



APLICAÇÕES INTERATIVAS

CORREÇÃO DE EXERCÍCIOS

Exercícios de **ngFor*

3. Crie o template de uma div contendo o título do post e seu conteúdo. Faça um array de títulos e um array de posts contendo 3 elementos cada. Faça com que os posts e títulos sejam exibidos na tela

Exercícios de **ngFor*

4. Crie um array de nomes vazio, junto a uma caixa de texto e um botão. Ao clicar no botão, o nome deve ser adicionado no array de nomes e aparecer numa lista de nomes ordenada (ol)

Exercícios de *ngSwitch*

1. Crie uma lista contendo canais do YouTube. Permita que o usuário pesquise o nome do canal numa caixa de texto. Se o canal estiver disponível, exiba apenas o nome do canal seguido de um link para acessá-lo

Exercícios de *ngSwitch*

2. Utilizando *ngSwitch*, construa um “menu” no *AppComponent* que permita ao usuário alternar entre os exercícios anteriores

CLASSES DE MODELO EM ANGULAR

Classes de Modelo

- Classes de modelo são utilizadas para representação de algum item do mundo real ou abstrato dentro do código
- Úteis para armazenamento, manipulação e passagem de valores entre os elementos da aplicação

Classes de Modelo

- Classes de modelo são representadas em Angular como uma classe pública (“exportada”) e contendo as propriedades correspondentes as necessidades do negócio
- Normalmente num arquivo separado

Exemplos de Classes de Modelo

```
export class Filme {  
  nome?: string;  
  nota?: number;  
}
```

```
export class Usuario {  
  nome?: string;  
  email?: string;  
}
```

```
export class Produto {  
  nome?: string;  
  preco?: number;  
  marca?: string;  
}
```

Propriedades Opcionais

```
export class Filme {  
  nome?: string;  
  nota?: number;  
}
```

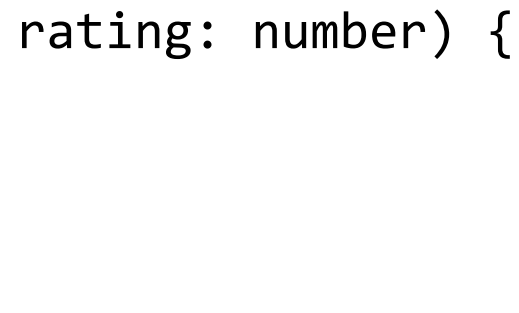
Indica uma propriedade opcional (não precisa ser inicializada ao ser declarada)
Também é possível usar notações como:

```
selMovie: Movie | null;  
ou  
selMovie: Movie | undefined;  
ou  
selMovie: Movie | null | undefined;
```

Propriedades Opcionais

```
export class Filme {  
    nome: string;  
    nota: number;
```

```
    constructor(name: string, rating: number) {  
        this.name = name;  
        this.rating = rating;  
    }  
}
```



Caso o construtor tenha parâmetros para inicializar as propriedades, não é necessário declará-las como opcionais

Utilizando Classes de Modelo

- Para serem utilizadas, classes de modelo precisam ser instanciadas
- Instanciamos uma classe em TypeScript utilizando a palavra reservada “new”

Utilizando Classes de Modelo

```
const alien = new Filme();
```

```
const parasita = new Filme();
```

CRUD

- Acrônimo para “Create, Read, Update, Delete” ou “Criar, Ler, Atualizar, Excluir”
- Normalmente refere-se a funcionalidade de cadastro (com todas as operações básicas padrão)

CRUD de Filmes

- Vamos trabalhar com os conceitos de classes de modelo através da construção de um CRUD simples de filmes
- O CRUD permitirá que o usuário cadastre, visualize, edite e exclua filmes de um catálogo

CRUD de Filmes

- Crie um novo projeto
- Chame-o de “movieapp”

