Assunto Prova

Criado em	@7 de dezembro de 2025 16:49
Tags	

Criar o projeto Angular

Digite:

```
ng add @angular/material
ng new nome-do-projeto
```

Rodar o projeto

```
ng serve --open
```

Criar um componente:

```
ng g c components/nome
```

Usando variáveis da classe no HTML

Na classe:

```
export class PrimeiroComponente {
  nome = "Ana";
  idade = 20;
}
```

No HTML:

```
<p>Nome: {{ nome }}</p>
<p>Idade: {{ idade }}</p>
```

👉 O `{{ }}` é o *interpolation*, usado para exibir valores direto do TS.

Usando métodos da classe no HTML

Na classe:

```
export class PrimeiroComponente {
  nome = "Ana";

  mostrarMensagem() {
    return "Bem-vinda, " + this.nome + "!";
  }
}
```

No HTML:

```
<p>{{ mostrarMensagem() }}</p>
```

👉 O Angular chama o método durante a renderização e mostra o retorno.

Ligando valores a atributos

Na classe:

```
imagem = "foto.jpg";
```

No HTML:

```
<img [src]="imagem">
```

Chamando métodos em eventos (Event Binding)

Na classe:

```
contador = 0;

incrementar() {
  this.contador++;
}
```

No HTML:

```
<p>{{ contador }}</p>
<button (click)="incrementar()">Somar</button>
```

Como criar um Service

Com o comando:

```
ng g s nome-do-service
```

Como usar um Service no componente

1. Importa o service

```
import { MeuService } from '../services/meu.service';
```

2. Injeta no construtor

```
constructor(private meuService: MeuService) {}
```

3. Usa a função do service

```
msg = this.meuService.mensagem();
```

Compartilhar dados entre componentes usando service

No service:

```
import { Injectable, signal } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class CompartilharService {
  dado = signal("valor inicial");
}
```

Em qualquer componente que usar:

```
constructor(public compartilhado: CompartilharService) {}
```

No HTML:

```
<p>{{ compartilhado.dado() }}</p>
```

ROTAS (Routing)

Arquivo de rotas

```
import { Routes } from '@angular/router';

export const routes: Routes = [
  {
```

```
    path: '',               // rota inicial (http://localhost:4200)
    component: HomeComponent // componente que aparece nessa rota
  },

  {
    path: 'courses',        // rota (http://localhost:4200/courses)
    component: CoursesComponent
  },

  {
    path: 'courses/:id',    // rota com parâmetro (id do curso)
    component: CoursesComponent
  }
];
```

Ativar rotas no app

```
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],      // permite usar <router-outlet>
  templateUrl: './app.component.html'
})
export class AppComponent {}
```

Links de navegação

```
<a routerLink="/">Home</a>       <!-- manda pra rota inicial -->
<a routerLink="/courses">Cursos</a>
```

Rota com parâmetros

Para caso tipo:

`/produto/10`

No arquivo de rotas:

```
{ path: 'produto/:id', component: ProdutoComponent }
```

No componente, para acessar o ID:

```
import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
  const id = this.route.snapshot.paramMap.get('id');
  console.log(id);
}
```

TOOLBAR (Material)

```
<mat-toolbar color="primary">

  <!-- nome da aplicação -->
  <span>Minha Aplicação</span>

  <!-- empurra os itens para o canto direito -->
  <span class="spacer"></span>

  <!-- links dentro da toolbar -->
  <a mat-button routerLink="/">Home</a>
  <a mat-button routerLink="/courses">Cursos</a>
</mat-toolbar>
```

```
.spacer {
  flex: 1 1 auto;   /* cria espaço flexível */
}
```

TABLE

```
<table mat-table [dataSource]="courses">

  <!-- Coluna: nome -->
  <ng-container matColumnDef="name">
    <th mat-header-cell *matHeaderCellDef> Nome </th>
    <td mat-cell *matCellDef="let c">{{ c.nome }}</td>
  </ng-container>

  <!-- cabeçalho -->
  <tr mat-header-row *matHeaderRowDef="['name']"></tr>
  <!-- linhas -->
  <tr mat-row *matRowDef="let row; columns: ['name'];"></tr>
</table>
```

INTERFACE

```
export interface Course {
  id: number;       // id do curso
  nome: string;     // nome do curso
  categoria: string; // categoria
}
```

SERVICE AJAX + SUBSCRIBE

```
this.service.getAll().subscribe({
  next: (data) => this.courses = data,   // sucesso: recebe os cursos
  error: () => alert("Erro ao buscar cursos") // erro: mostra alerta
});
```

PROGRESS SPINNER

HTML:

```
<mat-progress-spinner
  *ngIf="loading"        <!-- aparece só enquanto 'loading' = true -->
  mode="indeterminate">
</mat-progress-spinner>
```

TS:

```
loading = true; // começa carregando

this.service.getAll().subscribe({
  next: (data) => {
    this.courses = data;   // guarda os dados
    this.loading = false;  // some o spinner
  }
});
```

TRATAMENTO DE ERROS

TS:

```
this.service.getAll().subscribe({
  next: data => this.courses = data,
  error: err => {
    console.error(err);       // mostra erro no console
    this.msgErro = "Falha ao carregar dados"; // mensagem na tela
  }
});
```

