Programação WEB III

CRUD completo com Angular 18

Prof. Dr. Danilo Barbosa

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Conteúdo

- Criar Projeto Angular 18
- Instalar Bootstrap
- Criar Módulo de Postagem
- Criar Componentes para o Módulo
- Criar Rotas
- Criar Interface
- Criar Serviço
- Atualizar Lógica e Template do Componente
- Exportar provideHttpClient()
- Executar Aplicativo Angular
- Referências

Criar Projeto Angular 18

Você pode criar seu aplicativo Angular usando o seguinte comando:

ng new my-new-app

Instalar Bootstrap

Instale o Bootstrap para sua aplicação CRUD:

npm install bootstrap --save

Importe no arquivo angular.json:

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

Criar Módulo de Postagem

Após criar o aplicativo com sucesso, precisamos criar o módulo de postagem usando o comando da CLI do Angular:

ng generate module post

Após executar o comando com sucesso, serão criados arquivos no seguinte caminho: src/app/post/post.module.ts

Criar Componentes para o Módulo

Agora adicionaremos novos componentes ao nosso módulo de postagem:

ng generate component post/index
ng generate component post/view
ng generate component post/create
ng generate component post/edit

Criar Componentes para o Módulo

Após executar os comandos com sucesso, serão criados arquivos nos seguintes caminhos:

src/app/post/index/*
src/app/post/view/*
src/app/post/create/*
src/app/post/edit/*

Criar Rotas

Atualize o arquivo de rotas app.routes.ts:

```
import { Routes } from '@angular/router';
import { IndexComponent } from './post/index/index.component';
import { ViewComponent } from './post/view/view.component';
import { CreateComponent } from './post/create/create.component';
import { EditComponent } from './post/edit/edit.component';

export const routes: Routes = [
  { path: 'post', redirectTo: 'post/index', pathMatch: 'full'},
  { path: 'post/index', component: IndexComponent },
  { path: 'post/:postId/view', component: ViewComponent },
  { path: 'post/create', component: CreateComponent },
  { path: 'post/:postId/edit', component: EditComponent }
];
```

Criar Interface

Crie a interface para o módulo de postagem:

ng generate interface post/post

src/app/post/post.ts

```
export interface Post {
  id: number;
  title: string;
  body: string;
}
```

Criar Serviço

No home.component.html vou digitar o hello world

Criaremos o arquivo de serviço de postagem e implementaremos os métodos de serviço web getAll(), create(), find(), update() e delete().

ng generate service post/post

Criar Serviço

src/app/post/post.service.ts Código completo

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { Post } from './post';

@Injectable({
  providedIn: 'root'
})
export class PostService {
  private apiURL = "https://jsonplaceholder.typicode.com";

  httpOptions = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json'
    })
  }

constructor(private httpClient: HttpClient) { }

  getAll(): Observable<any> {
    return this.httpClient.get(this.apiURL + '/posts/')
      .pipe(
        catchError(this.errorHandler)
      )
  }

  create(post: Post): Observable<any> {
    return this.httpClient.post(this.apiURL + '/posts/', JSON.stringify(post), this.httpOptions)
      .pipe(
        catchError(this.errorHandler)
      )
  }

  find(id: number): Observable<any> {
    return this.httpClient.get(this.apiURL + '/posts/' + id)
      .pipe(
        catchError(this.errorHandler)
      )
  }

  update(id: number, post: Post): Observable<any> {
    return this.httpClient.put(this.apiURL + '/posts/' + id, JSON.stringify(post), this.httpOptions)
      .pipe(
        catchError(this.errorHandler)
      )
  }

  delete(id: number) {
    return this.httpClient.delete(this.apiURL + '/posts/' + id, this.httpOptions)
      .pipe(
        catchError(this.errorHandler)
      )
  }

  errorHandler(error: any) {
    let errorMessage = '';
    if (error.error instanceof ErrorEvent) {
      errorMessage = error.error.message;
    } else {
      errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;
    }
    return throwError(errorMessage);
  }
}
```

Criar Serviço

src/app/post/post.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { Post } from './post';

@Injectable({
  providedIn: 'root'
})
export class PostService {
  private apiURL = "https://jsonplaceholder.typicode.com";

  httpOptions = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json'
    })
  }
```

Criar Serviço

src/app/post/post.service.ts

```
 constructor(private httpClient: HttpClient) { }

 getAll(): Observable<any> {
  return this.httpClient.get(this.apiURL + '/posts/')
    .pipe(
      catchError(this.errorHandler)
    )
 }

 create(post: Post): Observable<any> {
  return this.httpClient.post(this.apiURL + '/posts/', JSON.stringify(post), this.httpOptions)
    .pipe(
      catchError(this.errorHandler)
    )
 }

 find(id: number): Observable<any> {
```

