

Olá!

Dando continuidade às atividades de integração com API REST CatalogoDeFilmes, em primeiro lugar, abra o VS Code, o próximo passo consiste em montar o seu formulário de cadastro de filme. Este deve conter cinco campos. São eles: nome do filme, gênero, ano de publicação, autor e data de nascimento todos do tipo texto. Já para o envio dos dados, você deve criar um botão do tipo submit. Lembre-se de que os dados informados aqui irão ser repassados ao serviço FilmeService e enviados a API REST CatalogoDeFilmes para cadastro. Com isso, o arquivo, para construção do formulário, encontra-se em `src/app/cadastrar-filme/cadastrar-filme.component.html` (`src barra app barra cadastrar traço filme barra cadastrar traço filme ponto component ponto html`).

Código

```
<h1>Cadastrar Filme</h1>
<form>
  <label> Nome do filme: </label>
  <br/>
  <input type="text" name="nome">
  <br/>

  <label> Gênero: </label>
  <br/>
  <input type="text" name="genero">
  <br/>

  <label> Ano publicação: </label>
  <br/>
  <input type="text" name="ano">
  <br/>

  <label> Autor: </label>
  <br/>
  <input type="text" name="autorNome">
  <br/>
```

```
<label> Data de Nascimento: </label>
<br/>
<input type="text" name="dataNascimento">
<br/>

<button type="submit">Salvar</button>
</form>
```

Portanto, fique atento! Para cadastro do formulário, é preciso informar que este será gerenciado pelo Angular. Isso é feito a partir da adição do atributo #f="ngForm" (hashtag f igual a aspas dupla ngForm aspas duplas) na tag de form. Diante disso, crie uma variável chamada f que contém todas as propriedades e valores vinculados ao formulário. Em seguida, inclua isso na tag form do formulário localizado em src/app/cadastrar-filme/cadastrar-filme.component.html (src barra app barra cadastrar traço filme barra cadastrar traço filme ponto component ponto html).

#### Código

```
<h1>Cadastrar Filme</h1>
<form #f="ngForm">
  <label> Nome do filme: </label>
  <br/>
  <input type="text" name="nome">
  <br/>

  <label> Gênero: </label>
  <br/>
  <input type="text" name="genero">
  <br/>

  <label> Ano publicação: </label>
  <br/>
  <input type="text" name="ano">
  <br/>
```

```
<label> Autor: </label>
```

```
<br/>
```

```
<input type="text" name="autorNome">
```

```
<br/>
```

```
<label> Data de Nascimento: </label>
```

```
<br/>
```

```
<input type="text" name="dataNascimento">
```

```
<br/>
```

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

```
</form>
```

O repasse dos dados do formulário para o componente é realizado a partir de evento. Para isso, o Angular possui a diretiva `ngSubmit`. Este é um atributo que está vinculado ao formulário e um método que se encontra declarado no componente. Na tag `form` do formulário, localizado em `src/app/cadastrar-filme/cadastrar-filme.component.html` (`src barra app barra cadastrar traço filme barra cadastrar traço filme ponto component ponto html`), inclua a diretiva `(ngSubmit)="onSubmit(f)"` (`abre parêntese ngSubmit fecha parêntese igual aspas duplas chamada da função onSubmit passando como parâmetro f aspas duplas`).

Ela é a responsável por repassar os dados do formulário para o método `onSubmit` do componente `cadastrar-filme` (`cadastrar traço filme`). Já o parâmetro `f` passado para o método `onSubmit` contém todas as propriedades e valores do formulário. Na prática, ele é um objeto que o representa.

Código

```
<h1>Cadastrar Filme</h1>
```

```
<form #f="ngForm" (ngSubmit)="onSubmit(f)">
```

```
<label> Nome do filme: </label>
```

```
<br/>
```

```
<input type="text" name="nome">
```

```
<br/>
```

```
<label> Gênero: </label>
```

```
<br/>
```

```
<input type="text" name="genero">
<br/>
```

```
<label> Ano publicação: </label>
<br/>
<input type="text" name="ano">
<br/>
```

```
<label> Autor: </label>
<br/>
<input type="text" name="autorNome">
<br/>
```

```
<label> Data de Nascimento: </label>
<br/>
<input type="text" name="dataNascimento">
<br/>
```

```
<button type="submit">Salvar</button>
</form>
```

Concluindo a configuração do formulário de cadastro de filmes, você precisa informar quais dos campos do formulário de cadastro devem ser enviados ao componente. Para isso, use a diretiva `ngModel`. Como é necessário enviar todos os campos para o componente, adicione a diretiva `ngModel` em todos os inputs do tipo `text`, como se fosse um atributo da tag.