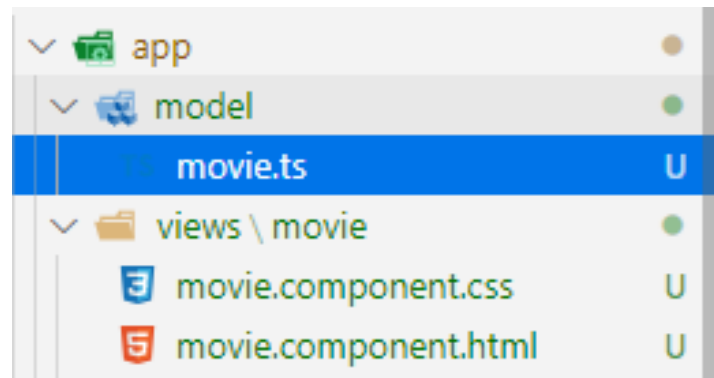
Componente

- Crie um novo componente na pasta “views”
- Chame-o de “movie”

Classe de Modelo

- Crie um pacote na pasta “app” chamado “model”
- Em seu interior, crie um arquivo chamado “movie.ts”

Classe de Modelo



Classe de Modelo

- Defina a classe de modelo “movie” como a seguir:

```
export class Movie {  
  name: string;  
  rating: number;  
  
  constructor(name: string, rating: number) {  
    this.name = name;  
    this.rating = rating;  
  }  
}
```

Vetor de Itens de Modelo

- Defina um vetor de itens de modelo na classe *“movie.component.ts”*, como a seguir

Vetor de Itens de Modelo

```
import { Component, OnInit } from '@angular/core';  
import { Movie } from 'src/app/model/movie';
```

```
@Component({  
  selector: 'app-movie',  
  templateUrl: './movie.component.html',  
  styleUrls: ['./movie.component.css']  
})  
export class MovieComponent implements OnInit {  
  
  movies = new Array<Movie>();  
  
  constructor() {  
  
  }  
  
  ngOnInit(): void {  
  
  }  
  
}
```

Adicionando Itens

- Para utilizar uma classe de modelo, é necessário instanciá-la (obter um objeto com valores)
- Cada objeto instanciado é único e independente dos demais

Adicionando Itens

- Como nosso vetor é um vetor de objetos da classe “Movie”, aceitará objetos desse tipo
- Desta forma, podemos criar vários filmes e adicioná-los ao vetor utilizando o método *push* do mesmo

Adicionando Itens

```
export class MovieComponent implements OnInit {  
  
    movies = new Array<Movie>();  
  
    constructor() {  
        // Criando um novo filme  
        const alien = new Movie('Alien - Oitavo Passageiro', 4);  
        this.movies.push(alien);  
  
        // Criando outro filme  
        const parasita = new Movie('Parasita', 5);  
        this.movies.push(parasita);  
    }  
  
    ...  
}
```

Listando Itens

- Utilize o `*ngFor` no HTML do componente de filmes para exibir uma listagem de filmes, como a seguir:

Listando Itens

```
<h2>Meus Filmes</h2>
```

```
<ul>
```

```
  <li *ngFor="let movie of movies">  
    {{movie.name}} - {{movie.rating}}
```