Criar Serviço

src/app/post/post.service.ts

```
find(id: number): Observable<any> {
  return this.httpClient.get(this.apiURL + '/posts/' + id)
    .pipe(
      catchError(this.errorHandler)
    )
}

update(id: number, post: Post): Observable<any> {
  return this.httpClient.put(this.apiURL + '/posts/' + id, JSON.stringify(post), this.httpOptions)
    .pipe(
      catchError(this.errorHandler)
    )
}
```

Criar Serviço

src/app/post/post.service.ts

```
  delete(id: number) {
    return this.httpClient.delete(this.apiURL + '/posts/' + id, this.httpOptions)
      .pipe(
        catchError(this.errorHandler)
      )
  }
  errorHandler(error: any) {
    let errorMessage = '';
    if (error.error instanceof ErrorEvent) {
      errorMessage = error.error.message;
    } else {
      errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;
    }
    return throwError(errorMessage);
  }
}
```

Atualizar Lógica e Template do Componente

Template e Componente da Página de Lista src/app/post/index/index.component.ts Código completo

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';
import { PostService } from '../post.service';
import { Post } from '../post';

@Component({
  selector: 'app-index',
  standalone: true,
  imports: [CommonModule, RouterModule],
  templateUrl: './index.component.html',
  styleUrl: './index.component.css'
})
export class IndexComponent {
  posts: Post[] = [];

  constructor(public postService: PostService) { }

  ngOnInit(): void {
    this.postService.getAll().subscribe((data: Post[]) => {
      this.posts = data;
      console.log(this.posts);
    })
  }

  deletePost(id: number) {
    this.postService.delete(id).subscribe(res => {
      this.posts = this.posts.filter(item => item.id !== id);
      console.log('Post deletado com sucesso!');
    })
  }
}
```

Atualizar Lógica e Template do Componente

Template e Componente da Página de Lista
src/app/post/index/index.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';
import { PostService } from '../post.service';
import { Post } from '../post';

@Component({
  selector: 'app-index',
  standalone: true,
  imports: [CommonModule, RouterModule],
  templateUrl: './index.component.html',
  styleUrl: './index.component.css'
})
export class IndexComponent {
  posts: Post[] = [];
```

Atualizar Lógica e Template do Componente

Template e Componente da Página de Lista
src/app/post/index/index.component.ts

```
  constructor(public postService: PostService) { }

  ngOnInit(): void {
    this.postService.getAll().subscribe((data: Post[]) => {
      this.posts = data;
      console.log(this.posts);
    })
  }

  deletePost(id: number) {
    this.postService.delete(id).subscribe(res => {
      this.posts = this.posts.filter(item => item.id !== id);
      console.log('Post deletado com sucesso!');
    })
  }
}
```

Atualizar Lógica e Template do Componente

Template e Componente da Página de Lista src/app/post/index/index.component.html código completo

```
<div class="container">
  <h1>Exemplo CRUD Angular 18</h1>
  <a href="#" routerLink="/post/create/" class="btn btn-success">Criar Nova Postagem</a>

  <table class="table table-striped">
    <thead>
      <tr>
        <th>ID</th>
        <th>Título</th>
        <th>Corpo</th>
        <th width="250px">Ação</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let post of posts">
        <td>{{ post.id }}</td>
        <td>{{ post.title }}</td>
        <td>{{ post.body }}</td>
        <td>
          <a href="#" [routerLink]="['/post/', post.id, 'view']" class="btn btn-info">Visualizar</a>
          <a href="#" [routerLink]="['/post/', post.id, 'edit']" class="btn btn-primary">Editar</a>
          <button type="button" (click)="deletePost(post.id)" class="btn btn-danger">Deletar</button>
        </td>
      </tr>
    </tbody>
  </table>
</div>
```

Atualizar Lógica e Template do Componente

Template e Componente da Página de Lista
src/app/post/index/index.component.html

```
<div class="container">
  <h1>Exemplo CRUD Angular 18</h1>
  <a href="#" routerLink="/post/create/" class="btn btn-success">Criar Nova Postagem</a>

  <table class="table table-striped">
    <thead>
      <tr>
        <th>ID</th>
        <th>Título</th>
        <th>Corpo</th>
        <th width="250px">Ação</th>
      </tr>
    </thead>
    <tbody>
```

Atualizar Lógica e Template do Componente

Template e Componente da Página de Lista
src/app/post/index/index.component.html

```
  <tbody>
   <tr *ngFor="let post of posts">
    <td>{{ post.id }}</td>
    <td>{{ post.title }}</td>
    <td>{{ post.body }}</td>
    <td>
     <a href="#" [routerLink]="['/post/', post.id, 'view']" class="btn btn-info">Visualizar</a>
     <a href="#" [routerLink]="['/post/', post.id, 'edit']" class="btn btn-primary">Editar</a>
     <button type="button" (click)="deletePost(post.id)" class="btn btn-danger">Deletar</button>
    </td>
   </tr>
  </tbody>
 </table>
</div>
```

Atualizar Lógica e Template do Componente

Criar Template e Componente de Página

Aqui, utilizaremos formulários reativos para armazenar dados no servidor utilizando serviços web. Vamos configurar o componente e o template.