#### Código

```
<h1>Cadastrar Filme</h1>
<form #f="ngForm" (ngSubmit)="onSubmit(f)">
  <label> Nome do filme: </label>
  <br/>
  <input type="text" name="nome" ngModel>
  <br/>

  <label> Gênero: </label>
```

```
<br/>
<input type="text" name="genero" ngModel>
<br/>

<label> Ano publicação: </label>
<br/>
<input type="text" name="ano" ngModel>
<br/>

<label> Autor: </label>
<br/>
<input type="text" name="autorNome" ngModel>
<br/>

<label> Data de Nascimento: </label>
<br/>
<input type="text" name="dataNascimento" ngModel>
<br/>

<button type="submit">Salvar</button>
</form>
```

Assim como a biblioteca HttpClient, as diretivas ngForm, ngSubmit, ngModel são disponibilizadas a partir de um módulo que é o FormsModule. Assim, navegue até o módulo AppModule em src/app/app.module.ts (src barra app barra app ponto module ponto ts) e, no decorator @NgModule (arroba NgModule), encontre a propriedade imports. Em seguida, adicione, a esta propriedade, o módulo FormsModule do pacote @angular/forms (arroba angular barra forms).

#### Código

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

import { AppComponent } from './app.component';
import { FilmeComponent } from './filmes/filme.component';
```

```
import { AutorComponent } from './autor/autor.component';
import { FilmeService } from './cadastrar-filme/filme.service';
import { HttpClientModule } from '@angular/common/http';
import { CadastrarFilmeComponent } from './cadastrar-filme/cadastrar-filme.component';
import { ConsultarFilmeComponent } from './consultar-filme/consultar-filme.component';

@NgModule({
  declarations: [
    AppComponent,
    FilmeComponent,
    AutorComponent,
    CadastrarFilmeComponent,
    ConsultarFilmeComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [FilmeService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Nessa etapa, injete, no componente cadastrar-filme (**cadastrar traço filme**), o serviço FilmeService a partir do construtor. Além disso, declare o método onSubmit e inclua nele um parâmetro chamado form. Este método é responsável por processar o evento de submit do botão do formulário de cadastro de filmes.

#### Código

```
import { Component, OnInit } from '@angular/core';
import { FilmeService } from './filme.service';
import { Filme } from './filme.model';
```

```
@Component({
```

```
selector: 'app-cadastrar-filme',
templateUrl: './cadastrar-filme.component.html',
styleUrls: ['./cadastrar-filme.component.css']
})
export class CadastrarFilmeComponent implements OnInit {

  filmeService: FilmeService;

  constructor(filmeService: FilmeService) {
    this.filmeService = filmeService;
  }

  ngOnInit(): void {
  }

  onSubmit(form) {
  }

}
```

Como o método `onSubmit` recebe, como parâmetro, o próprio formulário de cadastro de filmes, deve-se extrair, do parâmetro `form`, os dados encaminhados a partir do formulário pelo usuário e criar um objeto JSON do tipo `Filme` para que seja enviada uma requisição para a API REST `CatalogoDeFilmes` a partir do método `cadastrarFilmes` do serviço `FilmeService`. Para isso, crie uma variável local chamada `filme` do tipo `Filme`, utilizando o operador `let`, que possibilitará a você criar variáveis dentro de um escopo limitado que é o corpo do método `onSubmit`, ou seja, o ciclo de vida da variável `filme`, que está ligado a esse trecho de código.

#### Código

```
import { Component, OnInit } from '@angular/core';
import { FilmeService } from './filme.service';
import { Filme } from './filme.model';

@Component({
  selector: 'app-cadastrar-filme',
```

```
templateUrl: './cadastrar-filme.component.html',
styleUrls: ['./cadastrar-filme.component.css']
})
export class CadastrarFilmeComponent implements OnInit {

  filmeService:FilmeService;

  constructor(filmeService:FilmeService) {
    this.filmeService = filmeService;
  }

  ngOnInit(): void {
  }

  onSubmit(form) {
    let filme:Filme;

    filme = {
      id: null,
      nome: form.value.nome,
      genero: form.value.genero,
      ano: form.value.ano,
      autor: {
        id: null,
        nome: form.value.autorNome,
        dataNascimento: form.value.dataNascimento
      }
    }


    this.filmeService.cadastrarFilmes(filme).subscribe( dados => {
      console.log("Cadastrado com sucesso");
      console.log(dados);
    });
  }
}
```



```
}
```

Agora, chame o método `cadastrarFilmes` do serviço `FilmeService` para enviar uma requisição com os dados extraídos do formulário e enviá-los para API REST `CatalogoDeFilmes`. Para obter o retorno do servidor, use o método `subscribe`, pois ele irá fornecer o resultado. Em seguida, com o comando `console.log` (**console ponto log**) encaminhe os dados para o console do navegador.