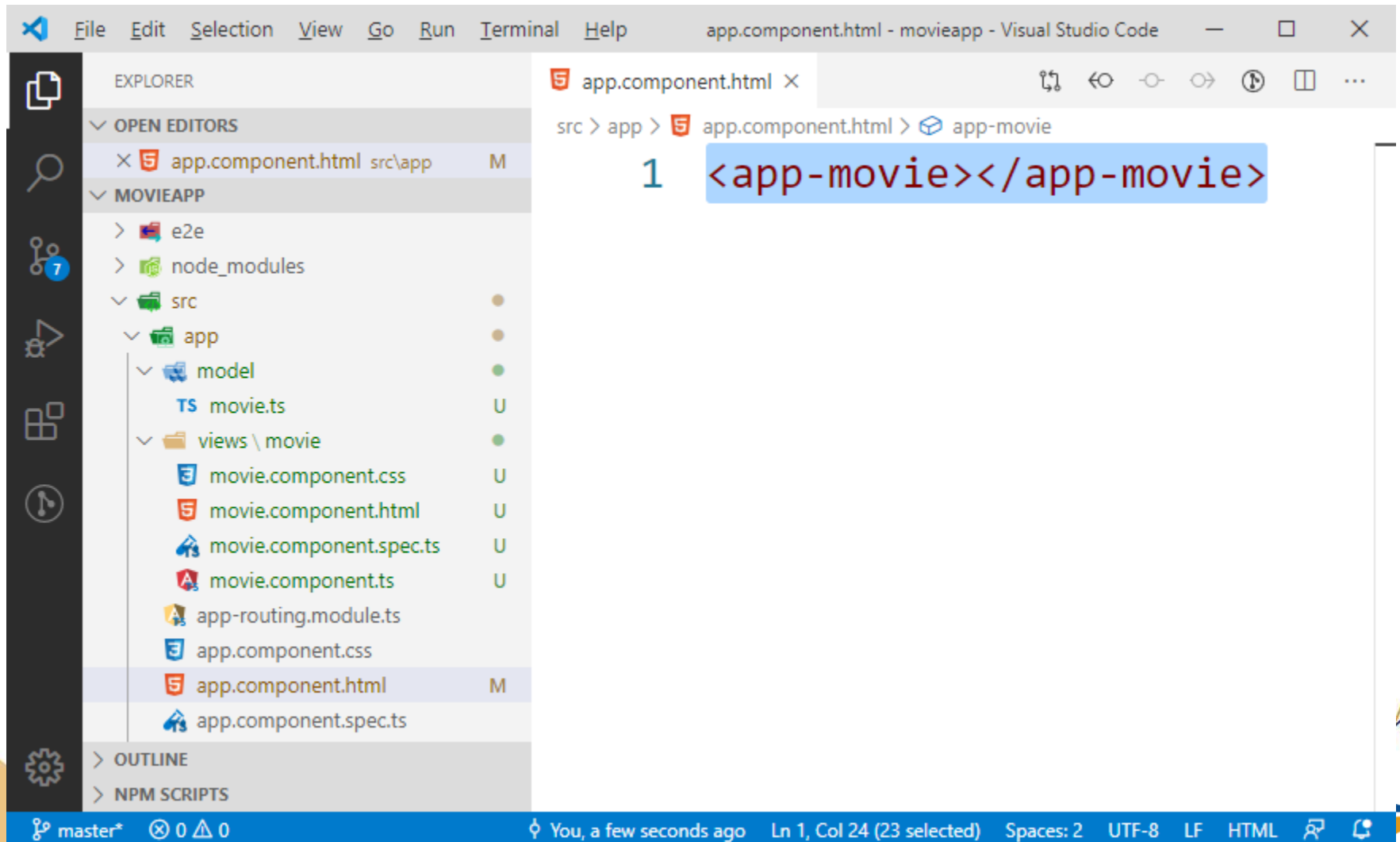
```
  </li>
```

```
</ul>
```

Utilizando Componente de Filmes

- Remova o conteúdo do componente principal (app-component) e adicione uma chamada ao componente de filmes, como a seguir:

Utilizando Componente de Filmes



The screenshot displays the Visual Studio Code interface. The Explorer sidebar on the left shows the project structure for 'movieapp'. The 'src' directory is expanded, revealing the 'app' directory, which contains a 'model' directory and a 'views' directory. The 'views' directory is further expanded, showing the 'movie' subdirectory. The file 'app.component.html' is selected in the Explorer and is also open in the editor. The editor shows the content of 'app.component.html', which is a single line of HTML: `<app-movie></app-movie>`. The status bar at the bottom indicates the current file is 'app.component.html' and the cursor is at line 1, column 24 (23 selected).