HTML:

```
<p *ngIf="msgErro" style="color:red">
  {{ msgErro }}
</p>
```

CUSTOMIZAR CATEGORIAS (Filtro)

HTML:

```
<select (change)="filtrar($event)"> <!-- dispara evento ao mudar -->
  <option value="">Todas</option>
  <option value="Front">Front</option>
  <option value="Back">Back</option>
</select>

<!-- lista filtrada na tela -->
<ul>
  <li *ngFor="let c of listaFiltrada">{{ c.nome }}</li>
</ul>
```

TS:

```
filtrar(event: any) {
  const cat = event.target.value;   // pega categoria escolhida

  this.listaFiltrada = cat
    ? this.courses.filter(c => c.categoria === cat) // filtra
    : this.courses;                                 // volta tudo
}
```

FORMULÁRIO

TS:

```
form = new FormGroup({
  nome: new FormControl(''),      // campo nome
  categoria: new FormControl('')  // campo categoria
});

//Para acessar:
console.log(form.value.nome);       // acessa o valor do input "nome"
console.log(form.value.categoria);  // acessa o valor do input "descricao"
```

HTML:

```
<form [formGroup]="form">       <!-- associa o FormGroup -->

  <input formControlName="nome" placeholder="Nome">       <!-- campo 1 -->
  <input formControlName="categoria" placeholder="Categoria"> <!-- campo 2 -->

  <button (click)="salvar()">Salvar</button>
</form>
```

VALIDAÇÃO

TS:

```
form = new FormGroup({
  nome: new FormControl('', Validators.required),         // obrigatório
  categoria: new FormControl('', [
    Validators.required,
    Validators.minLength(3)                              // mínimo 3 letras
  ])
});
```

HTML:

```
<!-- erro do nome -->
<div *ngIf="form.controls['nome'].invalid && form.controls['nome'].touched">
  Nome obrigatório
</div>
```

EDITAR

```
editar(curso: Course) {
  this.form.patchValue(curso); // joga os valores do curso no formulário
}
```

DELETE

```
apagar(id: number) {
  this.service.delete(id).subscribe(() => {

    // remove da lista sem precisar recarregar a página
    this.courses = this.courses.filter(c => c.id !== id);

  });
}
```

`ngIf` — Mostrar ou esconder algo

O `*ngIf` serve para **controlar se um elemento aparece ou não** no HTML.

Ele funciona assim:

```
<div *ngIf="condicao">
  Esse texto só aparece se a condicao for verdadeira.
</div>
```

Exemplo

No TS:

```
mostrar = true;
```

No HTML:

```
<p *ngIf="mostrar">Eu apareço!</p>
```

Se `mostrar = false`, **o elemento some do HTML**.

Não fica invisível — ele literalmente não existe mais na página.

`ngIf` com “senão”

```
<p *ngIf="logado; else naoLogado">Bem-vindo!</p>

<ng-template #naoLogado>
  <p>Faça login</p>
</ng-template>
```

`ngFor` — Repetir elementos a partir de uma lista

O `*ngFor` serve para **percorrer uma lista** e mostrar algo para cada item.

Estrutura básica:

```
<div *ngFor="let item of lista">
  {{ item }}
</div>
```

Exemplo

No TS:

```
nomes = ['Ana', 'Gabi', 'Lucas'];
```

No HTML:

```
<p *ngFor="let nome of nomes">
  {{ nome }}
</p>
```

ngFor com objetos

No TS:

```
produtos = [
  { nome: 'Teclado', preco: 120 },
  { nome: 'Mouse', preco: 80 }
];
```

No HTML:

```
<div *ngFor="let p of produtos">
  {{ p.nome }} - R$ {{ p.preco }}
</div>
```

Capturar quando o cliente aperta algo

HTML

```
<button (click)="fazerAlgo()">Clique aqui</button
```

TS

```
fazerAlgo() {
  console.log("O cliente clicou no botão!");
}
```

Capturar quando o cliente escolhe algo num select (combobox)

HTML

```
<select (change)="itemSelecionado($event)">
  <option value="pop">Pop</option>
  <option value="rock">Rock</option>
  <option value="kpop">K-pop</option>
</select>
```

TS

```
itemSelecionado(event: any) {
  const valor = event.target.value;
  console.log("O cliente escolheu:", valor);
  // ou tambem para input
  digitando(event: any) {
    console.log("O cliente digitou:", event.target.value);
   }

}
```

Se você quiser pegar o valor direto

Usando **ngModel**:

HTML

```
<input [(ngModel)]="nomeCliente" />

<button (click)="enviar()">Enviar</button>
```

TS

```
nomeCliente = "";

enviar() {
  console.log("O cliente digitou:", this.nomeCliente);
}
```

Como excluir um item específico de uma lista no TypeScript

Você mantém na lista **apenas os itens que NÃO são o que você quer excluir**.

Exemplo:

```
this.generos = this.generos.filter(item ⇒ item.id !== idParaExcluir);
```

👉 Se `idParaExcluir` for, por exemplo, `3`, então o item de id 3 sai da lista.