#### Código

```
  
import { Component, OnInit } from '@angular/core';  
import { FilmeService } from './filme.service';  
import { Filme } from './filme.model';  
  
@Component({  
  selector: 'app-cadastrar-filme',  
  templateUrl: './cadastrar-filme.component.html',  
  styleUrls: ['./cadastrar-filme.component.css']  
})  
export class CadastrarFilmeComponent implements OnInit {  
  
  filmeService: FilmeService;  
  
  constructor(filmeService: FilmeService) {  
    this.filmeService = filmeService;  
  }  
  
  ngOnInit(): void {  
  }  
  
  onSubmit(form) {  
    let filme: Filme;  
  
    filme = {  
      id: null,  
      nome: form.value.nome,
```

```
genero: form.value.genero,
ano: form.value.ano,
autor: {
  id: null,
  nome: form.value.autorNome,
  dataNascimento: form.value.dataNascimento
}
}

this.filmeService.cadastrarFilmes(filme).subscribe( dados => {
  console.log("Cadastrado com sucesso");
  console.log(dados);
});
}

}
```

Realizada a ação anterior, inclua o componente cadastrar-filme (**cadast**) no arquivo `src/app/app.component.html` (**src barra app barra app ponto component ponto html**). Para isso, utilize o seletor que é a tag `app-cadastrar-filme` (**app traço cadastrar traço filme**). Depois, o Angular irá exibir a HTML relacionada a tag `app-cadastrar-filme` (**app traço cadastrar traço filme**) no seu navegador.

#### Código

```
<filme></filme>
<autor></autor>
<app-cadastrar-filme></app-cadastrar-filme>
<consultar-filme></consultar-filme>
```



Depois que criar e configurar o componente `CadastrarFilmeComponent` e o serviço `FilmeService` no VS Code, você precisa incluir, nos controller `AutorController` e `FilmeController` da API REST `CatalogoDeFilmes`, a anotação `@CrossOrigin` (**arroba crossorigin**). Essa diretiva permite que não exista nenhum bloqueio de comunicação entre suas aplicações em Spring MVC e Angular, que estão executando na sua máquina local no domínio `localhost`.

Agora, abra o Eclipse e navegue até o pacote `br.com.lead.controller` (`br ponto com ponto lead ponto controller`). Estando nele, acesse a classe `AutorController`. Na classe `AutorController`, antes da diretiva `@Controller` (`arroba controller`), adicione a anotação `@CrossOrigin` (`arroba crossorigin`). Além disso, na anotação `@CrossOrigin` (`arroba crossorigin`), inclua as propriedades `origins` e `allowedHeaders`. Ambas devem receber, como valor, uma string contendo um asterisco. Depois, vá até a classe `FilmeController` e antes da diretiva `@Controller` (`arroba controller`) adicione a anotação `@CrossOrigin` (`arroba crossorigin`). Além disso, na anotação `@CrossOrigin` (`arroba crossorigin`), inclua as propriedades `origins` e `allowedHeaders`. Ambas devem receber, como valor, uma string contendo um asterisco.

#### Código

```
package br.com.lead.controller;
```

```
import javax.persistence.EntityManager;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
import br.com.lead.modelo.Filme;
import br.com.lead.util.JPAUtil;
```

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
```

```
@Controller
```

```
public class FilmeController {
```

```
    @RequestMapping(value = "/persistir-filme", method = RequestMethod.POST, produces =
"application/json")
```

```
        @ResponseBody
```

```
        public Filme persistirFilme(@RequestBody Filme filme) {
```

```
EntityManager entityManager = JPAUtil.getEntityManager();

entityManager.getTransaction().begin();
entityManager.persist(filme.getAutor());
entityManager.persist(filme);
entityManager.getTransaction().commit();
entityManager.close();

return filme;
}

@RequestMapping(value = "/consultar-filme", method = RequestMethod.GET, produces =
"application/json")
@ResponseBody
public Filme consultarFilme(@RequestParam Integer id) {
    EntityManager entityManager = JPAUtil.getEntityManager();