File Edit Selection View Go Run Terminal Help app.component.html - movieapp - Visual Studio Code

EXPLORER

OPEN EDITORS

- app.component.html src\app M

MOVIEAPP

- e2e
- node_modules
- src
 - app
 - model
 - TS movie.ts U
 - views \ movie
 - movie.component.css U
 - movie.component.html U
 - movie.component.spec.ts U
 - movie.component.ts U
 - app-routing.module.ts
 - app.component.css
 - app.component.html M
 - app.component.spec.ts

OUTLINE

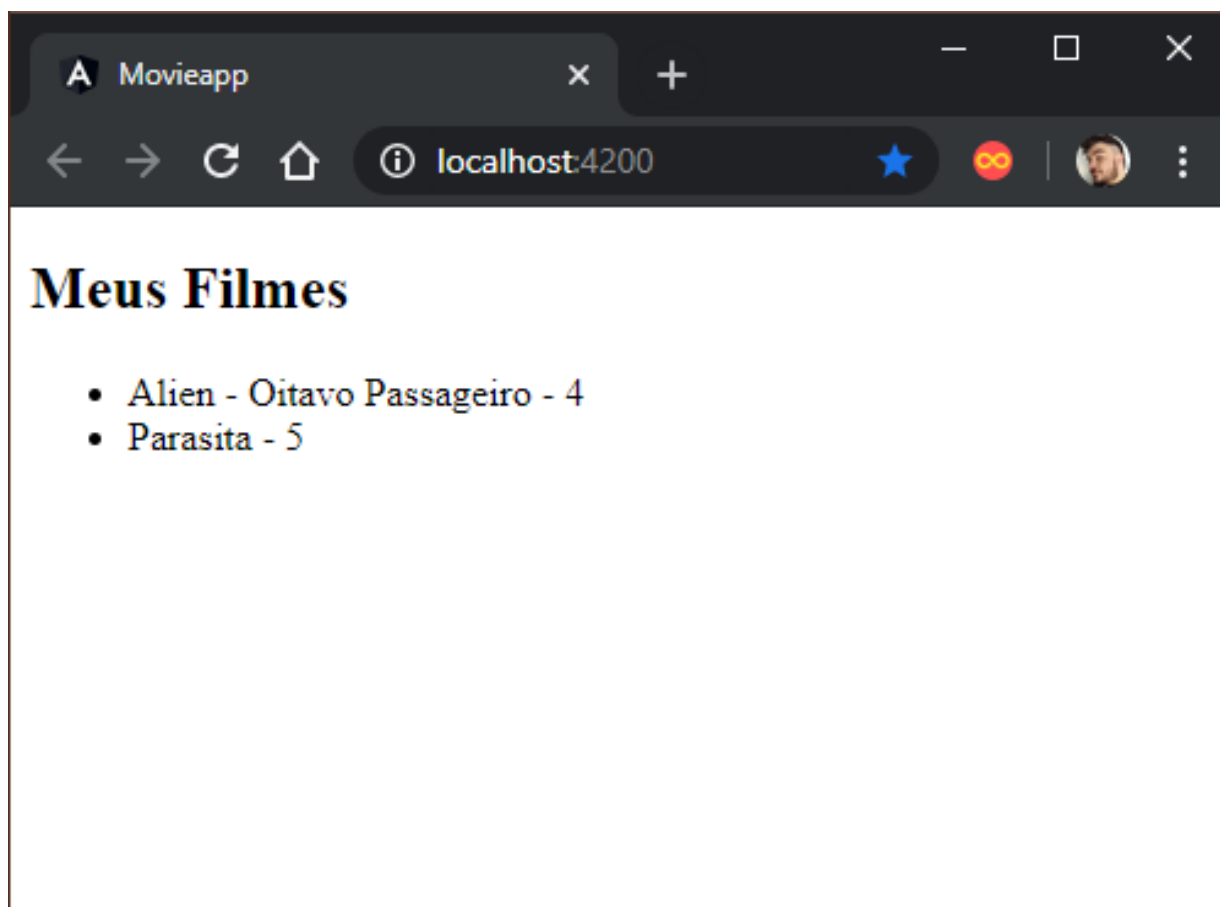
NPM SCRIPTS

src > app > app.component.html > app-movie

```
1 <app-movie></app-movie>
```

master* 0 0 You, a few seconds ago Ln 1, Col 24 (23 selected) Spaces: 2 UTF-8 LF HTML

