.8
- Jun/2013 – TypeScript 0.9
- Abr/2014 – TypeScript 1.0
- Set/2016 – TypeScript 2.0
- Jul/2018 – TypeScript 3.0
- Ago/2020 – TypeScript 4.0
- Nov/2020 – Typescript 4.1
- Fev/2021 – Typescript 4.2
- Mai/2021 – Typescript 4.3
- Ago/2021 – Typescript 4.4
- Nov/2021 – Typescript 4.5
- Fev/2022 – Typescript 4.6
  - <https://devblogs.microsoft.com/typescript/announcing-typescript-4-6/>
- Mai/2022 – Typescript 4.7
  - <https://devblogs.microsoft.com/typescript/announcing-typescript-4-7/>
- Mar/2023 – Typescript 5.0
  - <https://devblogs.microsoft.com/typescript/announcing-typescript-5-0/>

Prof. Dr. Razer A N R Montaño

Angular

20

20



## TypeScript

- Um código TS precisa ser compilado
  - Gera um código Javascript : ES5
  - Compilador TS faz toda a checagem de tipos e gera um Javascript clássico, sem tipagem
  - Pode-se intercalar código TS e Javascript

```
interface IComponent{
    getId() : string;
}

class Button implements IComponent{
    id:string;
    getId():string{
        return this.id;
    }
}
```

TS



```
var Button = (function () {
    function Button() {
    }
    Button.prototype.getId = function () {
        return this.id;
    };
    return Button;
})();
```

Javascript

FONTE: <https://tableless.com.br/diga-ola-ao-typescript-e-adeus-ao-javascript/>

Prof. Dr. Razer A N R Montaño

Angular

21

21



## TypeScript.

- Para testar o TS
  - Playground: <https://www.typescriptlang.org/Playground>
  - Verifica o código
  - Gera código Javascript compilado

Prof. Dr. Razer A N R Montaño

Angular

22

22

2

## Organização de uma Aplicação

Prof. Dr. Razer A N R Montaño

Angular

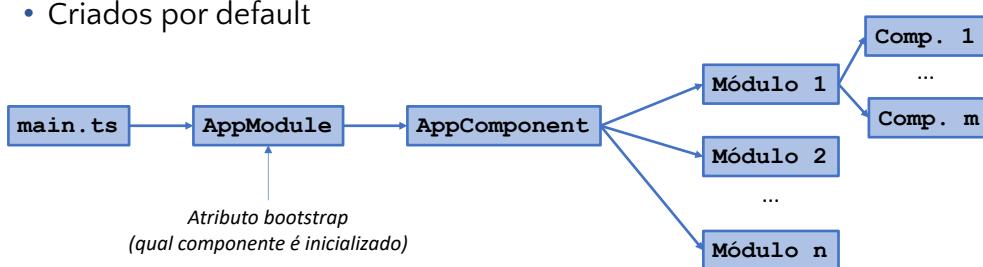
23

23



## Organização de uma Aplicação

- Aplicação é modularizada: separada em módulos
- Dentro dos módulos pode-se criar componentes
- A aplicação se inicia no **main.ts**, que carrega o **AppModule** que carrega o **AppComponent**
  - Criados por default



Prof. Dr. Razer A N R Montaño

Angular

24

24



## O que é um Componente

- Um trecho visual da aplicação e seu código subjacente
- Contém
  - HTML : a tela
  - CSS : o estilo
  - TS : o comportamento
- A ideia é encapsular sua lógica

Prof. Dr. Razer A N R Montaño

Angular

25

25



## O que é um Componente

- Por exemplo
  - Tela Inserção Pessoa – Componente **InserirPessoaComponent**:
    - `inserir-pessoa.component.html`
    - `inserir-pessoa.component.css`
    - `inserir-pessoa.component.ts`
- Para usar o Componente **InserirPessoaComponent**:
  - É criada uma *tag* personalizada
  - Atributo **selector** da classe do componente (arquivo .ts)

```
<app-inserir-pessoa></app-inserir-pessoa>
```

- Referencia todo o HTML, CSS, TS do componente

Prof. Dr. Razer A N R Montaño

Angular

26

26



## O que é um Componente

```
@Component({
  selector: 'app-inserir-pessoa',
  templateUrl: './inserir-pessoa.component.html',
  styleUrls: ['./inserir-pessoa.component.css']
})
export class InserirPessoaComponent implements OnInit {
  constructor() {}

  ngOnInit(): void {}
}
```

Prof. Dr. Razer A N R Montaño

Angular

27

27



## O que é um Componente

```
@Component({
  selector: 'app-inserir-pessoa',
  templateUrl: './inserir-pessoa.component.html',
  styleUrls: ['./inserir-pessoa.component.css']
})
export class InserirPessoaComponent implements OnInit {
  constructor() {}

  ngOnInit(): void {}
}
```

Tag HTML customizada

Template HTML

Estilo CSS

Prof. Dr. Razer A N R Montaño

Angular

28

28



## O que é um Módulo

- Elemento de organização
- Agrupador de Componentes
- Usado para controlar a visibilidade de componentes
  - Componentes visíveis somente dentro do módulo, ou visível para os demais
- Toda aplicação inicia com um módulo principal
  - **AppModule** : `app.module.ts`
- Pode-se criar componentes diretamente em **AppModule**
- Pode-se criar outros módulos abaixo
  - E então criar componentes



## O que é um Módulo

- Por exemplo, dentro de **AppModule**, pode-se criar o módulo de Pessoa
  - Contém todos os componentes de CRUD de Pessoa
- Será composto por um diretório **pessoa** com um arquivo
  - `pessoa.module.ts`
  - Classe **PessoaModule**
- Contendo
  - **declarations**: Todo que faz parte do módulo: Componentes, Diretivas e Pipes
  - **exports**: Todo que é exportado e será visível em outros módulos: Componentes, Diretivas e Pipes
  - **imports**: Todos os módulos que serão importados para serem usados aqui, módulos dos quais este depende
  - **providers**: Todos os serviços que serão usados no módulo
  - **bootstrap**: para o **AppModule**, se refere ao primeiro componente a ser carregado



## O que é um Módulo

```
@NgModule({
  declarations: [
    ],
  imports: [
    ],
  providers: [
    ],
  exports: [
    ]
})
export class PessoaModule { }
```

Prof. Dr. Razer A N R Montaño

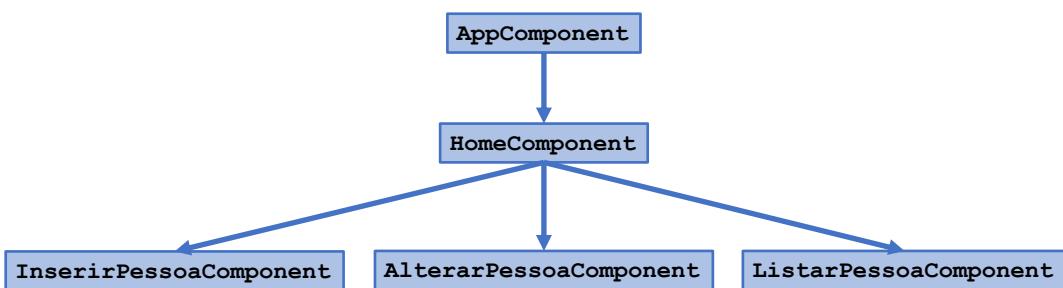
Angular

31

31



## Módulo x Componentes



Prof. Dr. Razer A N R Montaño

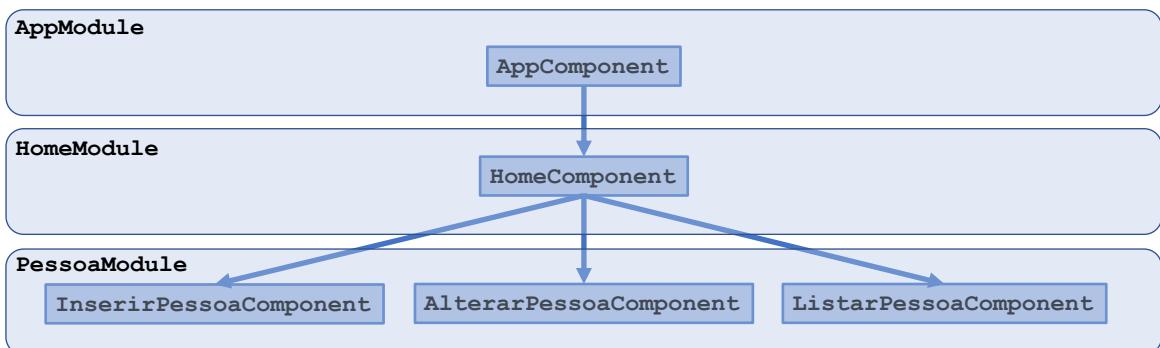
Angular

32

32



## Módulo x Componentes



Prof. Dr. Razer A N R Montaño

Angular

33

33



## Projeto em Produção..

- O navegador só entende: JS, HTML, CSS
- Então um projeto em Angular com TypeScript precisa ser **convertido, transpilado** para gerar os arquivos necessários
  - Verificação de tipos
  - Geração de JS a partir do TS
- Só há um HTML: SPA
  - Todas as telas são geradas pela manipulação do DOM
  - Telas inseridas dinamicamente na única página do sistema

Prof. Dr. Razer A N R Montaño

Angular

34

34



# Instalação do Ambiente

Prof. Dr. Razer A N R Montaño

Angular

35

35



## Instalação do Ambiente

### 💡 Objetivos

- ✓ Instalar Node.js
- ✓ Instalar o Angular CLI
- ✓ Comandos do Angular CLI



### 🌐 Referências

➤ <https://nodejs.org/pt-br/download/package-manager/>

Prof. Dr. Razer A N R Montaño

Angular

36

36



## Instalação do Ambiente

- Angular, Angular CLI e Aplicações dependem de pacotes NPM
  - Angular CLI – *Command Line Interface*
- NPM (*Node Packages Manager*) é instalado com o Node.js
- Deve-se então instalar o Node.js
  - Depende do seu SO
  - <https://nodejs.org/pt-br/download/package-manager/>
  - Binários em: <https://nodejs.org/en/download/>
- Instalar Node 16.14.2

**MÃOS À OBRA!!!!**



## Instalação do Ambiente

- Usa-se linha de comando
- Verificar versão do Node instalado

`$ node --version`

```
MacBook-Pro:~/ME/_/_3/Docente/Disciplinas_Material/Angular/lab13$ node --version
v16.14.2
```



## Instalação do Ambiente

- Verificar versão do NPM instalado

```
$ npm --version
```

```
[MacBook-Pro: ~/ME/_/___3/Docente/Disciplinas_Material/Angular/lab13] $ npm --version  
8.5.0
```



## Instalação do Ambiente

- Se já tem o ambiente montado
  - Atualização do Typescript e Angular CLI

```
$ npm install -g typescript@latest  
$ npm install -g @angular/cli@latest
```

- Para uma instalação nova
  - Instalação do Angular CLI

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



## Instalação do Ambiente

- Verificação da versão do Angular CLI instalada

```
$ ng --version
```

```
Angular CLI: 13.3.0
Node: 16.14.2
Package Manager: npm 8.5.0
OS: darwin x64

Angular: undefined
...
Package          Version
@angular-devkit/architect    0.1303.0 (cli-only)
@angular-devkit/core         13.3.0 (cli-only)
@angular-devkit/schematics   13.3.0 (cli-only)
@schematics/angular          13.3.0 (cli-only)
rxjs                         7.5.5
```

Prof. Dr. Razer A N R Montaño

Angular

41

41



## Instalação do Ambiente

- Verificação de pacotes NPM

```
$ npm list -global --depth 0
```

```
/usr/local/lib
└── @angular/cli@13.3.0
  ├── corepack@0.10.0
  ├── grunt-cli@1.3.2
  ├── json-server@0.16.3
  ├── n@8.1.0
  ├── npm@8.5.0
  └── typescript@4.6.3
    └── yarle-evernote-to-md@4.5.3
```

Prof. Dr. Razer A N R Montaño

Angular

42

42



## Instalação do Ambiente.

- Se o Angular avisar que a versão do Node não é suportada
  - Faça *downgrade* para a última versão suportada

```
$ npm install -g n  
$ n 16.14.2
```

Prof. Dr. Razer A N R Montaño

Angular

43

43



## Comandos do Angular CLI

- Comandos de Linha de Comando
- Criação de um novo projeto

```
$ ng new <nome-do-projeto>
```
- Criação de um novo módulo

```
$ ng generate module <nome-do-módulo>
```
- Criação de um novo componente

```
$ ng generate component <nome-do-componente>
```
- Criação de um novo serviço

```
$ ng generate service <nome-do-serviço>
```

Prof. Dr. Razer A N R Montaño

Angular

44

44



## Comandos do Angular CLI

- Subir um servidor com a aplicação, para testar localmente
  - Se os arquivos se alterarem, recompila automaticamente

```
$ ng serve
```

- Compilar o projeto (gera .js, .html, .css, etc), no diretório **./dist**

```
$ ng build --prod
```

Prof. Dr. Razer A N R Montaño

Angular

45

45



## Comandos do Angular CLI..

- Para obter ajuda, pode-se usar os comandos

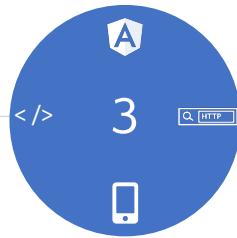
```
$ ng new --help  
$ ng generate --help  
$ ng serve --help  
$ ng build --help
```

Prof. Dr. Razer A N R Montaño

Angular

46

46



# Hello World!

Prof. Dr. Razer A N R Montaño

Angular

47

47



## Hello World

### ➡ Objetivos

- ✓ Criar seu primeiro projeto
- ✓ Executar seu primeiro projeto

### ➡ Referências

➤ <https://nodejs.org/pt-br/download/package-manager/>

Prof. Dr. Razer A N R Montaño

Angular

48

48



## Hello World

- Rodar

**\$ ng new hello-world**

- Na pergunta sobre *Angular routing*, pode dar enter
- Na pergunta sobre *CSS*, enter

- Resultado

- Cria o diretório **hello-world**
- Cria os arquivos para o projeto
- Instala dependências

```
Do you want to enforce strict type checking and stricter bundle budgets in the workspace?
This setting helps improve maintainability and catch bugs ahead of time.
For more information, see https://angular.io/strict No
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE hello-world/README.md (1019 bytes)
CREATE hello-world/.editorconfig (274 bytes)
CREATE hello-world/.gitignore (631 bytes)
CREATE hello-world/angular.json (3575 bytes)
CREATE hello-world/package.json (1201 bytes)
CREATE hello-world/tsconfig.json (458 bytes)
CREATE hello-world/tslint.json (3185 bytes)
CREATE hello-world/browserslistrc (763 bytes)
CREATE hello-world/karma.conf.js (1428 bytes)
CREATE hello-world/tsconfig.app.json (287 bytes)
CREATE hello-world/tsconfig.ng.e2e.json (333 bytes)
CREATE hello-world/src/favicon.ico (940 bytes)
CREATE hello-world/src/index.html (206 bytes)
CREATE hello-world/src/main.ts (372 bytes)
CREATE hello-world/src/polyfills.ts (2826 bytes)
CREATE hello-world/src/styles.css (80 bytes)
CREATE hello-world/src/test.ts (753 bytes)
CREATE hello-world/src/assets/.gitkeep (0 bytes)
CREATE hello-world/src/environments/environment.prod.ts (51 bytes)
CREATE hello-world/src/environments/environment.ts (662 bytes)
CREATE hello-world/src/app/app.module.ts (314 bytes)
CREATE hello-world/src/app/app.component.css (0 bytes)
CREATE hello-world/src/app/app.component.html (25725 bytes)
CREATE hello-world/src/app/app.component.spec.ts (955 bytes)
CREATE hello-world/src/app/app.component.ts (215 bytes)
CREATE hello-world/e2e/protractor.conf.js (904 bytes)
CREATE hello-world/e2e/tsconfig.e2e.json (274 bytes)
CREATE hello-world/e2e/src/app/e2e-spec.ts (662 bytes)
CREATE hello-world/e2e/src/app.po.ts (274 bytes)
✓ Packages installed successfully.
Successfully initialized git.
```



## Hello World

- Para testar a nova App
  - Entrar no diretório **hello-world**
  - Rodar

**\$ ng serve**

- Roda um servidor na porta 4200
- Então abrir o navegador



## Hello World

- Comando `ng serve`

```
~/ME/_/_3/Doce/D/Angular/lab13/hello-world ➜ master !1 ➜ ng serve
✓ Browser application bundle generation complete.

Initial Chunk Files | Names      | Raw Size
vendor.js           | vendor     | 1.70 MB |
polyfills.js        | polyfills  | 294.80 kB |
styles.css, styles.js | styles    | 173.23 kB |
runtime.js          | runtime   | 6.52 kB |
main.js             | main      | 5.57 kB |

| Initial Total | 2.17 MB

Build at: 2022-04-20T12:45:28.794Z - Hash: 8d19db2213e06e3b - Time: 6684ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

Prof. Dr. Razer A N R Montaño

Angular

51

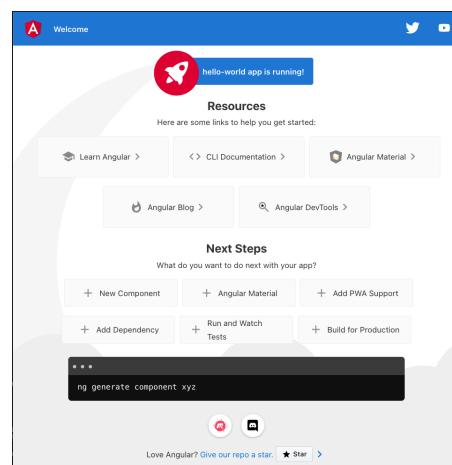
51



## Hello World

- Abrir navegador

`http://localhost:4200`



Prof. Dr. Razer A N R Montaño

52

52

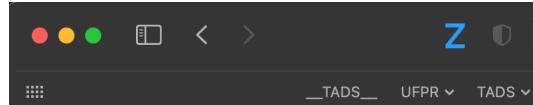


## Hello World..

- Para alterar o HTML sendo mostrado
- Alterar o arquivo *src/app/app.component.html*
  - Coloque seu HTML

```
hello-world > src > app > app.component.html > ...
1 |  <h1>Hello World!!!</h1>
2 |
```

- Resultado:



**Hello World!!!**



## EXERCÍCIOS..

1. Executar o projeto Hello World
2. Criar um novo projeto Página Pessoal
  - a) Alterar o HTML para conter seus dados pessoais
  - b) Adicione imagens no diretório **assets** subdiretório **img**
  - c) Adicione uma imagem na página da seguinte forma

```

```



# Estrutura do Projeto Hello World

Prof. Dr. Razer A N R Montaño

Angular

55

55



## Estrutura do Projeto

### 💡 Objetivos

- ✓ Apresentar a estrutura de diretórios criada para um projeto
- ✓ Apresentar os arquivos e seu propósito



### 🌐 Referências

- <https://angular.io/guide/file-structure>

Prof. Dr. Razer A N R Montaño

Angular

56

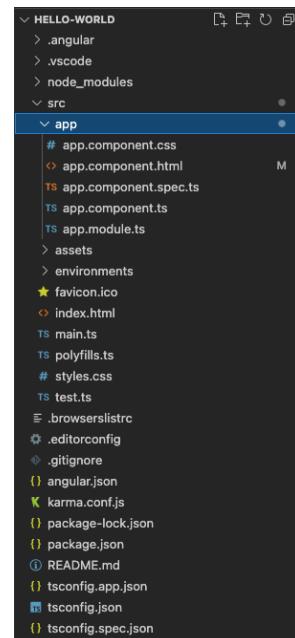
56



## Estrutura do Projeto

- Ao criar o projeto com  
  
    > **ng new hello-world**

- Tem-se a estrutura da imagem
- Editor usado é o VS Code



Prof. Dr. Razer A N R Montaño

Angular

57

57



## Estrutura do Projeto

- Arquivos dentro do diretório **hello-world**:
  - **.editorconfig**: Configuração para editores de texto
  - **.browserslistrc**: Informações de compatibilidade com navegadores
  - **.gitignore**: Arquivos que não devem ser versionados
  - **angular.json**: Arquivo de configuração do projeto (metadados)
  - **karma.conf.js**: Configuração da execução de testes unitários
  - **package.json**: Contém todas as dependências do projeto. São baixadas automaticamente
  - **package-lock.json**: Contém a árvore de dependências exatas (sem a sintaxe com \*)

Prof. Dr. Razer A N R Montaño

Angular

58

58



## Estrutura do Projeto

- Arquivos dentro do diretório ***hello-world***:
  - ***README.md***: Arquivo em formato *markdown* com informações textuais do projeto
  - ***tsconfig.json***: Informações sobre compilação do código *typescript*, geral
  - ***tsconfig.app.json***: Informações sobre compilação do código *typescript*, específico, usado quando se tem várias aplicações no mesmo espaço de trabalho
  - ***tsconfig.spec.json***: Informações sobre compilação do código de testes



## Estrutura do Projeto

- Diretórios criados
  - Diretório ***node\_modules***: Contém todas as bibliotecas usadas pelo projeto. Baixado automaticamente pelo **npm** (não deve ser versionado)
  - Diretório ***src***: Contém o código fonte da aplicação



## Estrutura do Projeto

- Arquivos dentro do diretório **src**:
  - **favicon.ico**: Ícone da aplicação
  - **main.ts**: Faz a inicialização da aplicação e carrega o módulo principal
  - **polyfills.ts**: Importação de bibliotecas para compatibilidade com diferentes navegadores
  - **styles.css**: Arquivo de CSS global
  - **test.ts**: Arquivo usado em testes, para carregar todas as classes de testes

Prof. Dr. Razer A N R Montaño

Angular

61

61



## Estrutura do Projeto

- Arquivos dentro do diretório **src**:
  - **index.html**: HTML principal da aplicação, usando uma *tag*

```
<app-root></app-root>
```

E é neste ponto que a tela da aplicação é renderizada

```
hello-world > src > index.html > ...
1  <!doctype html>
2  <html lang="en">
3  <head>
4  |  <meta charset="utf-8">
5  |  <title>HelloWorld</title>
6  |  <base href="/">
7  |  <meta name="viewport" content="width=device-width, initial-scale=1">
8  |  <link rel="icon" type="image/x-icon" href="favicon.ico">
9  </head>
10 <body>
11 |  <app-root></app-root>
12 </body>
13 </html>
14 |
```

Prof. Dr. Razer A N R Montaño

Angular

62

62



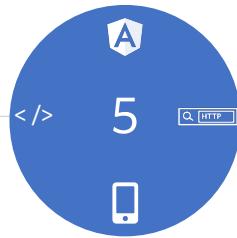
## Estrutura do Projeto..

- Subdiretórios dentro do diretório **src**:
  - Subdiretório **assets**: Local onde são colocados os recursos, como imagens
  - Subdiretório **environment**: Configurações de variáveis de ambiente, exemplo, para distinguir entre ambiente de desenvolvimento e produção
  - Subdiretório **app**: Arquivos da aplicação



## EXERCÍCIOS..

1. Abrir a estrutura do projeto Hello World e identificar todos os arquivos apresentados



# TypeScript

Prof. Dr. Razer A N R Montaño

Angular

65

65



## TypeScript

### ☰ Objetivos

- ✓ Apresentar o básico de TypeScript
- ✓ Mostrar elementos básicos de uma linguagem de programação, em TypeScript



### Referências

- <https://www.typescriptlang.org/docs/handbook/intro.html>

Prof. Dr. Razer A N R Montaño

Angular

66

66



## Conteúdo

1. Variáveis e Tipos
2. Operações
3. Comandos Condicionais
4. Comandos de Repetição
5. Funções
6. Orientação a Objetos
7. Módulos

Prof. Dr. Razer A N R Montaño

Angular

67

67



## TypeScript

- Será a linguagem básica para desenvolvimento em Angular
- Código aberto
- Microsoft em 2012
- Orientada a objetos
- Fortemente tipada
- Superconjunto do Javascript
- Tudo que é feito Javascript é aceito em TypeScript

Prof. Dr. Razer A N R Montaño

Angular

68

68



## TypeScript

- 2 maneiras para executar os testes
- Site TypeScript Playground
  - <https://www.typescriptlang.org/>
- Usando o Node.js e o TypeScript
  - Instalar Node.js (se ainda não estiver instalado):
    - <https://nodejs.org/en/download/>
  - Instalar o TypeScript:

```
$ npm install -g typescript
```



## TypeScript

- Com o Node.js e o TypeScript
  - Criar um arquivo, ex, **teste.ts**
  - Compilar para Javascript
    - Gera o arquivo **.js**

```
$ tsc teste.ts
```

- Rodar o JS criado com Node.js

```
$ node teste.js
```



## TypeScript.

- Exemplo, crie o arquivo **teste.ts** contendo o seguinte conteúdo

```
let idade : number = 20;  
let nome : string = "Razer";  
  
console.log(`Meu nome é ${nome} e tenho ${idade} anos. `);
```

- Execute

```
$ tsc teste.ts  
$ node teste.js
```

1

## Variáveis e Tipos



## Variáveis e Tipos

- Tipos

Nome	Descrição	
<b>string</b>	representa um texto	"texto", 'texto', `texto`
<b>number</b>	representa valores numéricos	64 bits: inteiros são bigint, números decimais em ponto flutuante padrão IEEE 754
<b>boolean</b>	representa valores lógicos: true e false	
<b>null</b>	representa o null	
<b>undefined</b>	representa o undefined	
<b>any</b>	qualquer coisa	Não faz checagem de tipo
<b>object</b>	todos os tipos que não são primitivos	



## Declarações

- Usa-se ":" para indicar o tipo
  - Se não for indicado, o tipo é inferido
- Exemplos simples

```
let idade = 18;      // infere um number
let nome : string = "Razer"; // é uma string
let preco : number = 950.8; // é um number
```



## Strings

- Definição
  - Podem ser definidas com " ou ' ou `
  - ` são usadas para interpolação e strings de várias linhas
- Exemplos simples

```
let descricao = `Olá gente
```

```
Precisamos conversar.
```

```
Abs, Razer`;
```

Prof. Dr. Razer A N R Montaño

Angular

75

75



## Strings

- Interpolação
  - Incluir variáveis no meio de strings
- Exemplos simples

```
let nome = "Razer";
let sobrenome = "Montaño";
let texto = `Olá ${nome}, da casa ${sobrenome}.
```

```
Bem-vindo!!!`;
```

Prof. Dr. Razer A N R Montaño

Angular

76

76



## Strings.

- Comparação
  - Operador ===
- Exemplos simples

```
let nome = "Razer";
let sobrenome = "Montaño";
if (nome === sobrenome) {
  console.log("iguais");
}
else {
  console.log("diferentes");
}
```

Prof. Dr. Razer A N R Montaño

Angular

77

77



## Arrays

- Armazena várias informações sequencialmente
  - Inicia em 0
  - Acessados via colchetes []
- Declaração:
  - string[]
  - Array<string>

```
let frutas : string[] = [ "Morango", "Maçã", "Laranja" ];
let planetas : Array<string> = [ "Terra", "Vênus", "Marte" ];
```

Prof. Dr. Razer A N R Montaño

Angular

78

78



## Arrays

- Declaração com vários tipos

```
let valores : (string | number)[] = [ "Razer", 20, "Teste", 77, 90];
```

ou

```
let valores : Array<string | number> = [ "Razer", 20, "Teste", 77, 90];
```

Prof. Dr. Razer A N R Montaño

Angular

79

79



## Arrays

- Acesso

```
let planetas : Array<string> = [ "Terra", "Vênus", "Marte" ];

console.log(planetas[2]); // obtém um elemento
planetas[1] = "Júpiter"; // seta um elemento
console.log(planetas); // mostra o array inteiro
```

- Tamanho do array

```
console.log(planetas.length);
```

Prof. Dr. Razer A N R Montaño

Angular

80

80



## Arrays.

- Métodos sobre array
  - **pop()**: remove o último elemento e o retorna
  - **push()**: insere um elemento no final e retorna o novo tamanho
  - **sort()**: ordena os elementos
  - **join()**: concatena todos os elementos em uma string e a retorna
- Exemplo

```
console.log(planetas.sort());
```



## Tipo any

- Typescript faz verificação de tipo
- Quando não se sabe o tipo
  - Tratar variáveis com tipos dinâmicos: **any**
- O tipo **any** desliga a verificação de tipos
- Exemplo

```
let aux : any = "Meu teste";
console.log(aux);
aux = 10.5;
console.log(aux);
```



## Tipo any.

- Quando é declarada uma variável sem tipo
  - Typescript infere o tipo **any**
- Exemplo

```
let aux1;  
aux1 = "Olá mundo";  
console.log(aux1);  
aux1 = 20.9;  
console.log(aux1);
```

Prof. Dr. Razer A N R Montaño

Angular

83

83



## Tipo Enum

- Introduzido no Typescript
  - Declara um conjunto de constantes
  - Podem ser: numéricas, strings ou heterogêneas
- Exemplo

```
enum Curso {  
    TADS,  
    TNI,  
    TCI  
}           { '0': 'TADS', '1': 'TNI', '2': 'TCI', TADS: 0, TNI: 1, TCI: 2 }
```

- Cada constante recebe um valor, no exemplo:
  - TADS = 0
  - TNI = 1
  - TCI = 2

Prof. Dr. Razer A N R Montaño

Angular

84

84



## Tipo Enum

- Pode-se inicializar de várias formas:
  - Ao setar um elemento, os demais recebem valores subsequentes
  - Pode-se inicializar todos os elementos
  - Pode-se inicializar com expressões

```
enum Curso {  
    TADS = 5,  
    TNI,  
    TCI  
}  
console.log(Curso);
```

Prof. Dr. Razer A N R Montaño

Angular

85

85



## Tipo Enum

- Acesso

```
enum Curso {  
    TADS = 5,  
    TNI,  
    TCI  
}  
console.log( Curso );  
  
console.log( Curso.TNI ); // retorna 6  
console.log( Curso[7] ); // retorna TCI
```

Prof. Dr. Razer A N R Montaño

Angular

86

86



## Tipo Enum.

- Com Strings:

```
enum Curso {
    TADS = "Análise e Desenvolvimento de Sistemas",
    TNI = "Negócios Imobiliários",
    TCI = "Comunicação Institucional"
}
console.log(Curso);
console.log(Curso.TADS);
```

Prof. Dr. Razer A N R Montaño

Angular

87

87



## Declarações com let, var e const

- **let**: declara uma variável dentro de um escopo, só é visível no escopo em que foi declarada, não pode ser usada antes de declarada

```
function teste(x: number) {
    // a não é visível aqui
    // ERRO - Cannot find name 'a'
    if (x > 10) {
        let a = 20;
        // a é visível aqui!!!!!!!
    }
    // a não é visível aqui
    // ERRO - Cannot find name 'a'
}
```

Prof. Dr. Razer A N R Montaño

Angular

88

88



## Declarações com let, var e const

- **var**: declara uma variável e iça sua declaração para o topo do escopo, pode ser usada antes de declarada

```
function teste(x: number) {
    // a é visível aqui!!!!!!
    // Mas é UNDEFINED
    if (x > 10) {
        var a = 20;
        // a é visível aqui!!!!!!
    }
    // a é visível aqui!!!!!!
    // Se não entrar no IF - Será UNDEFINED
}
```

Prof. Dr. Razer A N R Montaño

Angular

89

89



## Declarações com let, var e const

- **const**: o mesmo que **let**, mas declara uma **constante** dentro de um escopo, só é visível no escopo em que foi declarada
  - Não pode ser usada antes de declarada
  - Deve ser inicializada

```
function teste(x: number) {
    // a não é visível aqui
    if (x > 10) {
        const a = 20;
        // a é visível aqui
    }
    // a não é visível aqui
}
```

Prof. Dr. Razer A N R Montaño

Angular

90

90



## Declarações com let, var e const

- Não pode redefinir a constante

```
const c = 20;  
c = 10;
```

```
teste.ts:108:7 - error TS2588: Cannot assign to 'c' because it is a constant.  
108     c = 10;  
          ~
```



## Declarações com let, var e const

- Declara uma referência constante
  - Não pode redefinir
  - Mas pode mudar o que está sendo referenciado

```
const obj1 = { nome: "Razer", idade: 18 }  
console.log(obj1);  
obj1.nome = "Anthom";  
console.log(obj1);
```

```
{ nome: 'Razer', idade: 18 }  
{ nome: 'Anthom', idade: 18 }
```



## Declarações com let, var e const.

```
const obj1 = { nome: "Razer", idade: 18 }
obj1.nome = "Anthom";
obj1 = { nome: "Nizer", idade: 20 }
```

```
teste.ts:129:1 - error TS2588: Cannot assign to 'obj1' because it is a constant.
129 obj1 = { nome: "Nizer", idade: 20 }
~~~~~
```

2

## Operações



## Operações

- Operações aritméticas: assumindo  $y = 5$

Operador	Descrição	Exemplo	Resultado
+	Adição	<code>x = y + 2;</code>	$x = 7$
-	Subtração	<code>x = y - 2;</code>	$x = 2$
*	Multiplicação	<code>x = y * 2;</code>	$x = 10$
/	Divisão	<code>x = y / 2;</code>	$x = 2.5$
	Divisão inteira	<code>x = Math.trunc(y/2);</code>	$x = 2$
%	Resto da divisão	<code>x = y % 2;</code>	$x = 1$
**	Exponenciação	<code>x = y ** 2;</code>	$x = 25$
++	Incremento	<code>x = ++y; x = y++;</code>	$x = 6$ $x = 5$
--	Decremento	<code>x = --y; x = y--;</code>	$x = 4$ $x = 5$



## Operações

- Operações aritméticas podem ser contraídas com atribuição:
  - `x = x + 10` pode ser escrita como `x += 10`
- Contrações
  - `+=`
  - `-=`
  - `*=`
  - `/=`
  - `%=`



## Operações

- Operações relacionais

Operador	Descrição
==	Igual
===	Idênticos: tipo e valor
!=	Diferente
!==	Não idênticos
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a



## Operações

- Exemplo do operador Idêntico

```
let n1 : string = "3";
let n2 : number = 3;
console.log( n1 === n2 );    // TRUE
console.log( n1 == n2 );     // FALSE
```

- O mesmo para o operador Não Idêntico



## Operações

- Operações lógicas

Operador	Descrição	Exemplo	Tabela Verdade
!	Não	! a	$V \rightarrow F$ $F \rightarrow V$
&&	E	a && b	$V \&\& V \rightarrow V$ $V \&\& F \rightarrow F$ $F \&\& V \rightarrow F$ $F \&\& F \rightarrow F$
	Ou	a    b	$V    V \rightarrow V$ $V    F \rightarrow V$ $F    V \rightarrow V$ $F    F \rightarrow F$



## Operações

- Exemplo

```
let idade : number = 20;
console.log( (idade>18) && (idade<80) );      // TRUE
```



## Operações

- Operador ternário

```
res = (hora < 19) ? "dia" : "noite";
```

- Equivale a:

```
if (hora < 19) {  
    res = "dia";  
}  
else {  
    res = "noite";  
}
```

Prof. Dr. Razer A N R Montaño

Angular

101

101



## Operações.

- Exemplo

```
const agora = new Date();  
let hora : number = agora.getHours();  
  
res = (hora<19) ? "dia" : "noite";  
  
console.log(res);
```

Prof. Dr. Razer A N R Montaño

Angular

102

102

3

## Comandos Condicionais

Prof. Dr. Razer A N R Montaño

Angular

103

103



### Condisional IF

- Comando condicional IF
  - O bloco `else` é opcional

```
if (<expressão booleana>) {  
    <comandos se verdadeiro>  
}  
  
else {  
    <comandos se falso>  
}
```

Prof. Dr. Razer A N R Montaño

Angular

104

104



## Condisional IF

- Pode-se aninhar vários `if...else`
  - Não há comando `elseif`

```
if (<condição 1>) {  
    <comandos>  
}  
else if (<condição 2>) {  
    <comandos>  
}  
else if (<condição 3>) {  
    <comandos>  
}  
else {  
    <comandos>  
}
```

Prof. Dr. Razer A N R Montaño

Angular

105

105



## Condisional IF.

- Exemplo

```
let idade : number = 25;  
  
if (idade >= 18) {  
    console.log("De maior!!!");  
}  
else {  
    console.log("De menor...");  
}
```

Prof. Dr. Razer A N R Montaño

Angular

106

106



## Condisional SWITCH

- Comando condicional `switch...case`
- Avalia a expressão
  - Compara com os valores de cada `case` (usando `==`)
  - Pode ser usado com strings
- Procura o `case` que casa com o valor
  - Executa todos os comandos
  - Até encontrar um `break` ou terminar o comando `switch`
- Se nenhum `case` casa com o valor
  - Procura pelo `default`
- Se não houver `default`
  - Continua a execução após o `switch`

```
switch (<expressão>) {
  case <valor 1>:
    <comandos 1>
    break;
  case <valor 2>:
    <comandos 2>
    break;
  default:
    <comandos>
}
```

Prof. Dr. Razer A N R Montaño

Angular

107

107



## Condisional SWITCH

- Se vários `cases` casam com o valor
  - Somente o primeiro é executado
- O `break` é opcional
  - Ao executar os comandos dentro do `case`, só termina ao encontrar um `break`, que pode estar em um `case` adiante
- Pode-se ter mais de um `case` em sequência
  - Executa os comandos caso qualquer um dos dois seja satisfeito

```
switch (<expressão>) {
  case <valor 1>:
    <comandos 1>
  case <valor 2>:
  case <valor 3>:
    <comandos>
    break;
  default:
    <comandos>
}
```

Prof. Dr. Razer A N R Montaño

Angular

108

108



## Condisional SWITCH

```
let dia : number = new Date().getDay();
switch (dia) {
    case 0:
        console.log("Domingoooo!!!!");
        break;
    case 5:
        console.log("Sextou!!!!");
        break;
    case 6:
        console.log("Sábado de alegria....");
        break;
    default:
        console.log("Partiu trampo....");
}
```

Prof. Dr. Razer A N R Montaño

Angular

109

109



## Condisional SWITCH.

```
let dia : number = new Date().getDay();
switch (dia) {
    case 1:
        console.log("Segunda-feira, tudo de novo... ");
    case 2:
        console.log("Pegando no tranco... ");
        break;
    case 5:
        console.log("SEXTOU!!!!");
        break;
    case 6:
    case 0:
        console.log("Final de semana rolando... ");
        break;
    default:
        console.log("Dia de trabalho... ");
}
```

Prof. Dr. Razer A N R Montaño

Angular

110

110

## Comandos de Repetição

Prof. Dr. Razer A N R Montaño

Angular

111

111



### Laço FOR

- Comando de repetição **for**
  - Antes de entrar no laço, executa a inicialização
  - A cada volta do laço, no início, verifica a condição, se falsa sai do laço
  - A cada volta do laço, no final, executa o passo

```
for (<inicialização>; <condição>; <passo>) {  
    <comandos a serem repetidos>  
}
```

Prof. Dr. Razer A N R Montaño

Angular

112

112



## Laço FOR.

- Exemplo

```
for (let i=1; i<=10; i++) {  
    console.log(`Número: ${i}`);  
}
```

