6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

A – Criar uma aplicação Angular 4 com Node.js.

Vamos continuar aplicando o conceito do MEAN, só que desta vez implementaremos um projeto Angular 4. Este projeto também trata do cadastro de eventos. A diferença é que tanto a inclusão como o cadastro serão realizados através do webservice que desenvolvemos.

1. Na pasta **ProjetoEventos**, criar uma nova pasta chamada **Mean_Angular4**.
2. No prompt de comandos acessar a pasta recém criada.
3. Vamos criar um projeto chamado **appAngular4**, através do template **quickstart** disponível no github. Executar o comando:

```
git clone https://github.com/angular/quickstart.git appAngular4
```

4. Abrir o **VSCode** na pasta do projeto. No prompt de comandos, entrar nesta pasta também, pois executaremos a aplicação a partir dele. Lembrando que é possível usar o próprio VSCode para esta finalidade.
5. No prompt, executar (somente se o projeto foi gerado via quickstart):

```
npm install
```

6. Instalar a biblioteca do **bootstrap** e do **jquery**. Estes serão uteis na camada de visualização:

```
npm install bootstrap@3.3.7 --save
npm install jquery --save
```

7. Abrir o arquivo **index.html** (nossa página inicial) e configurar as bibliotecas **Bootstrap** e **JQuery** recém inseridas.

```
<!DOCTYPE html>
<html>
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
<head>
  <title>Angular QuickStart</title>
  <base href="/">
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet"
href="node_modules/bootstrap/dist/css/bootstrap.min.css">
  <script src="node_modules/jquery/dist/jquery.min.js"></script>
  <script src="node_modules/bootstrap/dist/js/bootstrap.min.js"></script>


  <link rel="stylesheet" href="styles.css">

  <!-- Polyfill(s) for older browsers -->
  <script src="node_modules/core-js/client/shim.min.js"></script>

  <script src="node_modules/zone.js/dist/zone.js"></script>
  <script src="node_modules/systemjs/dist/system.src.js"></script>

  <script src="systemjs.config.js"></script>
  <script>
    System.import('main.js').catch(function (err) { console.error(err);
});
  </script>
</head>

<body>
  <my-app>Loading AppComponent content here ...</my-app>
</body>

</html>
```

8. Nosso componente principal (**AppComponent**) define um texto, inserido no index.html. Vamos acrescentar dois componentes: um representando o menu de opções (**MenuComponent**) e outro, a página inicial de fato na nossa aplicação (**HomeComponent**), uma espécie de logotipo e apresentação da empresa. Para isso, criar pasta **menu**, abaixo de **app** (nossa raiz).
9. Na pasta **menu/views**, criar **menu.component.html** com o código abaixo:

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
<div class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle"
            data-toggle="collapse" data-target=".navbar-collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Impacta - Mean Stack</a>
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>
                    <a href="#">Home</a>
                </li>
                <li>
                    <a href="#">Gerenciar Eventos</a>
                </li>
            </ul>
        </div>
    </div>
</div>
```

10. Na pasta **menu**, definir o componente **menu.component.ts**:

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

@Component({
    moduleId: module.id,
    selector: 'menu',
    templateUrl: 'views/menu.component.html'
})
export class MenuComponent { }
```

11. abrir o arquivo **app.component.ts**, e realizar as alterações:

```
import { Component } from '@angular/core';
import { MenuComponent } from './menu/menu.component';
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
@Component({
  selector: 'my-app',
  template: '<menu></menu>'
})
export class AppComponent { }
```

12. Devemos agora configurar o novo componente em **app.module.ts**. Neste arquivo configuramos todos os componentes, serviços e módulos da aplicação:

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

import { AppComponent } from './app.component';
import { MenuComponent } from './menu/menu.component';

@NgModule({
  imports:      [ BrowserModule ],
  declarations: [ AppComponent, MenuComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

O componente principal incluirá outro componente, no caso, o **MenuComponent**. Para que isso seja possível, é necessário incluí-lo no metadata **declarations.**

13. Executar a aplicação. No prompt (na pasta do projeto) executar o comando (somente se criou o projeto pelo quickstart):

```
npm start
```

As alterações realizadas no projeto não requerem que este comando seja executado novamente: um dos componentes do Angular 4 é o BrowserLink, usado para recarregar a página automaticamente. Este componente monitora as alterações no código e recarrega o componente.