Utilizando Componente de Filmes



Edição

- Crie uma nova propriedade no componente de filmes
- Ela representará o filme selecionado (para edição)

Edição

```
export class MovieComponent implements OnInit {  
  
    movies = new Array<Movie>();  
    selMovie?: Movie;  
  
    constructor() {  
        ...  
    }  
}
```

Edição

```
export class MovieComponent implements OnInit {
```

```
  movies = new Array<Movie>();
```

```
  selMovie?: Movie;
```

```
  constructor() {
```

```
    ...
```

Indica uma propriedade opcional (não precisa ser inicializada ao ser declarada)
Também é possível usar notações como:

```
selMovie: Movie | null;
```

ou

```
selMovie: Movie | undefined;
```

ou

```
selMovie: Movie | null | undefined;
```

Campos de Edição

- Utilize o elemento selMovie em campos de edição na página para criar uma área de edição, conforme a seguir:

Campos de Edição

```
<h2>Meus Filmes</h2>
```

```
<ul>
```

```
  <li *ngFor="let movie of movies">
    {{movie.name}} - {{movie.rating}}
```

```
  </li>
```

```
</ul>
```

```
<div *ngIf="selMovie">
```

```
  <input type="text" [(ngModel)]="selMovie.name">
```

```
  <br>
```

```
  <input type="number" [(ngModel)]="selMovie.rating">
```

```
</div>
```

FormsModule

- Adicione a importação ao FormsModule no `app-module.ts`, conforme a seguir:

FormsModule

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { MovieComponent } from './views/movie/movie.component';
import { FormsModule } from '@angular/forms';
```

```
@NgModule({
  declarations: [
    AppComponent,
    MovieComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Método de Seleção

- Faça um método de seleção de filme e associe-o ao evento de clique no HTML dos itens de lista:

Método de Seleção

...

```
ngOnInit(): void {  
  
}
```

```
selectMovie(movie: Movie): void {  
    this.selMovie = movie;  
}  
}
```


Método de Seleção

```
<h2>Meus Filmes</h2>
```

```
<ul>
```

```
  <li *ngFor="let movie of movies" (click)="selectMovie(movie);">
    {{movie.name}} - {{movie.rating}}
```

```
  </li>
```

```
</ul>
```

```
...
```

Botões de Ação

- Crie dois botões abaixo dos campos de texto, visando permitir a criação de um novo item ou cancelar a ação atual

Botões de Ação

...

```
<div *ngIf="selMovie">  
  <input type="text" [(ngModel)]="selMovie.name">  
  <br>  
  <input type="number" [(ngModel)]="selMovie.rating">  
</div>
```

```
<div>  
  <button (click)="newMovie();">New</button>  
  <button (click)="closeEdit();">Close</button>  
</div>
```

Botões de Ação

```
...
    selectMovie(movie: Movie): void {
        this.selMovie = movie;
    }

    newMovie(): void {
        this.selMovie = new Movie('', 0);
        this.movies.push(this.selMovie);
    }

    closeEdit(): void {
        this.selMovie = undefined;
    }
}
```

Botão de Exclusão

- Adicione o índice das posições no array
- Crie um botão dentro do “li” para chamar um método de remoção do item usando o índice
- Use o método de array “splice” no método de remoção do componente para excluir o item

Botão de Exclusão

```
...  
<ul>  
  <li *ngFor="let movie of movies; let i = index" (click  
k)="selectMovie(movie);">  
    {{movie.name}} - {{movie.rating}}  
    <button (click)="remove(i);">x</button>  
  </li>  
</ul>  
...
```

```
...  
  remove(pos: number): void {  
    this.movies.splice(pos, 1);  
  }  
...
```

"That's all Folks!"