Prof. Dr. Razer A N R Montaño

Angular

113

113



## Laço WHILE

- Comando de repetição **while**
  - Repete enquanto a condição for verdadeira
  - A condição é verificada no início do laço

```
while (<condição>) {  
    <comandos a serem repetidos>  
}
```

Prof. Dr. Razer A N R Montaño

Angular

114

114



## Laço WHILE.

- Exemplo

```
let i : number = 1;
while (i<=10) {
    console.log(`Número: ${i}`);
    i++;
}
```

Prof. Dr. Razer A N R Montaño

Angular

115

115



## Laço DO..WHILE

- Comando de repetição **do .. while**
  - Repete enquanto a condição for verdadeira
  - A condição é verificada ao final, após a primeira execução dos comandos

```
do {
    <comandos a serem repetidos>
} while (<condição>);
```

Prof. Dr. Razer A N R Montaño

Angular

116

116



## Laço DO..WHILE.

- Exemplo

```
let i : number = 1;
do {
    console.log(`Número: ${i}`);
    i++;
} while (i<=10);
```



## Laço FOR..OF

- Comando de repetição **for..of**
  - Repete e obtém os elementos de uma lista, array ou coleção
  - A cada volta do laço a variável obtém um **elemento** da coleção

```
for (<variável> of <coleção>) {
    <comandos a serem repetidos>
}
```



## Laço FOR..OF.

- Exemplo

```
let arr = [ "teste1", "teste2", "teste3" ];  
for (let x of arr) {  
    console.log(`Elemento: ${x}`);  
}
```



## Laço FOR..IN

- Comando de repetição **for..in**
  - Repete e obtém os elementos de uma lista, array ou coleção
  - A cada volta do laço a variável obtém um **índice** de um elemento da coleção

```
for (<variável> in <coleção>) {  
    <comandos a serem repetidos>  
}
```



## Laço FOR..IN.

- Exemplo

```
let arr = [ "teste1", "teste2", "teste3" ];
for (let x in arr) {
    console.log(`Elemento: ${arr[x]}`);
}
```

Prof. Dr. Razer A N R Montaño

Angular

121

121



## Controle de Fluxo.

- Comandos: **break** e **continue**
  - **break**: sai do laço
  - **continue**: vai para a próxima iteração

```
let arr = [ "teste1", "teste2", "teste3" ];
for (let x of arr) {
    if (x === "teste2")
        continue;
    console.log(`Elemento: ${x}`);
}
```

Prof. Dr. Razer A N R Montaño

Angular

122

122

5

## Funções

Prof. Dr. Razer A N R Montaño

Angular

123

123



## Funções

- Blocos de execução de comandos
- Podem ser invocadas
- Podem ser:
  - Funções nomeadas
  - Funções anônimas
  - *Arrow functions, ou funções lambda*

Prof. Dr. Razer A N R Montaño

Angular

124

124



## Funções Nomeadas

- Declaradas com **function**
- Possuem um nome

```
function mostrar() {  
    console.log("Oi Mundo!!!!");  
}  
  
mostrar();
```

Prof. Dr. Razer A N R Montaño

Angular

125

125



## Funções Anônimas

- Não possuem um nome
  - São declaradas com uma expressão
  - São armazenadas em um variável
  - São invocadas pela variável

```
let ola = function () {  
    console.log("Olá Mundo!!!!");  
}  
  
ola();
```

Prof. Dr. Razer A N R Montaño

Angular

126

126



## Parâmetros

- Pode-se passar parâmetros, dentro dos parênteses
  - Define a quantidade exata de parâmetros a serem passados
  - Pode-se definir como opcionais e como valores default
- Pode-se receber um retorno

```
function operacao(x : number, y : number) : number {  
    return x * y;  
}  
  
let res = operacao(10, 30);  
  
console.log(res);
```

Prof. Dr. Razer A N R Montaño

Angular

127

127



## Parâmetros Opcionais

- Parâmetros opcionais são marcados com interrogação "?"
  - Quando não passados : `undefined`

```
function saudacoes(texto: string, nome?: string): string {  
    return texto + ' ' + nome + '.';  
}  
  
console.log( saudacoes("Olá", "Razer") ); // Olá Razer.  
console.log( saudacoes("Olá") ); // Olá undefined.
```

Prof. Dr. Razer A N R Montaño

Angular

128

128



## Parâmetros Com Valores Default

- Parâmetros podem ter valores *default*
  - Não são marcados com ?

```
function saudacoes(texto: string, nome: string = "Fulano"): string {  
    return texto + ' ' + nome + '.';  
}  
  
console.log( saudacoes("Olá", "Razer") ); // Olá Razer.  
console.log( saudacoes("Olá") ); // Olá Fulano.
```

Prof. Dr. Razer A N R Montaño

Angular

129

129



## Parâmetros Variáveis.

- Quantidade variável de parâmetros
  - Usa-se elipse : ...
  - Pode-se passar 0 ou mais argumentos
  - São transformados em um array

```
function saudacoes(texto: string, ...nomes: string[]): string {  
    return texto + ' ' + nomes.join(", ") + '.';  
}  
  
console.log( saudacoes("Olá", "Razer", "Guilherme", "Pedro") );
```

Prof. Dr. Razer A N R Montaño

Angular

130

130



## Arrow Function / Lambda Function

- Funções anônimas
  - Declaradas *inline*
- Expressões que definem uma função
- Sintaxe simplificada
  - Parâmetros entre ( )
  - Seta =>
  - Código entre { }

```
let oi = () => console.log("Oieee!!!");
let tchau = () => {
  console.log("bye bye");
  console.log("volte logo");
}

oi();
tchau();
```

Prof. Dr. Razer A N R Montaño

Angular

131

131



## Arrow Function / Lambda Function.

- Pode-se passar parâmetros e obter resultados

```
let soma = (x: number, y: number) => x * y;

let r = soma(10, 20);
```

Prof. Dr. Razer A N R Montaño

Angular

132

132

## 6

## Orientação a Objetos

Prof. Dr. Razer A N R Montaño

Angular

133

133



## Classes e Objetos

- Introduzidos no ECMAScript 6
- Typescript converte para funções Javascript, por compatibilidade
- Classes são definidas com `class`
- Construtores com `constructor`
- Acessar a instância dentro da classe com `this`
- Métodos são definidos como funções, dentro da classe

Prof. Dr. Razer A N R Montaño

Angular

134

134



## Classes e Objetos

- Definição de classes

```
class Pessoa {  
    nome: string;  
    idade: number;  
  
    constructor(n: string, i: number) {  
        this.nome = n;  
        this.idade = i;  
    }  
  
    getSalario(): number {  
        return 10000.0;  
    }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

135

135



## Classes e Objetos

- Instanciação é feita com `new`
  - Invoca o construtor
- Métodos são acessados com a referência (variável)

```
let r = new Pessoa("Razer", 22);  
  
console.log(r.getSalario());
```

Prof. Dr. Razer A N R Montaño

Angular

136

136



## Classes e Objetos

- Métodos
  - Declarados como funções dentro da classe
- Podem conter modificadores de acesso:
  - **public**: o padrão, método acessível dentro e fora da classe
  - **private**: método só é acessível dentro da classe
  - **protected**: igual ao **private**, mas também é acessível em subclasses

```
class Pessoa {
  ...
  private getSalario() : number {
    return 10000.0;
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

137

137



## Classes e Objetos

- Construtores
  - Similares a métodos em classes
  - Sem retorno, sempre retornam uma instância
  - Podem conter parâmetros
  - Podem ser sobreescritos

```
class Pessoa {
  ...
  constructor() {
    this.nome = "Razer";
    this.idade = 20;
  }
  constructor(n: string, i: number) {
    this.nome = n;
    this.idade = i;
  }
  ...
}
```

Prof. Dr. Razer A N R Montaño

Angular

138

138



## Classes e Objetos

- Atributos
  - Declarados como variáveis dentro da classe
- Podem conter modificadores de acesso:
  - **public**: o padrão, atributo acessível dentro e fora da classe
  - **private**: atributo só é acessível dentro da classe
  - **protected**: igual ao **private**, mas também é acessível em subclasses

```
class Pessoa {
  nome: string;
  public idade: number;
  ...
}
```

Prof. Dr. Razer A N R Montaño

Angular

139

139



## Classes e Objetos

- Atributos somente de leitura
  - Declarados com **readonly**
  - Só podem ser lidos e não escritos
  - Podem ser inicializados no construtor

```
class Calculo {
  readonly valor1 : number;
  valor2: number;
  constructor(valor1 : number, valor2: number) {
    this.valor1 = valor1;
    this.valor2 = valor2;
  }
}

let c = new Calculo(50, 100);
c.valor1 = 10; // ERRO!!! Valor1 é somente de leitura
c.valor2 = 20; // Permitido...
```

Prof. Dr. Razer A N R Montaño

Angular

140

140



## Classes e Objetos

- Definição de **Propriedades**
- Acessadores e Modificadores de Atributos: *setter/getter*
  - Declarados de forma similar a métodos dentro da classe
- Declarados com:
  - **get**: acessador do atributo
  - **set**: modificador do atributo, se inexistente é automaticamente **readonly**

```
class Pessoa {
  private _nome: string;

  get nome() {
    return this._nome;
  }
  set nome(valor) {
    this._nome = valor;
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

141

141



## Classes e Objetos

```
class Pessoa {
  private _nome: string;
  set nome(valor) {
    this._nome = valor;
  }
  get nome() {
    return this._nome;
  }
}
let x = new Pessoa();
x.nome = "Razer";
console.log( x.nome );
```

**Nota:** Se você tiver o erro

"error TS1056: Accessors are only available when targeting ECMAScript 5 and higher."

Compile com:

```
$ tsc --target es6 teste.ts
```

Prof. Dr. Razer A N R Montaño

Angular

142

142



## Classes e Objetos.

- Atributos estáticos
  - Declarados com `static`
  - São acessados via classe e não via instância

```
class Calculo {
  static pi: number = 3.14;
}

console.log( Calculo.pi );
```

Prof. Dr. Razer A N R Montaño

Angular

143

143



## Herança

- Herança é feita com `extends`

```
class Pessoa {
  nome: string;
  constructor(nome: string) {
    this.nome = nome;
  }
}
class Empregado extends Pessoa {
  matricula: number;
  constructor(matricula: number, nome: string) {
    super(nome);
    this.matricula = matricula;
  }
  mostrar(): void {
    console.log("Nome = " + this.nome + ", Matrícula = " + this.matricula);
  }
}
let emp = new Empregado(12, "Razer");
emp.mostrar();
```

Prof. Dr. Razer A N R Montaño

Angular

144

144



## Herança | Classe Abstrata

- Classe abstrata é definida com `abstract`
  - Não pode ser instanciada
  - Mas pode-se ter variáveis deste tipo

```
abstract class Pessoa {
  nome: string;
  constructor(nome: string) {
    this.nome = nome;
  }
}
class Empregado extends Pessoa {
  matricula: number;
  constructor(matricula: number, nome: string) {
    super(nome);
    this.matricula = matricula;
  }
  mostrar(): void {
    console.log("Nome = " + this.nome + ", Matrícula = " + this.matricula);
  }
}
let emp = new Empregado(12, "Razer");
emp.mostrar();
```

Prof. Dr. Razer A N R Montaño

Angular

145

145



## Herança | Classe Abstrata

- Se uma variável for do tipo da classe abstrata
  - Só acessa métodos que estão na classe abstrata

```
let emp : Pessoa = new Empregado(12, "Razer");
emp.mostrar(); // Erro, mostrar() não está em Pessoa
```

```
let emp : Pessoa = new Empregado(12, "Razer");
let emp2 = emp as Empregado; // emp2 é o emp considerado como Empregado
emp2.mostrar();
```

Prof. Dr. Razer A N R Montaño

Angular

146

146



## Herança | Classe Abstrata.

- Classe abstrata pode conter métodos abstratos
  - Não possuem implementação
  - Devem ser implementados na classe filha

```
abstract class Documento {  
    abstract mostrar();  
}  
  
class NotaFiscal extends Documento {  
    mostrar() {  
        console.log("Meu documento");  
    }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

147

147



## Interfaces

- Definem uma estrutura/contrato que deve ser seguida
  - Não é convertida para Javascript, é usada como checagem
  - Pode conter atributos e métodos
  - Devem ser implementados
- Uma interface pode estender outras interfaces (**extends**)
  - A classe deve implementar todos os elementos de toda a hierarquia

```
interface IEmpregado {  
    nome: string;  
    matricula: number;  
    getNovoSalario(number): number;  
}
```

Prof. Dr. Razer A N R Montaño

Angular

148

148



## Interfaces.

- Implementação de Interface

```
class Gerente implements IEmpregado {  
    nome: string;  
    matricula: number;  
    departamento: string;  
    getNovoSalario(valor: number) {  
        return valor * 1.25;  
    }  
}
```

7

## Módulos



## Módulos

- Pode-se escrever partes de código em vários arquivos
  - Cria-se um escopo local
  - O que é definido em um módulo só é visível no módulo
- Exemplo
  - *principal.ts* : contém o código principal
  - *modulo.ts* : o módulo com outros códigos implementados
- Dentro do módulo, **exporta-se** o que será visível para fora
- Dentro dos outros arquivos, **importa-se** o que se deseja usar

Prof. Dr. Razer A N R Montaño

Angular

151

151



## Módulos

- Exemplo: **modulo.ts**

```
export var nome : string = "Razer";
export class Pessoa {
    nome: string = "teste";
}
```

- Exemplo: **principal.ts**
  - Arquivo importado sem extensão

```
import { nome } from './modulo';
console.log( nome );
```

Prof. Dr. Razer A N R Montaño

Angular

152

152



## Módulos

- Pode-se importar e usar um apelido
  - Arquivo importado sem extensão

```
import { nome as nm } from './modulo';

console.log( nm );
```

Prof. Dr. Razer A N R Montaño

Angular

153

153



## Módulos.

- Pode-se importar todo o módulo
  - Usa-se um apelido

```
import * as modulo from './modulo';

let razer : modulo.Pessoa = new modulo.Pessoa();
razer.nome = "Razer";
```

Prof. Dr. Razer A N R Montaño

Angular

154

154



## Módulos em Diretórios

- Em uma aplicação Angular
  - O melhor é criar módulos dentro de diretórios
- Assuma um exemplo simples, como o Hello World
- Estrutura
  - Dentro de `src/app` crie o diretório `modulo1`
  - Dentro de `modulo1` crie o arquivo `modulo1.ts`
  - Adicione o conteúdo

```
export var nome : string = "Razer";
export class Pessoa {
  nome: string = "teste";
}
```

Prof. Dr. Razer A N R Montaño

Angular

155

155



## Módulos em Diretórios

- Para usar o módulo
  - No arquivo `src/app/app.component.ts` adicione o módulo e use a variável exportada

```
import { Component } from '@angular/core';
import { nome } from './modulo1/modulo1';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = nome;
```

Adiciona a importação  
Perceba que o arquivo não  
tem .ts no final

Uso da variável exportada  
no módulo

Prof. Dr. Razer A N R Montaño

Angular

156

156



## Módulos em Diretórios

- Para ver o **title** sendo mostrado altere o arquivo **src/app/app.component.html** para:

```
<h1>{{title}}</h1>
```

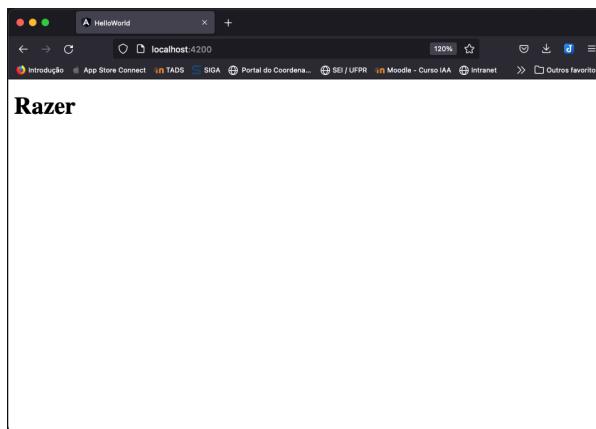
- Então execute no diretório do projeto

```
$ ng serve
```



## Módulos em Diretórios

- Resultado





## Módulos em Diretórios

- Pode-se usar *barrel files*
  - Explicita o que é exportado em uma subárvore de diretório
  - Serve para melhorar o esquema de importações
- Criação
  - Dentro de `src/app/modulo1` crie o arquivo `index.ts`
  - Adicione tudo que será exportado

```
export { nome } from './modulo1';
export { Pessoa } from './modulo1';
```

Prof. Dr. Razer A N R Montaño

Angular

159

159



## Módulos em Diretórios

- Para importar algo do módulo, só se usa o nome do diretório
  - No arquivo `src/app/app.component.ts` adicione o módulo e use a variável exportada

```
import { Component } from '@angular/core';
import { nome } from './modulo1';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = nome;
```

Adiciona a importação contendo  
só o nome do diretório  
Dentro do diretório precisa ter o  
*barrel file index.ts*

Uso da variável exportada no  
módulo

Prof. Dr. Razer A N R Montaño

Angular

160

160



## Módulos em Diretórios..

- Para **importar tudo** do módulo, use-se um *alias*
  - No arquivo `src/app/app.component.ts` adicione o módulo e use um *alias*

```
import { Component } from '@angular/core';
import * as mod1 from './modulo1'; ← Adiciona a importação contendo
                                         só o nome do diretório
                                         Dentro do diretório precisa ter o
                                         barrel file index.ts

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = mod1.nome; ← Uso da variável exportada no
                         módulo
}
```



## EXERCÍCIOS..

1. Criar o projeto ***hello-world-modulo***, da mesma forma que o ***hello-world***
  - Adicionar o módulo ***modulo1***
  - Fazer a importação e usar a variável nome
  - Adicionar e testar o *Barrel File*



# HTML

Prof. Dr. Razer A N R Montaño

Angular

163

163



## HTML

### ➡ Objetivos

- ✓ Apresentar o básico de HTML

### ➡ Referências

- <https://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>
- [https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\\_to\\_HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML)
- [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model/Introduction](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)
- <https://www.javascripttutorial.net/javascript-dom/>
- <https://www.w3schools.com/css/>

Prof. Dr. Razer A N R Montaño

Angular

164

164



## HTML

- HTML:
  - *Hypertext Markup Language*: Linguagem de Marcação de Hipertexto
- Linguagem de publicação de conteúdo
- Baseado em hipertexto: elementos, texto, ligados por conexões (links)
- Desenvolvido por Tim Berners-Lee
- Deslanchou em 1990 com o navegador Mosaic de Marc Andreessen
- Arquivos texto
  - Extensão **.html**
  - Devem ser abertos em um navegador (motor de renderização)

Prof. Dr. Razer A N R Montaño

Angular

165

165



## HTML

- Documento é baseado em *tags*

**<h1>Olá Mundo</h1>**

- O texto "Olá Mundo" sofre a formatação da *tag H1*

Prof. Dr. Razer A N R Montaño

Angular

166

166



## Estrutura de um documento

- Estrutura básica
  - **DOCTYPE**: informa o navegador sobre o tipo do documento
  - **<html>**: indica o início do documento HTML
  - **<head>**: indica o início do cabeçalho do documento
  - **<body>**: indica o início do corpo do documento

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

Prof. Dr. Razer A N R Montaño

Angular

167

167



## Estrutura de um documento.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
    <title>Meu site</title>
  </head>
  <body>
    <h1>Esse é o melhor site do multiverso</h1>
  </body>
</html>
```

Prof. Dr. Razer A N R Montaño

Angular

168

168



## HTML

- Exemplo de *tags*
  - **<h1>** : título de nível 1
  - **<br>** : pula linha
  - **<table>** : define uma tabela
  - **<p>** : define um parágrafo
  - **<div>** : define uma seção no documento
  - **<form>** : define um formulário
  - **<input>** : define uma entrada de dados de um formulário
- Essas *tags* vão dentro do **<body>**

Prof. Dr. Razer A N R Montaño

Angular

169

169



## HTML

- O comportamento das *tags* segue um padrão
  - Formato, tamanho, espaçamentos, etc
- Pode se mudado com CSS e Javascript
- **HTML** define a ESTRUTURA do documento
- **DOM** define a REPRESENTAÇÃO do documento
  - *Document Object Model*
- **CSS** define a FORMATAÇÃO do documento
  - *Cascading Style Sheet*
- **Javascript** define COMPORTAMENTO DINÂMICO

Prof. Dr. Razer A N R Montaño

Angular

170

170



## HTML.

- Algumas *tags* têm início e fechamento: `<h1> ... </h1>`
- Outras não têm fechamento: `<br/>`
- Algumas *tags* possuem atributos
  - Atributos possuem um nome e um valor

```
<input name="idade" value="18" />
```



## Tabelas

- Tag `<table>` define uma tabela:
- Possui *tags* para o conteúdo
  - `<thead>` : cabeçalho da tabela
  - `<tbody>` : corpo da tabela
  - `<tfoot>` : rodapé da tabela
  - `<tr>` : uma linha da tabela
  - `<td>`: um célula em uma linha
  - `<th>`: uma célula de cabeçalho na linha de cabeçalho

```
<table>
  <thead>
    <tr>
      <th>...</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>...</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>...</td>
    </tr>
  </tfoot>
</table>
```



## Tabelas.

```
<table border="1">
  <thead>
    <tr>
      <th>Nome</th>
      <th>Idade</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>João</td>
      <td>22</td>
    </tr>
    <tr>
      <th>Média</th>
      <td>20</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Razer</td>
      <td>18</td>
    </tr>
  </tfoot>
</table>
```

Prof. Dr. Razer A N R Montaño

Angular

173

173



## Links.

- Para links usa-se a tag: `<a href="#" . . .> . . . </a>`
- Exemplo

```
<a href="http://www.razer.net.br"> Site </a>
```

Prof. Dr. Razer A N R Montaño

Angular

174

174



## Listas.

- Lista Itemizada

```
<ul>
  <li> Arroz </li>
  <li> Carne </li>
  <li> Batata </li>
</ul>
```

- Lista numerada

```
<ol>
  <li> Sacar dinheiro </li>
  <li> Ir ao mercado </li>
  <li> Cozinhar </li>
</ol>
```



## Imagen

- Para adicionar uma imagem

```

```

- Atributos

- **src**: imagem a ser adicionada, pode ser um URL
- **alt**: texto alternativo, se a imagem não puder ser vista
- **width**: largura redimensionada da imagem em pixels
- **height**: altura redimensionada da imagem em pixels



## Imagen com Link.

- Imagem clicável

```
<a href="http://www.razer.net.br">  
      
</a>
```

Prof. Dr. Razer A N R Montaño

Angular

177

177



## Acentuação - Entidades

- Entidades
  - á: &aacute;
  - õ: &otilde;
  - ç: &ccedil;
  - è: &egrave;
  - ê: &ecirc;
  - &: &amp;
  - <: &lt;
  - >: &gt;
  - ©: &copy;

Prof. Dr. Razer A N R Montaño

Angular

178

178



## Codificação de Caracteres

- ASCII: *American Standard Code for Information Interchange*
  - Uma tabela : Número → Caractere a ser mostrado
  - 65 → 'A'
  - 66 → 'B'

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	&#64;	Ø	96	60	140	&#96;	`
65	41	101	&#65;	A	97	61	141	&#97;	a
66	42	102	&#66;	B	98	62	142	&#98;	b
67	43	103	&#67;	C	99	63	143	&#99;	c
68	44	104	&#68;	D	100	64	144	&#100;	d
69	45	105	&#69;	E	101	65	145	&#101;	e
70	46	106	&#70;	F	102	66	146	&#102;	f
71	47	107	&#71;	G	103	67	147	&#103;	g
72	48	110	&#72;	H	104	68	150	&#104;	h
73	49	111	&#73;	I	105	69	151	&#105;	i



## Codificação de Caracteres

- Às vezes:
  - Você escreve no HTML: São Paulo
  - No cliente aparece: S?o Paulo ou SÃ£o Paulo ou S?o Paulo
- Todos os textos são números:
  - “ABACAXI” => ASCII => 65 66 65 67 65 88 73
- Dependendo da codificação de caracteres tem-se números diferentes para a mesma String
- Exemplo
  - “LÁPIS” => ISO 8859-1 => 4C C1 50 49 53
  - “LÁPIS” => UTF-8 => 4C C3 81 50 49 53
- **Solução:** Usar sempre a mesma codificação de caracteres



## Codificação de Caracteres

- Duas mais usadas no Brasil:
  - ISO8859-1 (latin 1)
  - UTF-8
- Recomenda-se usar UTF-8, pois representa todos os símbolos e caracteres do mundo

Prof. Dr. Razer A N R Montaño

Angular

181

181



## Codificação de Caracteres

- ISO (International Standards Organization)
  - ISO8859-1 (latin 1): América do norte, Oeste europeu, América latina, Canadá e África
  - ISO8859-2 (latin 2): Leste europeu
  - ISO8859-3 (latin 3): Sudeste europeu, esperanto e etc
  - ISO8859-4 (latin 4): Escandinávia/Balcãs (outros que não estão no latin 1)
  - ISO8859-5: Cirílico – Bulgária, Bielorrússia, Rússia e Macedônia
  - ISO8859-6: Alfabeto Arábico
  - ISO8859-7: Alfabeto Grego
  - ISO8859-8: Alfabeto Hebraico
  - ISO8859-9: Alfabeto Turco, igual ao ISO8859-1, só que os caracteres turcos substituem os da Islândia
  - ISO8859-10: Línguas Nórdicas
  - ISO8859-15 (latin 9 ou latin 0): Substitui alguns elementos do Latin 1
  - ISO-2022-JP: Japonês
  - ISO-2022-JP-2: Japonês
  - ISO-2022-KR: Coreano

Prof. Dr. Razer A N R Montaño

Angular

182

182



## Codificação de Caracteres

- Unicode Consortium desenvolveu o padrão Unicode, cobrindo todos os caracteres, pontuações e símbolos do mundo
  - Unicode Transformation Format (UTF)
  - UTF-8 : caractere pode ter de 1 a 4 bytes, compatível com ASCII
  - UTF-16 : comprimento variável
  - Os 256 primeiros caracteres Unicode correspondem aos 256 primeiros do ISO8859-1
- Sempre que possível preferir o UTF-8

Prof. Dr. Razer A N R Montaño

Angular

183

183



## Codificação de Caracteres.

- Como não ter problemas:
  - **No HTML**
    - Escrever com um editor de texto em formato UTF-8
    - Indicar no HTML (`<head>`) que a codificação é UTF-8

```
<meta charset="utf-8">
<meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
```

- **No Banco de Dados**
  - Criar/Configurar seu banco de dados como UTF-8
- **Em outros arquivos**
  - Sempre usar um editor de texto que suporte UTF-8

Prof. Dr. Razer A N R Montaño

Angular

184

184



## Formulários

- Usados para entrada de dados  
`<form>...</form>`
- Atributos:
  - **name**: nome do formulário, opcional, usado em JS
  - **method**: forma de enviar os dados (**get** ou **post**)
  - **action**: página para onde será redirecionado

```
<form name="frm" method="post" action="resultado.html">  
...  
</form>
```



## Formulários

- Forma de envio dos dados: **get** e **post**
- Para entender, vamos entender um pouco de HTTP
- HTTP é o protocolo de comunicação
  - *HyperText Transfer Protocol*
  - Desde 1990 – Tim Berners-Lee : primeiro cliente/servidor
- Baseado em requisição e resposta
  - Diálogo entre Cliente (navegador) e Servidor
  - Navegador envia um texto para o servidor
  - Servidor interpreta o texto, faz o que tem que fazer e devolve um outro texto, com um documento HTML dentro



## Formulários

- Requisição (Cliente, navegador)

```
GET /sistema/teste.html HTTP/1.1  
Host: www.empressa.com.br
```

- Resposta (Servidor)

```
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 71  
  
<html><head><title>Teste</title></head>  
<body>  
Oi Mundo  
</body>  
</html>
```



## Formulários

- Requisição (Cliente, navegador)

```
GET /sistema/teste.html HTTP/1.1  
Host: www.empressa.com.br
```

- Resposta (Servidor)

```
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 71  
  
<html><head><title>Teste</title></head>  
<body>  
Oi Mundo  
</body>  
</html>
```

Método: GET / POST

Cabeçalho

Corpo



## Formulários

- Dois tipos de passagem de parâmetros
  - *Query parameters*: usado em links, consultas a dados
  - *Body parameters*: usado em formulários

Prof. Dr. Razer A N R Montaño

Angular

189

189



## Formulários

- **Query Parameters** : Parâmetros são passados na própria URL da requisição

```
GET clientes.html?id=10&nome=Maria HTTP/1.1  
Host: www.empresacom.br
```

- No navegador aparece  
<http://www.empresacom.br/clientes.html?id=10&nome=Maria>
  - Espaços são transformados para + ou %20
  - Caracteres especiais são codificados como Ascii / UTF-8 em hexadecimal
  - Limitação na URL

Prof. Dr. Razer A N R Montaño

Angular

190

190



## Formulários

- **Body Parameters** : Parâmetros são passados no corpo da requisição

```
POST clientes.html HTTP/1.1
Host: www.empresa.com.br
Content-Length: 16

id=10&nome=Maria
```

- No navegador aparece  
`http://www.empresa.com.br/clientes.html`
- Pode fazer *upload* de arquivos, arquivos binários e grandes



## Formulários

- Componentes de entrada de dados

```
<input type="tipo" name="nome" />
```

- Tipos:
  - **text, password, submit, reset, checkbox, radio**, etc
- Propriedades do componente
  - **type** : tipo, usado para definir seu formato de entrada de dados
  - **name** : nome, usado para recuperar o valor digitado
  - **value** : valor inicial
  - ...

```
<input type="text" name="endereco" />
```



## Formulário Completo.

```
<!DOCTYPE html>
<html>
<head><title>Formulário</title></head>
<body>
<form name="frm" method="post" action="resultado.html">
    Login: <input type="text" name="login" />      <br/>
    Senha: <input type="password" name="senha" /> <br/>
    <input type="submit" value="Entrar" />
    <input type="reset" value="Limpar" />
</form>
</body>
</html>
```

HTML Básico

193

193



## Componente: text

- Entrada de texto

```
<input type="text" name="nome" value="Razer"
       maxlength="50" size="10" />
```

- Atributos
  - **name**: indica o nome do componente
  - **value**: indica o texto que será o default
  - **size**: tamanho do controle (número caracteres)
  - **maxlength**: máximo de caracteres para digitação

HTML Básico

194

194



## Componente: password

- Entrada de texto sem mostrar o que está sendo digitado

```
<input type="password" name="nome" value=""  
       maxlength="10" size="10" />
```

- Atributos

- **name**: indica o nome do componente
- **value**: indica o texto que será o default
- **size**: tamanho do controle (número caracteres)
- **maxlength**: máximo de caracteres para digitação

HTML Básico

195

195



## Componente: checkbox

- Botão de Check (sim/não)

```
<input type="checkbox" name="fumante"  
       value="s" checked="checked" />
```

- Atributos:

- **name**: indica o nome do componente
- **value**: valor do controle quando está marcado
- **checked**: se inicia marcado ou não

- Se o checkbox estiver marcado, seu valor será o conteúdo no atributo **value**

HTML Básico

196

196



## Componente: radio

- Botão de radio – opções mutuamente exclusivas

```
<input type="radio" name="sexo" value="m"  
       checked="checked" />  
<input type="radio" name="sexo" value="f" />
```

- Atributos:

- **name**: indica o nome do componente. Para que se tenha um grupo de radio, mutuamente exclusivos, usa-se o mesmo nome
- **value**: valor do controle quando está marcado
- **checked**: se inicia marcado ou não

- O valor do componente será o valor (**value**) do item marcado no momento da submissão

HTML Básico

197

197



## Componente: submit

- Botão de Envio de Dados

```
<input type="submit" name="go" value="salvar" />
```

- Atributos:

- **name**: indica o nome do componente
- **value**: texto no caption do controle

- Quando clicado, todos os dados do formulário são “empacotados” e enviados para o recurso que está setado no atributo **action** do formulário (**<form>**)

HTML Básico

198

198



## Componente: reset

- Botão de limpeza de formulário

```
<input type="reset" name="limpar"
       value="limpar" />
```

- Atributos:

- **name**: indica o nome do componente
  - **value**: texto no caption do controle

- Quando pressionado, os componentes do formulário voltam a ter seu valor default, setado no atributo **value**.

HTML Básico

199

199



## Componente: button

- Botão que permite HTML dentro do caption

```
<button type="submit">Enviar</button>
```

- Atributo:

- **name**: indica o nome do componente
  - **value**: texto inicial do caption do controle
  - **type**: tipo do botão (**submit**, **reset**, **button**)
  - **disabled**: se o botão deve estar desabilitado

- Dentro da descrição do botão pode-se colocar elementos HTML, por exemplo, uma imagem:

```
<button type="submit">
  Enviar
</button>
```

HTML Básico

200

200



## Componente: hidden

- Controle de texto escondido, não aparece na tela

```
<input type="hidden" name="dados" value="dado" />
```

- Atributos:

- **name**: indica o nome do componente
- **value**: conteúdo do componente escondido

HTML Básico

201

201



## Componente: select

- Define ComboBox ou ListBox

```
<select name="estado" size="5" multiple>
  <option value="pr" selected="selected">paraná</option>
  <option value="sc">santa catarina</option>
</select>
```

- Atributos do **select**:

- **name**: indica o nome do componente
- **size**: é a quantidade de opções visíveis
- **multiple**: se setado, pode selecionar vários elementos de uma vez

- Atributos do **option**

- **value**: valor do controle quando o item estiver selecionado
- **selected**: se o item inicia selecionado
- **Conteúdo do option**: texto a ser mostrado para aquele item

HTML Básico

202

202



## Componente: select

- Para definir uma ComboBox
  - Manter o atributo **size="1"**
  - Só será mostrado um elemento
  - Será adicionado um botão para mostrar a lista suspensa com as opções

```
<select name="estado" size="1">
  <option value="pr" selected="selected">paraná</option>
  <option value="sc">santa catarina</option>
</select>
```

Paraná ▾

✓ Paraná  
Santa Catarina  
Rio Grande do Sul  
São Paulo  
Minas Gerais

HTML Básico

203

203



## Componente: select.

- Para definir uma ListBox
  - Manter o atributo **size** com um valor maior que 1
  - Serão mostrados vários elementos
  - Atributo **multiple** permite selecionar mais de um valor
  - Será adicionada uma barra de Scroll

```
<select name="estado" size="3">
  <option value="pr" selected="selected">paraná</option>
  <option value="sc">santa catarina</option>
</select>
```

Santa Catarina  
Rio Grande do Sul  
São Paulo  
...

HTML Básico

204

204



## DOM

- HTML DOM: *Document Object Model*
- Modelo de árvore para os elementos de um documento HTML
  - Elementos
  - Atributos
  - Texto
- Define Objetos e Propriedades para TODOS os elementos HTML
- Define Métodos de acesso
- W3C Standard
- Independente de Plataforma e Tecnologia

HTML Básico

205

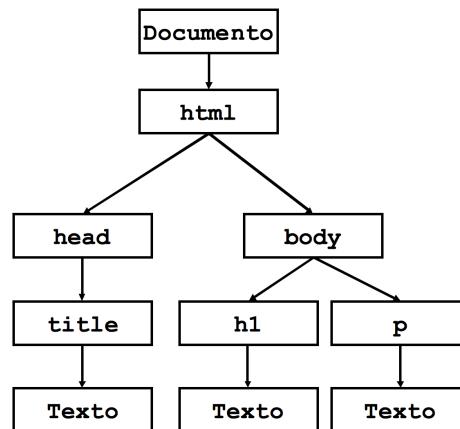
205



## DOM.

- Exemplo:

```
<html>
  <head>
    <title>
      Teste DOM
    </title>
  </head>
  <body>
    <h1>Teste DOM 1</h1>
    <p>Hello world!</p>
  </body>
</html>
```



HTML Básico

206

206



## CSS..

- Cascading Style Sheets
- Define COMO MOSTRAR elementos HTML
- Propriedade **style**
- Pode-se usar Folhas de Estilos Externas
- Folhas externas são armazenadas em arquivos CSS
- Usados para separar o conteúdo do documento (HTML) do seu leiaute (CSS)
- Pode-se definir mais de um estilo para um elemento, ocorrendo cascateamento:
  - Padrão do Navegador
  - Folha de Estilos Externa
  - Folha de Estilos Interna (dentro da tag `<head>`)
  - Estilo dentro do elemento HTML

HTML Básico

207

207



## EXERCÍCIOS..

1. Criar uma página pessoal contendo: sua foto, seu nome, seu e-mail, músicas favoritas. Use cores (CSS) para diferenciar as informações
2. Criar uma página contendo uma tabela de notas de 5 alunos da disciplina de Matemática. A tabela deve conter em cada linha: o nome do aluno, nota 1, nota 2, média. No final da tabela a média entre todos os alunos. Antes da tabela você deve adicionar uma imagem, o nome da disciplina e uma data.

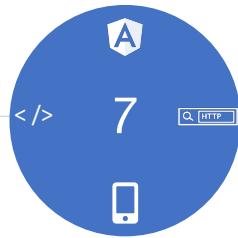
**DICA:** HTML não calcula as médias, então você deve colocar na mão

Prof. Dr. Razer A N R Montaño

Angular

208

208



# Projeto Soma

Prof. Dr. Razer A N R Montaño

Angular

209

209



## Projeto Soma

### ➡ Objetivos

- ✓ Criar Projeto Soma
- ✓ Criar Módulo, Componente e Serviço
- ✓ Alterações no HTML

### 🌐 Referências

- <https://angular.io/tutorial>

Prof. Dr. Razer A N R Montaño

Angular

210

210



## Projeto Soma

- Será composto pelo módulo **SomaModule**, contendo:
  - Componente **SomaComponent**, para apresentar o HTML
  - Serviço **SomaService**, para executar a lógica de negócio

Prof. Dr. Razer A N R Montaño

Angular

211

211



## Projeto Soma

- Os passos para criação do projeto são:
  1. Criar o projeto **Soma**
  2. Criar o módulo **SomaModule**
  3. Criar o componente **SomaComponent**
  4. Mostrar o componente na tela
  5. Criar o serviço **SomaService**
  6. Criar a tela HTML no Componente
  7. Implementar a funcionalidade no componente
  8. Teste da aplicação

Prof. Dr. Razer A N R Montaño

Angular

212

212

1

## Criar Projeto Soma

Prof. Dr. Razer A N R Montaño

Angular

213

213



## Projeto Soma.

- Para criar o projeto, executar

**\$ ng new soma**