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

14. Até agora, nossas classes não possuíam nenhum código. No nosso exemplo, o conteúdo dos links será obtido a partir de variáveis definidas na classe. Veja as alterações realizadas na classe **MenuComponent** e no arquivo **menu.component.html**:

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

@Component({
    moduleId: module.id,
    selector: 'menu',
    templateUrl: 'views/menu.component.html'
})
export class MenuComponent {
    titulo_empresa: string = "Impacta Treinamentos";
    titulo_home: string = "Home";
    titulo_principal: string = "Gestão de Eventos";
}
```

```
<div class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle"
data-toggle="collapse" data-target=".navbar-collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand"
                    [routerLink]="['/']">{{titulo_empresa}}</a>
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>
                    <a [routerLink]="['/home']">{{titulo_home}}</a>
                </li>
                <li>
                    <a [routerLink]="['/eventos']">{{titulo_principal}}</a>
                </li>
            </ul>
        </div>
    </div>
```

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

15. Alterar o conteúdo do arquivo **styles.css**:

```
h1 {
  color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
}

.margem {
  margin-top: 50px;
}
```

16. Vamos definir a página inicial da aplicação, o home. Criar uma pasta chamada **home**, abaixo de **app.** Nesta pasta definir o arquivo **home.component.ts**:

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

@Component({
    template: `
     <div class="container margem">
        <h1>PÁGINA INICIAL</h1>
     </div>
     `
})
export class HomeComponent { }
```

17. Criar a pasta **cadastro** e, nela, o arquivo **cadastro.component.ts**:

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

@Component({
    template: `
     <div class="container margem">
        <h1>CADASTRO DE EVENTOS</h1>
     </div>
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
    `
})
export class CadastroComponent { }
```

18. Incluir um componente para representar uma URL inválida (erro 404). Criar uma pasta chamada **erro** e, nela. o arquivo **notfound.component.ts**:

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

@Component({
    template: `
     <div class="container margem">
        <h1>ERRO 404 - PÁGINA NÃO LOCALIZADA</h1>
     </div>
     `
})
export class NotFoundComponent { }
```

19. Vamos definir as rotas da aplicação. Criar uma pasta chamada **rotas** e, nesta pasta, o arquivo **app.routes.ts**. Analise o conteúdo:

```
import { Routes } from '@angular/router';
import { HomeComponent } from '../home/home.component';
import { CadastroComponent } from '../cadastro/cadastro.component';
import { NotFoundComponent } from '../erro/notFound.component';

export const appRoutes: Routes = [
    { path: "", component: HomeComponent },
    { path: "eventos", component: CadastroComponent },
    { path: "home", component: HomeComponent },
    { path: "**", component: NotFoundComponent }
];
```

20. O próximo passo é alterar o módulo da aplicação, arquivo **app.module.ts**. Analise as alterações realizadas:

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

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
import { appRoutes } from './rotas/app.routes'; //deve vir primeiro

import { AppComponent } from './app.component';
import { MenuComponent } from './menu/menu.component';

//usado nas rotas
import { HomeComponent } from './home/home.component';
import { CadastroComponent } from './cadastro/cadastro.component';
import { NotFoundComponent } from './erro/notFound.component';


@NgModule({
  //lista os modulos que a aplicação necessitara
  imports: [BrowserModule, RouterModule.forRoot(appRoutes)],

  //lista os componentes que nossa aplicação utilizará
  declarations: [AppComponent,
    MenuComponent,
    HomeComponent,
    CadastroComponent,
    NotFoundComponent],

  //este é o componente inicial, incluído no index.html
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

21. Testar a aplicação até este ponto, executando cada uma das rotas. O que é possível verificar?
22. Para começar a manipular os dados, vamos criar uma estrutura para armazenar as informações do evento. Pra tanto, vamos criar uma interface. Esta interface ser chamará **IEvento**. Portanto, criar o arquivo **interface.evento.ts** na pasta **interfaces**:

```
export interface IEvento {
    descricao: string;
    data: string;
    preco: number;
}
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

23. Na pasta **cadastro**, criar uma subpasta chamada **views**. Nesta pasta criar o arquivo **cadastro.component.html**:

```
<div class="container margem">
    <h1>Cadastro de Eventos</h1>
    <hr/>
    <div class="row">
        <div class="col-md-6">
            <ul class="list_group">
                <li *ngFor="let item of listaEventos"
                          class="list-group-item">
                    <a href="#">{{item.descricao}}</a>
                </li>
            </ul>
        </div>

        <div class="col-md-6">
            <div class="form-group">
                <label for="descricao">Descrição:</label>
                <input type="text" class="form-control"
                   id="descricao" name="descricao">
            </div>
            <div class="form-group">
                <label for="data">Data:</label>
                <input type="text" class="form-control" id="data"
                   name="data">
            </div>
            <div class="form-group">
                <label for="preco">Preço:</label>
                <input type="number" class="form-control" id="preco"
                   name="preco">
            </div>

            <div class="form-group">
                <button type="button" class="btn btn-info">
                    <span class="glyphicon glyphicon-pencil"></span>
                    Incluir
                </button>
            </div>
        </div>
    </div>
</div>
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

24. No arquivo **cadastro.component.ts**, criar uma lista de eventos provisória. Faça as alterações sugeridas no código a seguir:

```
import { Component } from '@angular/core';
import { IEvento } from './../interfaces/interface.evento';

@Component({
    moduleId: module.id,
    templateUrl: 'views/cadastro.component.html'

})
export class CadastroComponent {
    //definindo um array de eventos

    public listaEventos: IEvento[] = [
        { descricao:'Avaliação Angular', data: '23/10/2018', preco:0 },
        { descricao: 'Formatura', data: '02/05/2020', preco: 140 },
        { descricao: 'Torneio de Tenis', data: '10/07/2018', preco: 210 },
        { descricao: 'Congresso de TI', data: '16/01/2019', preco: 400 }
    ];
 }
```

25. Execute a aplicação e acesse **Gestão de Eventos**. É possível visualizar a lista de eventos do lado esquerdo e um formulário do lado direito.
26. Tente acessar na URL um link inexistente. Verifique se o **componente NotFoundComponent** é renderizado.
27. Vamos transferir os dados da lista de eventos para um serviço. Para tanto, criar uma pasta chamada **services**. Nesta pasta criar o arquivo **eventos.service.ts**:

```
import { Injectable } from '@angular/core';
import { IEvento } from '../interfaces/interface.evento';

@Injectable()
export class EventosService {
    public getEventos(): IEvento[] {
        return [
            { descricao: 'Avaliação Angular', data: '23/10/2018', preco: 0 },
            { descricao: 'Formatura', data: '02/05/2020', preco: 140 },
            { descricao: 'Torneio de Tenis', data: '10/07/2018', preco: 210},
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
            { descricao: 'Congresso de TI', data: '16/01/2019', preco: 400 }
        ];
    }
}
```

28. No arquivo **cadastro.component.ts**, fazer as alterações indicadas:

```
import { Component } from '@angular/core';
import { IEvento } from './../interfaces/interface.evento';
import { EventosService } from '../services/eventos.service';

@Component({
    moduleId: module.id,
    templateUrl: 'views/cadastro.component.html'

})
export class CadastroComponent {
    public listaEventos: IEvento[];
    constructor(eventosService: EventosService) {
        this.listaEventos = eventosService.getEventos();
    }
 }
```

29. No arquivo **app.module.ts**, incluir o serviço com provider:

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

import { appRoutes } from './rotas/app.routes'; //deve vir primeiro

import { AppComponent } from './app.component';
import { MenuComponent } from './menu/menu.component';

//usado nas rotas
import { HomeComponent } from './home/home.component';
import { CadastroComponent } from './cadastro/cadastro.component';
import { NotFoundComponent } from './erro/notFound.component';
import { EventosService } from './services/eventos.service';

@NgModule({
```

```
  //lista os modulos que a aplicação necessitara
  imports: [BrowserModule, RouterModule.forRoot(appRoutes)],

  //lista os componentes que nossa aplicação utilizará
  declarations: [AppComponent,
    MenuComponent,
    HomeComponent,
    CadastroComponent,
    NotFoundComponent],

  providers : [ EventosService ],
  //este é o componente inicial, incluído no index.html
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