    Filme filme = entityManager.find(Filme.class, id);

    return filme;
}
}
```

No Eclipse, navegue até o menu Run. Na opção Run As, escolha Run on Server. Isso irá iniciar a aplicação do CatalogoDeFilmes, baseada no Spring MVC. Após o servidor Tomcat ter sido iniciado, você deve ir ao VS Code e, no terminal, executar o comando `ng serve` (ng espaço serve) para que seja inicializada a aplicação Angular. Feito isso, será exibido, na sua tela, o formulário de cadastro de filmes.

#Audiodescrição: A imagem apresenta uma página com o formulário de cadastro. No topo, na barra de endereço, está inserido "localhost:4200". Logo abaixo, aparece o formulário com as informações: "Catálogo de Filmes. Filme: Matrix. Publicação: 1999. Autor. Nome: Lilly Wachowski. Data de Nascimento: 29/12/1967". Em seguida, temos "Cadastrar Filme" e seus respectivos

campos de edição: Nome do filme; Gênero; Ano publicação; Autor; Data de Nascimento; e o botão “Salvar”.

Preencha o formulário e acione o botão salvar do formulário, como resultado será impresso no console do navegador a mensagem cadastrado com sucesso e, junto a ela, os dados vindos do servidor em formato JSON. Para visualizar o resultado da requisição, vá ao menu visualizar do seu navegador e selecione a opção desenvolvedor, após isso acione a alternativa console javascript.

#Audiodescrição: A imagem apresenta a tela dividida entre a página com o formulário de cadastro e o console. À esquerda, aparece o formulário com as informações: Catálogo de Filmes. Filme: Matrix. Publicação: 1999. Autor. Nome: Lilly Wachowski. Data de Nascimento: 29/12/1967”. Em seguida, temos “Cadastrar Filme” e seus respectivos campos de edição preenchidos: Nome do filme: Matrix; Gênero: Ação; Ano publicação: 2000; Autor: Lilian; Data de Nascimento: 1964-08-13. Há também o botão Salvar. À direita, o console exibe a mensagem “Cadastrado com sucesso” e, logo abaixo, o código da página.

Com isso, você concluiu a etapa de criação do formulário. Portanto, até aqui, você aprendeu sobre serviços e como realizar injeção de dependência. Além disso, conheceu as diretivas do Angular para manipulação de formulários e a biblioteca HttpClient, que é utilizada para realizar requisições a API REST. Na próxima videoaula, você irá conhecer o recurso de rotas do Angular.

Bons estudos!

## Referências

AGOSTINHO, Danilo. **Criando uma aplicação (Internet Banking) com Angular 6: na prática e sem complicações – Parte 04**. Disponível em: <<https://imasters.com.br/front-end/criando-uma-aplicacao-internet-banking-com-angular-6-na-pratica-e-sem-complicacoes-parte-04>>. Acesso em: 22 jun 2020.

Angular 10/9/8 Service Tutorial with Example. Disponível em: <<https://www.positronx.io/angular-service-tutorial-with-example/>>. Acesso em: 22 jun 2020.

Angular. Disponível em: <<https://angular.io/api/forms/NgForm>>. Acesso em: 22 jun 2020.

Angular. Disponível em: <<https://angular.io/tutorial/toh-pt4>>. Acesso em: 22 jun 2020.

FERNANDES, Diego. **TypeScript: Vantagens, mitos, dicas e conceitos fundamentais.** Disponível em: <<https://blog.rocketseat.com.br/typescript-vantagens-mitos-conceitos/>>. Acesso em: 22 jun 2020.

MARIANO, Matheus. **O mínimo que você precisa saber sobre TypeScript.** Disponível em: <<https://medium.com/@matheusmariano/o-m%C3%ADnimo-que-voc%C3%AA-precisa-saber-sobre-typescript-58d1b418f78b>>. Acesso em: 22 jun 2020.

MARX, Lukas. **How to use \*ngIf else in Angular.** Disponível em: <<https://malcoded.com/posts/angular-ngif-else/>>. Acesso em: 22 jun 2020.

PEREIRA, Manacés. **Angular — RactiveForms — Validações Customizadas com formulários reativos no Angular 6.** Disponível em: <<https://medium.com/manacespereira/angular-ractiveforms-valida%C3%A7%C3%B5es-customizadas-com-formul%C3%A1rios-reativos-no-angular-6-3a3338f9add9>>. Acesso em: 22 jun 2020.

RABELO, Eduardo. **TypeScript: O guia definitivo.** Disponível em: <<https://medium.com/@oieduardorabelo/typescript-o-guia-definitivo-1a63b04259cc>>. Acesso em: 22 jun 2020.