```
✖ ~/0/_1/UFPR_Angular/Lab ➤ ng new soma
? Do you want to enforce strict type checking and stricter bundle budgets in the workspace?
? This setting helps catch maintainability and catch bugs ahead of time.
? For more information, see https://angular.io/strict No
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE soma/README.md (1013 bytes)
CREATE soma/editorconfig (274 bytes)
CREATE soma/gitignore (631 bytes)
CREATE soma/angular.json (3519 bytes)
CREATE soma/package.json (1194 bytes)
CREATE soma/tsconfig.json (458 bytes)
CREATE soma/tslint.json (3185 bytes)
CREATE soma/browserslistrc (703 bytes)
CREATE soma/karma.conf.js (1421 bytes)
CREATE soma/tsconfig.lib.json (333 bytes)
CREATE soma/tsconfig.spec.json (333 bytes)
CREATE soma/src/favicon.ico (948 bytes)
CREATE soma/src/index.html (290 bytes)
CREATE soma/src/main.ts (372 bytes)
CREATE soma/src/polyfills.ts (2826 bytes)
CREATE soma/src/styles.css (80 bytes)
CREATE soma/src/test.ts (753 bytes)
CREATE soma/src/assets/.gitkeep (0 bytes)
CREATE soma/src/environments/environment.prod.ts (51 bytes)
CREATE soma/src/environments/environment.ts (662 bytes)
CREATE soma/src/app/app.module.ts (314 bytes)
CREATE soma/src/app/app.component.css (0 bytes)
CREATE soma/src/app/app.component.html (25725 bytes)
CREATE soma/src/app/app.component.spec.ts (934 bytes)
CREATE soma/src/app/app.component.ts (298 bytes)
CREATE soma/e2e/protractor.conf.js (994 bytes)
CREATE soma/e2e/tsconfig.json (274 bytes)
CREATE soma/e2e/src/app.e2e-spec.ts (655 bytes)
CREATE soma/e2e/src/app.po.ts (274 bytes)
✓ Packages installed successfully.
  Successfully initialized git.
```

Prof. Dr. Razer A N R Montaño

Angular

214

214



## Projeto Soma.

- Estrutura criada

```
└─ soma
    ├─ .vscode
    ├─ node_modules
    └─ src
        └─ app
            # app.component.css
            └─ app.component.html
            └─ app.component.spec.ts
            └─ app.component.ts
            └─ app.module.ts
            ├─ assets
            ├─ environments
            └─ favicon.ico
            └─ index.html
            └─ main.ts
            └─ polyfills.ts
            └─ styles.css
            └─ test.ts
            └─ .browserslistrc
            └─ .editorconfig
            └─ .gitignore
            └─ angular.json
            └─ karma.conf.js
            └─ package-lock.json
            └─ package.json
            └─ README.md
            └─ tsconfig.app.json
            └─ tsconfig.json
            └─ tsconfig.spec.json
```

Prof. Dr. Razer A N R Montaño

Angular

215

215

2

## Criar módulo SomaModule

Prof. Dr. Razer A N R Montaño

Angular

216

216



## Módulo SomaModule

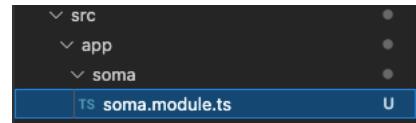
- Para criar o módulo **SomaModule**, executar

```
$ cd soma
$ ng g module soma
```

```
Apple ➜ ~ /G/ / 1/UFPR_A/l/soma ➜ git P master ➤ ng g module soma
CREATE src/app/soma/soma.module.ts (190 bytes)
```

- Resultado

- Cria o diretório **soma**
- Cria o arquivo **soma.module.ts**
  - Dentro a classe **SomaModule**



## Módulo SomaModule

- Arquivo **src/app/soma/soma.module.ts**

```
soma > src > app > soma > ts soma.module.ts > ...
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3
4
5
6   @NgModule({
7     declarations: [],
8     imports: [
9       CommonModule
10    ]
11  })
12  export class SomaModule {}
```



## Módulo SomaModule

- No arquivo ***src/app/app.module.ts***
  - Importar o ***soma.module.ts***

```
soma > src > app > TS app.module.ts > ...
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10   imports: [
11     BrowserModule
12   ],
13   providers: [],
14   bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```

***soma.module.ts***

```
soma > src > app > TS app.module.ts > ...
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { SomaModule } from './soma/soma.module';
6
7 @NgModule({
8   declarations: [
9     AppComponent
10   ],
11   imports: [
12     BrowserModule,
13     SomaModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

***app.module.ts***

Prof. Dr. Razer A N R Montaño

Angular

219

219



## Módulo SomaModule

- Pode-se usar *barrel file*
  - Para evitar a complicaçāo de pastas para importaçāo
- Cria-se um arquivo ***index.ts*** dentro do módulo
- Dentro de ***index.ts***, faz-se todas as exportaçōes

```
export * from './soma.module';
```

```
soma > src > app > soma > TS index.ts
1 export * from './soma.module';
```

Prof. Dr. Razer A N R Montaño

Angular

220

220



## Módulo SomaModule.

- Dentro de *app.module.ts*, simplifica-se a importação

```
import { SomaModule } from './soma';
```

- Automaticamente importa o *index.ts*

```
soma > src > app > TS app.module.ts > ...
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5  import { SomaModule } from './soma';
6
7  @NgModule({
8    declarations: [
9      AppComponent
10   ],
11   imports: [
12     BrowserModule,
13     SomaModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

3

## Criar componente SomaComponent



## Componente Soma

- Criar o componente **SomaComponent** dentro do módulo **soma**
- No diretório do projeto executar

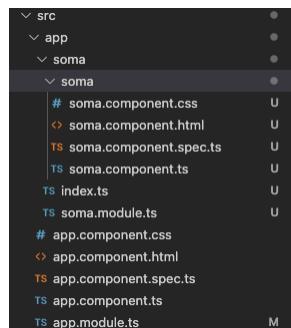
```
$ ng g component soma/soma
```

```
[iMac:~/G/_/__1/UFPR_A/l/soma] git master !1 ?1 ng g component soma/soma
CREATE src/app/soma/soma/soma.component.css (0 bytes)
CREATE src/app/soma/soma/soma.component.html (19 bytes)
CREATE src/app/soma/soma/soma.component.spec.ts (612 bytes)
CREATE src/app/soma/soma/soma.component.ts (267 bytes)
UPDATE src/app/soma/soma.module.ts (258 bytes)
```



## Componente Soma

- Resultado



- Criados os arquivos:
  - **soma.component.css**: Contém o CSS específico do componente
  - **soma.component.html**: Contém o HTML do componente
  - **soma.component.spec.ts**: Contém uma classe para teste unitário
  - **soma.component.ts**: Contém a classe do componente



## Componente Soma

- Usa-se *barrel file* para simplificar a importação
  - No diretório do componente (*src/app/soma/soma*) cria-se *index.ts*

```
soma > src > app > soma > soma > TS index.ts
1  export * from './soma.component';
2  |
```

- Em *soma.module.ts*, altera-se a importação

```
soma > src > app > soma > TS soma.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { SomaComponent } from './soma';
4  |
5  |
6  @NgModule({
7    declarations: [SomaComponent],
8    imports: [
9      CommonModule
10     ]
11   })
12   export class SomaModule {}
```

Prof. Dr. Razer A N R Montaño

225

225



## Componente Soma

- Arquivo *src/app/soma/soma/soma.component.ts* do componente soma:

```
soma > src > app > soma > soma > TS soma.component.ts > SomaComponent
1  import { Component, OnInit } from '@angular/core';
2  |
3  @Component({
4    selector: 'app-soma',
5    templateUrl: './soma.component.html',
6    styleUrls: ['./soma.component.css']
7  })
8  export class SomaComponent implements OnInit {
9    |
10    constructor() { }
11    |
12    ngOnInit(): void {
13    }
14  }
```

Prof. Dr. Razer A N R Montaño

Angular

226

226



## Componente Soma

- O Componente é uma classe
- A anotação **@Component** torna a classe um componente

```
@Component({
  selector: 'app-soma',
  templateUrl: './soma.component.html',
  styleUrls: ['./soma.component.css']
})
export class SomaComponent implements OnInit {
  ...
}
```

Prof. Dr. Razer A N R Montaño

Angular

227

227



## Componente Soma

- O atributo **selector** gera uma tag HTML

```
@Component({
  selector: 'app-soma',
  templateUrl: './soma.component.html',
  styleUrls: ['./soma.component.css']
})
export class SomaComponent implements OnInit {
```

- Pode-se usar

```
<app-soma></app-soma>
```

Para adicionar este componente em uma tela

Prof. Dr. Razer A N R Montaño

Angular

228

228



## Componente Soma

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

- Os atributos são:
  - **selector**: a tag que adiciona o componente na tela
  - **templateUrl**: o HTML a ser renderizado, quando o componente é incluído
  - **styleUrls**: o estilo do componente

Prof. Dr. Razer A N R Montaño

Angular

229

229



## Componente Soma.

- Interface **OnInit**: implementa método de ciclo de vida **ngOnInit()**
  - Construtor é invocado pelo TS para inicializar a classe
    - Mas não se sabe se o Angular terminou a inicialização do componente
  - **ngOnInit()** : contém inicializações, é sempre invocado pelo Angular
  - Executado depois que o Angular inicializou o componente
    - Injeções de dependências resolvidas

```
export class SomaComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

230

230

4

## Mostrar o Componente na Tela

Prof. Dr. Razer A N R Montaño

Angular

231

231



## Mostrar o componente Soma

- Exportar o componente no módulo
  - Editar `src/app/soma/soma.module.ts`
  - Adicionar dentro de `@NgModule`

```
exports: [
  SomaComponent
]
```

```
soma > src > app > soma > soma.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { SomaComponent } from './soma';
4
5
6
7  @NgModule({
8    declarations: [SomaComponent],
9    imports: [
10      CommonModule
11    ],
12    exports: [
13      SomaComponent
14    ]
15  })
16  export class SomaModule {}
```

Prof. Dr. Razer A N R Montaño

Angular

232

232



## Mostrar o componente Soma

- Dentro de *src/app/app.module.ts*
  - Dentro da anotação `@NgModule`, dentro de `imports`, adicionar `SomaModule`

```
soma > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppComponent } from './app.component';
5  import { SomaModule } from './soma';
6
7  @NgModule({
8    declarations: [
9      AppComponent
10    ],
11    imports: [
12      BrowserModule,
13      SomaModule
14    ],
15    providers: [],
16    bootstrap: [AppComponent]
17  })
18  export class AppModule {}
```

Prof. Dr. Razer A N R Montaño

Angular

233

233



## Mostrar o componente Soma

- Dentro de *src/app/app.component.html*
  - Apaga o conteúdo do arquivo
  - Adicionar a *tag* gerada pelo componente

```
<app-soma></app-soma>
```

```
soma > src > app > ◊ app.component.html > ⌂ app-soma
1 |   <app-soma></app-soma>
```

Prof. Dr. Razer A N R Montaño

Angular

234

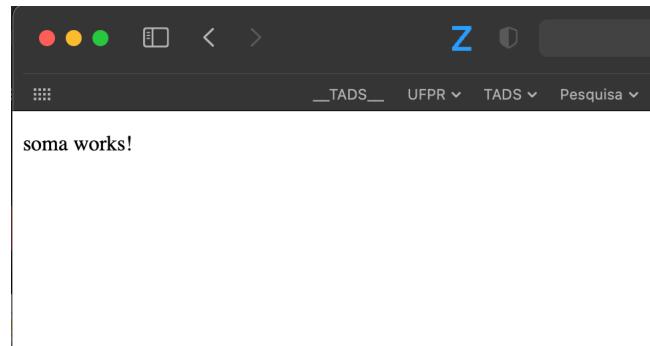
234



## Mostrar o componente Soma.

- No diretório do projeto, executar

```
$ ng serve
```



- Abrir o navegador

Prof. Dr. Razer A N R Montaño

Angular

235

235

5

## Criar o serviço SomaService

Prof. Dr. Razer A N R Montaño

Angular

236

236



## Serviço Soma

- Mantém a lógica de negócio do componente
- Criar o serviço **soma**, dentro do módulo **soma**, no subdiretório **services**
- No diretório do projeto, executar

```
$ ng g service soma/services/soma
```

```
| ⚡ ~/G/_/1/UFPR_A/l/soma > git master !1 ?1 > ng g service soma/services/soma
CREATE src/app/soma/services/soma.service.spec.ts (347 bytes)
CREATE src/app/soma/services/soma.service.ts (133 bytes)
```



## Serviço Soma

- Criado arquivo **src/app/soma/services/soma.service.ts**
  - É uma classe
  - Anotação **@Injectable** torna um serviço injetável em outras classes

```
soma > src > app > soma > services > TS soma.service.ts > ...
1   import { Injectable } from '@angular/core';
2
3   @Injectable({
4     providedIn: 'root'
5   })
6   export class SomaService {
7     |
8     constructor() { }
9   }
10
```



## Serviço Soma

- Dentro da classe **SomaService**
  - Arquivo *src/app/soma/services/soma.service.ts*
  - Implementar o método de negócios: **somar()**
  - Recebe dois números, soma e retorna o resultado

```
somar(numero1: number, numero2: number): number {  
  let resultado: number;  
  
  resultado = numero1 + numero2;  
  return resultado;  
}
```

Prof. Dr. Razer A N R Montaño

Angular

239

239



## Serviço Soma

- Sobre a sintaxe do método
  - **somar**: nome do método, retorna um número (: **number** no final)
  - **numero1**: primeiro parâmetro do tipo número (: **number**)
  - **numero2**: segundo parâmetro do tipo número (: **number**)
  - Comando **let** cria uma variável local, **resultado**, do tipo **number**

```
somar(numero1: number, numero2: number): number {  
  let resultado: number;  
  
  resultado = numero1 + numero2;  
  return resultado;  
}
```

Prof. Dr. Razer A N R Montaño

Angular

240

240



## Serviço Soma

- Resultado é

```
soma > src > app > soma > services > TS soma.service.ts > ...
1   import { Injectable } from '@angular/core';
2
3   @Injectable({
4     providedIn: 'root'
5   })
6   export class SomaService {
7
8     constructor() { }
9
10    somar(numero1: number, numero2: number): number {
11      let resultado: number;
12
13      resultado = numero1 + numero2;
14      return resultado;
15    }
16  }
17
```

Prof. Dr. Razer A N R Montaño

Angular

241

241



## Serviço Soma

- Para facilitar a importação, cria-se o *barrel file*
  - Dentro de **src/app/soma/services**, cria-se o **index.ts**

```
soma > src > app > soma > services > TS index.ts
1   export * from './soma.service';
2   |
```

- No **index.ts** no nível do módulo **SomaModule** (**src/app/soma/index.ts**), adicionam-se as exportações do componente e serviço

```
soma > src > app > soma > TS index.ts
1   export * from './soma.module';
2   export * from './soma';
3   export * from './services';
4
```

Prof. Dr. Razer A N R Montaño

Angular

242

242



## Serviço Soma

- No módulo `src/app/soma/soma.module.ts`
  - Importar o `SomaService`
  - Adicionar em `@NgModule`

```
providers: [
  SomaService
]
```

```
soma > src > app > soma > soma.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { SomaComponent } from './soma';
4  import { SomaService } from './services';

5
6  @NgModule({
7    declarations: [
8      SomaComponent
9    ],
10   imports: [
11     CommonModule
12   ],
13   exports: [
14     SomaComponent
15   ],
16   providers: [
17     SomaService
18   ]
19 })
20 export class SomaModule { }
```

Prof. Dr. Razer A N R Montaño

Angular

243

243



## Serviço Soma

- Para usar o serviço, no `SomaComponent` (`src/app/soma/soma/soma.component.ts`) deve-se:
  - Importar o serviço

```
import { SomaService } from './services';
```

- Adicionar uma injeção de dependência no construtor

```
constructor(private somaService: SomaService) { }
```

- O modificador `private` já cria automaticamente um atributo na classe
- Como está no construtor, uma instância de `SomaService` será automaticamente injetada

Prof. Dr. Razer A N R Montaño

Angular

244

244



## Serviço Soma.

- O resultado é como na imagem abaixo

```
soma > src > app > soma > soma > soma.component.ts > ...
1   import { Component, OnInit } from '@angular/core';
2
3   import { SomaService } from '../services';
4
5   @Component({
6     selector: 'app-soma',
7     templateUrl: './soma.component.html',
8     styleUrls: ['./soma.component.css']
9   })
10  export class SomaComponent implements OnInit {
11
12    constructor(private somaService: SomaService) { }
13
14    ngOnInit(): void {
15    }
16
17  }
18
```

6

## Criar a tela HTML no Componente



## Tela de Soma: HTML

- Alterações no HTML do **SomaCompoment** (*src/app/soma/soma/soma.component.html*)
- Serão adicionados:
  - Formulário
  - Dois campos de entrada de dados
  - Um botão para efetuar a operação
  - Um componente para mostrar o resultado

Prof. Dr. Razer A N R Montaño

Angular

247

247



## Tela de Soma: HTML

```
<form>
  Número 1: <input type="text" /><br/>
  Número 2: <input type="text" /><br/>
  <input type="button" value="Somar" /><br/><br/>
  Resultado: <input type="text" />
</form>
```

Prof. Dr. Razer A N R Montaño

Angular

248

248



## Tela de Soma: HTML.

- Resultado

```
soma > src > app > soma > soma > soma.component.html > ...
Go to component
<form>
1   Número 1: <input type="text" /><br/>
2   Número 2: <input type="text" /><br/>
3   <input type="button" value="Somar" /><br/><br/>
4   Resultado: <input type="text" />
5   </form>
6
7
8
```

7

## Implementar a funcionalidade no Componente



## Implementar a Funcionalidade na App

- Dentro de **SomaComponent**
  - Arquivo *src/app/soma/soma/soma.component.ts*
- Deve-se:
  - Implementar a funcionalidade na classe
    - Atributo resultado
    - Método para somar que invoca o **SomaService**
  - Alterar o formulário HTML para passar e receber valores da classe
    - Alterar os campos de entrada
    - Alterar o campo de resultado
    - Alterar o botão para invocar o método somar da classe

Prof. Dr. Razer A N R Montaño

Angular

251

251



## Implementar a Funcionalidade na App

- Dentro de **SomaComponent**
  - Arquivo *src/app/soma/soma/soma.component.ts*
  - Adicionar atributo (**private**) para conter o resultado
    - **res**: número, atributo contendo o resultado da soma

```
private res : number = 0;
```

```
export class SomaComponent implements OnInit {  
  private res : number = 0;  
  
  constructor(private somaService: SomaService) { }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

252

252



## Implementar a Funcionalidade

- Dentro de **SomaComponent**

- Arquivo *src/app/soma/soma/soma.component.ts*
- Adicionar método **somar**
  - Recebe dois números como **string**
  - Soma e retorna o resultado

```
somar(numero1: string, numero2: string): void {
  let n1: number;
  let n2: number;

  n1 = parseFloat(numero1);
  n2 = parseFloat(numero2);
  this.res = this.somaService.somar(n1, n2);
}
```

Prof. Dr. Razer A N R Montaño

Angular

253

253



## Implementar a Funcionalidade

- Resultado

```
soma > src > app > soma > soma > TS soma.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { SomaService } from '../services';
3
4 @Component({
5   selector: 'app-soma',
6   templateUrl: './soma.component.html',
7   styleUrls: ['./soma.component.css']
8 })
9 export class SomaComponent implements OnInit {
10   private res: number = 0;
11
12   constructor(private somaService: SomaService) { }
13
14   ngOnInit(): void {
15   }
16
17   somar(numero1: string, numero2: string): void {
18     let n1: number;
19     let n2: number;
20     n1 = parseFloat(numero1);
21     n2 = parseFloat(numero2);
22     this.res = this.somaService.somar(n1, n2);
23   }
24 }
```

Prof. Dr. Razer A N R Montaño

Angular

254

254



## Implementar a Funcionalidade

- Dentro de **SomaComponent**

- Arquivo *src/app/soma/soma/soma.component.ts*
- Adicionar o *GETTER* : **get resultado()**
  - Cria uma propriedade que é retornada por um método *getter*
  - Para retornar o resultado como uma **string**, pronta para ser mostrada

```
get resultado(): string {
    return this.res.toString();
}
```



## Implementar a Funcionalidade

- Resultado

```
soma > src > app > soma > soma > ts soma.component.ts > Soma
1 import { Component, OnInit } from '@angular/core';
2 import { SomaService } from '../services';
3
4 @Component({
5   selector: 'app-soma',
6   templateUrl: './soma.component.html',
7   styleUrls: ['./soma.component.css']
8 })
9 export class SomaComponent implements OnInit {
10   private res: number = 0;
11
12   constructor(private somaService: SomaService) { }
13
14   ngOnInit(): void {
15   }
16
17   somar(numero1: string, numero2: string): void {
18     let n1: number;
19     let n2: number;
20     n1 = parseFloat(numero1);
21     n2 = parseFloat(numero2);
22     this.res = this.somaService.somar(n1, n2);
23   }
24
25   get resultado(): string {
26     return this.res.toString();
27   }
28 }
29 }
```



## Implementar a Funcionalidade

- No HTML do componente
  - Arquivo `src/app/soma/soma/soma.component.html`
  - Efetuar o *binding*: ligar um componente de tela com um elemento do código
  - Alterações no código já colocado
  - Nas entradas de dados
    - Adicionar os *template reference variables* `#numero1` e `#numero2`, usados para acessar os valores digitados

Número 1: `<input #numero1 type="text" /><br/>`

Número 2: `<input #numero2 type="text" /><br/>`



## Implementar a Funcionalidade

- No HTML
  - Arquivo `src/app/soma/soma/soma.component.html`
  - No elemento que mostra o resultado
    - Adicionar `[value]="resultado"`, indicando ao Angular que deve preencher o `value` com o valor do atributo `resultado` do componente
    - É invocado sempre que uma alteração no Angular acontece
    - Adicionar o `readonly`

Resultado: `<input type="text" [value]="resultado"  
readonly />`



## Implementar a Funcionalidade

- No HTML

- Arquivo *src/app/soma/soma/soma.component.html*
- No botão
  - Adicionar `(click)="somar(numero1.value, numero2.value)"`
  - O tipo do botão (`type`) DEVE ser `button`
  - Indica ao Angular que ao clicar no botão o método `somar()` é invocado, passando os valores dos dois campos de entradas de dados (`#numero1` e `#numero2`)

```
<input type="button" value="Somar"
       (click)="somar(numero1.value, numero2.value)" /><br/><br/>
```



## Implementar a Funcionalidade

- HTML Completo

- Arquivo *src/app/soma/soma/soma.component.html*

```
<form>
    Número 1: <input #numero1 type="text" /><br/>
    Número 2: <input #numero2 type="text" /><br/>
    <input type="button" value="Somar"
          (click)="somar(numero1.value, numero2.value)" /><br/><br/>
    Resultado: <input type="text" [value]="resultado" readonly />
</form>
```



## Implementar a Funcionalidade.

- Resultado

```
soma > src > app > soma > soma > soma.component.html > ...
Go to component
1  <form>
2    Número 1: <input #numero1 type="text" /><br/>
3    Número 2: <input #numero2 type="text" /><br/>
4    <input type="button" value="Somar"
5    |   (click)="somar(numero1.value, numero2.value)"/><br/><br/>
6    Resultado: <input type="text" [value]="resultado" readonly />
7  </form>
8
9 |
```

8

## Teste da Aplicação



## Teste da Aplicação..

- Rodar **ng serve** no diretório do projeto
- Abrir o navegador : **http://localhost:4200**

Número 1:   
Número 2:   
**Somar**  
Resultado: 0

Número 1: 10  
Número 2: 20  
**Somar**  
Resultado: 30

Prof. Dr. Razer A N R Montaño

Angular

263

263



## EXERCÍCIOS..

1. Criar o projeto Soma mostrado
2. Criar o projeto Subtracao
  - a) Igual ao projeto Soma, mas efetua a subtração de dois números
  - b) Siga os mesmos passos do projeto Soma
  - c) NÃO copie o projeto anterior, este é um exercício de fixação dos passos e conceitos
3. Criar o projeto Multiplicacao
  - a) Idem
4. Criar o projeto Divisao
  - a) Idem

Prof. Dr. Razer A N R Montaño

Angular

264

264



# Elementos de Templates HTML

Prof. Dr. Razer A N R Montaño

Angular

265

265



## Elementos de Templates HTML

### ☰ Objetivos

- ✓ Apresentar a Sintaxe de *binding*, Unidirecional e Bidirecional
- ✓ Apresentar *Template Reference Variables*
- ✓ Apresentar Diretivas Estruturais *\*ngIf*, *\*ngFor* e *\*ngSwitch*



### Referências

- <https://angular.io/tutorial>
- <https://angular.io/guide/binding-syntax>

Prof. Dr. Razer A N R Montaño

Angular

266

266

1

## Sintaxe de Binding

Prof. Dr. Razer A N R Montaño

Angular

267

267



## Sintaxe de Binding

- O *binding* ou ligação mantém a tela atualizada, conforme o estado da aplicação
- Manipula propriedades DOM e eventos
  - Atributos HTML são usados para inicializar os elementos
  - Propriedades DOM mantém o estado atual do elemento
- Exemplo:

```
<input type="text" value="Razer">
```

- Se o usuário altera o conteúdo para "Juan"
  - Atributo HTML *value* continua "Razer"
  - Mas a propriedade DOM *value* estará como "Juan"

Prof. Dr. Razer A N R Montaño

Angular

268

268



## Sintaxe de Binding

- Outro exemplo: propriedade DOM *disabled*

```
<button disabled>Salvar</button>
```

- Por default, é falso

- Se o atributo HTML *disabled* for colocado, inicia como desabilitado
- O valor do atributo não importa, a presença dele é que importa:

```
<button disabled="false">Salvar</button>
```

não vai habilitar o botão



## Sintaxe de Binding

- Exemplo de manipulação de uma propriedade DOM

```
<input [disabled]="condicao">
```

- É possível controlar se o componente está ou não desabilitado
- Uma propriedade **condicao** deve estar presente na classe do componente (arquivo .ts)



## Sintaxe de Binding

- Assim, em Angular pode-se manipular OU a propriedade DOM OU o atributo HTML:

- Propriedade DOM

```
<input [disabled]="condition ? true : false">
```

- Atributo HTML

```
<input [attr.disabled]="condition ? 'disabled' : null">
```

- Prefere-se manipular a propriedade DOM

- Melhor desempenho

- Melhor legibilidade

- Atualização automática, sem *refresh*



## Sintaxe de Binding.

- Tem-se 3 categorias de *data binding* conforme a direção da ligação

- Unidirecional:** da fonte de dados para a tela

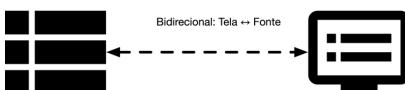
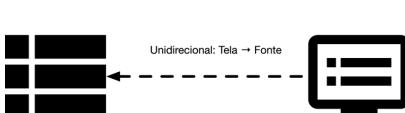
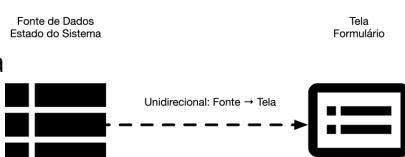
- Interpolação

- Binding* de propriedade

- Unidirecional:** da tela para a fonte de dados

- Binding* de evento

- Bidirecional:** entre a tela e a fonte de dados





## Interpolação : Fonte → Tela

- Interpolação
  - Apresenta um dado do componente no HTML
  - Usa a sintaxe `{}{}`
- Exemplo no *template* HTML  
`Meu nome é: <strong>{{nome}}</strong>`

Prof. Dr. Razer A N R Montaño

Angular

273

273



## Interpolação : Fonte → Tela

- Para isso o componente precisa ter um atributo público "nome"

```
@Component({  
    ...  
})  
export class PaginaComponent {  
    public nome: string = "";  
  
    ...  
}
```

Prof. Dr. Razer A N R Montaño

Angular

274

274



## Interpolação : Fonte → Tela

- Dentro da interpolação {{ }} pode-se:
  - Usar um atributo do componente. Ex: {{ nome }}
  - Efetuar uma operação: Ex: {{10 + 20}}
  - Invocar um método do componente: Ex: {{ getValor() }}
  - Etc

Prof. Dr. Razer A N R Montaño

Angular

275

275



## Interpolação : Fonte → Tela.

- Expressões possuem um contexto
  - Geralmente o componente onde está
  - Pode referenciar variáveis criadas no *template*. Ex:

```
<ul>
  <li *ngFor="let cliente of clientes">{{cliente.nome}}</li>
</ul>
```

- Pode referenciar variáveis de referência do *template*

```
<label>Digite algo:
  <input #entradaCliente>{{entradaCliente.value}}
</label>
```

Prof. Dr. Razer A N R Montaño

Angular

276

276



## Binding Propriedade : Fonte → Tela

- *Binding* de Propriedade do DOM

- Seta propriedades DOM do elemento HTML
  - Não confundir com atributos do elemento HTML
- Qualquer mudança nos objetos reflete nas propriedades
- Define: qual propriedade do DOM e qual expressão é avaliada
  - As expressões são avaliadas a cada evento do navegador
- Usa o operador []

```
[propriedade]="expressão"
```

- Sem o [] o angular entende o lado direito (entre aspas) como uma string estática

```
propriedade="string estática"
```



## Binding Propriedade : Fonte → Tela

- Exemplo no *template* HTML

```
<span [title]="titulo">Meu texto lindo</span>
<span [innerHTML]='html'></span>
<span [style.color]='color'>Texto em vermelho</span>
```



## Binding Propriedade : Fonte → Tela

- Para isso o componente precisa ter um atributo público `titulo`, `html` e `color`

```
@Component({
  ...
})
export class PaginaComponent {
  public titulo: string = "";
  public html: string = "";
  public color: string = "";

  constructor() {
    this.titulo = "Oi";
    this.html = "Meu <strong>texto</strong>";
    this.color = "red";
  }
  ...
}
```

Prof. Dr. Razer A N R Montaño

Angular

279

279



## Binding Propriedade : Fonte → Tela.

- Pode-se usar expressões

```
<span [style.color]="estaAtrasado ? 'red' : 'blue'">Texto da Tarefa</span>
```

Prof. Dr. Razer A N R Montaño

Angular

280

280



## Binding de Evento : Tela → Fonte

- Unidirecional: da tela para a fonte de dados
- *Binding* de Evento
  - Liga um evento na tela a um método do componente
  - Quando o evento ocorre, o método do componente executa
  - Sintaxe:

```
(evento)="expressão"
```

Prof. Dr. Razer A N R Montaño

Angular

281

281



## Binding de Evento : Tela → Fonte

- Exemplo

```
<button class='btn' (click)='salvar()'>Confirma</button>
```

- Precisa-se do método **salvar()** no componente

```
@Component({
  ...
})
export class PaginaComponent {
  ...

  salvar() {
    // faz alguma coisa
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

282

282



## Binding de Evento : Tela → Fonte.

- Se for necessário acesso a dados do evento, pode-se passá-lo como parâmetro
  - Parâmetro **\$event**
  - Exemplo: em um evento de mouse pode-se verificar sua posição

```
<button class='btn' (click)='salvar($event)'>Confirma</button>
```

- No método **salvar()** no componente

```
@Component({
  ...
})
export class PaginaComponent {
  ...

  salvar($event: Event) {
    // faz alguma coisa
  }
}
```



## Binding Bidirecional : Tela ↔ Fonte

- Bidirecional: entre a tela e a fonte de dados
  - Combinação dos dois *bindings* unidirecionais
  - Usado para sincronizar um dado na classe do componente com o dado que está na tela
  - Sintaxe

`[ (alvo) ]="expressão"`

- Usado também para comunicar dados entre componentes Pai e Filho
  - **@Input** : atributo que recebe dados
  - **@Output** : atributo que emite dados para o pai



## Binding Bidirecional : Tela ↔ Fonte

- Para formulários

- Diretiva `ngModel`
  - No `template` HTML

```
Nome: <input type="text" [(ngModel)]="nome" />
<div>{{ nome }}</div>
```

- O campo **nome** está ligado bidirecionalmente

- Quando o input é mostrado, obtém o dado do componente para mostrar
  - Quando o input é modificado na tela, atualiza seu valor, inclusive na interpolação ( `{{ nome }}` )



## Binding Bidirecional : Tela ↔ Fonte.

- Para usar o `ngModel`

- Deve-se importar **FormsModule** na aplicação (`app.module.ts` ou nos módulos)
  - O componente deve ter uma propriedade **nome**

```
@Component({
  ...
})
export class PaginaComponent {
  public nome: string = "";

  ...
}
```

2

## *Template Reference Variable*

Prof. Dr. Razer A N R Montaño

Angular

287

287



## *Template Reference Variables*

- Usadas para **referenciar** elementos do template HTML
  - Elemento DOM
  - Diretiva (Ex, ngForm)
  - Um elemento
- Sintaxe
  - Usa-se #
  - Exemplo

```
<input #telefone placeholder="número do telefone" />
```

- No mesmo HTML pode-se usar o valor do telefone

```
<button (click)="salvar(telefone.value)">Salvar</button>
```

Prof. Dr. Razer A N R Montaño

Angular

288

288



## Template Reference Variables

- Um caso comum de uso é com a diretiva `ngForm`
  - Pode-se obter o valor e a validade do formulário

```
<form #meuForm="ngForm" (ngSubmit)="onSubmit(meuForm)">
  <label for="nome">Nome:
  <input class="form-control" name="nome" ngModel required />
  </label>
  <button type="submit">Salvar</button>
</form>

<div [hidden]="!meuForm.form.valid">
  <p>{{ errorMessage }}</p>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

289

289



## Template Reference Variables.

- Consegue-se até mesmo desabilitar o botão de submit, caso o formulário não seja válido

```
<form #meuForm="ngForm" (ngSubmit)="onSubmit(meuForm)">
  <label for="nome">Nome:
  <input class="form-control" name="nome" ngModel required />
  </label>
  <button type="submit" [disabled]="meuForm.form.invalid" >
    Salvar </button>
</form>

<div [hidden]="!meuForm.form.valid">
  <p>{{ errorMessage }}</p>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

290

290

3

## Diretivas Estruturais

Prof. Dr. Razer A N R Montaño

Angular

291

291



### Diretivas Estruturais.

- Três diretivas disponíveis
  - **\*ngIf**
  - **\*ngFor**
  - **\*ngSwitch**
- Pode-se construir suas próprias diretivas

Prof. Dr. Razer A N R Montaño

Angular

292

292



## \*ngIf

- Usado para mostrar ou esconder partes do *template* HTML
  - Adiciona ou remove um elemento do DOM
  - É diferente de "esconder" um elemento, pois ao esconder (*hidden*) o elemento está lá, só invisível
- Exemplo:

```
<div *ngIf="mostrar"><p>Texto a ser mostrado</p></div>
```

- Se a variável "mostrar" for verdadeira, o **<div>** estará presente no DOM
- Caso contrário, o **<div>** não estará presente



## \*ngIf

- O componente precisa de um atributo "mostrar"
- Exemplo:

```
@Component({  
    ...  
})  
export class PaginaComponent {  
    public mostrar: boolean = true;  
  
    ...  
}
```



## \*ngIf.

- Pode-se usar operadores lógicos: !, && e ||
- E relacionais: >, <, >=, <=, ==, !=
- Exemplo:

```
<div *ngIf="mostrar && nota > 70">
    <p>Texto a ser mostrado</p>
</div>
```



## \*ngFor

- Usado para repetir partes do *template* HTML
- Exemplo:

```
<div *ngFor="let item of listaItens">{{item.nome}}</div>
```

- Cria uma variável de loop chamada "**item**"
- Cada elemento de "**listaItens**" será atribuído a "**item**"
- Dentro do **<div>** mostra "**item.nome**"



## \*ngFor.

- Pode-se usar com um índice
- Exemplo:

```
<div *ngFor="let item of listaItens; let i=index">
    {{i+1}} - {{item.nome}}
</div>
```

- A variável **i** recebe o índice do elemento



## \*ngSwitch

- Como um comando **switch**, apresenta um elemento conforme várias opções
- É um atributo de diretiva, portanto escreve-se com **[ngSwitch]**
- Possui duas diretivas: **\*ngSwitchCase** e **\*ngSwitchDefault**
- Exemplo:

```
<ul [ngSwitch]="superhero">
    <li *ngSwitchCase="'Groot'">Groot</li>
    <li *ngSwitchCase="'Ironman'">Ironman</li>
    <li *ngSwitchDefault>Batman</li>
</ul>
```

- Dependendo do valor do atributo "superhero" do componente, um texto será mostrado



## \*ngSwitch..

- Aqui um exemplo combinando `*ngFor` e `*ngSwitch`

```
<div *ngFor="let carro of listaCarros" [ngSwitch]="carro.cor">
  <div *ngSwitchCase="'blue'" class="blue">
    {{ carro.nome }} ({{ carro.cor }})
  </div>
  <div *ngSwitchCase="'yellow'" class="yellow">
    {{ carro.nome }} ({{ carro.cor }})
  </div>
  <div *ngSwitchCase="'silver'" class="silver">
    {{ carro.nome }} ({{ carro.cor }})
  </div>
  <div *ngSwitchDefault class="text-warning">
    {{ carro.nome }} ({{ carro.cor }})
  </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

299

299



# Bootstrap

Prof. Dr. Razer A N R Montaño

Angular

300

300



## Bootstrap

### Objetivos

- ✓ Instalar o Bootstrap no Projeto Soma
- ✓ Adicionar Bootstrap no formulário



### Referências

➤ <https://getbootstrap.com>

Prof. Dr. Razer A N R Montaño

Angular

301

301



## Bootstrap

- Instalar o Bootstrap
- Executar dentro do diretório do projeto

**\$ npm install --save bootstrap**

- **--save** vai atualizar o *package.json* incluindo suas dependências

Prof. Dr. Razer A N R Montaño

Angular

302

302



## Bootstrap

- Habilitar o Bootstrap na aplicação
- Atualizar o arquivo **angular.json**, dentro de:

```
{ ...  
  "projects" : {  
    ...  
    "architect": {  
      "build": {  
        ...  
        "options": {  
          ...  
          <AQUI>  
          ...  
        }  
      }  
    }  
  }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

303

303



## Bootstrap

- Adicionar:

```
"styles": [  
  "src/styles.css",  
  "node_modules/bootstrap/dist/css/bootstrap.min.css"  
],
```

Prof. Dr. Razer A N R Montaño

Angular

304

304



## Bootstrap

```
22     "assets": [
23         "src/favicon.ico",
24         "src/assets"
25     ],
26     "styles": [
27         "src/styles.css"
28     ],
29     "scripts": []
30 ]
```



```
22     "assets": [
23         "src/favicon.ico",
24         "src/assets"
25     ],
26     "styles": [
27         "src/styles.css",
28         "node_modules/bootstrap/dist/css/bootstrap.min.css"
29     ],
30     "scripts": []
```

Prof. Dr. Razer A N R Montaño

Angular

305

305



## Bootstrap

- Adicionar um título (barra de navegação) à tela
- Adicionar o seguinte trecho HTML no topo do arquivo `src/app/soma/soma/soma.component.html`

```
<nav class="navbar navbar-default">
  <div class="conteiner-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        Projeto Soma
      </a>
    </div>
  </div>
</nav>
```

Prof. Dr. Razer A N R Montaño

Angular

306

306



## Bootstrap

- Arrumar a formatação do formulário no mesmo arquivo
  - Logo abaixo da barra de navegação
- Colocar o formulário dentro de um container, com 6 colunas

```
<div class="container">
  <div class="col-md-6 col-md-offset-6">
    <form>
      ...
    </form>
  </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