30. Agora vamos selecionar um item na lista de eventos e vê-lo no formulário e, na medida que alterarmos alguma função no formulário, vê-la na lista. Este processo caracteriza um vínculo bidirecional, ou um binding bidirecional. Para usar o binding bidirecional é necessário habilitar o uso do atributo **ngModel**. Para tanto, realizar as alterações no arquivo **app.module.ts:**

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

import { appRoutes } from './rotas/app.routes'; //deve vir primeiro

import { AppComponent } from './app.component';
import { MenuComponent } from './menu/menu.component';

//usado nas rotas
import { HomeComponent } from './home/home.component';
import { CadastroComponent } from './cadastro/cadastro.component';
import { NotFoundComponent } from './erro/notFound.component';
import { EventosService } from './services/eventos.service';

@NgModule({
  //lista os modulos que a aplicação necessitara
  imports: [BrowserModule, RouterModule.forRoot(appRoutes), FormsModule],
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
  //lista os componentes que nossa aplicação utilizará
  declarations: [AppComponent,
    MenuComponent,
    HomeComponent,
    CadastroComponent,
    NotFoundComponent],

  providers : [ EventosService ],
  //este é o componente inicial, incluído no index.html
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

31. Realizar agora as alterações em **cadastro.component.ts**:

```
import { Component } from '@angular/core';
import { IEvento } from './../interfaces/interface.evento';
import { EventosService } from '../services/eventos.service';

@Component({
    moduleId: module.id,
    templateUrl: 'views/cadastro.component.html'

})
export class CadastroComponent {
    //para um evento selecionado
    public eventoSelecionado: IEvento;

    public selecionar(item: IEvento): void {
        this.eventoSelecionado = item;
    }

    //lista de eventos
    public listaEventos: IEvento[];
    constructor(eventosService: EventosService) {
        this.listaEventos = eventosService.getEventos();
    }
 }
```

32. Em seguida, providenciar as alterações no arquivo **cadastro.component.html**:

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
<div class="container margem">
    <h1>Cadastro de Eventos</h1>
    <hr/>
    <div class="row">
        <!-- incluindo evento click na lista -->
        <div class="col-md-6">
            <ul class="list_group">
                <li *ngFor="let item of listaEventos"
                        (click)="selecionar(item)" class="list-group-item">
                    <a [class.selecionado]="item === eventoSelecionado"
                    [routerLink]="['/eventos']">{{item.descricao}}</a>
                </li>
            </ul>
        </div>

        <!-- incluindo o formulário somente se houver elemento selecionado --
>
        <!-- nova div envolvendo <div class="col-md-6"> -->
        <div *ngIf="eventoSelecionado">
        <div class="col-md-6">
            <div class="form-group">
                <label for="descricao">Descrição:</label>
                <input type="text" [(ngModel)]="eventoSelecionado.descricao"
                    class="form-control" id="descricao" name="descricao">
            </div>
            <div class="form-group">
                <label for="data">Data:</label>
                <input type="text" [(ngModel)]="eventoSelecionado.data"
                    class="form-control" id="data" name="data">
            </div>
            <div class="form-group">
                <label for="preco">Preço:</label>
                <input type="number" [(ngModel)]="eventoSelecionado.preco"
                    class="form-control" id="preco" name="preco">
            </div>

            <div class="form-group">
                <button type="button" class="btn btn-info">
                    <span class="glyphicon glyphicon-pencil"></span>
                    Incluir
                </button>
            </div>
        </div>
        </div>
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
    </div>
</div>
```

33. É interessante que o usuário tenha uma pista de qual item foi selecionado. Vamos tornar negrito o item selecionado. Adicionar em **styles.css** o código:

```
h1 {
  color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
}

.margem {
  margin-top: 50px;
}
```

```
.selecionado {
  font-weight: bold;
}
```

34. Testar esta nova funcionalidade.
35. Vamos agora permitir a inclusão de um novo item a ser cadastrado (pois só podemos visualizar uma lista pronta!). No arquivo **cadastro.component.ts**, realizar a seguinte alteração:

```
import { Component } from '@angular/core';
import { IEvento } from './../interfaces/interface.evento';
import { EventosService } from '../services/eventos.service';

@Component({
    moduleId: module.id,
    templateUrl: 'views/cadastro.component.html'

})
export class CadastroComponent {
    //para um evento selecionado
    public eventoSelecionado: IEvento;
```