Arquivo: src/app/post/create/create.component.ts

Arquivo: src/app/post/create/create.component.html

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.ts Código completo

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PostService } from '../post.service';
import { Router } from '@angular/router';
import { ReactiveFormsModule, FormGroup, FormControl, Validators } from '@angular/forms';

@Component({
 selector: 'app-create',
 standalone: true,
 imports: [CommonModule, ReactiveFormsModule],
 templateUrl: './create.component.html',
 styleUrl: './create.component.css'
})
export class CreateComponent {

 form!: FormGroup;
 constructor(
  public postService: PostService,
  private router: Router
 ) { }

 ngOnInit(): void {
  this.form = new FormGroup({
   title: new FormControl('', [Validators.required]),
   body: new FormControl('', Validators.required)
  });
 }

 get f() {
  return this.form.controls;
 }

 submit() {
  console.log(this.form.value);
  this.postService.create(this.form.value).subscribe((res: any) => {
   console.log('Post criado com sucesso!');
   this.router.navigateByUrl('post/index');
  });
 }
}
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PostService } from '../post.service';
import { Router } from '@angular/router';
import { ReactiveFormsModule, FormGroup, FormControl, Validators } from '@angular/forms';

@Component({
  selector: 'app-create',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule],
  templateUrl: './create.component.html',
  styleUrl: './create.component.css'
})
export class CreateComponent {
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.ts

```
export class CreateComponent {

 form!: FormGroup;
 constructor(
   public postService: PostService,
   private router: Router
 ) { }

 ngOnInit(): void {
   this.form = new FormGroup({
     title: new FormControl('', [Validators.required]),
     body: new FormControl('', Validators.required)
   });
 }
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.ts

```
  get f() {
    return this.form.controls;
  }

  submit() {
    console.log(this.form.value);
    this.postService.create(this.form.value).subscribe((res: any) => {
      console.log('Post criado com sucesso!');
      this.router.navigateByUrl('post/index');
    });
  }
}
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.html Código completo

```
<div class="container">
 <h1>Criar Nova Postagem</h1>

 <a href="#" routerLink="/post/index" class="btn btn-primary">Voltar</a>

 <form [formGroup]="form" (ngSubmit)="submit()">

  <div class="form-group">
   <label for="title">Título:</label>
   <input
    formControlName="title"
    id="title"
    type="text"
    class="form-control">
   <div *ngIf="f['title'].touched && f['title'].invalid" class="alert alert-danger">
    <div *ngIf="f['title'].errors && f['title'].errors['required']">
     O título é obrigatório.
    </div>
   </div>
  </div>

  <div class="form-group">
   <label for="body">Conteúdo</label>
   <textarea
    formControlName="body"
    id="body"
    type="text"
    class="form-control">
   </textarea>
   <div *ngIf="f['body'].touched && f['body'].invalid" class="alert alert-danger">
    <div *ngIf="f['body'].errors && f['body'].errors['required']">
     O conteúdo é obrigatório.
    </div>
   </div>
  </div>

  <button class="btn btn-primary" type="submit" [disabled]="!form.valid">Enviar</button>
 </form>
</div>
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.html

```
<div class="container">
 <h1>Criar Nova Postagem</h1>

 <a href="#" routerLink="/post/index" class="btn btn-primary">Voltar</a>

 <form [formGroup]="form" (ngSubmit)="submit()">

  <div class="form-group">
   <label for="title">Título:</label>
   <input
    formControlName="title"
    id="title"
    type="text"
    class="form-control">
   <div *ngIf="f['title'].touched && f['title'].invalid" class="alert alert-danger">
    <div *ngIf="f['title'].errors && f['title'].errors['required']">
     O título é obrigatório.
    </div>
   </div>
  </div>
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/create/create.component.html

```
  <div class="form-group">
    <label for="body">Conteúdo</label>
    <textarea
      formControlName="body"
      id="body"
      type="text"
      class="form-control">
    </textarea>
    <div *ngIf="f['body'].touched && f['body'].invalid" class="alert alert-danger">
      <div *ngIf="f['body'].errors && f['body'].errors['required']">
        O conteúdo é obrigatório.
      </div>
    </div>
  </div>

  <button class="btn btn-primary" type="submit" [disabled]="!form.valid">Enviar</button>
 </form>
</div>
```

Atualizar Lógica e Template do Componente

Editar Template e Componente de Página
Aqui, utilizaremos formulários reativos para atualizar informações de postagens no servidor usando serviços web.