307

307



## Bootstrap

- Dentro de `<form>...</form>`, adicionar o formulário
- O primeiro número fica:

```
<div class="form-group row">
  <label for="numero1" class="col-sm-2 col-form-label">
    Número 1:</label>
  <div class="col-sm-4">
    <input id="numero1" class="form-control"
      placeholder="Número 1" #numero1 type="text" />
    <small class="form-text text-muted">
      Digite o 1º. número da soma</small>
  </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

308

308



## Bootstrap

- O segundo número fica:

```
<div class="form-group row">
    <label for="numero2" class="col-sm-2 col-form-label">
        Número 2:</label>
    <div class="col-sm-4">
        <input id="numero2" class="form-control"
            placeholder="Número 2" #numero2 type="text" />
        <small class="form-text text-muted">
            Digite o 2º. número da soma</small>
    </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

309

309



## Bootstrap

- O Botão e Resultado ficam

```
<button type="button" class="btn btn-primary"
    (click)="somar(numero1.value, numero2.value)">
    Somar</button>

<div class="form-group row">
    <label for="resultado" class="col-sm-2 col-form-label">
        Resultado:</label>
    <div class="col-sm-4">
        <input id="resultado" class="form-control"
            type="text" [value]="resultado" readonly />
    </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

310

310



## Bootstrap

```
<div class="container">
  <div class="col-md-6 col-md-offset-6">
    <form>
      <div class="form-group row">
        <label for="numero1" class="col-sm-2 col-form-label">Número 1:</label>
        <div class="col-sm-4">
          <input id="numero1" class="form-control" placeholder="Número 1" #numero1 type="text" />
          <small class="form-text text-muted">Digite o 1o. número da soma</small>
        </div>
      </div>
      <div class="form-group row">
        <label for="numero2" class="col-sm-2 col-form-label">Número 2:</label>
        <div class="col-sm-4">
          <input id="numero2" class="form-control" placeholder="Número 2" #numero2 type="text" />
          <small class="form-text text-muted">Digite o 2o. número da soma</small>
        </div>
      </div>
      <button type="button" class="btn btn-primary" (click)="somar(numero1.value, numero2.value)">Somar</button>
      <div class="form-group row">
        <label for="resultado" class="col-sm-2 col-form-label">Resultado:</label>
        <div class="col-sm-4">
          <input id="resultado" class="form-control" type="text" [value]="resultado" readonly />
        </div>
      </div>
    </form>
  </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

311

311



## Bootstrap..

- Resultado na Tela

### Projeto Soma

Número 1:  Número 2:  <b>Somar</b>	<input type="text" value="Número 1"/> <small>Digite o 1o. número da soma</small> <input type="text" value="Número 2"/> <small>Digite o 2o. número da soma</small>  <b>Resultado:</b> <input type="text" value="0"/>
--	---

Prof. Dr. Razer A N R Montaño

Angular

312

312



## EXERCÍCIOS..

1. Adicionar o Bootstrap no projeto Soma e personalizar o formulário
2. Personalizar com Bootstrap os projetos Subracao, Multiplicacao e Divisao
3. Criar uma nova aplicação Calculadora, contendo o mesmo formulário mas com 4 botões para efetuar as 4 operações

Projeto Calculadora

Número 1:   
Digite o 1o. número da soma

Número 2:   
Digite o 2o. número da soma

Resultado:

Prof. Dr. Razer A N R Montaño

Angular

313

313



## Material Design

Prof. Dr. Razer A N R Montaño

Angular

314

314



## Material Design

### Objetivos

- ✓ Adicionar Material Design na aplicação
- ✓ Alterar a aplicação para usar os componentes



### Referências

- <https://angular.io>
- <https://material.angular.io>



## Material Design

- Aqui faremos um clone do projeto Soma
  - Instalar Material no projeto
  - Adicionar as importações corretas
  - Alterar o formulário



## Material Design

- Para instalar em um projeto

```
$ ng add @angular/material
```

- Escolha as opções default

```
MacBook-Pro:~/ME/_...3/Doce/0/Ang/lab13/soma-material % ng add @angular/material
i Using package manager: npm
✓ Found compatible package version: @angular/material@13.3.2.
✓ Package information loaded.

The package @angular/material@13.3.2 will be installed and executed.
Would you like to proceed? Yes
✓ Package successfully installed.
? Choose a prebuilt theme name, or "custom" for a custom theme: Indigo/Pink      [ Preview: https://material.angular.io?theme=indigo-pink ]
? Set up global Angular Material typography styles? No
? Set up browser animations for Angular Material? Yes
UPDATE package.json (1169 bytes)
✓ Packages installed successfully.
UPDATE src/app/app.module.ts (476 bytes)
UPDATE angular.json (3305 bytes)
UPDATE src/index.html (549 bytes)
UPDATE src/styles.css (181 bytes)
```

Prof. Dr. Razer A N R Montaño

Angular

317

317



## Material Design

- Importar os módulos do Material no módulo que irá usar
  - Arquivo `src/app/soma/soma.module.ts`

```
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatIconModule } from '@angular/material/icon';
import { MatButtonModule } from '@angular/material/button';
```

Prof. Dr. Razer A N R Montaño

Angular

318

318



## Material Design

- Adicionar os **imports** e **exports** no módulo
  - Arquivo *src/app/soma/soma.module.ts*
  - Dentro de **@NgModule**

```
soma-material > src > app > soma > TS soma.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { SomaComponent } from './soma';
4 import { SomaService } from './services';
5 import { MatFormFieldModule } from '@angular/material/form-field';
6 import { MatInputModule } from '@angular/material/input';
7 import { MatIconModule } from '@angular/material/icon';
8 import { MatButtonModule } from '@angular/material/button';
9
10 @NgModule({
11   declarations: [
12     SomaComponent
13   ],
14   imports: [
15     CommonModule,
16     MatFormFieldModule,
17     MatInputModule,
18     MatIconModule,
19     MatButtonModule
20   ],
21   exports: [
22     SomaComponent,
23     MatFormFieldModule,
24     MatInputModule,
25     MatIconModule,
26     MatButtonModule
27   ],
28   providers: [
29     SomaService
30   ]
31 })
32 export class SomaModule {}
```

Prof. Dr. Razer A N R Montaño

Angular

319

319



## Material Design

- Atualizar o formulário no Componente Soma
  - Arquivo *src/app/soma/soma/soma.component.html*
- Estrutura
  - **<form>**
    - **<mat-form-field>**
      - **<mat-label>**
      - **<input>**
      - **<mat-icon>**
      - **<mat-hint>**
    - **<mat-form-field>**
      - **<mat-label>**
      - **<input>**
      - **<mat-icon>**
      - **<mat-hint>**
    - **<button>**
    - **<input>**

Prof. Dr. Razer A N R Montaño

Angular

320

320



## Material Design

- Um campo tem a estrutura

```
<p>
  <mat-form-field>
    <mat-label>Número 1</mat-label>
    <input matInput #numero1 placeholder="Digite o 1o. número da soma">
    <mat-icon matSuffix>sentiment_very_satisfied</mat-icon>
    <mat-hint>1o. número da soma</mat-hint>
  </mat-form-field>
</p>
```



## Material Design

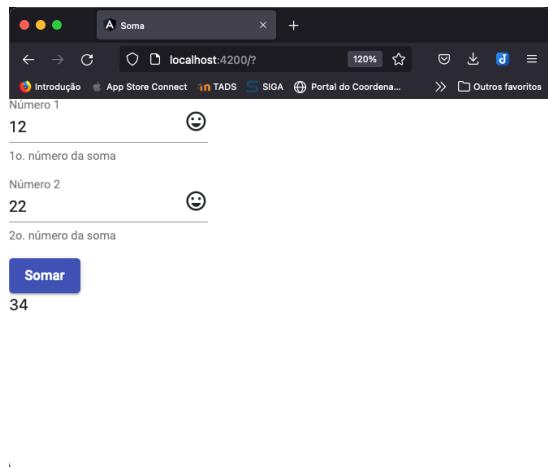
```
<form>
  <p>
    <mat-form-field>
      <mat-label>Número 1</mat-label>
      <input matInput #numero1 placeholder="Digite o 1o. número da soma">
      <mat-icon matSuffix>sentiment_very_satisfied</mat-icon>
      <mat-hint>1o. número da soma</mat-hint>
    </mat-form-field>
  </p>
  <p>
    <mat-form-field>
      <mat-label>Número 2</mat-label>
      <input matInput #numero2 placeholder="Digite o 2o. número da soma">
      <mat-icon matSuffix>sentiment_very_satisfied</mat-icon>
      <mat-hint>2o. número da soma</mat-hint>
    </mat-form-field>
  </p>
  <button mat-raised-button type="button" color="primary"
    (click)="somar(numero1.value, numero2.value)">Somar</button>
  <p><input matInput [value]="resultado" readonly></p>
</form>
```



## Material Design

- Testar o projeto

**\$ ng serve**



Prof. Dr. Razer A N R Montaño

Angular

323

323



## Material Design..

- Alguns componentes do Material
  - **mat-form-field**: agrupa componentes como um campo de formulário
  - **mat-label**: label do campo dentro de um **form-field**
  - **mat-hint**: dica para o campo dentro de um **form-field**
  - **mat-error**: mostra um erro dentro de um **form-field**
  - **matInput**: atributo adicionado a **<input>** e **<textarea>** para estarem dentro de um **mat-form-field**
  - **mat-button**, **mat-button-raised**: atributo adicionado a **<button>** ou **<a>** para aplicar material no botão
  - **mat-select**, **mat-option**: cria combo box e suas opções
  - **mat-radio-group**, **mat-radio-button**: cria botões de rádio
  - **mat-datepicker**, **mat-datepicker-toggle**: cria um **datepicker** e o botão para acioná-lo

Prof. Dr. Razer A N R Montaño

Angular

324

324



## EXERCÍCIOS..

1. Adicionar o Material no projeto Soma e personalizar o formulário
2. Personalizar com Material os projetos Subracao, Multiplicacao e Divisao
3. Criar uma nova aplicação Calculadora, contendo o mesmo formulário mas com 4 botões para efetuar as 4 operações

Prof. Dr. Razer A N R Montaño

Angular

325

325



# CRUD

Prof. Dr. Razer A N R Montaño

Angular

326

326



## CRUD

### Objetivos

- ✓ Implementar um CRUD de Pessoa em Angular
- ✓ Usar o Local Storage



### Referências

➤ <https://angular.io>



## CRUD

- Serão adicionados:
  - Formulário de Inserção, Remoção, Atualização e Consulta
- Passos
  1. Criar projeto e Instalar Bootstrap
  2. Módulo de Rotas
  3. Criar Módulo Pessoa
  4. Criar Modelo para Pessoa
  5. Criar e Implementar Serviço para Pessoa
  6. Implementar o Componente para Listagem
  7. Implementar o Componente para Inserção
  8. Implementar o Componente para Edição
  9. Implementar a Remoção



## CRUD

- Estrutura a ser criada
- AppModule
  - PessoaModule
    - EditarPessoaComponent
    - InserirPessoaComponent
    - ListarPessoaComponent
    - services/PessoaService
  - SharedModule
    - models/PessoaModel

Prof. Dr. Razer A N R Montaño

Angular

329

329

1

## Criar Projeto e Instalar o Bootstrap

Prof. Dr. Razer A N R Montaño

Angular

330

330



## Criar Projeto

- Criar o projeto

```
$ ng new crud-pessoa
```

- Na pergunta se quer **Criar o Angular Routing**, responder Y (Yes)
  - Vai criar o arquivo *src/app/app-routing.module.ts*

- Logo em seguida

```
$ cd crud-pessoa
```

```
$ ng serve
```

- Testar em

```
http://localhost:4200
```



## Adicionar Bootstrap

- Instalar o Bootstrap no projeto
- Dentro do diretório do projeto

```
$ npm install --save bootstrap
```

- Configurar Bootstrap no projeto
  - Adicionar o CSS no *angular.json*

```
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
```



## Adicionar Bootstrap

- Adicionar o <nav> em *src/app/app.component.html*

```
<nav class="navbar navbar-default">
  <div class="conteiner-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        {{title}}
      </a>
    </div>
  </div>
</nav>
```

Prof. Dr. Razer A N R Montaño

Angular

333

333



## Adicionar Bootstrap.

- O template {{title}} se refere ao atributo **title** em **AppComponent** (*src/app/app.component.ts*)

```
export class AppComponent {
  title = 'crud-pessoa';
}
```

Prof. Dr. Razer A N R Montaño

Angular

334

334

2

## Módulo de Rotas

Prof. Dr. Razer A N R Montaño

Angular

335

335



## Rotas

- Deve-se configurar as rotas da aplicação
  - Esquema de navegação
  - Gerenciado pelo **RouterModule**
- Foi criado o módulo de rotas da aplicação
  - Arquivo `src/app/app-routing.module.ts`
  - Usando o **RouterModule**
  - Já foi importado no módulo raiz
- O Angular
  - Mantém rotas como um módulo, *Singleton*
  - Inicializa com as rotas configuradas

Prof. Dr. Razer A N R Montaño

Angular

336

336



## Rotas

- Arquivo `src/app/app-routing.module.ts`
  - Como ainda não temos rotas, deixamos `routes` vazio

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];

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

Prof. Dr. Razer A N R Montaño

Angular

337

337



## Rotas

- No arquivo a constante `routes`:
  - Armazena a estrutura de rotas, como uma lista
  - Sintaxe:

```
{ path: 'login', component: LoginComponent }
```

- Indica que ao acessar `http://localhost:4200/login`, o `LoginComponent` tratará a requisição

Prof. Dr. Razer A N R Montaño

Angular

338

338



## Rotas

- Já está importado no módulo principal (*src/app/app.module.ts*)

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5
6  import { AppRoutingModule } from './app-routing.module'; [Red box]
7
8  @NgModule({
9    declarations: [
10      AppComponent
11    ],
12    imports: [
13      BrowserModule,
14      AppRoutingModule [Red box]
15    ],
16    providers: [],
17    bootstrap: [AppComponent]
18  })
19  export class AppModule { }
```

Prof. Dr. Razer A N R Montaño

Angular

339

339



## Rotas

- Deve-se indicar ao Angular ONDE renderizar o HTML que o componente irá gerar
- Ex.:

- Ao acessar **http://localhost:4200/login**
- O Angular procura na sua estrutura de rotas e encontra

```
{ path: 'login', component: LoginComponent }
```

- Sinalizando que **LoginComponent** tratará a requisição
- O HTML gerado por **LoginComponent** será colocado na tela onde estiver uma tag especial

```
<router-outlet></router-outlet>
```

- Deve-se adicionar esta tag no *src/app/app.component.html* da aplicação

Prof. Dr. Razer A N R Montaño

Angular

340

340



## Rotas.

- Atualizando *src/app/app.component.html*

```
<div class="conteiner-fluid">
<nav class="navbar navbar-default">
  <div class="conteiner-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        {{title}}
      </a>
    </div>
  </div>
</nav>
<router-outlet></router-outlet>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

341

341

3

## Criar Módulo Pessoa

Prof. Dr. Razer A N R Montaño

Angular

342

342



## Módulo Pessoa

- Criar o módulo de Pessoa para gerenciar os dados

**\$ ng g module pessoa**

- Arquivo gerado: *src/app/pessoa/pessoa.module.ts*

Prof. Dr. Razer A N R Montaño

Angular

343

343



## Módulo Pessoa

- Deve-se registrar o módulo na aplicação:
  - Arquivo *src/app/app.module.ts*
  - Adicionar um **import**
  - Adicionar a importação dentro de **@NgModule**

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5
6  import { AppRoutingModule } from './app-routing.module';
7  import { PessoaModule } from './pessoa/pessoa.module';
8
9  @NgModule({
10    declarations: [
11      AppComponent
12    ],
13    imports: [
14      BrowserModule,
15      AppRoutingModule,
16      PessoaModule
17    ],
18    providers: [],
19    bootstrap: [AppComponent]
20  })
21  export class AppModule { }
```

Prof. Dr. Razer A N R Montaño

Angular

344

344



## Módulo Pessoa.

- Vai agrupar as operações e telas de pessoa
  - Componente de Inserção: **InserirPessoaComponent**
  - Componente de Edição: **EditarPessoaComponent**
  - Componente de Lista: **ListarPessoaComponent**
  - Lógica de Negócio/Serviço: **PessoaService**
- E o Modelo?
  - Fica fora do módulo **Pessoa**
  - Será compartilhado em outros módulos
  - Módulo específico : **SharedModule**, será criado mais para frente

Prof. Dr. Razer A N R Montaño

Angular

345

345

4

## Criar Modelo para Pessoa

Prof. Dr. Razer A N R Montaño

Angular

346

346



## Modelo Pessoa

- Representa o dado de Pessoa
  - Id - número
  - Nome - string
  - Idade - número
- Será armazenado localmente
  - Por enquanto!!

Prof. Dr. Razer A N R Montaño

Angular

347

347



## Modelo Pessoa

- Criar a classe Pessoa
- Executar no diretório do projeto

```
$ ng g class shared/models/pessoa --type=model
```

- Arquivo gerado: **src/app/shared/models/pessoa.model.ts**
- Foi criada dentro de um diretório **shared**
  - Será um módulo usado para compartilhamento
  - Vários módulos/componentes do projeto poderão importar
  - Não deve depender de nenhum outro elemento

Prof. Dr. Razer A N R Montaño

Angular

348

348



## Modelo Pessoa.

- Adicione os atributos no construtor da classe

```
1  export class Pessoa {  
2      constructor(  
3          public id?: number,  
4          public nome?: string,  
5          public idade?: number) {  
6      }  
7  }  
8
```

- Automaticamente cria **id**, **nome** e **idade** como atributos
- O "?" indica que são opcionais na chamada do construtor

5

## Criar e Implementar Serviço para Pessoa



## Serviço Pessoa

- Criar a classe de serviço para Pessoa
  - Vai efetuar a lógica de negócio

```
$ ng g service pessoa/services/pessoa
```

- Arquivo gerado: *src/app/pessoa/services/pessoa.service.ts*



## Serviço Pessoa.

- Deve-se importar o serviço no módulo:  
*src/app/pessoa/pessoa.module.ts*
  - Adicionar um **import**
  - Adicionar um **provider** dentro de **@NgModule**

```
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3
4  import { PessoaService } from './services/pessoa.service';
5
6  @NgModule({
7    declarations: [],
8    imports: [
9      CommonModule
10    ],
11    providers: [
12      PessoaService
13    ]
14  })
15  export class PessoaModule { }
```



## Métodos do Serviço Pessoa

- Serão implementados 5 métodos:
  - `listarTodos()`
  - `inserir()`
  - `buscarPorId()`
  - `alterar()`
  - `remover()`

Prof. Dr. Razer A N R Montaño

Angular

353

353



## Métodos do Serviço Pessoa

- Primeiramente: Preparar a classe de serviço
  - Importar o *model* Pessoa
  - Definir uma chave para acesso ao *LocalStorage*
- *LocalStorage*
  - Um array pré-definido : `localStorage`
  - Acessado diretamente no código
  - Armazena Strings
  - Será usado para armazenar JSON de pessoas
- Para manipular JSON: Objeto JSON
  - `JSON.parse(string)`: dada uma string JSON, retorna um objeto javascript
  - `JSON.stringify(objeto)`: dado um objeto javascript, retorna uma string JSON

Prof. Dr. Razer A N R Montaño

Angular

354

354



## Preparação da classe serviço

```
1 import { Injectable } from '@angular/core';
2
3 import { Pessoa } from '../../../../../shared/models/pessoa.model';
4
5 const LS_CHAVE: string = "pessoas";
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class PessoaService {
11
12   constructor() { }
13
14 }
```

Prof. Dr. Razer A N R Montaño

Angular

355

355



## Método: listarTodos()

```
1 import { Injectable } from '@angular/core';
2
3 import { Pessoa } from '../../../../../shared/models/pessoa.model';
4
5 const LS_CHAVE: string = "pessoas";
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class PessoaService {
11
12   constructor() { }
13
14   listarTodos(): Pessoa[] {
15     const pessoas = localStorage[LS_CHAVE];
16     // Precisa do condicional, pois retorna undefined se a chave não existe
17     return pessoas ? JSON.parse(pessoas) : [];
18   }
19
20 }
```

Prof. Dr. Razer A N R Montaño

Angular

356

356



## Método: inserir()

```
20  inserir(pessoa: Pessoa): void {
21    // Obtém a lista completa de pessoas
22    const pessoas = this.listarTodos();
23
24    // Seta o ID único
25    // Para não precisar gerenciar, será usado o Timestamp
26    // Quantidade de segundos desde 1970
27    pessoa.id = new Date().getTime();
28
29    // Adiciona no final da lista
30    pessoas.push(pessoa);
31
32    // Armazena no Local Storage
33    localStorage[LS_CHAVE] = JSON.stringify(pessoas);
34 }
```



## Método: buscarPorId()

```
buscarPorId(id: number): Pessoa | undefined {
  // Obtém a lista completa de pessoas
  const pessoas: Pessoa[] = this.listarTodos();

  // Efetua a busca
  // find() : retorna o primeiro elemento da lista que satisfaz a condição
  //           caso contrário, undefined
  return pessoas.find(pessoa => pessoa.id === id);
}
```



## Método: buscarPorId()

- Sobre o método **find()**
  - Retorna o primeiro elemento do array que satisfaz a condição
  - Se não encontrar, retorna **undefined**
- No exemplo

```
pessoas.find( pessoa => pessoa.id === id)
```

Nome do elemento para ser usado na condição

Condição a ser satisfeita  
Usa o nome definido antes do  
"=>"



## Método: buscarPorId()

- => é chamada de *Arrow Function*
  - Introduzido no ES6
  - É simplificação na escrita de funções
- Sintaxe:

```
() => {}
```

  - Antes do => tem-se os parâmetros, se for só um dá pra tirar os parênteses
  - Depois do => tem-se o código, se for só uma linha, dá pra tirar as chaves



## Método: buscarPorId()

- Assim

```
pessoas.find( pessoa => pessoa.id === id)
```

- Equivale a

```
function verificar(pessoa) {  
    return pessoa.id === id;  
}  
  
pessoas.find( verificar )
```

- Para cada elemento do *array*, a função **verificar()** é chamada



## Método atualizar()

```
46  atualizar(pessoa: Pessoa): void {  
47      // Obtém a lista completa de pessoas  
48      const pessoas: Pessoa[] = this.listarTodos();  
49  
50      // Varre a lista de pessoas  
51      // Quando encontra a pessoa com o mesmo id, altera a lista  
52      pessoas.forEach( (obj, index, objs) => {  
53          if (pessoa.id === obj.id) {  
54              objs[index] = pessoa  
55          }  
56      });  
57  
58      // Armazena a nova lista no Local Storage  
59      localStorage[LS_CHAVE] = JSON.stringify(pessoas);  
60  
61 }
```



## Método: atualizar()

- Sobre o método **forEach()**

- Varre todos os elementos do *array* e chama a *Arrow Function* definida
- 3 parâmetros (antes do =>):
  - Objeto corrente, Índice do objeto corrente, Array completo
  - O código (depois do =>) pode usar estes 3 parâmetros

- No exemplo

```
pessoas.forEach( (obj, index, objs) => {
  if (pessoa.id === obj.id) {
    objs[index] = pessoa;
  }
});
```

Prof. Dr. Razer A N R Montaño

Angular

363

363



## Método: atualizar()

- No exemplo

```
pessoas.forEach( (obj, index, objs) => {
  if (pessoa.id === obj.id) {
    objs[index] = pessoa;
  }
});
```

- Equivale a

```
function alterarPessoa(obj, index, objs) {
  if (pessoa.id === obj.id) {
    objs[index] = pessoa;
  }
}
pessoas.forEach( alterarPessoa );
```

Prof. Dr. Razer A N R Montaño

Angular

364

364



## Método remover()

```
63 |   remover(id: number): void {  
64 |     // Obtém a lista completa de pessoas  
65 |     // Feito com let para poder ser alterada  
66 |     let pessoas: Pessoa[] = this.listarTodos();  
67 |  
68 |     // filter(): retorna a mesma lista, com os registros que  
69 |     //           satisfazem a condição, isto é, cujo  
70 |     //           id é diferente do passado na função  
71 |     pessoas = pessoas.filter(pessoa => pessoa.id !== id);  
72 |  
73 |     // Atualiza a lista de pessoas  
74 |     localStorage[LS_CHAVE] = JSON.stringify(pessoas);  
75 | }
```



## Método: remover().

- Sobre o método **filter()**
  - Retorna um *array* contendo todos os elementos que satisfazem a condição
- No exemplo

```
pessoas = pessoas.filter(pessoa => pessoa.id !== id);
```

## 6

## Implementar o Componente para Listagem

Prof. Dr. Razer A N R Montaño

Angular

367

367



## Componente de Listagem

- Executar

```
$ ng g component pessoa/listar-pessoa
```

- Criou o componente **ListarPessoaComponent**
- Componente é um elemento a ser mostrado, precisa registrar uma rota
  - Quando o usuário acessar `http://localhost:4200/pessoas/listar`
  - Deve redirecionar para o componente **ListarPessoaComponent**
- Também deve-se registrar a rota vazia
  - Isto é, quando o usuário acessar `http://localhost:4200`
  - Dever redirecionar para o componente **ListarPessoaComponent**

Prof. Dr. Razer A N R Montaño

Angular

368

368



## Componente de Listagem

- Deve-se
  - Importar o componente
- Adicionar as rotas em **routes** do arquivo: *src/app/app-routing.module.ts*

```

1   { path: '',
2     redirectTo: 'pessoas/listar',
3     pathMatch: 'full' },
4   { path: 'pessoas',
5     redirectTo: 'pessoas/listar' },
6   { path: 'pessoas/listar',
7     component: ListarPessoaComponent }

```



## Componente de Listagem

```

1  import { NgModule } from '@angular/core';
2  import { Routes, RouterModule } from '@angular/router';
3
4  import { ListarPessoaComponent } from './pessoa/listar-pessoa/listar-pessoa.component';
5
6  const routes: Routes = [
7    { path: '',
8      redirectTo: 'pessoas/listar',
9      pathMatch: 'full' },
10   { path: 'pessoas',
11     redirectTo: 'pessoas/listar' },
12   { path: 'pessoas/listar',
13     component: ListarPessoaComponent }
14 ];
15
16 @NgModule({
17   imports: [RouterModule.forRoot(routes)],
18   exports: [RouterModule]
19 })
20 export class AppRoutingModule { }
21

```



## Componente de Listagem

- Ao rodar a aplicação, abre o HTML de **ListarPessoaComponent**
  - Indicando que a rota funcionou
- Por causa da rota:

```
{     path: '',
      redirectTo: 'pessoas/listar',
      pathMatch: 'full' },
```

crud-pessoa

listar-pessoa works!

- Quando acessa **http://localhost:4200**, redireciona para  
**http://localhost:4200/pessoas/listar**



## Componente de Listagem

- Para criar o HTML da listagem de Pessoas
  - Instalar e adicionar o **Font-Awesome**

```
$ npm install --save @fortawesome/fontawesome-free
```

- Adicionar nos *styles* em *angular.json*

```
26          "styles": [
27            "src/styles.css",
28            "node_modules/bootstrap/dist/css/bootstrap.min.css",
29            "node_modules/@fortawesome/fontawesome-free/css/all.css"
30          ],
```



## Componente de Listagem

- Criar o HTML da listagem de Pessoas
  - Altera o arquivo: *src/app/pessoa/listar-pessoa/listar-pessoa.component.html*
- Outras alterações
  - Altera o título da aplicação: *src/app/app.component.ts*
  - Alterar arquivos de testes:
    - *src/app/app.component.spec.ts*

Prof. Dr. Razer A N R Montaño

Angular

373

373



## listar-pessoa.component.html (1/2)

```
<h1>Pessoas</h1>

<table class="table table-striped table-bordered table-hover">
<tbody>
    <tr>
        <th>Nome</th>
        <th>Idade</th>
        <th class="text-center">
            <a href="#" title="Novo" alt="Novo"
               class="btn btn-xs btn-success">
                <i class="fa fa-plus" aria-hidden="true"></i> Novo
            </a>
        </th>
    </tr>
```

Prof. Dr. Razer A N R Montaño

Angular

374

374



## listar-pessoa.component.html (2/2)

```
<tr>
  <td> </td>
  <td> </td>
  <td class="text-center" style="width: 300px">
    <a href="#" title="Editar" alt="Editar" class="btn btn-xs btn-info">
      <i class="fa fa-edit" aria-hidden="true"></i> Editar
    </a>
    <a href="#" title="Remover" alt="Remover" class="btn btn-xs btn-danger">
      <i class="fa fa-times" aria-hidden="true"></i> Remover
    </a>
  </td>
</tr>
</tbody>
</table>
<p>Nenhuma pessoa cadastrada.</p>
```

Prof. Dr. Razer A N R Montaño

Angular

375

375



## Resultado

[crud-pessoa](#)

### Pessoas

Nome	Idade	
		<button>+ Novo</button> <button>Editar</button> <button>Remover</button>

Nenhuma pessoa cadastrada.

Prof. Dr. Razer A N R Montaño

Angular

376

376



## Componente de Listagem

- Altera o título da aplicação: *src/app/app.component.ts*
  - Alterar o atributo **title**

```

1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'Cadastro de Pessoas';
10 }
11

```



## Componente de Listagem

- Alterar arquivos de testes:
  - Remover as linhas que testam por "crud-pessoa" (as assinaladas abaixo)
  - *src/app/app.component.spec.ts*

```

1  import { TestBed } from '@angular/core/testing';
2  import { AppComponent } from './app.component';
3
4  describe('AppComponent', () => {
5    beforeEach(async () => {
6      await TestBed.configureTestingModule({
7        declarations: [
8          AppComponent
9        ],
10       }).compileComponents();
11     });
12
13   it('should create the app', () => {
14     const fixture = TestBed.createComponent(AppComponent);
15     const app = fixture.componentInstance;
16     expect(app).toBeTruthy();
17   });
18
19   it('should have as title "crud-pessoa"', () => {
20     const fixture = TestBed.createComponent(AppComponent);
21     const app = fixture.componentInstance;
22     expect(app.title).toEqual('crud-pessoa');
23   });
24
25   it('should render title', () => {
26     const fixture = TestBed.createComponent(AppComponent);
27     fixture.detectChanges();
28     const compiled = fixture.nativeElement;
29     expect(compiled.querySelector('.content span').textContent).toContain('crud-pessoa app is running!'));
30   });
31 });
32

```



## Resultado

Cadastro de Pessoas

### Pessoas

Nome	Idade	
		<span style="color: green; border: 1px solid green; padding: 2px;">+ Novo</span> <span style="color: cyan; border: 1px solid cyan; padding: 2px;">Edita</span> <span style="color: red; border: 1px solid red; padding: 2px;">Remover</span>

Nenhuma pessoa cadastrada.

Prof. Dr. Razer A N R Montaño

Angular

379

379



## Implementar a Listagem

- Importar **RouterModule** e **FormsModule** em **src/app/pessoa/pessoa.module.ts**
  - Usados para implementar o formulário e as rotas

```

1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3
4 import { RouterModule } from '@angular/router';
5 import { FormsModule } from '@angular/forms';
6
7 import { PessoaService } from './services/pessoa.service';
8 import { ListarPessoaComponent } from './listar-pessoa/listar-pessoa.component';
9
10 @NgModule({
11   declarations: [ListarPessoaComponent],
12   imports: [
13     CommonModule,
14     RouterModule,
15     FormsModule
16   ],
17   providers: [
18     PessoaService
19   ]
20 })
21 export class PessoaModule { }
22

```

Prof. Dr. Razer A N R Montaño

Angular

380

380



## Implementar a Listagem

- Adicionar o serviço em *src/app/pessoa/listar-pessoa/listar-pessoa.component.ts*
  - Adicionar construtor com a injeção do serviço

```

1  import { Component, OnInit } from '@angular/core';
2
3  import { PessoaService } from '../services/pessoa.service';
4
5  @Component({
6    selector: 'app-listar-pessoa',
7    templateUrl: './listar-pessoa.component.html',
8    styleUrls: ['./listar-pessoa.component.css']
9  })
10 export class ListarPessoaComponent implements OnInit {
11
12   constructor(private pessoaService : PessoaService) { }
13
14   ngOnInit(): void {
15   }
16
17 }
18

```

Prof. Dr. Razer A N R Montaño

Angular

381

381



## Implementar a Listagem

- Implementar os métodos em *app/pessoa/listar-pessoa/listar-pessoa.component.ts*
  - Importar o *model*
  - Criar um atributo *pessoas*, que ficará disponível no HTML
    - Inicializar com []
    - Inicializar no **ngOnInit()**
  - Implementar o método **listarTodos()**

```

1  import { Component, Input, OnInit } from '@angular/core';
2
3  import { PessoaService } from '../services/pessoa.service';
4  import { Pessoa } from '../../../../../shared/models/pessoa.model';
5
6  @Component({
7    selector: 'app-listar-pessoa',
8    templateUrl: './listar-pessoa.component.html',
9    styleUrls: ['./listar-pessoa.component.css']
10 })
11 export class ListarPessoaComponent implements OnInit {
12
13   pessoas: Pessoa[] = [];
14
15   constructor(private pessoaService : PessoaService) { }
16
17   ngOnInit(): void {
18     this.pessoas = this.listarTodos();
19   }
20
21   listarTodos(): Pessoa[] {
22     return this.pessoaService.listarTodos();
23   }
24
25 }
26

```

Prof. Dr. Razer A N R Montaño

Angular

382

382



## Implementar a Listagem

- Alterar o HTML para mostrar as Pessoas *app/pessoa/listar-pessoa/listar-pessoa.component.html*
  - Primeiro `<tr>` : adicionar `*ngFor` para iterar sobre a lista
  - Nas colunas: adicionar `{{ pessoa.nome }}` e `{{ pessoa.idade }}` para mostrar os dados
  - No `<p>` ao final : adicionar `*ngIf` para só mostrar quando a lista for vazia

Prof. Dr. Razer A N R Montaño

Angular

383

383



## Implementar a Listagem

```

1  <h1>Pessoas</h1>
2
3  <table class="table table-striped table-bordered table-hover">
4    <thead>
5      <tr>
6        <th>Nome</th>
7        <th>Idade</th>
8        <th class="text-center">
9          <a href="#" title="Novo" alt="Novo" class="btn btn-xs btn-success">
10            <i class="fa fa-plus" aria-hidden="true"></i> Novo
11          </a>
12        </th>
13      </tr>
14    <tr *ngFor="let pessoa of pessoas">
15      <td> {{pessoa.nome}} </td>
16      <td> {{pessoa.idade}} </td>
17      <td class="text-center" style="width: 300px">
18        <a href="#" title="Editar" alt="Editar" class="btn btn-xs btn-info">
19          <i class="fa fa-edit" aria-hidden="true"></i> Editar
20        </a>
21        <a href="#" title="Remover" alt="Remover" class="btn btn-xs btn-danger">
22          <i class="fa fa-times" aria-hidden="true"></i> Remover
23        </a>
24      </td>
25    </tr>
26  </tbody>
27 </table>
28
29  <p *ngIf="pessoas.length==0">Nenhuma pessoa cadastrada.</p>
30

```

Prof. Dr. Razer A N R Montaño

Angular

384

384



## Implementar a Listagem

- Criar algumas Pessoas para teste em `src/app/pessoa/listar-pessoa/listar-pessoa.component.ts`
  - Dentro no `listarTodos()`

```
listarTodos(): Pessoa[] {  
    // return this.pessoaService.listarTodos();  
  
    return [  
        new Pessoa(1, "Razer", 20),  
        new Pessoa(2, "Brunna", 52),  
        new Pessoa(3, "Guilherme", 33),  
        new Pessoa(4, "Juan", 88)  
    ];  
}
```



## Implementar a Listagem.

Cadastro de Pessoas

### Pessoas

Nome	Idade	<button>+ Novo</button>
Razer	20	<button>Editar</button> <button>Remover</button>
Brunna	52	<button>Editar</button> <button>Remover</button>
Guilherme	33	<button>Editar</button> <button>Remover</button>
Juan	88	<button>Editar</button> <button>Remover</button>

7

## Implementar o Componente para Inserção

Prof. Dr. Razer A N R Montaño

Angular

387

387



## Componente de Inserção

- Executar

```
$ ng g component pessoa/inserir-pessoa
```

- Criou o componente **InserirPessoaComponent**
  - Arquivos `src/app/pessoa/inserir-pessoa/*`
- É necessário registrar uma rota para esse componente
  - Quando o usuário acessar `http://localhost:4200/pessoas/novo`
  - Deve redirecionar para o componente **InserirPessoaComponent**

Prof. Dr. Razer A N R Montaño

Angular

388

388



## Componente de Inserção

- Adiciona-se a nova rota no arquivo: `src/app/app-routing.module.ts`
  - Não esquecer de importar o `InserirPessoaComponent`

```
{ path: 'pessoas/novo',
  component: InserirPessoaComponent }
```

Prof. Dr. Razer A N R Montaño

Angular

389

389



## Componente de Inserção

```
1  import { NgModule } from '@angular/core';
2  import { Routes, RouterModule } from '@angular/router';
3
4  import { ListarPessoaComponent } from './pessoa/listar-pessoa/listar-pessoa.component';
5  import { InserirPessoaComponent } from './pessoa/inserir-pessoa/inserir-pessoa.component';
6
7  const routes: Routes = [
8    { path: '',
9      redirectTo: 'pessoas/listar',
10     pathMatch: 'full' },
11    { path: 'pessoas',
12      redirectTo: 'pessoas/listar' },
13    { path: 'pessoas/listar',
14      component: ListarPessoaComponent },
15    { path: 'pessoas/novo',
16      component: InserirPessoaComponent }
17  ];
18
19  @NgModule({
20    imports: [RouterModule.forRoot(routes)],
21    exports: [RouterModule]
22  })
23  export class AppRoutingModule { }
```

Prof. Dr. Razer A N R Montaño

Angular

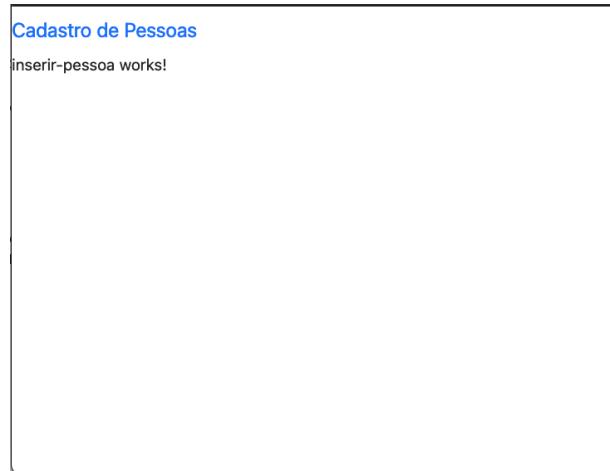
390

390



## Componente de Inserção

- Acessando: `http://localhost:4200/pessoas/novo`



Prof. Dr. Razer A N R Montaño

Angular

391

391



## Componente de Inserção

- Para fazer o botão "Novo" da tela de listagem ir para a rota `pessoas/novo`
  - Editar `src/app/pessoa/listar-pessoa/listar-pessoa.component.html`
  - No elemento `<a>` do botão novo, adicionar`[routerLink] = "[ '/pessoas/novo' ]"`

```
8   <th class="text-center">
9     <a [routerLink] = "[ '/pessoas/novo' ]"
10    href="#" title="Novo" alt="Novo" class="btn btn-xs btn-success">
11      <i class="fa fa-plus" aria-hidden="true"></i> Novo
12    </a>
13  </th>
```

Prof. Dr. Razer A N R Montaño

Angular

392

392



## Componente de Inserção

- Basta testar se o botão Novo vai para a tela de Inserção

Cadastro de Pessoas

### Pessoas

Nome	Idade	
Razer	20	<button>+ Novo</button>
Brunna	52	<button>Editar</button> <button>X Remover</button>
Guilherme	33	<button>Editar</button> <button>X Remover</button>
Juan	88	<button>Editar</button> <button>X Remover</button>

Cadastro de Pessoas

Inserir-pessoa works!



Prof. Dr. Razer A N R Montaño

Angular

393

393



## inserir-pessoa.component.html (1/2)

- Adicionar o HTML puro

```
<h1>Nova Pessoa</h1>

<div class="well">
  <form>
    <div class="form-group">
      <label for="nome">Nome:</label>
      <input type="text" class="form-control" id="nome" name="nome"
             minlength="2" required>
      <div class="alert alert-danger">
        <div> Digite o nome da pessoa. </div>
        <div> O nome deve conter ao menos 2 caracteres. </div>
      </div>
    </div>
  </form>
```

Prof. Dr. Razer A N R Montaño

Angular

394

394



## inserir-pessoa.component.html (2/2)

```
<div class="form-group">
  <label for="idade">Idade:</label>
  <input type="text" class="form-control" id="idade" name="idade" required>
  <div class="alert alert-danger">
    <div> Digite a idade da pessoa. </div>
  </div>
</div>
<div class="form-group">
  <button type="button" class="btn btn-primary">
    <i class="fa fa-save" aria-hidden="true"></i> Salvar
  </button>
  <a class="btn btn-secondary">
    <i class="fa fa-arrow-left" aria-hidden="true"></i> Voltar
  </a>
</div>
</form>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

395

395



## Resultado

Cadastro de Pessoas

### Nova Pessoa

Nome:

Digite o nome da pessoa.  
O nome deve conter ao menos 2 caracteres.