```
    private novoEvento: IEvento;

    //para a inclusão de um novo evento
    public novo() {
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
        this.novoEvento = { descricao: '', data:'',preco:0 }
        this.eventoSelecionado = this.novoEvento;
    }

    public incluir(evento: IEvento) {
        this.listaEventos.push(evento);
        alert('Evento incluído com sucesso');
    }

    public selecionar(item: IEvento): void {
        this.eventoSelecionado = item;
    }

    //lista de eventos
    public listaEventos: IEvento[];
    constructor(eventosService: EventosService) {
        this.listaEventos = eventosService.getEventos();
    }
}
```

36. no arquivo **cadastro.component.html**, realizar as alterações (observe a inclusão de um novo botão e da chamada aos eventos):

```
<div class="container margem">
    <h1>Cadastro de Eventos</h1>
    <hr/>
    <div class="row">
        <!-- incluindo evento click na lista -->
        <div class="col-md-6">
            <ul class="list_group">
                <li *ngFor="let item of listaEventos"
                        (click)="selecionar(item)" class="list-group-item">
                    <a [class.selecionado]="item === eventoSelecionado"
                    [routerLink]="['/eventos']">{{item.descricao}}</a>
                </li>
            </ul>
```

```
            <button type="button" (click)="novo()" class="btn btn-info">
                <span class="glyphicon glyphicon-plus"></span>
                Novo
            </button>
```

```
        </div>

        <!-- incluindo o formulário somente se houver elemento selecionado --
>
        <!-- nova div envolvendo <div class="col-md-6"> -->
        <div *ngIf="eventoSelecionado">
        <div class="col-md-6">
            <div class="form-group">
                <label for="descricao">Descrição:</label>
                <input type="text" [(ngModel)]="eventoSelecionado.descricao"
                    class="form-control" id="descricao" name="descricao">
            </div>
            <div class="form-group">
                <label for="data">Data:</label>
                <input type="text" [(ngModel)]="eventoSelecionado.data"
                    class="form-control" id="data" name="data">
            </div>
            <div class="form-group">
                <label for="preco">Preço:</label>
                <input type="number" [(ngModel)]="eventoSelecionado.preco"
                    class="form-control" id="preco" name="preco">
            </div>

            <div class="form-group">
                <button type="button" (click)="incluir(eventoSelecionado)"
                    class="btn btn-info">
                    <span class="glyphicon glyphicon-pencil"></span>
                    Incluir
                </button>
            </div>
        </div>
        </div>

    </div>
</div>
```

B – Aplicar filtros nos componentes.

Filtros são elementos que auxiliam na apresentação da view para o usuário. Através de filtros podemos formatar datas, horas, apresentar subconjuntos, retornar textos em maiúsculo, etc. Podemos usar filtros padrão, ou criarmos os nossos próprios. Os principais filtros padrão são:

uppercase: retorna a propriedade em maiúsculo.

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

lowercase: retorna a propriedade em minúsculo.

percent: apresenta um valor decimal com símbolo de porcentagem (%).

currency: especifica o valor da propriedade com moeda local.

date: apresenta um dado no formato de data (se compatível).

1. No arquivo **cadastro.component.html**, aplicar o filtro uppercase na descrição:

```
<!-- incluindo evento click na lista -->
<div class="col-md-6">
    <ul class="list_group">
        <li *ngFor="let item of listaEventos"
                (click)="selecionar(item)" class="list-group-item">
            <a [class.selecionado]="item == eventoSelecionado"
            [routerLink]="['/eventos']">
                {{item.descricao | uppercase}}</a>
        </li>
    </ul>

    <button type="button" (click)="novo()" class="btn btn-info">
        <span class="glyphicon glyphicon-plus"></span>
        Novo
    </button>

</div>
```

2. Visualize.
3. Vamos criar nosso próprio filtro. O objetivo é obter uma sublista de eventos, com base em um texto digitado em um campo de textos. Criar uma pasta chamada **filters**. Na pasta **filters**, criar o arquivo **sublista.filter.ts:**

```
import { Pipe, PipeTransform } from '@angular/core';
import { IEvento } from '../interfaces/interface.evento';
@Pipe({
    name: 'sublista'
})
export class SubLista implements PipeTransform {
    transform(eventos: IEvento[], input: string) : IEvento[] {
        //usando aerofunction (similar a delegates do c#)
```