Idade:

Digite a idade da pessoa.

Salvar ← Voltar

Prof. Dr. Razer A N R Montaño

Angular

396

396



## Implementar a Inserção de Pessoa

- Em `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts`
- Adicionar atributo `formPessoa`: é uma representação do formulário no componente

```
// Recebe uma referência do formulário aqui no componente  
// 'formPessoa' deve ser o nome do formulário no HTML  
@ViewChild('formPessoa') formPessoa!: NgForm;
```

- Importar `ViewChild` do `@angular/core`

Prof. Dr. Razer A N R Montaño

Angular

397

397



## Implementar a Inserção de Pessoa

- Em `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts`
  - Adicionar atributo `pessoa`: para fazer o *binding* com a tela
- ```
// Atributo de binding, os dados digitados no formulário vêm para  
// este atributo  
pessoa!: Pessoa;
```
- Importar o model `Pessoa`

Prof. Dr. Razer A N R Montaño

Angular

398

398



## Implementar a Inserção de Pessoa

- Resultado

```
crud-pessoa > src > app > pessoa > inserir-pessoa > inserir-pessoa.component.ts > ...
1 import { Component, OnInit, ViewChild } from '@angular/core';
2 import { NgForm } from '@angular/forms';
3 import { Pessoa } from 'src/app/shared/models/pessoa.model';
4
5 @Component({
6   selector: 'app-inserir-pessoa',
7   templateUrl: './inserir-pessoa.component.html',
8   styleUrls: ['./inserir-pessoa.component.css']
9 })
10 export class InserirPessoaComponent implements OnInit {
11   @ViewChild('formPessoa') formPessoa!: NgForm;
12   pessoa!: Pessoa;
13
14   constructor() {}
15
16   ngOnInit(): void {
17   }
18
19 }
```

Prof. Dr. Razer A N R Montaño

Angular

399

399



## Implementar a Inserção de Pessoa

- Em *src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts*
  - Adicionar as injecções no construtor
  - Fazer as importações (VS Code já as adiciona)

```
// Deve-se injetar no construtor:
// - service, para efetuar a operação
// - Router, para redirecionar para a tela de listagem depois da
// inserção
constructor(
  private pessoaService: PessoaService,
  private router: Router) { }
```

Prof. Dr. Razer A N R Montaño

Angular

400

400



## Implementar a Inserção de Pessoa

- Em `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts`
  - Criar uma pessoa no `ngOnInit()`

```
ngOnInit(): void {  
    // Cria uma instância vazia, para não dar erro de referência  
    this.pessoa = new Pessoa();  
}
```

Prof. Dr. Razer A N R Montaño

Angular

401

401



## Implementar a Inserção de Pessoa

- Em `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts`
  - Criar o método `inserir()` que será invocado do HTML

```
// Para inserir:  
// - Verifica se o formulário é válido, se não deu nenhum erro  
// - Se OK  
// . Chama o inserir do Service, this.pessoa está preenchida (binding)  
// . Redireciona para /pessoas  
inserir(): void {  
    if (this.formPessoa.form.valid) {  
        this.pessoaService.inserir(this.pessoa);  
        this.router.navigate( ["/pessoas"] );  
    }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

402

402



## Implementar a Inserção de Pessoa

- Em *src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts*
  - Cuidar com as importações

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

import { PessoaService } from '../services/pessoa.service';
import { Pessoa } from '../../shared/models/pessoa.model';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
```



## Implementar a Inserção de Pessoa

- Em *src/app/pessoa/inserir-pessoa/inserir-pessoa.component.html*
- Acertar o HTML com as diretivas do Angular
- O nome do formulário
  - **#formPessoa** : o mesmo que está em **@ViewChild**
  - **ngForm** : diretiva para tratamento de formulários

```
<form #formPessoa="ngForm">
```



## Implementar a Inserção de Pessoa

- Efetuar o *binding* (*two way binding*) do campo nome

- **[ (ngModel) ]="pessoa.nome"**
  - Se o usuário alterar na tela, automaticamente altera no componente
  - Se for alterado no componente, automaticamente altera na tela
- **#nome="ngModel"**
  - Cria uma variável "nome" para usar no HTML (ex, verificar erros)

```
<input type="text" class="form-control" id="nome" name="nome"
       [ (ngModel) ]="pessoa.nome"
       #nome="ngModel"
       minlength="2" required>
```

Prof. Dr. Razer A N R Montaño

Angular

405

405



## Implementar a Inserção de Pessoa

- Para controlar se as mensagens de erro devem ser mostradas
- **#nome="ngModel"**
  - Dentro de "nome" tem o array "errors"
  - \*ngIf na <div> mais de fora de erro
  - Só mostra se houve erro (**errors**), se o usuário chegou a editar o campo (**dirty**) e se ele já saiu do componente (**touched**)

```
<div *ngIf="nome.errors && (nome.dirty || nome.touched)"
      class="alert alert-danger">
```

Prof. Dr. Razer A N R Montaño

Angular

406

406



## Implementar a Inserção de Pessoa

- Adicionar diretivas nas `<div>` internas de erro, para mostrar o erro específico

- `#nome="ngModel"`

- Dentro de "nome" tem o array "errors" com os erros para as validações definidas: `minlength` e `required`

- Usa-se para mostrar/esconder a mensagem de erro

- Propriedade `hidden` do `<div>`

```
<div [hidden]="!nome.errors?.['required']">
    Digite o nome da pessoa.
</div>
<div [hidden]="!nome.errors?.['minlength']">
    O nome deve conter ao menos 2 caracteres.
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

407

407



## Implementar a Inserção de Pessoa

- Fazer a mesma coisa para o campo Idade

```
<input type="text" class="form-control" id="idade"
       name="idade"
       [(ngModel)]="pessoa.idade"
       #idade="ngModel"
       required>
<div *ngIf="idade.errors && (idade.dirty || idade.touched)"
      class="alert alert-danger">
    <div [hidden]="! idade.errors?.['required']">
        Digite a idade da pessoa.
    </div>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

408

408



## Implementar a Inserção de Pessoa

- Acertar o botão Salvar
  - Adicionar o click para o método do componente
  - Adicionar controle para habilitar só quando o formulário é válido

```
<button type="button" class="btn btn-primary"
        (click)="inserir()"
        [disabled]="!formPessoa.form.valid">
    <i class="fa fa-save" aria-hidden="true"></i> Salvar
</button>
```

Prof. Dr. Razer A N R Montaño

Angular

409

409



## Implementar a Inserção de Pessoa

- Acertar o botão Voltar
  - Adicionar a navegação para a listagem de pessoa

```
<a class="btn btn-secondary"
    [routerLink]="['/pessoas']">
    <i class="fa fa-arrow-left" aria-hidden="true"></i> Voltar
</a>
```

Prof. Dr. Razer A N R Montaño

Angular

410

410



## Implementar a Inserção de Pessoa

- Remover as Pessoas de teste inseridas em  
*src/app/pessoa/listar-pessoa/listar-pessoa.component.ts*
  - Método `listarTodos()`

```
listarTodos(): Pessoa[] {  
    return this.pessoaService.listarTodos();  
}
```

Prof. Dr. Razer A N R Montaño

Angular

411

411



## Implementar a Inserção de Pessoa.

- Testar se a aplicação está salvando as pessoas

[Cadastro de Pessoas](#)

### Pessoas

Nome	Idade		
Razer	18	<input checked="" type="button"/> Editar	<input type="button"/> Remover
Guilherme	33	<input checked="" type="button"/> Editar	<input type="button"/> Remover

Prof. Dr. Razer A N R Montaño

Angular

412

412

## 8

## Implementar o Componente para Edição

Prof. Dr. Razer A N R Montaño

Angular

413

413



## Componente de Edição

- Executar

```
$ ng g component pessoa/editar-pessoa
```

- Criou o componente **EditarPessoaComponent**
  - Arquivos *src/app/pessoa/editar-pessoa/\**
- É necessário registrar uma rota para esse componente
  - Quando o usuário acessar <http://localhost:4200/pessoas/editar>
  - Deve redirecionar para o componente **EditarPessoaComponent**

Prof. Dr. Razer A N R Montaño

Angular

414

414



## Componente de Edição

- Adiciona-se a nova rota no arquivo: `src/app/app-routing.module.ts`
  - Não esquecer de importar o `EditarPessoaComponent`

```
{ path: 'pessoas/editar/:id',
  component: EditarPessoaComponent }
```

- Para passar parâmetro (`id`) de qual pessoa está sendo editada
  - No `path` passar: `:id`
  - Dentro de `EditarPessoaComponent` pode-se obter o parâmetro passado



## Componente de Edição

- Para fazer o botão "Editar" da tela de listagem ir para a rota `pessoas/editar`
  - Editar o arquivo `src/app/pessoa/listar-pessoa/listar-pessoa.component.html`
  - No elemento `<a>` do botão editar, adicionar

```
[routerLink]="/pessoas/editar"]"
```

- Para passar o `id` como parâmetro, adiciona-se o `id` na lista do `routerLink`, que fará a concatenação, ficando:

```
[routerLink]="/pessoas/editar", pessoa.id]"
```

- É possível porque o `<a>` está dentro do `<tr>` que possui uma iteração

```
<tr *ngFor="let pessoa of pessoas">
```



## editar-pessoa.component.html (1/2)

- Adiciona-se o HTML puro em `src/app/pessoa/editar-pessoa/editar-pessoa.component.html`

```
<h1>Editar Pessoa</h1>
<div class="well">
<form >
  <div class="form-group">
    <label for="nome">Nome:</label>
    <input type="text" class="form-control" id="nome" name="nome"
           minlength="2" required>
    <div class="alert alert-danger">
      <div>Digite o nome da pessoa.</div>
      <div>O nome deve conter ao menos 2 caracteres.</div>
    </div>
  </div>
```



## editar-pessoa.component.html (2/2)

```
<div class="form-group">
  <label for="idade">Idade:</label>
  <input type="text" class="form-control" id="idade" name="idade"
         required>
  <div class="alert alert-danger">
    <div> Digite a idade da pessoa.</div>
  </div>
</div>
<div class="form-group">
  <button type="button" class="btn btn-primary">
    <i class="fa fa-save" aria-hidden="true"></i> Atualizar
  </button>
  <a class="btn btn-secondary">
    <i class="fa fa-arrow-left" aria-hidden="true"></i> Voltar
  </a>
</div>
</form>
</div>
```



## Implementação da Edição de Pessoa

- Em `src/app/pessoa/editar-pessoa/editar-pessoa.component.ts`
  - Adicionar os atributos: `formPessoa` e `pessoa`
  - Acertar as importações

```
export class EditarPessoaComponent implements OnInit {  
  
    @ViewChild("formPessoa") formPessoa!: NgForm;  
    pessoa!: Pessoa;
```

Prof. Dr. Razer A N R Montaño

Angular

419

419



## Implementação da Edição de Pessoa

- Em `src/app/pessoa/editar-pessoa/editar-pessoa.component.ts`
  - Implementar o construtor com as injeções
    - `pessoaService`: para fazer a operação de edição
    - `route`: `ActivatedRoute`, para obter o parâmetro passado na URL
    - `router`: para fazer o redirecionamento, após a edição

```
export class EditarPessoaComponent implements OnInit {  
  
    ...  
    constructor(  
        private pessoaService: PessoaService,  
        private route: ActivatedRoute,  
        private router: Router  
    ) { }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

420

420



## Implementação da Edição de Pessoa

- Em `src/app/pessoa/editar-pessoa/editar-pessoa.component.ts`
  - Implementar o `ngOnInit()` para obter a pessoa, via `id`

```
export class EditarPessoaComponent implements OnInit {
    ...
    ngOnInit(): void {
        // snapshot.params de ActivatedRoute dá acesso aos parâmetros passados
        // Operador + (antes do this) converte para número
        let id = +this.route.snapshot.params['id'];
        // Com o id, obtém a pessoa
        const res = this.pessoaService.buscarPorId(id);
        if (res !== undefined)
            this.pessoa = res;
        else
            throw new Error ("Pessoa não encontrada: id = " + id);
    }
}
```

Prof. Dr. Razer A N R Montaño

Angular

421

421



## Implementação da Edição de Pessoa

- Em `src/app/pessoa/editar-pessoa/editar-pessoa.component.ts`
  - Implementar o `atualizar()`

```
export class EditarPessoaComponent implements OnInit {
    ...
    atualizar(): void {
        // Verifica se o formulário é válido
        if (this.formPessoa.form.valid) {
            // Efetivamente atualiza a pessoa
            this.pessoaService.atualizar(this.pessoa);
            // Redireciona para /pessoas/listar
            this.router.navigate(['/pessoas']);
        }
    }
}
```

Prof. Dr. Razer A N R Montaño

Angular

422

422



## HTML de Inserção com Marcações Angular

- Alterar o HTML em *src/app/pessoa/editar-pessoa/editar-pessoa.component.html*
  - Adicionar o nome do formulário

```
<form #formPessoa="ngForm">
```

Prof. Dr. Razer A N R Montaño

Angular

423

423



## HTML de Inserção com Marcações Angular

- Alterar o HTML em *src/app/pessoa/editar-pessoa/editar-pessoa.component.html*
  - Alteração do `<input>` do nome

```
<input type="text" class="form-control" id="nome" name="nome"  
      [(ngModel)]="pessoa.nome"  
      #nome="ngModel"  
      minlength="2" required>
```

Prof. Dr. Razer A N R Montaño

Angular

424

424



## HTML de Inserção com Marcações Angular

- Alterar o HTML em *src/app/pessoa/editar-pessoa/editar-pessoa.component.html*
  - Alteração nas <div> que mostram erro do campo nome

```
<div *ngIf="nome.errors && (nome.dirty || nome.touched)"  
      class="alert alert-danger">  
    <div [hidden]="!nome.errors?.['required']">  
      Digite o nome da pessoa.  
    </div>  
    <div [hidden]="!nome.errors?.['minlength']">  
      O nome deve conter ao menos 2 caracteres.  
    </div>  
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

425

425



## HTML de Inserção com Marcações Angular

- Alterar o HTML em *src/app/pessoa/editar-pessoa/editar-pessoa.component.html*
  - Alteração do <input> do idade

```
<input type="text" class="form-control" id="idade" name="idade"  
      [(ngModel)]="pessoa.idade"  
      #idade="ngModel"  
      required>
```

Prof. Dr. Razer A N R Montaño

Angular

426

426



## HTML de Inserção com Marcações Angular

- Alterar o HTML em *src/app/pessoa/editar-pessoa/editar-pessoa.component.html*
  - Alteração nas `<div>` que mostram erro do campo idade

```
<div *ngIf="idade.errors && (idade.dirty || idade.touched)"  
      class="alert alert-danger">  
    <div [hidden]="!idade.errors?.['required']">  
      Digite a idade da pessoa.  
    </div>  
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

427

427



## HTML de Inserção com Marcações Angular

- Alterar o HTML em *src/app/pessoa/editar-pessoa/editar-pessoa.component.html*
  - Alteração nos botões de Atualizar e Voltar

```
<button type="button" class="btn btn-primary"  
       (click)="atualizar()"  
       [disabled]="!formPessoa.form.valid">  
  <i class="fa fa-save" aria-hidden="true"></i> Atualizar  
</button>  
  
<a class="btn btn-secondary"  
   [routerLink]="['/pessoas']">  
  <i class="fa fa-arrow-left" aria-hidden="true"></i> Voltar  
</a>
```

Prof. Dr. Razer A N R Montaño

Angular

428

428



## Teste da Edição.

- Neste ponto a edição está funcionando
- Testar!!!

Prof. Dr. Razer A N R Montaño

Angular

429

429

9

## Implementar a Remoção

Prof. Dr. Razer A N R Montaño

Angular

430

430



## Implementar a Remoção

- Está dentro da listagem
  - Não tem uma tela específica
  - Não tem um componente para ele
- Deve-se alterar diretamente no componente de listagem de pessoa
  - *src/app/pessoa/listar-pessoa/listar-pessoa.component.ts*
  - *src/app/pessoa/listar-pessoa/listar-pessoa.component.html*

Prof. Dr. Razer A N R Montaño

Angular

431

431



## Implementar a Remoção

- Alterar o componente de listagem de pessoa
  - *src/app/pessoa/listar-pessoa/listar-pessoa.component.ts*
  - Adicionar um método para remoção

```
remover($event: any, pessoa: Pessoa): void {
    $event.preventDefault();
    if (confirm(`Deseja realmente remover a pessoa ${pessoa.nome}?`)) {
        this.pessoaService.remover(pessoa.id);
        this.pessoas = this.listarTodos();
    }
}
```

Prof. Dr. Razer A N R Montaño

Angular

432

432



## Implementar a Remoção

- Nota sobre o **\$event.preventDefault()**

- O botão de remover é um *link*, que tem um comportamento *default*: abrir no browser o que está no **href**
- Então, setamos **href** para "#" e no evento disparado colocamos

```
$event.preventDefault();
```

- Para não termos esse comportamento!!!

```
remover($event: any, pessoa: Pessoa): void {
  $event.preventDefault();
  ...
}
```

- Mais detalhes aqui:

- <https://medium.com/@jacobwarduk/how-to-correctly-use-preventdefault-stoppropagation-or-return-false-on-events-6c4e3f31aedb>



## Implementar a Remoção

- Alterar o componente de listagem de pessoa

*src/app/pessoa/listar-pessoa/listar-pessoa.component.html*

- Adicionar o click no botão de Remoção

```
<a href="#" title="Remover" alt="Remover" class="btn btn-xs btn-danger"
  (click)="remover($event, pessoa)">
  <i class="fa fa-times" aria-hidden="true"></i> Remover
</a>
```



## Teste da Aplicação Toda..

- Neste ponto a remoção está funcionando
- Testar toda a aplicação

Prof. Dr. Razer A N R Montaño

Angular

435

435



## EXERCÍCIOS

1. Criar e executar o CRUD de Pessoa
2. Ao projeto de CRUD de Pessoa, adicionar o campo Data Nascimento, do tipo String
3. OBRIGATÓRIO: No mesmo projeto, criar um CRUD de Endereço com os seguintes dados:
  - Id (Numérico)
  - Rua (String)
  - Número (Numérico)
  - Complemento (String)
  - Bairro (String)
  - CEP (String)
  - Cidade (String)
  - Estado (String)
  - Não precisa relacionar com Pessoa, Cidade, Estado, será feito mais tarde

Prof. Dr. Razer A N R Montaño

Angular

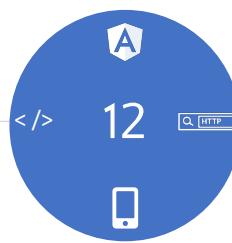
436

436



## EXERCÍCIOS..

4. OBRIGATÓRIO: No mesmo projeto, criar um CRUD de Cidade com os seguintes dados:
  - Id (Numérico)
  - Nome (String)
  - Estado (String)
  - Não precisa relacionar com Estado, será feito mais tarde
5. OBRIGATÓRIO: No mesmo projeto, criar um CRUD de Estado com os seguintes dados:
  - Id (Numérico)
  - Nome (String)
  - Sigla (String)



## Validação e Formatação de Campos



## Validação de campos

### Objetivos

- ✓ Validação de campos numéricos
- ✓ Máscara de data
- ✓ Organização do Código



### Referências

➤ <https://angular.io/guide/form-validation>



## Validação de Campos

- Validação de Campos:
  - Será usado o projeto de CRUD de Pessoa anterior contendo o campo data de nascimento adicionado
- Passos
  1. Criar e adicionar uma diretiva: só aceitar valores numéricos em idade
  2. Diretiva para Validação de Valor de Idade
  3. Máscara de Datas
  4. Pipes



## Validação de Campos.

- Diretiva de Atributo
  - Cria um atributo para elementos HTML
  - Um recurso do Angular
  - Manipular o DOM
  - Estender a funcionalidade de um elemento HTML
    - Conteúdo
    - Estilo
    - Visibilidade
    - Formato

Prof. Dr. Razer A N R Montaño

Angular

441

441

1

## Criar e Adicionar uma Diretiva

Prof. Dr. Razer A N R Montaño

Angular

442

442



## Diretiva de Validação

- Executar

```
$ ng g directive shared/directives/numerico
```

- Criou os arquivos:

- *src/app/shared/directives/numerico.directive.ts* : arquivo contendo a classe da diretiva
- *src/app/shared/directives/numerico.directive.spec.ts* : teste da diretiva

Prof. Dr. Razer A N R Montaño

Angular

443

443



## Diretiva de Validação

- Arquivo criado

- *src/app/shared/directives/numerico.directive.ts*

```
import { Directive } from '@angular/core';
@Directive({
  selector: '[appNumerico]'
```

Atributo a ser usado na tag  
HTML para aplicar a diretiva  
Trocá-lo para "numerico"

Prof. Dr. Razer A N R Montaño

Angular

444

444



## Diretiva de Validação

- Arquivo alterado

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

@Directive({
  selector: '[numerico]'
})
export class NumericoDirective {

  constructor() { }

}
```

Prof. Dr. Razer A N R Montaño

Angular

445

445



## Diretiva de Validação

- Deve-se adicionar um **HostListener**
  - Dentro de *src/app/shared/directives/numerico.directive.ts*
  - Após o construtor
  - Declara um evento do DOM a ser escutado e um método que o trata
  - Será usado para tratar o **KeyUp**: só permitir números

```
@HostListener('keyup', ['$event'])
onKeyUp($event: any) {
  let valor = $event.target.value;

  // expressão regular: remove tudo que não é número
  valor = valor.replace(/[^D]/g, '');

  $event.target.value = valor;
}
```

Prof. Dr. Razer A N R Montaño

Angular

446

446



## Diretiva de Validação

- Acertar as importações
  - Importar `HostListener` e `ElementRef`

```
import { Directive, HostListener, ElementRef } from '@angular/core';
```

Prof. Dr. Razer A N R Montaño

Angular

447

447



## Diretiva de Validação

- Adicionar a diretiva no módulo Pessoa
  - `src/app/pessoa/pessoa.module.ts`

```
import { NumericoDirective } from '../shared/directives/numerico.directive';
```

- Dentro de `declarations` do `@NgModule`

```
@NgModule({
  declarations: [
    ListarPessoaComponent,
    InserirPessoaComponent,
    EditarPessoaComponent,
    NumericoDirective
  ],
  ...
```

Prof. Dr. Razer A N R Montaño

Angular

448

448



## Diretiva de Validação

- Para adicionar a diretiva em um campo, o HTML:
  - `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.html`
  - `src/app/pessoa/editar-pessoa/editar-pessoa.component.html`
- Alterar o trecho de código, adicionando o atributo **numerico**

```
<input type="text" class="form-control" id="idade" name="idade"
       [ngModel]="pessoa.idade"
       #idade="ngModel"
       numerico
       required>
```

*"selector" definido na diretiva*

- O campo agora só aceita dígitos



## Diretiva de Validação

- Mas só altera o dado que está na tela
- Deve-se agora sincronizar o *Model* com o dado alterado na tela
- Usar:
  - **ControlValueAccessor**: Uma ponte entre os formulários do Angular e os elementos nativos do DOM



## Diretiva de Validação

- No arquivo da diretiva
  - `src/app/shared/directives/numerico.directive.ts`
- Importar:

```
import { ControlValueAccessor, NG_VALUE_ACCESSOR } from '@angular/forms';
```

- Registrar como um *provider* na anotação `@Directive`

```
@Directive({
  selector: '[numerico]',
  providers: [
    {
      provide: NG_VALUE_ACCESSOR,
      useExisting: NumericoDirective,
      multi: true
    }
])

```

Prof. Dr. Razer A N R Montaño

Angular

451

451



## Diretiva de Validação

- Na classe da diretiva
  - `src/app/shared/directives/numerico.directive.ts`
  - Implementar a interface `ControlValueAccessor`

```
export class NumericoDirective implements ControlValueAccessor {
```

- Adicionar dois atributos auxiliares

```
  onChange: any;
  onTouched: any;
```

- Implementar o construtor para injetar um `ElementRef` (referência a um elemento HTML da tela, não esqueça de importar de `@angular/core`)

```
  constructor(private el: ElementRef) { }
```

Prof. Dr. Razer A N R Montaño

Angular

452

452



## Diretiva de Validação

- Implementar os métodos da Interface, dentro da classe **NumericoDirective**

```
registerOnChange(fn: any): void {
  this.onChange = fn;
}
registerOnTouched(fn: any): void {
  this.onTouched = fn;
}
writeValue(value: any): void {
  this.el.nativeElement.value = value;
}
```

Prof. Dr. Razer A N R Montaño

Angular

453

453



## Diretiva de Validação.

- Invocar o método `this.onChange()` dentro do *listener* do evento

```
@HostListener('keyup', ['$event'])
onKeyUp($event: any) {
  let valor = $event.target.value;

  // expressão regular: remove tudo que não é número
  valor = valor.replace(/[^D]/g, '');

  $event.target.value = valor;
  // atualiza o model
  this.onChange(valor); ← Atualização do Model
}
```

Prof. Dr. Razer A N R Montaño

Angular

454

454

2

## Diretiva para Validação de Valor de Idade

Prof. Dr. Razer A N R Montaño

Angular

455

455



### Validação de Idade

- Deseja-se que o valor válido da idade seja  $\geq 18$
- Será criada uma diretiva que efetua a validação
- Executar

```
$ ng g directive shared/directives/minimo-validator
```

- Criou os arquivos:
  - `src/app/shared/directives/minimo-validator.directive.ts`
  - `src/app/shared/directives/minimo-validator.directive.spec.ts`

Prof. Dr. Razer A N R Montaño

Angular

456

456



## Validação de Idade

- Dentro de: `src/app/shared/directives/minimo-validator.directive.ts`

```
import { Validator, NG_VALIDATORS, FormControl } from '@angular/forms';
import { Directive, OnInit } from '@angular/core';

@Directive({
  selector: '[minimoValidator]',
  providers: [{  
    provide: NG_VALIDATORS,  
    useExisting: MinimoValidatorDirective,  
    multi: true
  }]
})
export class MinimoValidatorDirective implements Validator, OnInit {
```

Acertar as importações

Acertar o nome do atributo

e

Registrar o provider

Acertar as implementações  
de interface

Prof. Dr. Razer A N R Montaño

Angular

457

457



## Validação de Idade

```
constructor() { }
ngOnInit() { }

validate(c: FormControl) {
  let v: number = +c.value;
  if (isNaN(v)) {
    return { 'minimo': true, 'requiredValue': 18 }
  }
  if (v < 18) {
    return { 'minimo': true, 'requiredValue': 18 }
  }
  return null;
}
```

Implementar o validate()

Se der tudo certo, retorna  
NULL

Se der erro, retorna o  
nome do erro (minimo) que  
será usado para verificar se  
houve erro

Prof. Dr. Razer A N R Montaño

Angular

458

458



## Validação de Idade

- Acertar o HTML para a validação
  - `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.html`
  - `src/app/pessoa/editar-pessoa/editar-pessoa.component.html`

```

<input type="text" class="form-control" id="idade" name="idade"
       [(ngModel)]="pessoa.idade"
       #idade="ngModel"
       numero
       minimoValidator
       required>
<div *ngIf="idade.errors && (idade.dirty || idade.touched)"
      class="alert alert-danger">
    <div [hidden]="!idade.errors?.['required']">
      Digite a idade da pessoa.
    </div>
    <div [hidden]="!idade.errors?.['minimo']"> ←
      Idade deve ser maior ou igual a 18
    </div>
  </div>

```

*Adiciona o atributo do validador*

*Adiciona a mensagem de erro, se houver*

Prof. Dr. Razer A N R Montaño

Angular

459

459



## Validação de Idade

- Adicionar as diretivas no módulo Pessoa
  - `src/app/pessoa/pessoa.module.ts`

```

import { NumericoDirective } from './shared/directives/numerico.directive';
import { MinimoValidatorDirective } from './shared/directives/minimo-
validator.directive';

```

- Dentro de **declarations** do **@NgModule**

```

@NgModule({
  declarations: [
    ListarPessoaComponent,
    InserirPessoaComponent,
    EditarPessoaComponent,
    NumericoDirective,
    MinimoValidatorDirective
  ],

```

Prof. Dr. Razer A N R Montaño

Angular

460

460



## Validação de Idade

- Pode-se passar um parâmetro para a diretiva
  - Torna a validação mais genérica
- Adicionar um atributo no HTML: ex, **valorMinimo="18"**

```
<input type="text" class="form-control" id="idade" name="idade"
       [(ngModel)]="pessoa.idade"
       #idade="ngModel"
       numero
       minimoValidator
       valorMinimo="18"
       required>
```

Prof. Dr. Razer A N R Montaño

Angular

461

461



## Validação de Idade

- Na diretiva de validação, adicionar um atributo ligado à **valorMinimo**

```
export class MinimoValidatorDirective implements Validator, OnInit {
  @Input("valorMinimo") valorMinimo: string = "0";
  constructor() { }
```

Prof. Dr. Razer A N R Montaño

Angular

462

462



## Validação de Idade

- Alterar o método de validação para usar o atributo

```
validate(c: FormControl) {  
  let v: number = +c.value;  
  if (isNaN(v)) {  
    return { 'minimo': true, 'requiredValue': +this.valorMinimo }  
  }  
  if (v < +this.valorMinimo) {  
    return { 'minimo': true, 'requiredValue': +this.valorMinimo }  
  }  
  return null;  
}
```



## Validação de Idade.

- Atualizar os formulários de inserção e atualização
- Testar!!!!

3

## Máscara de Data

Prof. Dr. Razer A N R Montaño

Angular

465

465



## Máscara de Data

- Primeiramente adicionar data de nascimento no model
- Nestes exemplos a data será representada como **string**
  - Mas há o tipo **Date** no Javascript
- Adicionar em *src/app/shared/models/pessoa.model.ts*

```
public dataNascimento?: string
```

Prof. Dr. Razer A N R Montaño

Angular

466

466



## Máscara de Data

- Em `src/app/pessoa/editar-pessoa/editar-pessoa.component.html`
- Em `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.html`
  - Adicionar o campo de data de nascimento

```
<div class="form-group">
    <label for="data">Data Nascimento:</label>
    <input type="text" class="form-control" id="data" name="data"
        [(ngModel)]="pessoa.dataNascimento"
        #dataNascimento="ngModel"
        required>
    <div *ngIf="dataNascimento.errors && (dataNascimento.dirty || dataNascimento.touched)">
        class="alert alert-danger">
            <div [hidden]="!dataNascimento.errors?.['required']">
                Digite a data de nascimento da pessoa.
            </div>
    </div>
</div>
</div>
```



## Máscara de Data

- Instalar o `ngx-mask`
  - <https://github.com/JsDaddy/ngx-mask>

```
$ npm i --save ngx-mask
```



## Máscara de Data

- Adicionar no módulo: `src/app/pessoa/pessoa.module.ts`

```
import { NgxMaskModule, IConfig } from 'ngx-mask'
export const options: Partial<IConfig> | ((() => Partial<IConfig>) = {});
```

- E no `imports` do `@NgModule`, adicionar:

```
@NgModule({
  imports: [
    NgxMaskModule.forRoot()
  ],
})
```



## Máscara de Data

- No HTML
  - `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.html`
  - `src/app/pessoa/editar-pessoa/editar-pessoa.component.html`
  - Adicionar um campo com a máscara:

```
<input type="text" class="form-control" id="data" name="data"
       [(ngModel)]="pessoa.dataNascimento"
       #dataNascimento="ngModel"
       mask="d0/M0/0000"
       required>
```



## Máscara de Data

- No HTML
  - `src/app/pessoa/listar-pessoa/listar-pessoa.component.html`
  - Adicionar mais uma coluna
  - Adicionar a máscara como um *Pipe*, para formatar a data ao mostrar:

```
<td> {{pessoa.dataNascimento! | mask: 'd0/M0/0000'}} </td>
```

Prof. Dr. Razer A N R Montaño

Angular

471

471



## Máscara de Data.

- Atualizar a aplicação
- Testar!!!!

Prof. Dr. Razer A N R Montaño

Angular

472

472

4

## Pipes

Prof. Dr. Razer A N R Montaño

Angular

473

473



## Pipes

- Um Pipe é um elemento do Angular usado para transformação de dados

```
<td> {{pessoa.dataNascimento | mask: 'd0/M0/0000'}} </td>
```

- Alguns Pipes disponíveis

- **DatePipe**
- **UpperCasePipe**
- **LowerCasePipe**
- **CurrencyPipe**
- **DecimalPipe**
- **PercentPipe**

- Podem ser encadeados mais de um

```
 {{ birthday | date | uppercase}}
```

Prof. Dr. Razer A N R Montaño

Angular

474

474



## Pipes

- Pode-se construir seu próprio Pipe

```
$ ng g pipe shared/pipes/meu-pipe
```

- Gera o código: *src/app/shared/pipes/meu-pipe.pipe.ts*

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'meuPipe'
})
export class MeuPipePipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

475

475



## Pipes

- Faz-se a customização: ex, colocar em caixa alta

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'caixaAlta'
})
export class MeuPipePipe implements PipeTransform {

  transform(value: string | undefined): string {
    if (value)
      return value.toUpperCase();
    else
      return "";
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

476

476



## Pipes

- Importar no módulo em `src/app/pessoa/pessoa.module.ts`
  - Se ele foi automaticamente importado para `src/app/app.module.ts`, pode remover e inserir em `src/app/pessoa/pessoa.module.ts`

```
import { MeuPipePipe } from './shared/pipes/meu-pipe.pipe';
...

@NgModule({
  declarations: [
    ...
    MeuPipePipe
  ],
  ...
})
```

Prof. Dr. Razer A N R Montaño

Angular

477

477



## Pipes

- Adicionar no campo desejado

```
<tr *ngFor="let pessoa of pessoas">

  <td> {{pessoa.nome! | caixaAlta }} </td>

  <td> {{pessoa.idade}} </td>
```

Prof. Dr. Razer A N R Montaño

Angular

478

478



## Pipes.

- Agora os nomes na listagem estarão sendo mostrados em caixa alta.

Cadastro de Pessoas

### Pessoas

Nome	Idade	Data Nascimento		
RAZER	20	10/10/2010	<input checked="" type="button"/> Editar	<input type="button"/> Remover
BRUNNA	52	10/10/2010	<input checked="" type="button"/> Editar	<input type="button"/> Remover
GUILHERME	22	1/1/2001	<input checked="" type="button"/> Editar	<input type="button"/> Remover
JUAN	88	04/04/2004	<input checked="" type="button"/> Editar	<input type="button"/> Remover

Prof. Dr. Razer A N R Montaño

Angular

479

479

5

## Organização do Código

Prof. Dr. Razer A N R Montaño

Angular

480

480



## Organização

- Antes de continuar, vamos organizar o módulo **shared**
  - Usar *barrel files* para facilitar as importações
  - Criar o módulo
  - Acertar as importações nos outros módulos

Prof. Dr. Razer A N R Montaño

Angular

481

481



## Organização: barrel files

- Usar *barrel files*
- Dentro de **src/app/shared/directives**
  - Criar arquivo **index.ts** com o conteúdo

```
export * from './minimo-validator.directive';
export * from './numerico.directive';
```

Prof. Dr. Razer A N R Montaño

Angular

482

482



## Organização: barrel files

- Usar *barrel files*
- Dentro de *src/app/shared/models*
  - Criar arquivo *index.ts* com o conteúdo

```
export * from './pessoa.model';
```

Prof. Dr. Razer A N R Montaño

Angular

483

483



## Organização: barrel files

- Usar *barrel files*
- Dentro de *src/app/shared/pipes*
  - Criar arquivo *index.ts* com o conteúdo

```
export * from './meu-pipe.pipe';
```

Prof. Dr. Razer A N R Montaño

Angular

484

484



## Organização: SharedModule

- Criar módulo **SharedModule**

```
$ ng generate module shared
```

- No arquivo *src/app/shared/shared.module.ts* terão as declarações das Diretivas e Pipes
  - Todas as **diretivas e pipes** serão fornecidas por este módulo
  - **LEMBRETE:** Um elemento só pode estar declarado em somente um módulo
  - Se eles estiverem dentro de outras seções **declarations** de outros **@NgModule**, retirar, devem ficar só aqui

Prof. Dr. Razer A N R Montaño

Angular

485

485



## Organização: SharedModule

- Usar *barrel files* no **SharedModule**
- Dentro de *src/app/shared*
  - Criar arquivo *index.ts* com o conteúdo abaixo
  - Os Pipes, Diretivas serão declaradas no **SharedModule**

```
// Models
export * from './models';

// SharedModule
export * from './shared.module';
```

Prof. Dr. Razer A N R Montaño

Angular

486

486



## Organização

- O arquivo: `src/app/shared/shared.module.ts` contém:

```
import { NgModule } from '@angular/core';
// Diretivas
import { NumericoDirective, MinimoValidatorDirective } from './directives';
// Pipes
import { MeuPipePipe } from './pipes';

@NgModule({
  declarations: [
    MinimoValidatorDirective,
    NumericoDirective,
    MeuPipePipe
  ],
  exports: [
    MinimoValidatorDirective,
    NumericoDirective,
    MeuPipePipe
  ]
})
export class SharedModule {}
```



## Organização

- Acertar as importações, no arquivo `src/app/app.module.ts`

```
import { SharedModule } from './shared';
```

- Dentro de `@NgModule`, no atributo `imports`, adicionar

```
 SharedModule
```



## Organização

- Acertar as importações, no arquivo *src/app/pessoa/pessoa.module.ts*

```
import { SharedModule } from './shared';
```

- Dentro de **@NgModule**, no atributo **imports**, adicionar

**SharedModule**



## Organização

- Acertar as importações, no arquivo *src/app/pessoa/pessoa.module.ts*
- Remover as importações das **diretivas e pipes**
  - Das importações
  - Dos **declarations** dentro de **@NgModule**



## Organização..

- As importações de modelos:
  - `src/app/pessoa/editar-pessoa/editar-pessoa.component.ts`
  - `src/app/pessoa/inserir-pessoa/inserir-pessoa.component.ts`
  - `src/app/pessoa/listar-pessoa/listar-pessoa.component.ts`

- Trocar de:

```
import { Pessoa } from '.../.../shared/models/pessoa.model';
```

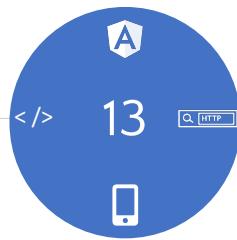
- Para:

```
import { Pessoa } from 'src/app/shared';
```



## EXERCÍCIOS..

1. Aplicar as validações mostradas ao CRUD de Pessoa
2. Transfira as diretivas para o diretório `shared`, se já não estiverem lá. Arrume as importações, coloque `barrel files` e crie o `SharedModule`.
3. Aplicar as seguintes validações no CRUD de Endereço:
  - Rua (String) – Obrigatório
  - Número (Numérico) – Obrigatório e Aceitar somente números
  - Complemento (String) – Opcional
  - Bairro (String) – Obrigatório
  - CEP (String) – Obrigatório e Máscara (00000-000)
  - Cidade (String) – Obrigatório
  - Estado (String) – Obrigatório
4. Aplicar as seguintes validações no CRUD de Cidade:
  - Nome (String) – Obrigatório
  - Estado (String) – Obrigatório
5. Aplicar as seguintes validações no CRUD de Estado:
  - Nome (String) – Obrigatório
  - Sigla (String) – Obrigatório, exatamente 2 letras



# Modal para Mostrar Pessoa

Prof. Dr. Razer A N R Montaño

Angular

493

493



## Modal

### ➡ Objetivos

- ✓ Criar uma janela Modal para mostrar Pessoa

### ➡ Referências

- <https://ng-bootstrap.github.io/#/home>

Prof. Dr. Razer A N R Montaño

Angular

494

494



## Modal

- Instalar **jQuery**

```
$ npm install --save jquery
```

- Instalar **Ng-Bootstrap**

- Componentes Angular usando Bootstrap 4

```
$ npm install --save @ng-bootstrap/ng-bootstrap
```

- Se der erro de versão, adicionar **--legacy-peer-deps**

- Instalar **PopperJS**

- Biblioteca de posicionamento

```
$ npm install --save @popperjs/core
```

Prof. Dr. Razer A N R Montaño

Angular

495

495



## Modal

- Adicionar **jQuery** na aplicação (*angular.json*)

```
"styles": [  
  "src/styles.css",  
  "node_modules/bootstrap/dist/css/bootstrap.min.css",  
  "node_modules/@fortawesome/fontawesome-free/css/all.css"  
,  
  "scripts": [  
    "node_modules/jquery/dist/jquery.min.js",  
    "node_modules/bootstrap/dist/js/bootstrap.min.js"  
,  
  ]
```

Prof. Dr. Razer A N R Montaño

Angular

496

496



## Modal

- Criar o componente para a modal

```
$ ng generate component pessoa/modal-pessoa
```

- Gera os arquivos
  - *src/app/pessoa/modal-pessoa/modal-pessoa.css*
  - *src/app/pessoa/modal-pessoa/modal-pessoa.html*
  - *src/app/pessoa/modal-pessoa/modal-pessoa.spec.ts*
  - *src/app/pessoa/modal-pessoa/modal-pessoa.ts*



## Modal

- Alterar o HTML da modal
  - *src/app/pessoa/modal-pessoa/modal-pessoa.component.html*

```
<div class="modal-header">
  <button type="button" class="close" aria-label="Fechar"
    (click)="activeModal.dismiss()">
    <span aria-hidden="true">&times;/<span>
  </button>
  <h4 class="modal-title" id="modalLabel"> Pessoa </h4>
</div>
<div class="modal-body text-center">
  <div>
    <p>Nome: {{ pessoa.nome }} </p>
    <p>Idade: {{ pessoa.idade }}</p>
    <p>Data Nascimento: {{ pessoa.dataNascimento | mask: 'd0/M0/0000'}} </p>
  </div>
</div>
<div class="modal-footer">
  <button type="button" class="btn btn-outline-dark"
    (click)="activeModal.close()">Fechar</button>
</div>
```



## Modal

- Aqui tanto **close()** como **dismiss()** fecharão a modal
- Diferem no retorno (que aqui não está sendo usado):
  - **close(x)** : fecha modal passando um resultado
  - **dismiss(x)** : fecha modal passando uma razão
- O resultado das chamadas é um **promise**
  - Representa uma execução assíncrona, que pode ter sucesso ou falhar
  - No **close()**, o **promise** é resolvido (roda o código de sucesso)
  - No **dismiss()**, o **promise** é rejeitado (roda o código de falha)

Prof. Dr. Razer A N R Montaño

Angular

499

499



## Modal

- Alterar o componente da modal
  - [src/app/pessoa/modal-pessoa/modal-pessoa.component.ts](#)

```
import { Component, OnInit, Input } from '@angular/core';
import { Pessoa } from 'src/app/shared';

import { NgbActiveModal } from '@ng-bootstrap/ng-bootstrap';

@Component({
  selector: 'modal-pessoa',
  templateUrl: './modal-pessoa.component.html',
  styleUrls: ['./modal-pessoa.component.css']
})
export class ModalPessoaComponent {
  @Input() pessoa!: Pessoa;

  constructor(public activeModal: NgbActiveModal) {}

  ngOnInit(): void {}

}
```

Prof. Dr. Razer A N R Montaño

Angular

500

500



## Modal

- Alterar componente de listagem de pessoa para abrir a modal
  - `src/app/pessoa/listar-pessoa/listar-pessoa.component.ts`

```
import { NgbModal } from '@ng-bootstrap/ng-bootstrap';
import { ModalPessoaComponent } from './modal-pessoa/modal-pessoa.component';

...
constructor(private pessoaService : PessoaService,
            private modalService: NgbModal) { }

...
abrirModalPessoa(pessoa: Pessoa) {
  const modalRef = this.modalService.open(ModalPessoaComponent);
  modalRef.componentInstance.pessoa = pessoa;
}
```

Prof. Dr. Razer A N R Montaño

Angular

501

501



## Modal..

- Alterar o HTML de listagem de pessoa: adicionar um botão antes do *link*  
Editar para abrir a modal
  - `src/app/pessoa/listar-pessoa/listar-pessoa.component.html`

```
<button type="button" class="btn btn-xs btn-info"
        (click)="abrirModalPessoa(pessoa) " >
  <i class="fa fa-eye" aria-hidden="true"></i> Ver
</button>
```

Prof. Dr. Razer A N R Montaño

Angular

502

502



## EXERCÍCIOS..

1. Implementar a modal no CRUD de Pessoa
2. Implementar a modal no CRUD de Endereço
3. Implementar a modal no CRUD de Cidade
4. Implementar a modal no CRUD de Estado

Prof. Dr. Razer A N R Montaño

Angular

503

503



## Menu para os Formulários

Prof. Dr. Razer A N R Montaño

Angular

504

504

 **Menu**

 **Objetivos**

- ✓ Criar um Menu para os 4 formulários criados
- ✓ Usar um *dropdown* menu

 **Referências**

➤ <https://angular.io>

Prof. Dr. Razer A N R Montaño Angular 505

505

 **Menu**

- Objetivo é este Menu

 **Pessoas** Endereços Cidades Estados Cadastros ▾

**Pessoas**

Nome	Idade	Data
RAZER	20	10/10

*Icone.  
Ao pressionar vai para a rota "/pessoas"*

*Links.  
Ao pressionar vão para suas rotas específicas*

*Menu Dropdown.  
Ao pressionar abre várias opções:  
Pessoas, Endereços, Cidades, Estados  
Cada um vai para sua rota*

Prof. Dr. Razer A N R Montaño Angular 506

506



## Menu

- Atualizar o HTML
  - Arquivo `src/app/app.component.html`
  - Adicionar um `<nav>` configurado para apresentar um menu
  - Adicionar o link com ícone
  - Adicionar os links para as rotas
    - Atributos `routerLink` dos `<a>`
  - Adicionar um menu `dropdown`
    - Usar o `ngbDropdown` e auxiliares
    - Adicionar os links
- Importar `NgbModule` em `src/app/app.module.ts`

Prof. Dr. Razer A N R Montaño

Angular

507

507



## Menu

- Diretiva `ngbDropdown`
  - Disponibilizada no pacote `ng-bootstrap`
  - Portanto já deve estar instalada
- Se precisar instalar

```
$ npm install --save jquery
$ npm install --save --legacy-peer-deps @ng-bootstrap/ng-bootstrap
$ npm install --save @popperjs/core
```

Prof. Dr. Razer A N R Montaño

Angular

508

508



## Menu: Atualizar o HTML

- Arquivo `src/app/app.component.html`

```
<div class="conteiner-fluid">

  <nav class="navbar navbar-default">
    <div class="conteiner-fluid">
      <div class="navbar-header">
        <a class="navbar-brand" href="#">
          {{title}}
        </a>
      </div>
    </div>
  </nav>

  <router-outlet></router-outlet>
</div>
```



Remover este trecho



## Menu: Atualizar o HTML

```
<div class="conteiner-fluid">

  <router-outlet></router-outlet>
</div>
```

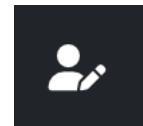


Adicionar aqui o novo Menu



## Menu: Atualizar o HTML

```
<nav class="navbar navbar-expand-md n navbar-dark bg-dark">
  <a class="navbar-brand nav-link" href="#" routerLink="/pessoas">
    <i class="fa fa-user-edit"></i>
  </a>
```



Prof. Dr. Razer A N R Montaño

Angular

511

511



## Menu: Atualizar o HTML

```
<div class="navbar-collapse collapse" id="navbarContent">
  <ul class="navbar-nav ml-auto">
    <li class="nav-item">
      <a class="nav-link" href="#" routerLink="/pessoas">Pessoas</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#" routerLink="/enderecos">Endereços</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#" routerLink="/cidades">Cidades</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#" routerLink="/estados">Estados</a>
    </li>
  </ul>
```

Pessoas Endereços Cidades Estados

Prof. Dr. Razer A N R Montaño

Angular

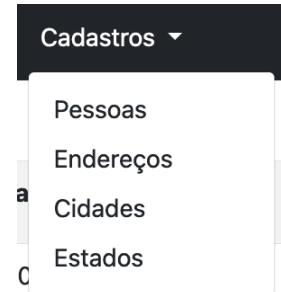
512

512



## Menu: Atualizar o HTML

```
<li class="nav-item" ngbDropdown>
  <a class="nav-link" tabindex="0" ngbDropdownToggle id="navbarDropdown1"
     role="button"> Cadastros </a>
  <div ngbDropdownMenu aria-labelledby="navbarDropdown1"
       class="dropdown-menu">
    <a ngbDropdownItem href="#" routerLink="/pessoas"
       (click)="event.preventDefault()">Pessoas</a>
    <a ngbDropdownItem href="#" routerLink="/enderecos"
       (click)="event.preventDefault()">Endereços</a>
    <a ngbDropdownItem href="#" routerLink="/cidades"
       (click)="event.preventDefault()">Cidades</a>
    <a ngbDropdownItem href="#" routerLink="/estados"
       (click)="event.preventDefault()">Estados</a>
  </div>
</li>
</ul>
</div>
</nav>
```



## Menu: Importar NgbModule

- No arquivo `src/app/app.module.ts`

```
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
```

- Em `imports` da anotação `@NgModule`
  - Adicionar `NgbModule`



## Menu: Testar..

- Testar a aplicação

Nome	Idade	Data Nascimento	Pessoas	Endereços	Cidades	Estados
PESSOA ESQUISITA	22	20/2/2000	+ Novo			
RAZER	33	25/06/1975	Ver	Editar	Remover	

Prof. Dr. Razer A N R Montaño

Angular

515



## EXERCÍCIOS..

1. Implementar a nova barra de menus na aplicação

Prof. Dr. Razer A N R Montaño

Angular

516

516



# Outros Elementos de Formulários

Prof. Dr. Razer A N R Montaño

Angular

517

517



## Outros Elementos de Formulários

### ☰ Objetivos

- ✓ Usar outros componentes de formulários
- ✓ Check Box, Radio Button, Combo Box
- ✓ Cidade x Estado



### Referências

- <https://angular.io>
- <https://github.com/ng-select/ng-select>

Prof. Dr. Razer A N R Montaño

Angular

518

518



## Outros Elementos de Formulários.

- Para continuar aqui deve-se ter os módulos criados: Endereço, Cidade, Estado
- Exemplo: para criação do módulo Endereço:

```
$ ng g module endereco
$ ng g component endereco/editar-endereco
$ ng g component endereco/inserir-endereco
$ ng g component endereco/listar-endereco
$ ng g component endereco/modal-endereco
$ ng g service endereco/services/endereco
$ ng g class shared/models/endereco --type=model
```

- Deve-se então implementar estes módulos
- Acertar os *barrel files*

1

## Checkbox



## Checkbox

- Adicionar um Checkbox no endereço, indicando se é residencial
- Na model (*src/app/shared/models/endereco.model.ts*)

```
export class Endereco {
    constructor(
        public id?: number,
        public rua?: string,
        public numero: number = 0,
        public complemento?: string,
        public bairro?: string,
        public cep?: string,
        public cidade?: string,
        public estado?: string,
        public residencial?: boolean) {
    }
}
```



## Checkbox

- Na listagem (*src/app/endereco/listar-endereco/listar-endereco.component.html*)

```
<td> {{endereco.residencial! ? "Residencial" : "Comercial"}} </td>
```



## Checkbox

- Na inserção (*src/app/endereco/inserir-endereco/inserir-endereco.component.html*)

```
<div class="form-group">
    <label for="residencial">Residencial:</label>
    <input type="checkbox" class="form-check-input"
        id="residencial" name="residencial"
        [(ngModel)]="endereco.residencial"
        #residencial="ngModel">
</div>
```



## Checkbox

- Na edição (*src/app/endereco/editar-endereco/editar-endereco.component.html*)

```
<div class="form-group">
    <label for="residencial">Residencial:</label>
    <input type="checkbox" class="form-check-input"
        id="residencial" name="residencial"
        [(ngModel)]="endereco.residencial"
        #residencial="ngModel">
</div>
```



## Checkbox.

- Na Modal (*src/app/endereco/modal-endereco/modal-endereco.component.html*)

```
<p>{{ endereco.residencial ? "Residencial" : "Comercial" }} </p>
```

2

## Radio Button



## Radio Button

- Adicionar um botão de rádio na Pessoa, indicando se tem e o tipo da carteira de motorista
- Na model (*src/app/shared/models/pessoa.model.ts*)

```
export class Pessoa {  
    constructor(  
        public id?: number,  
        public nome?: string,  
        public idade: number = 0,  
        public dataNascimento?: string,  
        public motorista?: string) {  
    }  
}
```



## Radio Button

- Na listagem (*src/app/pessoa/listar-pessoa/listar-pessoa.component.html*)

```
<td> {{pessoa.motorista! ? pessoa.motorista : "Não" }} </td>
```



## RadioButton

- Na inserção (*src/app/pessoa/listar-pessoa/inserir-pessoa.component.html*)

```
<div class="form-group">
    <label for="motorista">Cart. Motorista:</label>
    <input type="radio" class="form-check-input" id="motorista1"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista1="ngModel" value="Não"> Não <br/>
    <input type="radio" class="form-check-input" id="motorista2"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista2="ngModel" value="A"> A <br/>
    <input type="radio" class="form-check-input" id="motorista3"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista3="ngModel" value="B"> B <br/>
    <input type="radio" class="form-check-input" id="motorista4"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista4="ngModel" value="AB"> AB <br/>
</div>
```



## RadioButton

- Na edição (*src/app/pessoa/listar-pessoa/editar-pessoa.component.html*)

```
<div class="form-group">
    <label for="motorista">Cart. Motorista:</label>
    <input type="radio" class="form-check-input" id="motorista1"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista1="ngModel" value="Não"> Não <br/>
    <input type="radio" class="form-check-input" id="motorista2"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista2="ngModel" value="A"> A <br/>
    <input type="radio" class="form-check-input" id="motorista3"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista3="ngModel" value="B"> B <br/>
    <input type="radio" class="form-check-input" id="motorista4"
        name="motorista" [(ngModel)]="pessoa.motorista"
        #motorista4="ngModel" value="AB"> AB <br/>
</div>
```



## RadioButton.

- Na Modal (*src/app/pessoa/listar-pessoa/modal-pessoa.component.html*)

```
<p>Cart. Motorista: {{pessoa.motorista! ? pessoa.motorista : "Não" }} </p>
```

Prof. Dr. Razer A N R Montaño

Angular

531

531

3

## ComboBox

Prof. Dr. Razer A N R Montaño

Angular

532

532



## Combo Box

- Adicionar uma Combo Box em Cidade para selecionar o estado
- Vamos usar um componente: **ng-select**
  - <https://github.com/ng-select/ng-select>

```
$ npm install --save @ng-select/ng-select
```

Prof. Dr. Razer A N R Montaño

Angular

533

533



## Combo Box

- Importar no módulo principal `src/app/app.module.ts`)

```
import { NgSelectModule } from '@ng-select/ng-select';
...
@NgModule({
  ...
  imports: [
    ...
    NgSelectModule
  ]
})
export class AppModule { }
```

Prof. Dr. Razer A N R Montaño

Angular

534

534



## Combo Box

- Adicionar no **CidadeModule** (*src/app/cidade/cidade.module.ts*)

```
import { NgSelectModule } from '@ng-select/ng-select';
...
@NgModule({
  ...
  imports: [
    ...
    NgSelectModule
  ]
})
export class CidadeModule { }
```



## Combo Box

- Adicionar o estilo em *src/styles.css*

```
@import "~@ng-select/ng-select/themes/default.theme.css";
```



## Combo Box

- Na model Cidade (*src/app/shared/models/cidade.model.ts*), adiciona o estado como um objeto do tipo **Estado**

```
import { Estado } from ".";

export class Cidade {
    constructor(
        public id?: number,
        public nome?: string,
        public estado?: Estado) {}
}
```



## Combo Box

- Na listagem de cidade (*src/app/cidade/listar-cidade/listar-cidade.component.html*), mostrar o sigla de dentro do objeto **Estado**
  - Usar a notação "?" para tratar objeto indefinido

```
<td> {{cidade?.estado?.sigla}} </td>
```



## Combo Box

- Na inserção de cidade ([src/app/cidade/inserir-cidade/inserir-cidade.component.html](#)), trocar a entrada de estado pelo **ng-select**:

```
<ng-select name="estado" [items]="estados"
           bindLabel="sigla"
           [(ngModel)]="cidade.estado"
           placeholder="Selecione um estado">
</ng-select>
```

Prof. Dr. Razer A N R Montaño

Angular

539

539



## Combo Box

- Na inserção de cidade ([src/app/cidade/inserir-cidade/inserir-cidade.component.html](#)), trocar a entrada de estado pelo **ng-select**:

```
<ng-select name="estado" [items]="estados"
           bindLabel="sigla"
           [(ngModel)]="cidade.estado"
           placeholder="Selecione um estado">
</ng-select>
```

Itens a serem apresentados no combo box  
Atributo "estados" do componente (classe no .ts)

O que vai ser apresentado na lista da combo box  
Atributo "sigla" de cada elemento que está dentro de "estados"

Onde será armazenado o estado selecionado  
Atributo "cidade.estado" do componente (classe no .ts)

Prof. Dr. Razer A N R Montaño

Angular

540

540



## Combo Box

- Inserção de cidade ([src/app/cidade/inserir-cidade/inserir-cidade.component.ts](#)), adicionar o suporte à lista de estados

```
export class InserirCidadeComponent implements OnInit {
  @ViewChild('formCidade') formCidade!: NgForm;
  cidade: Cidade = new Cidade();
  estados: Estado[] = [];

  constructor(
    private cidadeService: CidadeService,
    private estadoService: EstadoService,
    private router: Router) { }

  ngOnInit(): void {
    this.cidade = new Cidade();
    this.estados = this.estadoService.listarTodos();
  }
  inserir(): void {
    if (this.formCidade.form.valid) {
      this.cidadeService.inserir(this.cidade);
      this.router.navigate( ['/cidades' ] );
    }
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

541

541



## Combo Box

- Na modal ([src/app/cidade/modal-cidade/modal-cidade.component.html](#)), mostrar a sigla do objeto

```
<p>Estado: {{ cidade?.estado?.sigla }}</p>
```

Prof. Dr. Razer A N R Montaño

Angular

542

542



## Combo Box

- Na edição de cidade (*src/app/cidade/editar-cidade/editar-cidade.component.html*), trocar a entrada de estado pelo **ng-select**:

```
<ng-select name="estado" [items]="estados"
           bindLabel="sigla"
           [(ngModel)]="cidade.estado"
           placeholder="Selecione um estado">
</ng-select>
```

Prof. Dr. Razer A N R Montaño

Angular

543

543



## Combo Box..

- Na edição de cidade (*src/app/cidade/editar-cidade/editar-cidade.component.ts*), adicionar o suporte à lista de estados
  - Adicionar atributo  
**estados: Estado[]**
  - Injetar no construtor o  
**EstadoService**
  - No **ngOnInit()** inicializar a lista de estados com todos os estados
  - Acertar importações

Prof. Dr. Razer A N R Montaño

Angular

544

544



## EXERCÍCIOS..

1. Implementar as alterações apresentadas
2. Adicionar um dado "fumante" (checkbox) no cadastro de Pessoa
3. Adicionar um dado "tipo" (radio button) no cadastro de Endereço, para armazenar : "Avenida", "Rua", "Travessa"
4. Adicionar uma combo box no cadastro de Endereço, para escolher uma Cidade da base
  - a. Na listagem de Endereço deve mostrar a cidade e a sigla do estado



## Programação Reativa



## Programação Reativa

### Objetivos

- ✓ Apresentar RxJS
- ✓ Entender Observables e Subjects
- ✓ Entender operadores
- ✓ Exemplos



### Referências

➤ <https://rxjs-dev.firebaseio.com>

Prof. Dr. Razer A N R Montaño

Angular

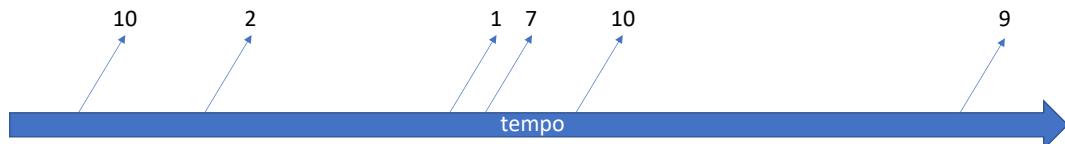
547

547



## Programação Reativa

- Imagine um elemento que gera valores ao longo do tempo



- Este componente gera/emite um fluxo assíncrono de dados
- Como consumir esses valores????
  - A maneira mais comum : síncrona com **CALLBACKs**

Prof. Dr. Razer A N R Montaño

Angular

548

548



## Programação Reativa

- **Callback**

- Processo síncrono
- Implementar uma **função** no gerador do evento
- É executada quando um novo dado chega: **Callback**

- Problemas com as *Callbacks*

- Pode ser necessário "propagar"/"aninhar" callback:
- Perda do controle do que está sendo executado
- Código confuso
- **Callback Hell**

*Realtime Code implies ⇒ Asynchronous Code implies ⇒ Callbacks implies ⇒ Callback Hell implies ⇒ :( } } } }*

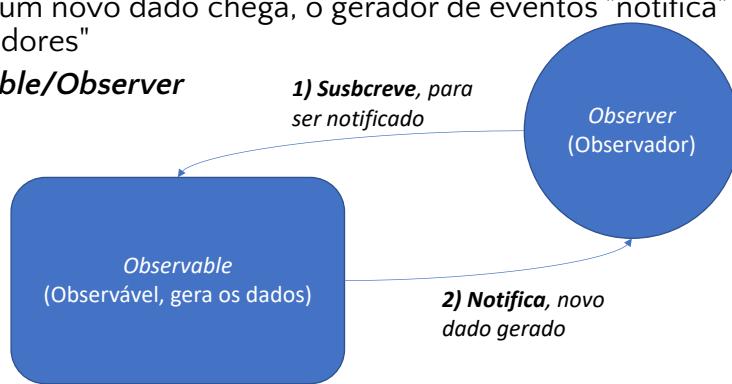


## Programação Reativa

- **Design Pattern Observer**

- Assíncrono
- Implementar um componente que "observa" o gerador de dados
- Quando um novo dado chega, o gerador de eventos "notifica" seus "observadores"

- **Observable/Observer**





## Programação Reativa

- Programar fazendo uso de fluxos assíncronos de dados
  - Um exemplo clássico: Chamada a API REST
  - Escreve-se o software para "reagir" à mudança nos dados
- **RxJS: Reactive Extensions for Javascript**
  - Biblioteca JS que leva a programação reativa para Web
  - Usada para escrever programas assíncronos
  - Baseada em eventos
- Fornece implementação do *design pattern Observer*
  - Tipos: **Observable**, **Subject**
- Fornece operadores para manipular os dados assíncronos
  - `map()`, `filter()`, `reduce()`, `take()`, `distinct()`, etc

Prof. Dr. Razer A N R Montaño

Angular

551

551



## Programação Reativa

- Assim, tem-se:
  - **Observable**: um objeto que pode **emitir eventos**, por exemplo, chegou uma nova ligação. É uma abstração de um **fluxo assíncrono de dados**
  - **Observer**: um objeto que é **notificado** sempre que um evento ocorre
    - Um *observer* precisa se **subscrever** no *observable*, para poder receber as notificações
- Outros conceitos:
  - **Operadores**: são funções aplicadas ao fluxo de dados
  - **Subject**: igual a um *Observable*, exceto que o mesmo dado é notificado a todos os *Observers*, assim que é emitido

Prof. Dr. Razer A N R Montaño

Angular

552

552



## Programação Reativa

- Um **Observable**
  - Pode emitir valores ao longo do tempo
  - Pode emitir valores de forma síncrona e assíncrona
- Com um objeto **Observable**
  - Pode-se subscrever nele para escutar todos os dados que ele transmite
  - Método **subscribe (função)**
- Exemplo:

```
meu_observable.subscribe(  
    valor => console.log('teste: ' + valor)  
);
```

Prof. Dr. Razer A N R Montaño

Angular

553

553



## Programação Reativa

- O método **subscribe ( observer )**
  - Recebe um *observer* como parâmetro
- Um *observer* possui 3 funções
  - **next**: função executada quando um valor é emitido
  - **error**: função executada quando houver erro
  - **complete**: função executada quando o processamento terminar
- Exemplo:

```
meu_observable.subscribe({  
    next: valor => console.log('teste: ' + valor),  
    error: err => console.error('Erro: ' + err),  
    complete: () => console.log('Terminou.')  
});
```

Prof. Dr. Razer A N R Montaño

Angular

554

554



## Programação Reativa.

- Quando passa somente uma função, assume **next**
- Exemplo:

```
meu_observable.subscribe(  
    valor => console.log('teste: ' + valor)  
) ;
```

Prof. Dr. Razer A N R Montaño

Angular

555

555



## Pipeline

- Pipeline de dados
  - Uma sequência de transformação dos dados
  - Tornar os dados mais legíveis
  - Filtrá-los
  - Aplicam-se operadores, que são funções
- Usa-se o método **pipe( operadores )**
- Exemplo:

```
meu_observable.pipe(  
    filter( valor => valor > 2) ,  
    map( valor => valor * 2 )  
) ;
```

Prof. Dr. Razer A N R Montaño

Angular

556

556



## Pipeline

- **filter()** : só emite os valores que satisfazem a condição

```
filter( valor => valor > 2),
```

- **map()** : aplica uma função a cada valor do fluxo e emite seu resultado

```
map( valor => valor * 2 )
```



## Pipeline.

- Assim:

```
meu_observable.pipe(  
    filter( valor => valor > 2),  
    map( valor => valor * 2 )  
).subscribe( x => console.log(x) );
```

- Vai obter os valores gerados por **meu\_observable**, considerar somente os que são **maiores que 2**, aplicar a função **\* 2** e só daí disponibilizar para o **subscribe**



## Operadores.

- Vários operadores estão disponíveis, são de dois tipos: Criacionais e de Pipe
- **Criacionais:** usados para criar novos *Observables*
  - `from()`: cria um *observable* a partir de um *array*
  - `of()`: cria um *observable* a partir de vários valores
  - `concat()`: concatena dois ou mais *observables*
- **De Pipe:** geram um novo *Observable* aplicando alguma transformação
  - `map()`: aplica uma função nos valores emitidos e emite esses resultados
  - `filter()`: só emite os valores que satisfazem uma condição
  - `distinct()`: só emite os valores que são diferentes dos já emitidos
  - `take(n)`: só emite os *n* primeiros elementos
  - `max() / min()`: emite o maior/menor valor
  - `find()`: emite somente o primeiro elemento que satisfaz uma condição
  - `count()`: emite a quantidade de elementos foram emitidos, quando o *observable* termina
- Ver: <https://rxjs-dev.firebaseio.com/guide/operators>



## Projeto de Exemplo

- Criar um projeto simples
  1. Criar um diretório com o arquivo de teste
  2. Instalar o rxjs
  3. Configurar o arquivo `package.json`
  4. Configurar o arquivo `tsconfig.json`
  5. Executar um exemplo



## Projeto de Exemplo

1. Criar o diretório com o arquivo de teste (**teste.ts**)

```
import { from } from 'rxjs';

from(["Curitiba", "SJP", "Pinhais", "Colombo"]).subscribe(
  valor => console.log(valor)
);
```

Prof. Dr. Razer A N R Montaño

Angular

561

561



## Projeto de Exemplo

2. Instalar o RxJS

```
$ npm install rxjs --save
```

- As importações se parecem com:

```
import { Observable, of, from } from 'rxjs';
```

Prof. Dr. Razer A N R Montaño

Angular

562

562



## Projeto de Exemplo

### 3. Criar o arquivo *package.json*

- Conteúdo:

```
{  
  "type": "module",  
  "dependencies": {  
    "rxjs": "^7.3.0"  
  }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

563

563



## Projeto de Exemplo

### 4. Configurar o arquivo *tsconfig.json*

- Criar o arquivo no mesmo diretório e adicionar:

```
{  
  "compilerOptions": {  
    "moduleResolution": "node",  
    "lib": ["es2016", "dom"],  
    "target": "es6"  
  }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

564

564



## Projeto de Exemplo.

### 5. Executar o exemplo

- Para transpilar o projeto

```
$ tsc -p tsconfig.json
```

- Para executar o código

```
$ node teste.js
```



## Exemplo: from()

- Criação de um Observable: **from()**

```
let arr = ["Ibicaré", "Joaçaba", "Treze Tílias", "Luzerna"]
from(arr).subscribe({
  next: valor => {
    let x = 'Cidade: ' + valor;
    console.log(x)
  },
  error: valor => console.log('Erro: ' + valor),
  complete: () => console.log(' acabou')
});
```



## Exemplo: from()

- Outra sintaxe do **from()**

```
let arr = ["Ibicaré", "Joaçaba", "Treze Tílias", "Luzerna"]
from(arr).subscribe({
  next(valor) {
    let x = 'Cidade: ' + valor;
    console.log(x);
  },
  error(valor) {
    console.log('Erro: ' + valor);
  },
  complete() {
    console.log(' acabou');
  }
});
```

Prof. Dr. Razer A N R Montaño

Angular

567

567



## Exemplo: of()

- Criação de um Observable: **of()**

```
of("Curitiba", 200, 4.5, true).subscribe({
  next: valor => {
    let x = 'Valor: ' + valor;
    console.log(x)
  },
  error: valor => console.log('Erro: ' + valor),
  complete: () => console.log(' acabou')
});
```

Prof. Dr. Razer A N R Montaño

Angular

568

568



## Exemplo: pipe()

- Filtro: `pipe(filter())`

```
import { from, of, filter } from 'rxjs';

let valores = [5, 10, 15, 20, 25, 30, 35, 40];

const obs1 = from(valores).pipe(
  filter(valor => valor % 2 == 0)
);
obs1.subscribe({
  next(valor) {
    console.log(valor);
  }
});
```

Gera um novo Observable  
Só emite os valores que atendem a condição de filter()

> Neste exemplo, só os valores pares

\$ 10 20 30 40

Prof. Dr. Razer A N R Montaño

Angular

569

569



## Exemplo: pipe()

- Filtro: `pipe(filter(), map())`

```
let valores = [5, 10, 15, 20, 25, 30, 35, 40];

const obs2 = from(valores).pipe(
  filter(valor => valor % 2 == 0),
  map(valor => valor / 10)
);
obs2.subscribe({
  next(valor) {
    console.log(valor);
  }
});
```

Gera um novo Observable  
1) Só emite os valores que atendem a condição de filter()  
2) Obtém o valor emitido e transforma pelo resultado de map()

> Neste exemplo, só os valores pares  
> Depois os valores pares são divididos por 10

\$ 1 2 3 4

Prof. Dr. Razer A N R Montaño

Angular

570

570



## Exemplo: pipe()

- Filtro: `pipe(filter(), map(), take())`

```
let valores = [5, 10, 15, 20, 25, 30, 35, 40];

const obs3 = from(valores).pipe(
  filter(valor => valor % 2 == 0),
  map(valor => valor / 10),
  take(2)
);

obs3.subscribe({
  next(valor) {
    console.log(valor);
  }
});
```

Gera um novo Observable  
 1) Só emite os valores que atendem a condição de filter()  
 2) Obtém o valor emitido e transforma pelo resultado de map()  
 3) Só emite os n primeiros valores, passados em take()

> Neste exemplo, só os valores pares  
 > Depois os valores pares são divididos por 10  
 > Depois emite só os 2 primeiros valores  
 \$ 1 2



## Exemplo: pipe()

- Filtro: `pipe(filter(), count())`

```
let valores = [5, 10, 15, 20, 25, 30, 35, 40];

const obs4 = from(valores).pipe(
  filter(valor => valor % 2 == 0),
  count()
);

obs4.subscribe({
  next(valor) {
    console.log(valor);
  }
});
```

Gera um novo Observable  
 1) Só emite os valores que atendem a condição de filter()  
 2) Conta a quantidade de valores emitidos e emite só esta quantidade

> Neste exemplo, só os valores pares  
 > Depois são contados e emite a quantidade  
 \$ 4



## Exemplo: pipe()

- Filtro: **pipe(count())**

```
let valores = [5, 10, 15, 20, 25, 30, 35, 40];

const obs5 = from(valores).pipe(
  count(valor => valor % 2 == 0)
);

obs5.subscribe({
  next(valor) {
    console.log(valor);
  }
});
```

Gera um novo Observable  
 1) Só emite a quantidade de valores que atendem à condição  
 > Neste exemplo, só os valores pares  
 \$ 4



## Exemplo: pipe()

- Filtro: **pipe(max())**

```
let valores = [5, 10, 15, 20, 25, 30, 35, 40];

const obs6 = from(valores).pipe(
  max()
);

obs6.subscribe({
  next(valor) {
    console.log(valor);
  }
});
```

Gera um novo Observable  
 1) Só emite o maior valor  
 > Neste exemplo é o 40  
 \$ 40



## Exemplo: concat()..

- Concatenação de *Observables*

```
const obs7 = concat(obs2, obs3, obs4);  
obs7.subscribe({  
    next(valor) {  
        console.log(valor);  
    }  
});
```

Gera um novo Observable da concatenação dos passados como parâmetro

- 1) Emite o observable obs2
- 2) Emite o observable obs3
- 3) Emite o observable obs4

\$ 1 2 3 4 1 2 4



## EXERCÍCIOS..

1. Implementar os exemplos de *Observables* apresentados



# Login

Prof. Dr. Razer A N R Montaño

Angular

577

577



## Login

### ➡ Objetivos

- ✓ Criar uma página de login
- ✓ Autenticar o usuário e manter o usuário logado
- ✓ Permitir acesso a páginas conforme o perfil do usuário



### ➡ Referências

- <https://angular.io>

Prof. Dr. Razer A N R Montaño

Angular

578

578



## Login

- Passos:

1. **Criar modelo de usuário:** dados de usuário como login, senha, nome e perfil
2. **Criar módulo de autenticação:** módulo que conterá toda a lógica de login
3. **Criar modelo para login:** dados da tela de login: login e senha
4. **Criar serviço de login:** validar o login/senha e retornar o usuário logado
5. **Criar componente de login:** tela de login e lógica de tela
6. **Criar a rota de login:** para poder invocar a tela de login
7. **Criar o serviço de guarda:** verificar se o usuário logado tem acesso ao componente
8. **Configurar as permissões das rotas:** indicar qual perfil tem acesso em cada rota
9. **Criar uma página principal:** Página Home, mostrada após o login
10. **Menu, Usuário logado e Logout:** alterar o componente principal para menu contextualizado, manter o usuário logado e efetuar logout

Prof. Dr. Razer A N R Montaño

Angular

579

579



## 1) Model de Usuário

- Criar a model de Usuário

```
$ ng g class shared/models/usuario --type=model
```

- Adicionar os atributos

```
export class Usuario {  
    constructor(  
        public id?: number,  
        public nome?: string,  
        public login?: string,  
        public senha?: string,  
        public perfil?: string  
    ) {}  
}
```

Prof. Dr. Razer A N R Montaño

Angular

580

580



## 1) Model de Usuário.

- O perfil do usuário, neste exemplo, poderá ser:
  - Administrador: "ADMIN"
  - Gerente: "GERENTE"
  - Funcionário: "FUNC"
- Arrumar os *barrel files* de models

Prof. Dr. Razer A N R Montaño

Angular

581

581



## 2) Módulo de Autenticação

- Criar um módulo de autenticação

**\$ ng g module auth**

- Em **AppModule** (*src/app/app.module.ts*)
  - Importar **AuthModule**
  - Adicionar **AuthModule** nos *imports* do **@NgModule**
- Em **AuthModule** (*src/app/auth/auth.module.ts*)
  - Importar e adicionar nos *imports* do **@NgModule**
  - **FormsModule**
  - **RouterModule**

Prof. Dr. Razer A N R Montaño

Angular

582

582



## 2) Módulo de Autenticação.