```
            return eventos.filter(
                evento =>

evento.descricao.toLowerCase().includes(input.toLowerCase()));
    }
}
```

4. Alterar o arquivo **app.module.ts:**

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

import { appRoutes } from './rotas/app.routes'; //deve vir primeiro

import { AppComponent } from './app.component';
import { MenuComponent } from './menu/menu.component';
import { SubLista } from './filters/sublista.filter';

//usado nas rotas
import { HomeComponent } from './home/home.component';
import { CadastroComponent } from './cadastro/cadastro.component';
import { NotFoundComponent } from './erro/notFound.component';
import { EventosService } from './services/eventos.service';

@NgModule({
  //lista os modulos que a aplicação necessitara
  imports: [BrowserModule, RouterModule.forRoot(appRoutes), FormsModule],

  //lista os componentes que nossa aplicação utilizará
  declarations: [AppComponent,
    MenuComponent,
    HomeComponent,
    CadastroComponent,
    NotFoundComponent, SubLista],

  providers : [ EventosService ],
  //este é o componente inicial, incluído no index.html
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

5. No arquivo **cadastro.component.html**, realizar as alterações a seguir (incluir a caixa de textos indicada):

```
<div class="container margem">
    <h1>Cadastro de Eventos</h1>
    <hr/>
    <div class="row">
        <!-- incluindo evento click na lista -->
        <div class="col-md-6">

            <div class="form-group">
                <label for="codigo">Informe a descrição para filtrar:</label>
                <input type="text" #busca (keyup)="0" class="form-control" >
            </div>


            <ul class="list_group">
                <li *ngFor="let item of listaEventos | sublista:busca.value"
                    (click)="selecionar(item)" class="list-group-item">
                    <a [class.selecionado]="item == eventoSelecionado"
                    [routerLink]="['/eventos']">{{item.descricao |
                          uppercase}}</a>
                </li>
            </ul>

            <button type="button" (click)="novo()" class="btn btn-info">
                <span class="glyphicon glyphicon-plus"></span>
                Novo
            </button>

        </div>

       //continua....
```

6. Testar a aplicação com filtros.

C – Obter a lista de eventos do webservice appEventos.

7. Em **app.module.ts**, importar e configurar o modulo **HttpModule**:

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

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

import { HttpModule } from '@angular/http';

import { appRoutes } from './rotas/app.routes'; //deve vir primeiro

import { AppComponent } from './app.component';
import { MenuComponent } from './menu/menu.component';
import { SubLista } from './filters/sublista.filter';

//usado nas rotas
import { HomeComponent } from './home/home.component';
import { CadastroComponent } from './cadastro/cadastro.component';
import { NotFoundComponent } from './erro/notFound.component';
import { EventosService } from './services/eventos.service';

@NgModule({
  //lista os modulos que a aplicação necessitara
  imports: [BrowserModule, RouterModule.forRoot(appRoutes),
    FormsModule, HttpModule],

  //lista os componentes que nossa aplicação utilizará
  declarations: [AppComponent,
    MenuComponent,
    HomeComponent,
    CadastroComponent,
    NotFoundComponent, SubLista],

  providers : [ EventosService ],
  //este é o componente inicial, incluído no index.html
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

8. Em **eventos.service.ts**, realizar as alterações:

```
import { Injectable } from '@angular/core';
import { IEvento } from '../interfaces/interface.evento';
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import 'rxjs/Rx';
```

```
@Injectable()
export class EventosService {
```

```
    //acesso ao HTTP
    public constructor(private _http: Http) { }
    private url: string = "http://localhost:3200/eventos";

    public getEventosWS(): Observable<IEvento[]> {
        return this._http.get(this.url)
            .map(res => res.json());
    }
```

```
    public getEventos(): IEvento[] {
        return [
            { descricao: 'Avaliação Angular', data: '23/10/2018', preco: 0 },
            { descricao: 'Formatura', data: '02/05/2020', preco: 140 },
            { descricao: 'Torneio de Tenis', data: '10/07/2018', preco:210 },
            { descricao: 'Congresso de TI', data: '16/01/2019', preco: 400 }
        ];
    }
}
```

9. Em **cadastro.component.ts**, realizar as alterações:

```
import { Component } from '@angular/core';
import { IEvento } from './../interfaces/interface.evento';
import { EventosService } from '../services/eventos.service';

@Component({
    moduleId: module.id,
    templateUrl: 'views/cadastro.component.html'

})
export class CadastroComponent {
    //para um evento selecionado
    public eventoSelecionado: IEvento;

    private novoEvento: IEvento;
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
    //para a inclusão de um novo evento
    public novo() {
        this.novoEvento = { descricao: '', data:'',preco:0 }
        this.eventoSelecionado = this.novoEvento;
    }

    public incluir(evento: IEvento) {
        this.listaEventos.push(evento);
        alert('Evento incluído com sucesso');
    }

    public selecionar(item: IEvento): void {
        this.eventoSelecionado = item;
    }

    //lista de eventos
    public listaEventos: IEvento[];
    constructor(eventosService: EventosService) {
        //this.listaEventos = eventosService.getEventos();

        eventosService.getEventosWS()
            .subscribe(res => this.listaEventos = res,
            error => alert(error),
            () => console.log('finalizado'));
    }
}
```

10. Verificar na execução da aplicação que agora a lista contempla os eventos do webservice, e não mais da lista fictícia.
11. Nossa tarefa agora é enviar dados para o webservice com Angular 4. No arquivo **eventos.service.ts**, realizar as alterações:

```
import { Injectable } from '@angular/core';
import { IEvento } from '../interfaces/interface.evento';

import { Http, Response, Headers, RequestOptions } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import 'rxjs/Rx';


@Injectable()
export class EventosService {
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
    //acesso ao HTTP
    public constructor(private _http: Http) { }

    private url: string = "http://localhost:3200/eventos";

    public getEventosWS(): Observable<IEvento[]> {

        return this._http.get(this.url)
            .map(res => res.json());
    }
```

```
    public setEventoWS(evento: IEvento): Observable<IEvento> {
        let header = new Headers({ 'Content-Type': 'application/json' });
        let options = new RequestOptions({ headers: header });

        let json = JSON.stringify(
            {
                descricao: evento.descricao,
                data: evento.data,
                preco: evento.preco
            });

        return this._http.post(this.url, json, options)
            .map(res => res.json());
    }
```

```
    public getEventos(): IEvento[] {
        return [
            { descricao: 'Avaliação Angular', data: '23/10/2018', preco: 0 },
            { descricao: 'Formatura', data: '02/05/2020', preco: 140 },
            { descricao: 'Torneio de Tenis', data: '10/07/2018', preco:210 },
            { descricao: 'Congresso de TI', data: '16/01/2019', preco: 400 }
        ];
    }
}
```

12. Em **cadastro.component.ts**, alterar o método incluir() para acessar o método do webservice. Verifique que existem outras alterações a serem feitas:

```
import { Component } from '@angular/core';
import { IEvento } from './../interfaces/interface.evento';
import { EventosService } from '../services/eventos.service';

@Component({
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

```
    moduleId: module.id,
    templateUrl: 'views/cadastro.component.html'

})
export class CadastroComponent {
    //para um evento selecionado
    public eventoSelecionado: IEvento;
    private novoEvento: IEvento;

    //para a inclusão de um novo evento
    public novo() {
        this.novoEvento = { descricao: '', data:'',preco:0 }
        this.eventoSelecionado = this.novoEvento;
    }
```

```
    public incluir(evento: IEvento) {
        //this.listaEventos.push(evento);
        this.eventosService.setEventoWS(evento)
            .subscribe(res => JSON.stringify(res),
            error => alert(error),
            () => this.listar());
        alert('Evento incluído com sucesso');
    }
```

```
    public selecionar(item: IEvento): void {
        this.eventoSelecionado = item;
    }

    //lista de eventos
    public listaEventos: IEvento[];
```

```
    constructor(private eventosService: EventosService) {
        //this.listaEventos = eventosService.getEventos();
        this.listar();
    }

    public listar(): void{
        this.eventosService.getEventosWS()
            .subscribe(res => this.listaEventos = res,
            error => alert(error),
            () => console.log('finalizado'));
    }
}
```

6. Definindo uma aplicação com Angular 4 - MEAN (Parte 2)

13. Testar a aplicação.