```
import { FormsModule } from '@angular/forms';
import { RouterModule } from '@angular/router';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

@NgModule({
  ...
  imports: [
    ...
    FormsModule,
    RouterModule,
    NgbModule,
    ...
  ]
})
export class AuthModule {}
```

Prof. Dr. Razer A N R Montaño

Angular

583

583



## 3) Model para Login.

- Criar um model para login

```
$ ng g class shared/models/login --type=model
```

- Armazena os dados da tela de login, digitado pelo usuário

```
export class Login {
  constructor(
    public login?: string,
    public senha?: string
  ) {}
}
```

Prof. Dr. Razer A N R Montaño

Angular

584

584



## 4) Serviço de Login

- Criar um serviço de login

```
$ ng g service auth/services/login
```

- No serviço de registro de usuários, criar os métodos:
  - **usuarioLogado()**: retorna o usuário logado
  - **login()**: efetua o login
  - **logout()**: remove o usuário do registro de usuário logado



## 4) Serviço de Login

- No arquivo de serviço: **src/app/auth/services/login.service.ts**
- Fazer a importação de **Observable** e **of**, e dos modelos

```
import { Observable, of } from 'rxjs';
import { Usuario, Login } from 'src/app/shared';
```

- Para poder retornar um *Observable* com os dados de login
- Dentro da classe, adicionar o código a seguir



## 4) Serviço de Login

```
const LS_CHAVE: string = "usuarioLogado";

export class LoginService {

    public get usuarioLogado(): Usuario {
        let usu = localStorage[LS_CHAVE];
        return (usu ? JSON.parse(localStorage[LS_CHAVE]) : null);
    }

    public set usuarioLogado(usuario: Usuario) {
        localStorage[LS_CHAVE] = JSON.stringify(usuario);
    }

    logout() {
        delete localStorage[LS_CHAVE];
    }
}
```

Prof. Dr. Razer A N R Montaño

Angular

587

587



## 4) Serviço de Login.

```
...
login(login: Login): Observable<Usuario | null> {
    let usu = new Usuario(1, "Razer-Func",
        login.login, login.senha, "FUNC");
    if (login.login == login.senha) {
        if (login.login == "admin") {
            usu = new Usuario(1, "Razer-Admin",
                login.login, login.senha, "ADMIN");
        }
        else if (login.login == "gerente") {
            usu = new Usuario(1, "Razer-Gerente",
                login.login, login.senha, "GERENTE");
        }
        return of(usu);
    }
    else {
        return of(null);
    }
}
...
```

Aqui será trocado por uma consulta aos usuários cadastrados em uma API REST.

O retorno é um *Observable<Usuario>*

Para criá-lo, usa-se *of()*

Prof. Dr. Razer A N R Montaño

Angular

588

588



## 5) Componente de Login

- Criar um componente de login

```
$ ng g component auth/login
```

- Responsável pela tela de login

Prof. Dr. Razer A N R Montaño

Angular

589

589



## 5) Componente de Login (HTML - 1/4)

- Criar a tela de login (*src/app/auth/login/login.component.html*)

```
<h1>Login</h1>

<div *ngIf="message" class="alert alert-danger
  alert-dismissible fade show" role="alert">
  {{message}}
  <a href="#" class="close" data-dismiss="alert"
    aria-label="close"> &times; </a>
  <button type="button" class="close" data-dismiss="alert"
    aria-label="Close" (click)="$event.preventDefault()">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

590

590



## 5) Componente de Login (HTML - 2/4)

- Criar a tela de login (*src/app/auth/login/login.component.html*)

```
<h1>Login</h1>
<div class="well">
  <form #formLogin="ngForm">
    <div class="form-group">
      <label for="usuario">Usuário:</label>
      <input type="text" class="form-control" id="usuario" name="usuario"
             [(ngModel)]="login.login" #usuario="ngModel" required>
      <div *ngIf="usuario.errors && (usuario.dirty || usuario.touched)"
           class="alert alert-danger">
        <div [hidden]="!usuario.errors?.['required']">
          Digite o usuário
        </div>
      </div>
    </div>
  </div>
```



## 5) Componente de Login (HTML - 3/4)

- Criar a tela de login (*src/app/auth/login/login.component.html*)

```
<div class="form-group">
  <label for="senha">Senha:</label>
  <input type="text" class="form-control" id="senha" name="senha"
         [(ngModel)]="login.senha" #senha="ngModel"
         required>
  <div *ngIf="senha.errors && (senha.dirty || senha.touched)"
       class="alert alert-danger">
    <div [hidden]="!senha.errors?.['required']">
      Digite a senha.
    </div>
  </div>
</div>
```



## 5) Componente de Login (HTML - 4/4)

- Criar a tela de login (*src/app/auth/login/login.component.html*)
- ```
<button type="button" class="btn btn-primary"
        (click)="logar()"
        [disabled]="!formLogin.form.valid">
    <i class="fa fa-save" aria-hidden="true"></i> Logar
</button>

<a class="btn btn-secondary"
   [routerLink]="/autocadastro">
    <i class="fa fa-arrow-left" aria-hidden="true"></i> Registrar
</a>
</form>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

593

593



## 5) Componente de Login (TS - 1/4)

- Inserir o código para efetuar o login
  - Arquivo *src/app/auth/login/login.component.ts*

```
export class LoginComponent implements OnInit {

  @ViewChild('formLogin') formLogin!: NgForm;
  login: Login = new Login();
  loading: boolean = false;
  message!: string;

  constructor(
    private loginService: LoginService,
    private router: Router,
    private route: ActivatedRoute) {
    if (this.loginService.usuarioLogado) {
      this.router.navigate( ["/home"] );
    }
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

594

594



## 5) Componente de Login (TS - 2/4)

- Inserir o código para efetuar o login
  - Arquivo *src/app/auth/login/login.component.ts*

```
ngOnInit(): void {  
    this.route.queryParams  
        .subscribe(params => {  
            this.message = params['error'];  
        }) ;  
}
```



## 5) Componente de Login (TS - 3/4)

- Inserir o código para efetuar o login (*src/app/auth/login/login.component.ts*)
  - A rota `/home` será criada mais tarde. Para testar agora pode trocar por `/pessoas`

```
logar(): void {  
    this.loading = true;  
    if (this.formLogin.form.valid) {  
        this.loginService.login(this.login).subscribe((usu) => {  
            if (usu != null) {  
                this.loginService.usuarioLogado = usu;  
                this.loading = false;  
                this.router.navigate( ["/home"] );  
            }  
            else {  
                this.message = "Usuário/Senha inválidos.";  
            }  
        });  
    }  
    this.loading = false;  
}
```



## 5) Componente de Login (TS - 4/4).

- Acertar as importações
  - Arquivo `src/app/auth/login/login.component.ts`

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router, ActivatedRoute } from '@angular/router';
import { LoginService } from '../services/login.service';
import { Login } from 'src/app/shared';
```

Prof. Dr. Razer A N R Montaño

Angular

597

597



## 6) Rota de Login

- Adiciona-se a rota para o login
  - Criar arquivo `src/app/auth/auth-routing.module.ts`

```
import { Routes } from "@angular/router";
import { LoginComponent } from "./login/login.component";

export const LoginRoutes: Routes = [
  {
    path: 'login',
    component: LoginComponent
  }
];
```

Prof. Dr. Razer A N R Montaño

Angular

598

598



## 6) Rota de Login

- Adiciona-se a rota de login dentro das rotas da aplicação
  - Arquivo `src/app/app-routing.module.ts`
  - Não esquecer a importação de `LoginRoutes`

```
const routes: Routes = [
  <todas as rotas aqui>,
  ...LoginRoutes
];
```



## 6) Rota de Login.

- Arrumar a rota vazia "" para ir para "login" (`src/app/app-routing.module.ts`)

```
const routes: Routes = [
  {
    path: '',
    redirectTo: 'login',
    pathMatch: 'full'
  },
  <todas as outras rotas aqui>,
  ...LoginRoutes
];
```



## 7) Serviço de Guarda

- Implementa-se o serviço de guarda
  - Interceptador de rotas
  - Invocado sempre que uma rota é acionada
  - Pode verificar se o perfil do usuário é compatível
- Rodar o comando a seguir:
  - Quando perguntar, escolha **CanActivate**

```
$ ng g guard auth/auth
```

Prof. Dr. Razer A N R Montaño

Angular

601

601



## 7) Serviço de Guarda (1/3)

- Gerou o seguinte arquivo (*src/app/auth/auth.guard.ts*)

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree
} from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |
      Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

602

602



## 7) Serviço de Guarda (2/3)

- Implementar o construtor, injetando o serviço de login e o **Router**
  - Arquivo `src/app/auth/auth.guard.ts`

```
import { LoginService } from './services/login.service';
import { Router } from '@angular/router';

export class AuthGuard implements CanActivate {
    constructor(
        private loginService: LoginService,
        private router: Router
    ) {}
```

Prof. Dr. Razer A N R Montaño

Angular

603

603



## 7) Serviço de Guarda (3/3).

- Dentro do método `canActivate()`, remover o "`return true`" e adicionar:

```
const usuarioLogado = this.loginService.usuarioLogado;
let url = state.url;
if (usuarioLogado) {
    if (route.data?.['role'] && route.data?.['role'].indexOf(usuarioLogado.perfil) === -1) {
        // Se o perfil do usuário não está no perfil da rota
        // vai para login
        this.router.navigate(['/login'],
            { queryParams: { error: "Proibido o acesso a " + url } });
        return false;
    }
    // em qualquer outro caso, permite o acesso
    return true;
}
// Se não está logado, vai para login
this.router.navigate(['/login'],
    { queryParams: { error: "Deve fazer o login antes de acessar " + url } });
return false;
```

Prof. Dr. Razer A N R Montaño

Angular

604

604



## 8) Permissões das Rotas

- Adiciona-se **AuthGuard** em todas as rotas:
  - Editar o arquivo `src/app/app-routing.module.ts`
- Permissões são dadas por:
  - **canActivate**: indicando o **AuthGuard** (deve-se fazer a importação), só um usuário logado acessa
  - **data**: contendo a informação de perfil autorizado

```
canActivate: [AuthGuard],  
data: {  
  role: 'ADMIN,GERENTE,FUNC'  
},
```

- As rotas que não possuírem essas informações poderão ser acessadas por qualquer usuário, logado ou não



## 8) Permissões das Rotas

- Rotas de Pessoa: Todos podem acessar, mas devem estar logados
- Exemplo:
  - Arrumar em todas as rotas, menos as que possuem **redirectTo**

```
{  
  path: 'pessoas/novo',  
  component: InserirPessoaComponent,  
  canActivate: [AuthGuard],  
  data: {  
    role: 'ADMIN,GERENTE,FUNC'  
  },  
},
```



## 8) Permissões das Rotas

- Rotas de Endereço: ADMIN e GERENTE
- Exemplo:
  - Arrumar em todas as rotas, menos as que possuem `redirectTo`

```
{  
  path: 'enderecos/novo',  
  component: InserirCidadeComponent,  
  canActivate: [AuthGuard],  
  data: {  
    role: 'ADMIN,GERENTE'  
  }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

607

607



## 8) Permissões das Rotas

- Rotas de Cidade: Somente GERENTE
- Exemplo:
  - Arrumar em todas as rotas, menos as que possuem `redirectTo`

```
{  
  path: 'cidade/novo',  
  component: InserirCidadeComponent,  
  canActivate: [AuthGuard],  
  data: {  
    role: 'GERENTE'  
  }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

608

608



## 8) Permissões das Rotas.

- Rotas de Estado: ADMIN e FUNC
- Exemplo:
  - Arrumar em todas as rotas, menos as que possuem `redirectTo`

```
{  
  path: 'cidade/novo',  
  component: InserirCidadeComponent,  
  canActivate: [AuthGuard],  
  data: {  
    role: 'ADMIN,FUNC'  
  }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

609

609



## 9) Página principal

- Criar uma página principal da aplicação "Home"
  - A ser mostrada logo após o login
  - Criar rota para `/home` em `src/app/app-routing.module.ts`

```
$ ng g component home
```

- Em `src/app/home/home.component.html` adicione seu HTML da página principal
  - Neste exemplo só vai ter este código mesmo!!!

```
<p>Página inicial</p>
```

Prof. Dr. Razer A N R Montaño

Angular

610

610



## 9) Página principal.

- Criar uma rota para `/home` em `src/app/app-routing.module.ts`

```
{  
  path: 'home',  
  component: HomeComponent,  
  canActivate: [AuthGuard],  
  data: {  
    role: 'ADMIN,GERENTE,FUNC'  
  }  
,
```



## 10) Menu, Usuário logado e Logout

- Alterar o `AppComponent`
  - Manter o usuário logado e fazer logout, em `src/app/app.component.ts`
  - Na Barra de menus mostrar somente os itens autorizados, em `src/app/app.component.html`



## 10) Menu, Usuário logado e Logout (1/4)

- Alteração do HTML principal para a nova barra de menu (*src/app/app.component.html*)

```
<div class="conteiner-fluid">
  <nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <a class="navbar-brand nav-link" href="#" routerLink="/pessoas">
      <i class="fa fa-user-edit"></i> {{ title }} </a>
    <div *ngIf="usuarioLogado"
        class="navbar-collapse collapsed justify-content-between"
        id="navbarContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="#" routerLink="/pessoas">Pessoas</a>
        </li>
        <li *ngIf="usuarioLogado.perfil==='ADMIN' || usuarioLogado.perfil==='GERENTE'" class="nav-item">
          <a class="nav-link" href="#" routerLink="/enderecos">Endereços</a>
        </li>
      </ul>
    </div>
  </nav>
</div>
```



## 10) Menu, Usuário logado e Logout (2/4)

```
<li *ngIf="usuarioLogado.perfil==='GERENTE'" class="nav-item">
  <a class="nav-link" href="#" routerLink="/cidades">Cidades</a>
</li>
<li *ngIf="usuarioLogado.perfil==='ADMIN' || usuarioLogado.perfil==='FUNC'" class="nav-item">
  <a class="nav-link" href="#" routerLink="/estados">Estados</a>
</li>
```



## 10) Menu, Usuário logado e Logout (3/4)

```
<li class="nav-item" ngbDropdown>
  <a class="nav-link" tabindex="0" ngbDropdownToggle id="navbarDropdown1" role="button">
    Cadastros </a>
  <div ngbDropdownMenu aria-labelledby="navbarDropdown1" class="dropdown-menu">
    <a ngbDropdownItem href="#" routerLink="/pessoas"
       (click)="$_event.preventDefault()">Pessoas</a>
    <a *ngIf="usuarioLogado.perfil==='ADMIN' || usuarioLogado.perfil==='GERENTE'">
      ngbDropdownItem href="#" routerLink="/enderecos"
      (click)="$_event.preventDefault()">Endereços</a>
    <a *ngIf="usuarioLogado.perfil==='GERENTE'">
      ngbDropdownItem href="#" routerLink="/cidades"
      (click)="$_event.preventDefault()">Cidades</a>
    <a *ngIf="usuarioLogado.perfil==='ADMIN' || usuarioLogado.perfil==='FUNC'">
      ngbDropdownItem href="#" routerLink="/estados"
      (click)="$_event.preventDefault()">Estados</a>
    </div>
  </li>
</ul>
```

Prof. Dr. Razer A N R Montaño

Angular

615

615



## 10) Menu, Usuário logado e Logout (4/4)

```
<ul *ngIf="usuarioLogado" class="navbar-nav ml-auto">
  <li class="nav-item">
    <a class="nav-link" href="#" (click)="logout()">Logout</a>
  </li>

  <li class="nav-item">
    <p class="navbar-text mb-0 text-white">
      <i class="fa fa-user"></i> {{ usuarioLogado.nome }}
    </p>
  </li>
</ul>
</div>
</nav>

<router-outlet></router-outlet>

</div>
```

Prof. Dr. Razer A N R Montaño

Angular

616

616



## 10) Menu, Usuário logado e Logout

- Alteração do componente principal para manter o usuário logado e fazer logout  
(*src/app/app.component.ts*)

```
constructor(  
    private router: Router,  
    private loginService: LoginService  
) { }  
  
get usuarioLogado(): Usuario | null {  
    return this.loginService.usuarioLogado;  
}  
  
logout() {  
    this.loginService.logout();  
    this.router.navigate(['/login']);  
}
```

Prof. Dr. Razer A N R Montaño

Angular

617

617



## 10) Menu, Usuário logado e Logout..

- Arrumar as importações

```
import { Component } from '@angular/core';  
import { Router } from '@angular/router';  
import { LoginService } from './auth/services/login.service';  
import { Usuario } from './shared';
```

Prof. Dr. Razer A N R Montaño

Angular

618

618



## EXERCÍCIOS..

1. Implementar o mecanismo de login e permissões na aplicação

Prof. Dr. Razer A N R Montaño

Angular

619

619



## Web Services

Prof. Dr. Razer A N R Montaño

Angular

620

620



## Web Services

### Objetivos

- ✓ Acesso à uma API REST
- ✓ Rodar um servidor JSON fake, para cadastro de usuários
- ✓ Implementar o cadastro de Usuário via REST
- ✓ Implementar o Login via REST em uma API em Java



### Referências

➤ <https://angular.io/guide/http>

Prof. Dr. Razer A N R Montaño

Angular

621

621



## Conteúdo

1. Web Service de Usuário com json-server
2. Web Service de Login em Java

Prof. Dr. Razer A N R Montaño

Angular

622

622



## Web Services

- Serviços
  - Executam alguma função
  - Executam uma lógica de negócio
  - Transformam dados
  - Podem rotear mensagens
  - Consultar bancos de dados
  - etc

Prof. Dr. Razer A N R Montaño

Angular

623

623



## Web Services

- Características de Serviços
  - Reutilizáveis
  - Compartilham um contrato formal
  - Baixo acoplamento
  - Abstraem lógica de negócio
  - Podem ser compostos
  - Evitam alocação de recursos por longos períodos de tempo

Prof. Dr. Razer A N R Montaño

Angular

624

624



## Web Services

- Antes, haviam tecnologias proprietárias para comunicação entre serviços
  - OMG CORBA (1991): *Common Object Request Broker Architecture*
  - Microsoft DCOM (1996): *Distributed Component Object Model*
- Em 1998 surge o SOAP (W3C)
  - <https://www.w3.org/TR/soap/>
  - *Simple Object Access Protocol*
  - Protocolo de troca de informações em ambiente descentralizado e distribuído
  - Baseado em XML
  - Descoberta de serviços

Prof. Dr. Razer A N R Montaño

Angular

625

625



## Web Services

- REST surgiu juntamente com o protocolo HTTP 1.1
- Tese de Roy Fielding, 2000
- Ao invés de definir um protocolo de comunicação
  - Estabeleceu conceitos de como usar os métodos HTTP para criar e invocar serviços
  - Não usa encapsulamento em XML
- Não é um protocolo, mas um *design pattern* arquitetural
- É baseado em Recursos: um elemento de informação
  - Recursos são identificados por uma URI
  - Recursos podem ser manipulados, por HTTP e seus métodos (POST, GET, PUT, DELETE, etc)

Prof. Dr. Razer A N R Montaño

Angular

626

626



## Web Services

- Microsserviços

<https://martinfowler.com/articles/microservices.html>

- Desenvolvimento de aplicativos com um conjunto de pequenos serviços
- Componentes mínimos e independentes
- Se comunicam através de um mecanismo leve, ex HTTP
- Mecanismos de *deploy* independente
- Mínimo de centralização possível
  - Linguagens diferentes
  - Diferentes tecnologias
  - Armazenamentos independentes

Prof. Dr. Razer A N R Montaño

Angular

627

627



## Web Services

- Vantagens
  - Ciclos de desenvolvimento reduzido
  - Altamente escalável
  - Resiliente: serviços independentes, um não afeta o outro
  - Fácil implementação: modulares e menores
  - Facilidade de manutenção, entendimento, atualização
  - Heterogeneidade de tecnologias

Prof. Dr. Razer A N R Montaño

Angular

628

628



## Web Services

- Desafios

- Espalhamento de informações
- Dependência entre serviços na hora de compilá-los
- Testes de integração
- Controle de versão: cuidar com a compatibilidade
- Implantação: deve-se investir em automação
- Geração de logs: logs centralizados
- Monitoramento: ter uma visão centralizada para identificar problemas
- Depuração: Remotamente? Centenas de serviços?
- Conectividade: serviços "on-line"
- Acompanhamento de transações de uma única sessão

Prof. Dr. Razer A N R Montaño

Angular

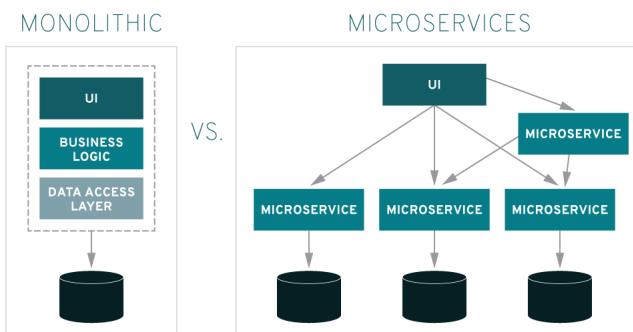
629

629



## Web Services

- Diferenças entre aplicação Monolítica e Microsserviços

FONTE: <https://pretius.com/blog/benefits-of-microservices/>

Prof. Dr. Razer A N R Montaño

Angular

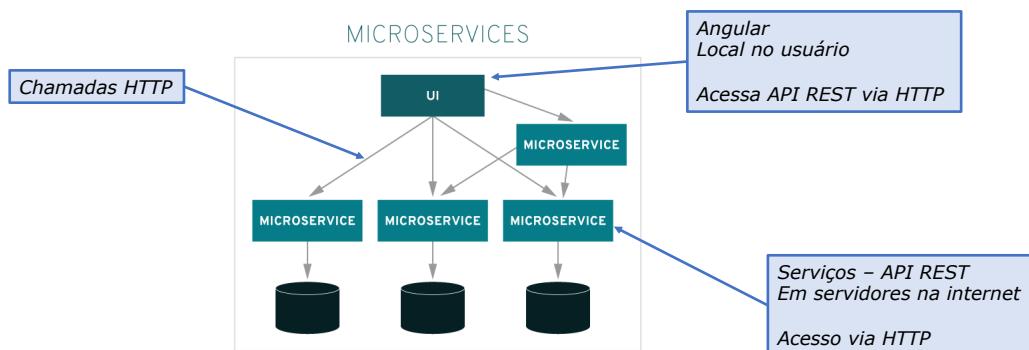
630

630



## Web Services

- Diferenças entre aplicação Monolítica e Microsserviços



FONTE: <https://pretius.com/blog/benefits-of-microservices/>

Prof. Dr. Razer A N R Montaño

Angular

631

631



## Web Services

- Consumir uma API REST
- Usa **HttpClient**
  - Simplifica a interface **XMLHttpRequest**, disponível em navegadores
  - Fácil de enviar requisições
  - Fácil de tratar o retorno
  - Usa **RxJS Observables** para tratar chamadas assíncronas
- RxJS : *framework* de programação reativa
  - Facilita a escrita de código assíncrono

Prof. Dr. Razer A N R Montaño

Angular

632

632



## Web Services

- **Observables** são objetos que podem gerar/emitir valores ao longo do tempo (conhecidos como *streams*)
  - Seguem o *Design Pattern Observer* do GoF
  - <http://www.blackwasp.co.uk/Observer.aspx>
- **Observers** são elementos que executam algum tipo de código quando um novo dado é gerado por um **Observable**
- **Subscrição**
  - Para que um *observer* possa receber os dados de *Observables*

Prof. Dr. Razer A N R Montaño

Angular

633

633



## Web Services

- Métodos de **HttpClient**
  - **get()**
  - **delete()**
  - **post()**
  - **put()**
- Retornam um **Observable<tipo>**
  - Para receber o retorno da API, deve-se subscrever neste *Observable*

Prof. Dr. Razer A N R Montaño

Angular

634

634



## Web Services

- Por exemplo, seja o método:

```
buscarPorId(id: number): Observable<Usuario> {
    return this.httpClient.get<Usuario>(this.BASE_URL + id,
        this.httpOptions);
}
```

- Tem a seguinte estrutura:

```
this.httpClient.get<Usuario>(this.BASE_URL + id, this.httpOptions);
```

Método: get, post, put, delete

URL de acesso:  
http://localhost:3000/usuarios/10

Tipo do retorno do WS

Headers da invocação, ex.:  
'Content-Type': 'application/json'

Prof. Dr. Razer A N R Montaño

Angular

635

635



## Web Services

- Por exemplo, seja o método:

```
buscarPorId(id: number): Observable<Usuario> {
    return this.httpClient.get<Usuario>(this.BASE_URL + id,
        this.httpOptions);
}
```

- Para receber o valor de retorno, deve-se chamar o método e subscrever no seu *Observable* de retorno:

```
buscarPorId(10).subscribe( (data: Usuario) => {
    // pode-se acessar o usuário retornado aqui
    // por meio do parâmetro "data"
})
;
```

Prof. Dr. Razer A N R Montaño

Angular

636

636



## Web Services

- Para receber os dados de uma lista, pode-se usar

```
this.usuarioService.listarTodos().subscribe( {  
    next: (data: Usuario[]) => {  
        if (data == null) {  
            this.usuarios = [];  
        }  
        else {  
            this.usuarios = data;  
        }  
    }  
});
```

Prof. Dr. Razer A N R Montaño

Angular

637

637



## Web Services.

- É possível implementar um serviço em Angular
  - Acessa uma API REST
  - Retorna os resultados como *Observables*
- A API pode estar em qualquer servidor em qualquer linguagem
  - Servidor de teste: **json-server**
  - Servidor em Java com **SpringBoot**

Prof. Dr. Razer A N R Montaño

Angular

638

638

1

## Web Service de Usuário com json-server

Prof. Dr. Razer A N R Montaño

Angular

639

639



## Web Service de Usuário com json-server

- Objetivos desta subseção
  1. Instalar o json-server
  2. Criar o serviço de acesso a usuários
  3. Criar o módulo de manutenção de usuários

Prof. Dr. Razer A N R Montaño

Angular

640

640



## 18.1.1) Servidor JSON

- Pacote json-server do NPM
  - <https://www.npmjs.com/package/json-server>
- Gerencia um BD JSON para efeitos de teste
- Dentro do seu projeto execute

```
$ npm install -g json-server
```

- Crie um diretório chamado **server** e lá dentro um arquivo **db.json**



## 18.1.1) Servidor JSON

- Conteúdo de **db.json**

```
{
  "usuarios": [
    {
      "id": 1,
      "nome": "Admin",
      "login": "admin",
      "senha": "admin",
      "perfil": "ADMIN"
    },
    {
      "id": 2,
      "nome": "Gerente",
      "login": "gerente",
      "senha": "gerente",
      "perfil": "GERENTE"
    },
    {
      "id": 3,
      "nome": "Func",
      "login": "func",
      "senha": "func",
      "perfil": "FUNC"
    }
  ]
}
```



## 18.1.1) Servidor JSON

- Para rodar o servidor, execute

```
$ json-server --no-cors --watch server/db.json
```

- A API estará disponível em **http://localhost:3000**
- Os dados de usuários em **http://localhost:3000/usuarios**



## 18.1.1) Servidor JSON.

The image shows two side-by-side browser windows displaying JSON data for users. Both windows have the title 'localhost:3000/usuarios' and show the same three user entries. The left window has tabs for 'JSON', 'Dados brutos', and 'Cabeçalhos'. The right window has tabs for 'JSON', 'Dados brutos', and 'Cabeçalhos'. The 'Dados brutos' tab is active in both.

**User Data (JSON Format):**

```
[{"id": 1, "nome": "Admin", "login": "admin", "senha": "admin", "perfil": "ADMIN"}, {"id": 2, "nome": "Gerente", "login": "gerente", "senha": "gerente", "perfil": "GERENTE"}, {"id": 3, "nome": "Func", "login": "func", "senha": "func", "perfil": "FUNC"}]
```



## 18.1.2) Serviço de Usuários

- No `src/app/app.module.ts`, importar o módulo `HttpClientModule`

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

- Adicionar também a importação na diretiva `@NgModule`



## 18.1.2) Serviço de Usuários

- Criar um serviço de registro de usuários

```
$ ng g service auth/services/usuario
```

- Aqui serão implementados os métodos de acesso à API REST



## 18.1.2) Serviço de Usuários

- No `src/app/auth/services/usuario.service.ts`, fazer as importações

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Usuario } from 'src/app/shared';
```

- Adicionar a injeção do `HttpClient` no construtor:

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



## 18.1.2) Serviço de Usuários

- Criar um atributo com o endereço base
- Criar um atributo com as opções do HTTP (`headers`)

```
BASE_URL = "http://localhost:3000/usuarios/";

httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
};
```



## 18.1.2) Serviço de Usuários

- No serviço `src/app/auth/services/usuario.service.ts`, criar os métodos:
  - `listarTodos()`: retorna todos os usuários cadastrados
  - `buscarPorId()`: retorna um usuário dado seu id
  - `inserir()`: insere um novo usuário
  - `remover()`: remove um usuário da base
  - `alterar()`: altera um usuário da base

Prof. Dr. Razer A N R Montaño

Angular

649

649



## 18.1.2) Serviço de Usuários

```
listarTodos(): Observable<Usuario[]> {  
  
    return this.httpClient.get<Usuario[]>(this.BASE_URL,  
                                              this.httpOptions);  
}  
  
buscarPorId(id: number): Observable<Usuario> {  
    return this.httpClient.get<Usuario>(this.BASE_URL + id,  
                                              this.httpOptions);  
}
```

Prof. Dr. Razer A N R Montaño

Angular

650

650



## 18.1.2) Serviço de Usuários.

```
inserir(usuario: Usuario): Observable<Usuario> {

    return this.httpClient.post<Usuario>(this.BASE_URL,
        JSON.stringify(usuario),
        this.httpOptions);
}

remover(id: number): Observable<Usuario> {

    return this.httpClient.delete<Usuario>(this.BASE_URL + id,
        this.httpOptions);
}

alterar(usuario: Usuario): Observable<Usuario> {

    return this.httpClient.put<Usuario>(this.BASE_URL + usuario.id,
        JSON.stringify(usuario),
        this.httpOptions);
}
```

Prof. Dr. Razer A N R Montaño

Angular

651

651



## 18.1.3) Módulo de Usuários

- Deve-se criar o módulo de usuários, com um CRUD, da mesma forma que os demais
  - Criar o módulo, os componentes
  - *Model* Usuario já existe do módulo de login
  - Serviço de acesso a Usuário foi criado na seção anterior
- Comandos

```
ng g module usuario
ng g component usuario/inserir-editar-usuario
ng g component usuario/listar-usuario
ng g component usuario/modal-usuario
```

Prof. Dr. Razer A N R Montaño

Angular

652

652



### 18.1.3) Módulo de Usuários

- Acertar as rotas e dar permissão somente ao ADMIN
  - Mesma forma que foi feito na rota de Login
  - Criar `src/app/usuario/usuario-routing.module.ts`
  - Adicionar em `src/app/app-routing.module.ts`
- Adicionar ao menu em `src/app/app.component.html`
- Importar **UsuarioModule** em `src/app/app.module.ts`

Prof. Dr. Razer A N R Montaño

Angular

653

653



### usuario.model.ts

```
export class Usuario {  
    constructor(  
        public id?: number,  
        public nome?: string,  
        public login?: string,  
        public senha?: string,  
        public perfil?: string  
    ) {}  
}
```

Prof. Dr. Razer A N R Montaño

Angular

654

654



## usuario-routing.module.ts

```
export const UsuarioRoutes: Routes = [
  {
    path: 'usuarios',
    redirectTo: 'usuarios/listar',
  },
  {
    path: 'usuarios/listar',
    component: ListarUsuarioComponent,
    canActivate: [AuthGuard],
    data: {
      role: 'ADMIN'
    }
  },
  {
    path: 'usuarios/novo',
    component: InserirEditarUsuarioComponent,
    canActivate: [AuthGuard],
    data: {
      role: 'ADMIN'
    }
  },
  {
    path: 'usuarios/editar/:id',
    component: InserirEditarUsuarioComponent,
    canActivate: [AuthGuard],
    data: {
      role: 'ADMIN'
    }
  }
];
```

Prof. Dr. Razer A N R Montaño

Angular

655

655



## modal-usuario.component.html

```
<div class="modal-header">
  <button type="button" class="close" aria-label="Fechar"
    (click)="activeModal.dismiss() ">
    <span aria-hidden="true">&times;</span>
  </button>
  <h4 class="modal-title" id="modalLabel"> Usuário </h4>
</div>

<div class="modal-body text-center">
<div>
  <p>Nome: {{ usuario.nome }} </p>
  <p>Login: {{ usuario.login }}</p>
  <p>Perfil: {{ usuario.perfil }} </p>
</div>

<div class="modal-footer">
  <button type="button" class="btn btn-outline-dark"
    (click)="activeModal.close() ">Fechar</button>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

656

656



## modal-usuario.component.ts

```
export class ModalUsuarioComponent implements OnInit {

  @Input() usuario: Usuario = new Usuario();

  constructor(public activeModal: NgbActiveModal) {}

  ngOnInit() {

  }

}
```

Prof. Dr. Razer A N R Montaño

Angular

657

657



## listar-usuario.component.html

Nome	Login	Perfil	
<a href="#">Novo</a>			<a (click)="remover(\$event, usuario)" alt="Remover" class="btn btn-xs btn-danger" href="#" title="Remover"><i aria-hidden="true" class="fa fa-times"></i> Remover</a>

Nenhum usuário cadastrado.

Prof. Dr. Razer A N R Montaño

Angular

658

658



## listar-usuario.component.ts

```

export class ListarUsuarioComponent implements OnInit {
  usuarios: Usuario[] = [];

  constructor(private usuarioService: UsuarioService,
    private modalService: NgbModal) { }

  ngOnInit(): void {
    this.usuarios = [];
    this.listarTodos();
  }

  listarTodos(): Usuario[] {
    this.usuarioService.listarTodos().subscribe({
      next: (data: Usuario[]) => {
        if (data == null) {
          this.usuarios = [];
        }
        else {
          this.usuarios = data;
        }
      }
    });
    return this.usuarios;
  }

  remover($event: any, usuario: Usuario): void {
    $event.preventDefault();
    if (confirm('Deseja realmente remover o usuário "' + usuario.nome + '?')) {
      this.usuarioService.remover(usuario.id!).subscribe({
        complete: () => { this.listarTodos(); }
      });
    }
  }

  abrirModal(usuario: Usuario) {
    const modalRef = this.modalService.open(ModalUsuarioComponent);
    modalRef.componentInstance.usuario = usuario;
  }
}

```

Prof. Dr. Razer A N R Montaño

Angular

659

659



## inserir-editar-usuario.component.html (1/2)

```

<h1 *ngIf="!novoUsuario">Novo Usuário</h1>
<h1 *ngIf="novoUsuario">Editar Usuário</h1>
<div class="well">
  <form #formUsuario="ngForm">
    <div class="form-group">
      <label for="nome">Nome:</label>
      <input type="text" class="form-control" id="nome" name="nome" [(ngModel)]="usuario.nome" #nome="ngModel"
        minlength="2" required>
      <div *ngIf="!nome.errors && (nome.dirty || nome.touched)" class="alert alert-danger">
        <div [hidden]="!nome.errors?.['required']"> Digite o nome do usuário. </div>
        <div [hidden]="!nome.errors?.['minlength']"> O nome deve conter ao menos 2 caracteres. </div>
      </div>
    </div>

    <div class="form-group">
      <label for="login">Login:</label>
      <input type="text" class="form-control" id="login" name="login" [(ngModel)]="usuario.login" #login="ngModel"
        minlength="2" required>
      <div *ngIf="!login.errors && (login.dirty || login.touched)" class="alert alert-danger">
        <div [hidden]="!login.errors?.['required']"> Digite o login da pessoa. </div>
        <div [hidden]="!login.errors?.['minlength']"> O login deve conter ao menos 2 caracteres. </div>
      </div>
    </div>

    <div *ngIf="!novoUsuario"><p>Deixe a senha em branco para não alterá-la.</p></div>
    <div class="form-group">
      <label for="senha">Senha:</label>
      <input type="text" class="form-control" id="senha" name="senha" [(ngModel)]="usuario.senha" #senha="ngModel" >
    </div>
  </form>
</div>

```

Prof. Dr. Razer A N R Montaño

Angular

660

660



## inserir-editar-usuario.component.html (2/2)

```
<div class="form-group">
  <label for="perfil">Perfil:</label>
  <input type="radio" class="form-check-input" id="perfil" name="perfil"
    [(ngModel)]="usuario.perfil"
    #perfil="ngModel"
    value="ADMIN"> Administrador <br/>
  <input type="radio" class="form-check-input" id="perfil" name="perfil"
    [(ngModel)]="usuario.perfil"
    #perfil="ngModel"
    value="GERENTE"> Gerente <br/>
  <input type="radio" class="form-check-input" id="perfil" name="perfil"
    [(ngModel)]="usuario.perfil"
    #perfil="ngModel"
    value="FUNC"> Funcionário <br/>
</div>
<button type="button" class="btn btn-primary" (click)="salvar()" [disabled]="!formUsuario.form.valid">
  <i *ngIf="!loading" class="fa fa-save" aria-hidden="true"></i>
  <span *ngIf="loading" class="spinner-border spinner-border-sm mr-1"></span> Salvar
</button>
<a class="btn btn-secondary" [routerLink]=["'/usuarios']>
  <i class="fa fa-arrow-left" aria-hidden="true"></i> Voltar
</a>
</form>
</div>
```

Prof. Dr. Razer A N R Montaño

Angular

661

661



## inserir-editar-usuario.component.ts (1/2)

```
export class InserirEditarUsuarioComponent implements OnInit {
  @ViewChild('formUsuario') formUsuario!: NgForm;
  novoUsuario: boolean = true;
  usuario: Usuario = new Usuario();
  id!: string;
  loading!: boolean;

  constructor(
    private usuarioService: UsuarioService,
    private route: ActivatedRoute,
    private router: Router) { }

  ngOnInit(): void {
    this.usuario = new Usuario();
    this.loading = false;

    this.id = this.route.snapshot.params['id'];
    this.novoUsuario = !this.id;

    if (!this.novoUsuario) {
      this.usuarioService.buscarPorId(+this.id).subscribe(
        usuario => {
          this.usuario = usuario;
          this.usuario.senha = "";
        }
      );
    }
  }
}
```

Prof. Dr. Razer A N R Montaño

Angular

662

662



## inserir-editar-usuario.component.ts (2/2)

```
salvar(): void {
  this.loading = true;
  if (this.formUsuario.form.valid) {
    if (this.novoUsuario) {
      this.usuarioService.inserir(this.usuario).subscribe(
        usuario => {
          this.loading = false;
          this.router.navigate( ["/usuarios"] );
        }
      );
    } else {
      this.usuarioService.alterar(this.usuario).subscribe(
        usuario => {
          this.loading = false;
          this.router.navigate( ["/usuarios"] );
        }
      );
    }
  }
  this.loading = false;
}
```

Prof. Dr. Razer A N R Montaño

Angular

663

663



## 18.1.3) Módulo de Usuários..

- Neste ponto todo o acesso ao CRUD de usuários está sendo feito no servidor JSON *fake*
  - Pode-se conferir os dados no arquivo **server/db.json**
- Pode-se então migrar para um servidor válido
  - Desenvolvido em Java com **Framework Spring**

Prof. Dr. Razer A N R Montaño

Angular

664

664



## EXERCÍCIOS.

1. Implementar o módulo de usuários com o uso da API REST (no json-server)

Prof. Dr. Razer A N R Montaño

Angular

665

665

2

## Web Service de Login em Java

Prof. Dr. Razer A N R Montaño

Angular

666

666



## Web Service de Login em Java

- Objetivos desta subseção
  - Criar o projeto de Login usando Spring Boot
  - Acessar o Web Service de Login

Prof. Dr. Razer A N R Montaño

Angular

667

667



## Web Service de Login em Java

- Você pode construí-lo facilmente usando **SpringBoot**
  - Baixe e instale o JDK 11 (OpenJDK - <https://jdk.java.net/archive/>)
  - Baixe o SpringTools (<https://spring.io/tools>)
    - Neste material foi usada a versão 4.14.0
    - IDE estava atualizada
  - Crie um novo projeto
    - Na criação não esqueça de adicionar suporte a Spring Web
  - Crie as classes *model* Login e Usuário
  - Crie a classe de Serviço
- Este tutorial pode também ajudar:  
<https://spring.io/guides/gs/rest-service/>

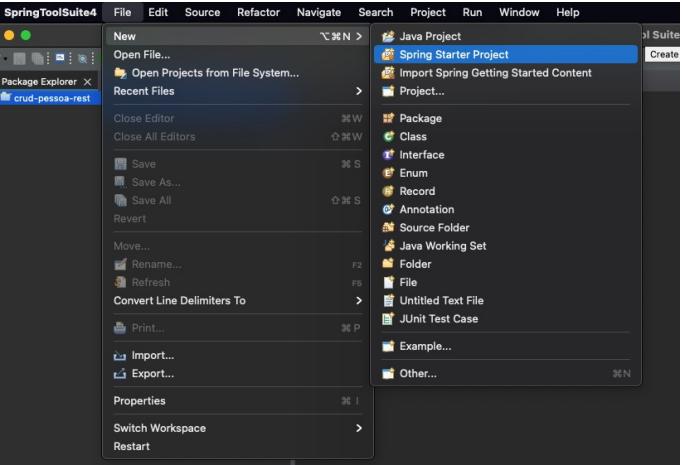
Prof. Dr. Razer A N R Montaño

Angular

668

668

## Criar o Projeto



The screenshot shows the SpringToolSuite4 interface. The 'File' menu is open, displaying various options like 'New', 'Open File...', and 'Recent Files'. The 'Spring Starter Project' option is highlighted with a blue selection bar. The 'Package Explorer' view on the left shows a project named 'crud-pessoa-rest'.

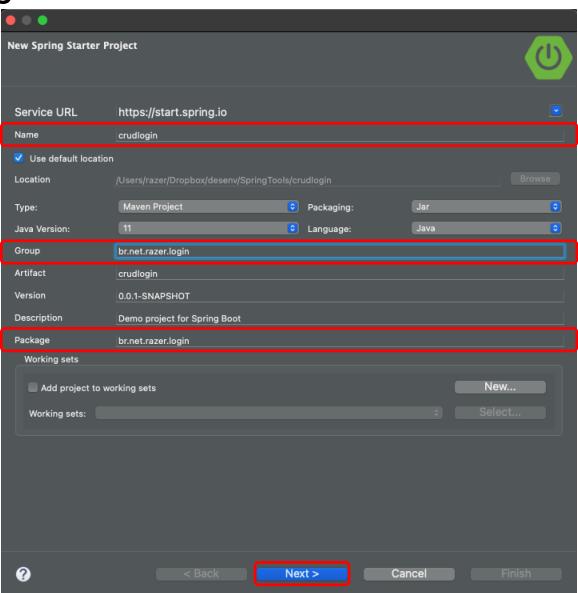
Prof. Dr. Razer A N R Montaño

Angular

669

669

## Criar Projeto



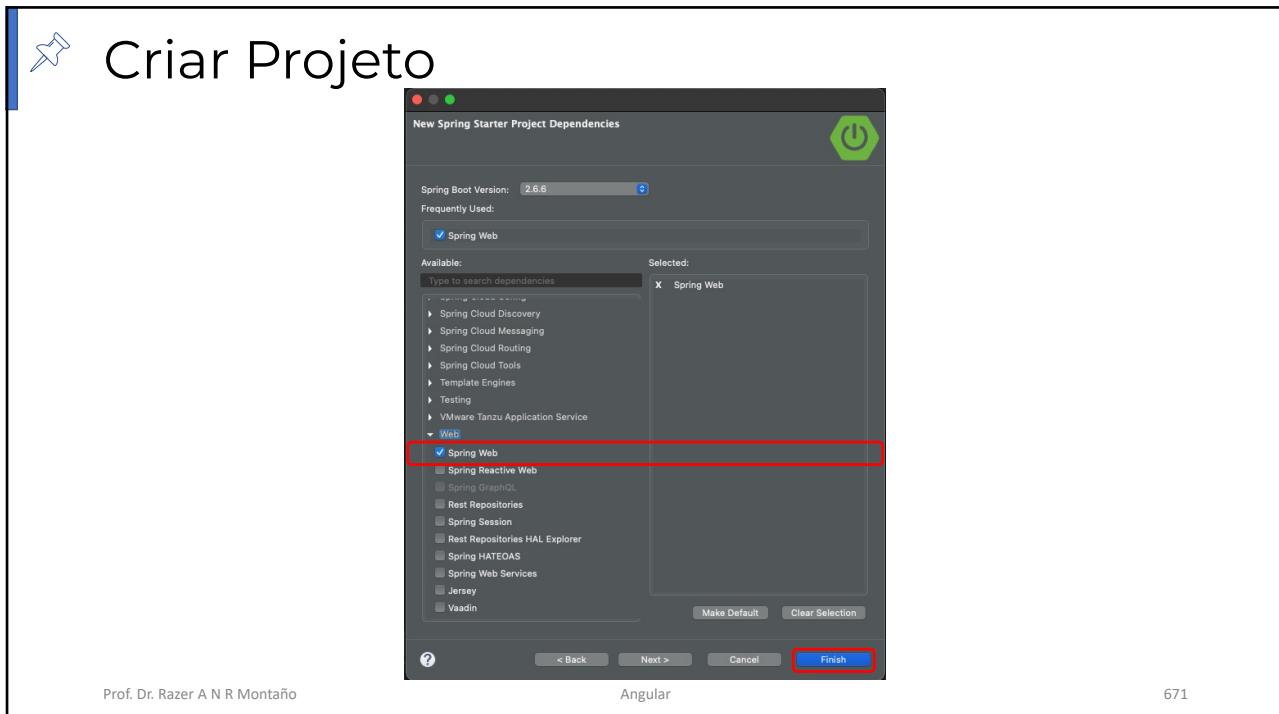
The screenshot shows the 'New Spring Starter Project' dialog. The 'Service URL' field contains 'https://start.spring.io'. The 'Name' field is set to 'crudlogin'. The 'Group' field is set to 'br.net.razer.login'. The 'Package' field is also set to 'br.net.razer.login'. The 'Next >' button at the bottom is highlighted with a red border.

Prof. Dr. Razer A N R Montaño

Angular

670

670

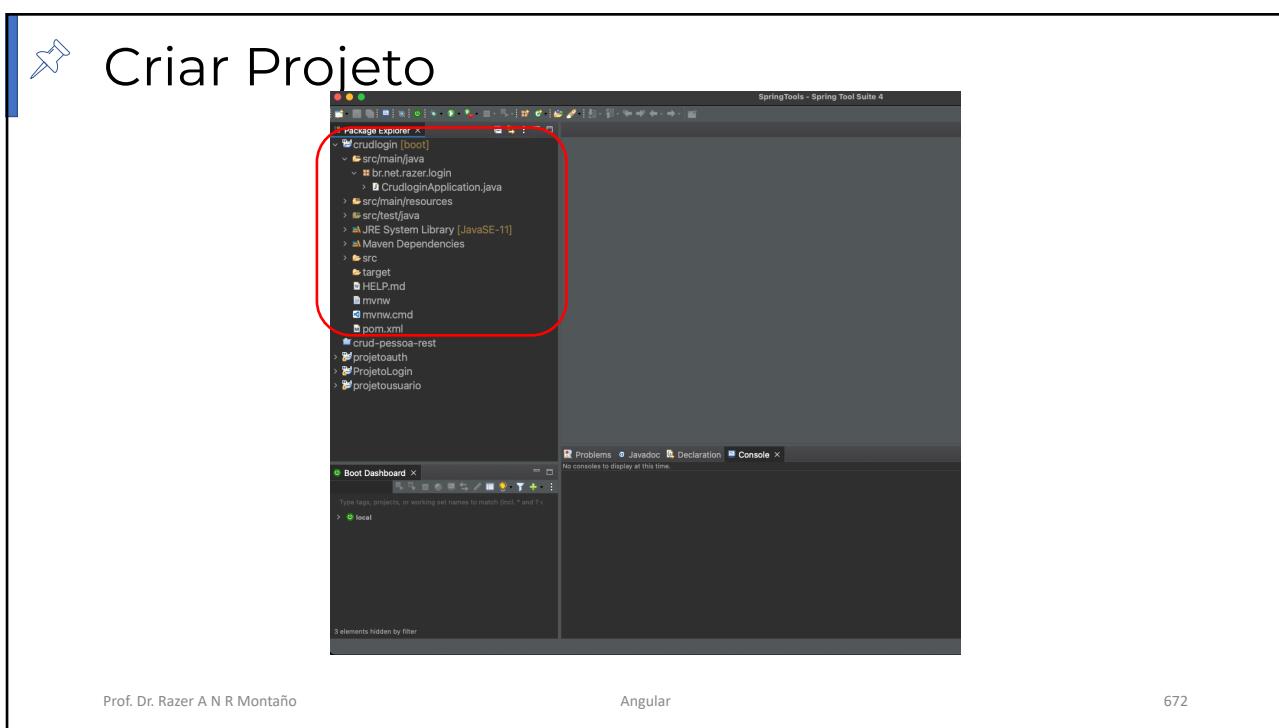


Prof. Dr. Razer A N R Montaño

Angular

671

671

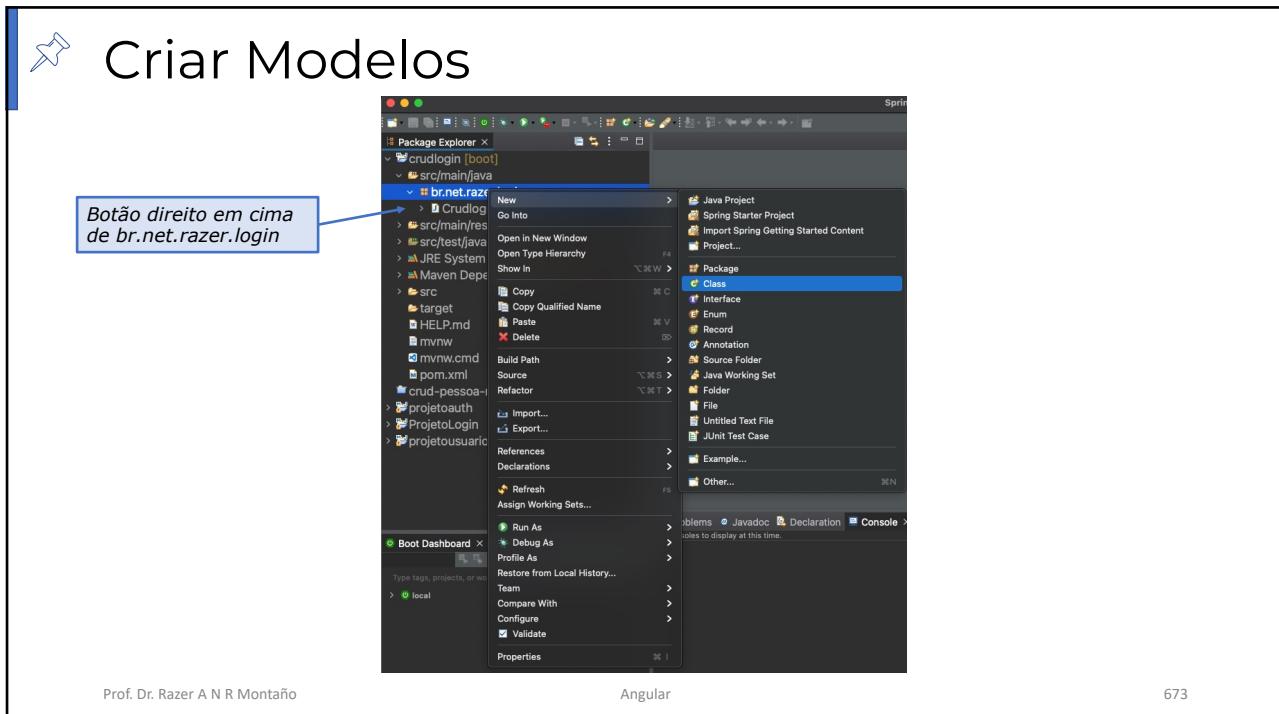


Prof. Dr. Razer A N R Montaño

Angular

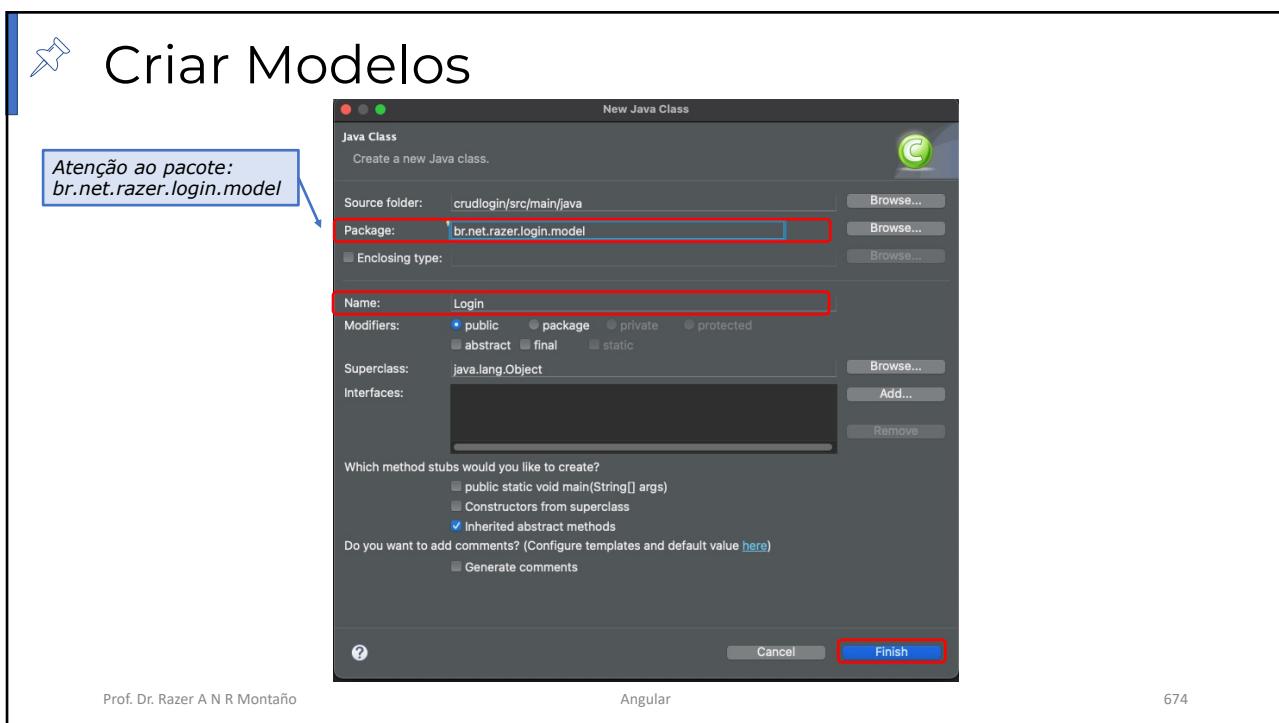
672

672



673

673



674

674



## Criar Modelos

The screenshot shows the Spring Tools IDE interface. On the left, the Package Explorer shows a project structure for 'crudlogin [boot]'. The 'src/main/java' folder contains packages 'br.net.razer.login' and 'br.net.razer.login.model'. Within 'br.net.razer.login.model', there is a file named 'Login.java'. On the right, the code editor window displays the content of 'Login.java':

```
1 package br.net.razer.login.model;
2
3 public class Login {
4
5 }
```

Prof. Dr. Razer A N R Montaño

Angular

675

675



## Criar Modelos

- Classe model Login

```
package br.net.razer.login.model;
import java.io.Serializable;
public class Login implements Serializable {
    private String login;
    private String senha;

    // setters/getters
}
```

The screenshot shows a code editor with a file named 'login.model.ts'. The code defines a class 'Login' with properties 'login' and 'senha' and a constructor:

```
1 export class Login {
2     constructor(
3         public login?: string,
4         public senha?: string
5     ) {}
6 }
7
```

Prof. Dr. Razer A N R Montaño

Angular

676

676



## Criar Modelos

```

1 package br.net.razer.login.model;
2
3 import java.io.Serializable;
4
5 public class Login implements Serializable {
6     private String login;
7     private String senha;
8
9     public Login() {
10         super();
11     }
12
13     public Login(String login, String senha) {
14         super();
15         this.login = login;
16         this.senha = senha;
17     }
18
19     public String getLogin() {
20         return login;
21     }
22
23     public void setLogin(String login) {
24         this.login = login;
25     }
26
27     public String getSenha() {
28         return senha;
29     }
30
31     public void setSenha(String senha) {
32         this.senha = senha;
33     }
34 }
35 }
```

Prof. Dr. Razer A N R Montaño

Angular

677

677



## Criar Modelos

- Classe *model* Usuário, no mesmo pacote

```

package br.net.razer.login.model;
import java.io.Serializable;
public class Usuario implements Serializable {
    private int id;
    private String nome;
    private String login;
    private String senha;
    private String perfil;

    // setters/getters e construtores
}
```

src > app > shared > models > **ts** usuario.model.ts >

```

1 export class Usuario {
2     constructor(
3         public id?: number,
4         public nome?: string,
5         public login?: string,
6         public senha?: string,
7         public perfil?: string
8     ) {}
9
10 }
```

Prof. Dr. Razer A N R Montaño

Angular

678

678

## Criar Modelos

```

1 package br.net.razer.login.model;
2
3 import java.io.Serializable;
4
5 public class Usuario implements Serializable {
6     private int id;
7     private String nome;
8     private String login;
9     private String senha;
10    private String perfil;
11
12    public Usuario() {
13        super();
14    }
15
16    public Usuario(int id, String nome, String login, String senha, String perfil) {
17        super();
18        this.id = id;
19        this.nome = nome;
20        this.login = login;
21        this.senha = senha;
22        this.perfil = perfil;
23    }
24
25    public int getId() {
26        return id;
27    }
28
29    public void setId(int id) {
30        this.id = id;
31    }
32
33    public String getName() {
34        return nome;
35    }
36
37    public void setName(String nome) {
38        this.nome = nome;
39    }
40
41    public String getLogin() {
42        return login;
43    }
44
45    public void setLogin(String login) {
46        this.login = login;
47    }
48
49    public String getSenha() {
50        return senha;
51    }
52
53    public void setSenha(String senha) {
54        this.senha = senha;
55    }
56
57    public String getPerfil() {
58        return perfil;
59    }
60
61    public void setPerfil(String perfil) {
62        this.perfil = perfil;
63    }
64}
65

```

Prof. Dr. Razer A N R Montaño

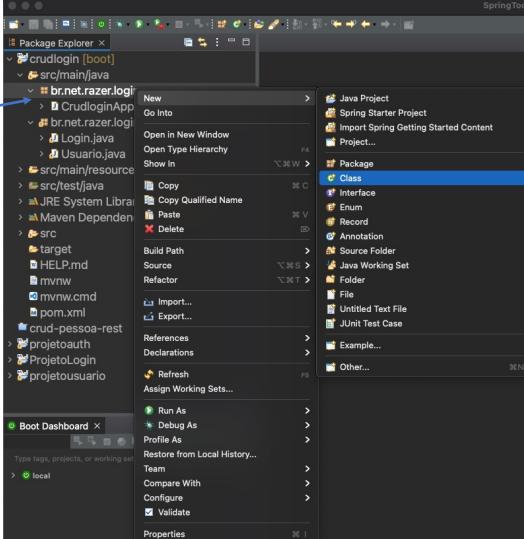
Angular

679

679

## Criar WebService de Login

Botão direito em cima de `br.net.razer.login`

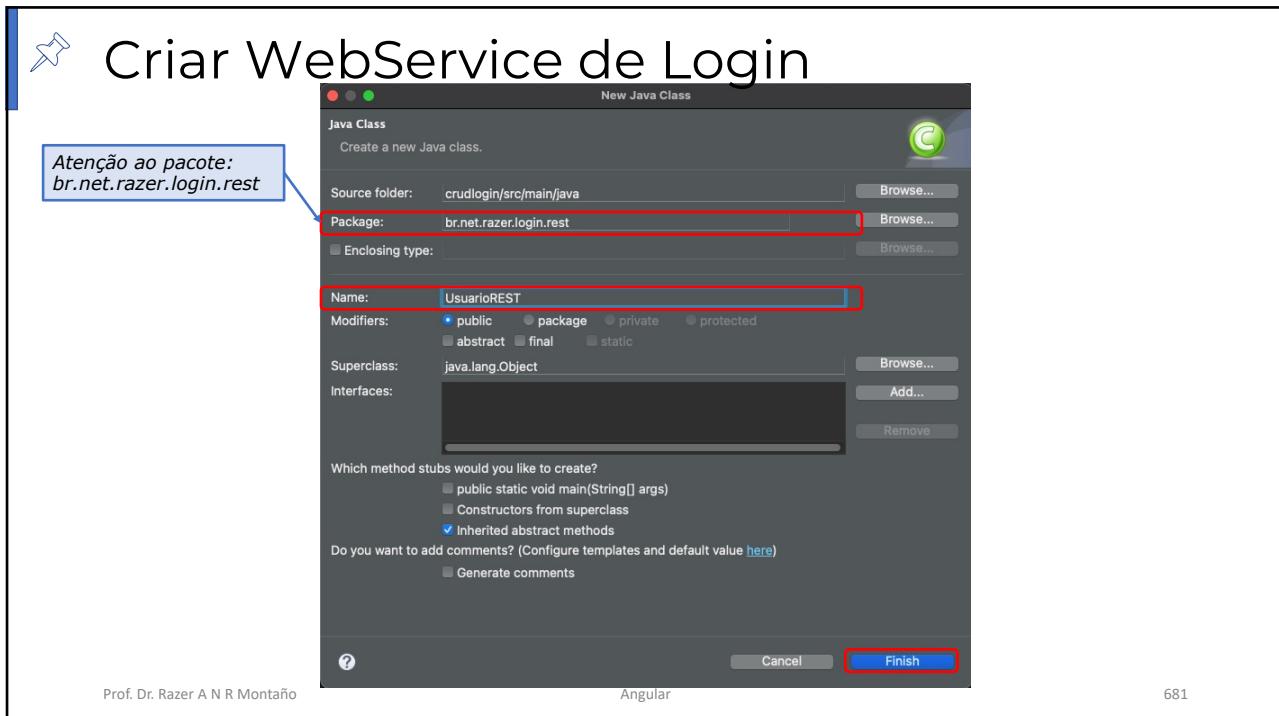


Prof. Dr. Razer A N R Montaño

Angular

680

680



Prof. Dr. Razer A N R Montaño

Angular

681

## WebService de Login

- Classe Serviço REST
  - Acertar as importações

```
package br.net.razer.login.rest;

import java.util.ArrayList;
import java.util.List;

import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.net.razer.login.model.Usuario;
import br.net.razer.login.model.Login;
```

Prof. Dr. Razer A N R Montaño Angular 682

682



## WebService de Login

```

@CrossOrigin
@RestController
public class UsuarioREST {
    public static List<Usuario> lista = new ArrayList<>();
    @PostMapping("/login")
    Usuario login(@RequestBody Login login) {
        Usuario usuario = lista.stream().
            filter(usu -> usu.getLogin().equals(login.getLogin()) &&
                   usu.getSenha().equals(login.getSenha())).
            findAny().orElse(null);
        return usuario;
    }
    @GetMapping("/login")
    List<Usuario> listarTodos() {
        return lista;
    }
    static {
        lista.add(new Usuario(1, "administr", "admin", "admin", "ADMIN"));
        lista.add(new Usuario(2, "gerent", "gerente", "gerente", "GERENTE"));
        lista.add(new Usuario(3, "funcion", "func", "func", "FUNC"));
    }
}

```

Prof. Dr. Razer A N R Montaño

Angular

683

683



## Criar WebService de Login

```

1 package br.net.razer.login.rest;
2
3 import java.util.ArrayList;
4
5 @CrossOrigin
6 @RestController
7 public class UsuarioREST {
8     public static List<Usuario> lista = new ArrayList<>();
9
10    @PostMapping("/login")
11    Usuario login(@RequestBody Login login) {
12        Usuario usuario = lista.stream().
13            filter(usu -> usu.getLogin().equals(login.getLogin()) &&
14                   usu.getSenha().equals(login.getSenha())).
15            findAny().orElse(null);
16        return usuario;
17    }
18    @GetMapping("/login")
19    List<Usuario> listarTodos() {
20        return lista;
21    }
22    static {
23        lista.add(new Usuario(1, "administr", "admin", "admin", "ADMIN"));
24        lista.add(new Usuario(2, "gerent", "gerente", "gerente", "GERENTE"));
25        lista.add(new Usuario(3, "funcion", "func", "func", "FUNC"));
26    }
27}

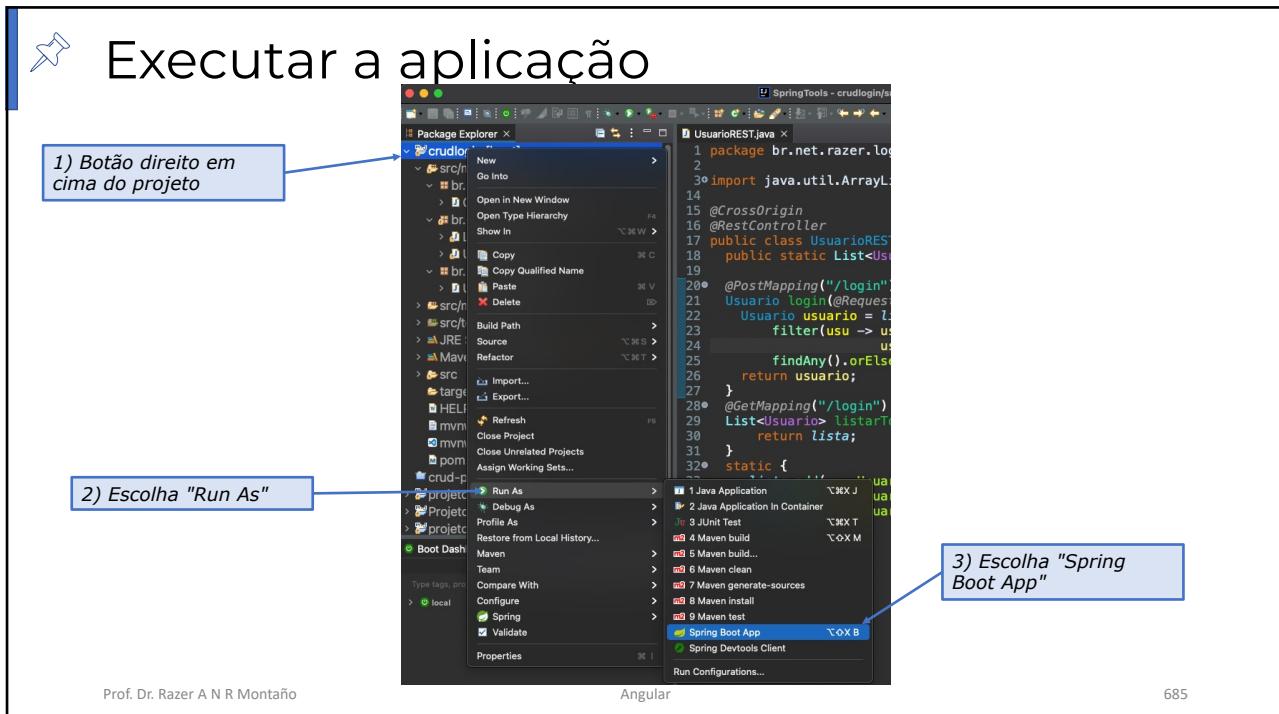
```

Prof. Dr. Razer A N R Montaño

Angular

684

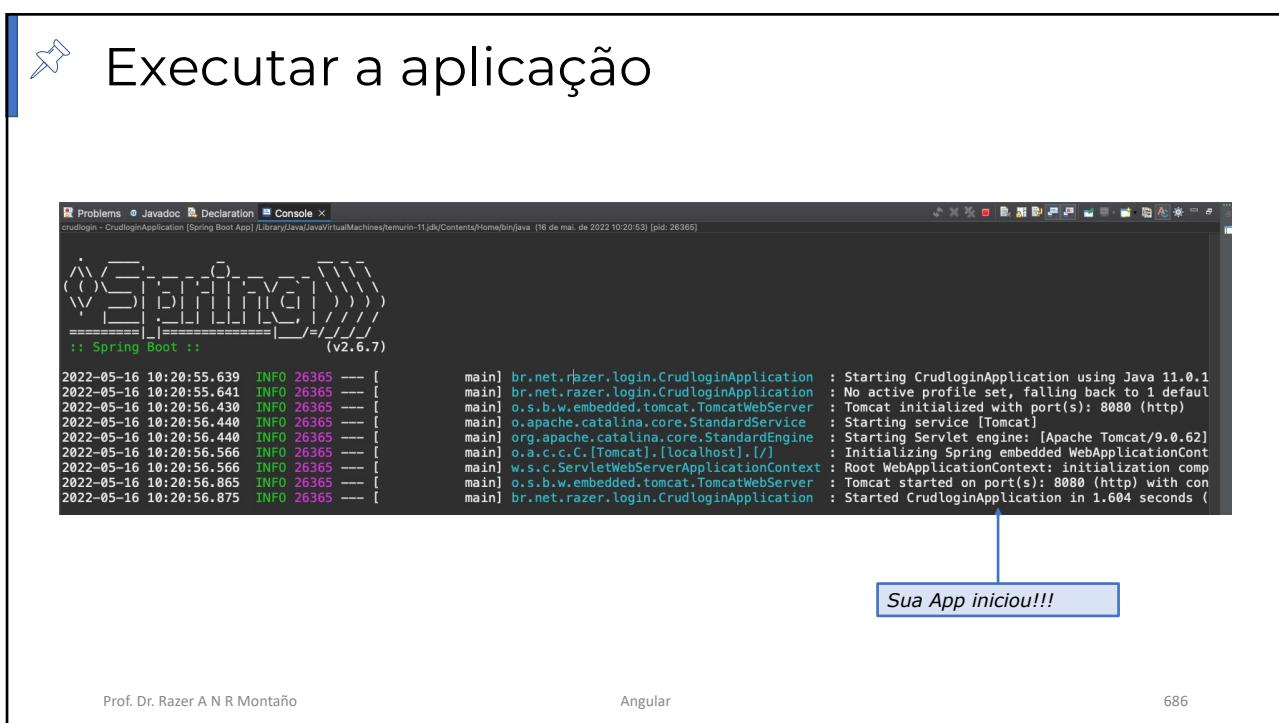
684



Prof. Dr. Razer A N R Montaño

Angular

685



Prof. Dr. Razer A N R Montaño

Angular

686



## Executar a aplicação

- Ao rodar o projeto, o WS REST estará disponível em:  
**http://localhost:8080/login**
- Via POST para login
- Via GET como teste para ver todos os Usuários

Prof. Dr. Razer A N R Montaño

Angular

687

687



## Executar a aplicação.



A screenshot of a web browser window titled "localhost:8080/login". The content is displayed in a JSON viewer. The JSON structure shows three objects (0, 1, and 2) representing user data:

```
JSON Dados brutos Cabeçalhos
Salvar Copiar Recolher tudo Expandir tudo
▼ 0:
  id: 1
  nome: "administr"
  login: "admin"
  senha: "admin"
  perfil: "ADMIN"
▼ 1:
  id: 2
  nome: "gerent"
  login: "gerente"
  senha: "gerente"
  perfil: "GERENTE"
▼ 2:
  id: 3
  nome: "funcion"
  login: "func"
  senha: "func"
  perfil: "FUNC"
```

Prof. Dr. Razer A N R Montaño

Angular

688

688



## Acessar o Web Service de Login

- Deve-se alterar o serviço que acessa o login:  
*src/app/auth/login/services/login.service.ts*
- Retirar o código *fake* e adicionar uma chamada ao WS recém implementado

Prof. Dr. Razer A N R Montaño

Angular

689

689



## Acessar o Web Service de Login

- Adicionar atributos e **HttpClient** no construtor
- Acertar as importações

```
export class LoginService {  
  BASE_URL = "http://localhost:8080/login/";  
  httpOptions = {  
    headers: new HttpHeaders({  
      'Content-Type': 'application/json'  
    })  
  };  
  
  constructor(private httpClient: HttpClient) { }  
}
```

Prof. Dr. Razer A N R Montaño

Angular

690

690



## Acessar o Web Service de Login

- Alteração no método **login()**

```
login(login: Login): Observable<Usuario> {  
    return this.httpClient.post<Usuario>(this.BASE_URL,  
        login,  
        this.httpOptions);  
}
```



Já retorna um  
*Observable<Usuario>*



## Acessar o Web Service de Login.

- Neste ponto o LOGIN já está sendo feito no seu Web Service na porta 8080
- Deve-se implementar as demais funcionalidades de USUÁRIO para que todo o módulo de usuário acesse o Web Service



## EXERCÍCIOS.

1. Implementar os dois WS
  - a) WS de usuário com o json-server
  - b) WS de login com o SpringBoot

Prof. Dr. Razer A N R Montaño

Angular

693

693

3

## REST API Completa de Usuários em Java

Prof. Dr. Razer A N R Montaño

Angular

694

694



## REST API de Usuário

- Migrar todo o módulo de usuário para REST API Java
- Criar uma tabela no PostgreSQL para manter usuários
- Usar Spring Data para acessar a base
- Prover todo o CRUD de usuários
- Prover o LOGIN

Prof. Dr. Razer A N R Montaño

Angular

695

695



## REST API de Usuário

- Passos a serem efetuados
1. Criar a tabela no PostgreSQL

Prof. Dr. Razer A N R Montaño

Angular

696

696



## 1) Criar a Tabela no PostgreSQL

```
CREATE TABLE tb_usuarios (
    id_usu serial PRIMARY KEY,
    nome_usu character varying(50),
    login_usu character varying(50),
    senha_usu character varying(50),
    perfil_usu character varying(10)
)
```

Prof. Dr. Razer A N R Montaño

Angular

697

697



## EXERCÍCIOS..

1. Implementar o WS de usuário completo com Spring
  - Não precisa usar banco de dados, basta armazenar uma lista estática com todos os usuários
  - Alterar a aplicação Angular para acessar este WS
2. Implementar todo o CRUD de usuários e LOGIN em API REST Java com Banco de Dados PostgreSQL

Prof. Dr. Razer A N R Montaño

Angular

698

698



# Variáveis de Ambiente

Prof. Dr. Razer A N R Montaño

Angular

699

699



## Variáveis de Ambiente

- O servidor de DEV pode ser diferente do PROD
  - Outras variáveis podem diferenciar
  - Usuário/senha
  - Cor
  - Etc
- Quando construir a aplicação Angular com

```
$ ng build --configuration=production
```

  - Usa as variáveis de produção
- Para executar com a configuração de produção

```
$ ng serve --configuration=production
```

Prof. Dr. Razer A N R Montaño

Angular

700

700



## Variáveis de Ambiente

- Instalação/*deploy* da aplicação
- Várias integrações com o comando:

**\$ ng deploy**

- Firebase, Azure, Amazon S3, etc
- Pode ser usado em servidores de HTML estático, como **Apache** e **Nginx**
- Consultar: <https://angular.io/guide/deployment>

Prof. Dr. Razer A N R Montaño

Angular

701

701

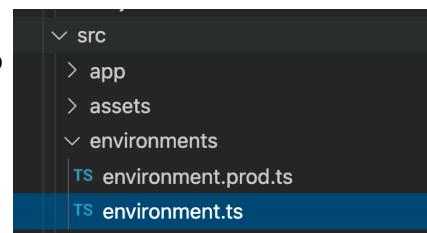


## Variáveis de Ambiente

- Deve-se importar o **environment** nos componentes necessários

```
import { environment as env } from 'src/environments/environment';
```

- Quando a aplicação é construída em desenvolvimento
  - Usa **environment.ts**
- Quando a aplicação é construída para produção
  - Usa o **environment.prod.ts**
- Usa-se **env** para acessar os dados de ambiente



Prof. Dr. Razer A N R Montaño

Angular

702

702



## Variáveis de Ambiente

- O arquivo **src/environments/environment.ts** fica

```
export const environment = {
  production: false,
  BASE_URL: "http://localhost:3000",
  LOGIN_BASE_URL: "http://localhost:8080"
};
```

Prof. Dr. Razer A N R Montaño

Angular

703

703



## Variáveis de Ambiente

- O arquivo **src/environments/environment.prod.ts** fica com as mesmas variáveis

```
export const environment = {
  production: true,
  BASE_URL: "http://www.servidor.com.br:3000",
  LOGIN_BASE_URL: "http://auth.servidor.com.br:8080"
};
```

Prof. Dr. Razer A N R Montaño

Angular

704

704



## Variáveis de Ambiente..

- No arquivo `src/app/usuario/usuario.service.ts`:

```
...
import { environment as env } from 'src/environments/environment';
...
@Injectable({
  providedIn: 'root'
})
export class UsuarioService {
  ...
  BASE_URL: string = env.BASE_URL + "usuarios/";
  ...
}
```

Aqui está o uso  
das variáveis de  
ambiente



## EXERCÍCIOS..

1. Implementar as variáveis de ambiente para o servidor REST
2. Testar em ambiente de DEV e PROD.



## REFERÊNCIAS

[1] ANGULAR. Angular: The modern web developer's platform Disponível em: <<https://angular.io/>>. Acesso em 09 de jan. de 2021.

[2] MACHADO, Kheronn. Angular 8 e Firebase: Construindo uma aplicação integrada com a plataforma do Google. Casa do Código. 2019.

Prof. Dr. Razer A N R Montaño

Angular

707

707



## Licenças

<https://thenounproject.com/term/science-fiction/>

Icons made by [Frepik](#) from [www.flaticon.com](http://www.flaticon.com)

Prof. Dr. Razer A N R Montaño

Angular

708

708