Arquivo: src/app/post/edit/edit.component.ts
Arquivo: src/app/post/edit/edit.component.html

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.ts Código completo

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PostService } from '../post.service';
import { ActivatedRoute, Router } from '@angular/router';
import { Post } from '../post';
import { ReactiveFormsModule, FormGroup, FormControl, Validators } from '@angular/forms';

@Component({
 selector: 'app-edit',
 standalone: true,
 imports: [CommonModule, ReactiveFormsModule],
 templateUrl: './edit.component.html',
 styleUrl: './edit.component.css'
})
export class EditComponent {

 id!: number;
 post!: Post;
 form!: FormGroup;

 constructor(
  public postService: PostService,
  private route: ActivatedRoute,
  private router: Router
 ) { }

 ngOnInit(): void {
  this.id = this.route.snapshot.params['postId'];
  this.postService.find(this.id).subscribe((data: Post) => {
   this.post = data;
  });

  this.form = new FormGroup({
   title: new FormControl('', [Validators.required]),
   body: new FormControl('', Validators.required)
  });
 }

 get f() {
  return this.form.controls;
 }

 submit() {
  console.log(this.form.value);
  this.postService.update(this.id, this.form.value).subscribe((res: any) => {
   console.log('Post atualizado com sucesso!');
   this.router.navigateByUrl('post/index');
  });
 }
}
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PostService } from '../post.service';
import { ActivatedRoute, Router } from '@angular/router';
import { Post } from '../post';
import { ReactiveFormsModule, FormGroup, FormControl, Validators } from '@angular/forms';

@Component({
  selector: 'app-edit',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule],
  templateUrl: './edit.component.html',
  styleUrl: './edit.component.css'
})
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.ts

```
export class EditComponent {

  id!: number;
  post!: Post;
  form!: FormGroup;

  constructor(
    public postService: PostService,
    private route: ActivatedRoute,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.id = this.route.snapshot.params['postId'];
    this.postService.find(this.id).subscribe((data: Post) => {
      this.post = data;
    });

    this.form = new FormGroup({
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.ts

```
  this.form = new FormGroup({
    title: new FormControl('', [Validators.required]),
    body: new FormControl('', Validators.required)
  });
 }

 get f() {
  return this.form.controls;
 }

 submit() {
  console.log(this.form.value);
  this.postService.update(this.id, this.form.value).subscribe((res: any) => {
    console.log('Post atualizado com sucesso!');
    this.router.navigateByUrl('post/index');
  });
 }
}
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.html Código completo

```
<div class="container">
  <h1>Atualizar Postagem</h1>

  <a href="#" routerLink="/post/index" class="btn btn-primary">Voltar</a>

  <form [formGroup]="form" (ngSubmit)="submit()">

    <div class="form-group">
      <label for="title">Título:</label>
      <input
        formControlName="title"
        id="title"
        type="text"
        [(ngModel)]="post.title"
        class="form-control">
      <div *ngIf="f['title'].touched && f['title'].invalid" class="alert alert-danger">
        <div *ngIf="f['title'].errors && f['title'].errors['required']">
          O título é obrigatório.
        </div>
      </div>
    </div>

    <div class="form-group">
      <label for="body">Conteúdo</label>
      <textarea
        formControlName="body"
        id="body"
        type="text"
        [(ngModel)]="post.body"
        class="form-control">
      </textarea>
      <div *ngIf="f['body'].touched && f['body'].invalid" class="alert alert-danger">
        <div *ngIf="f['body'].errors && f['body'].errors['required']">
          O conteúdo é obrigatório.
        </div>
      </div>
    </div>

    <button class="btn btn-primary" type="submit" [disabled]="!form.valid">Atualizar</button>
  </form>
</div>
```

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.html

```
<div class="container">
 <h1>Atualizar Postagem</h1>

 <a href="#" routerLink="/post/index" class="btn btn-primary">Voltar</a>

 <form [formGroup]="form" (ngSubmit)="submit()">

  <div class="form-group">
   <label for="title">Título:</label>
   <input
    formControlName="title"
    id="title"
    type="text"
    [(ngModel)]="post.title"
    class="form-control">
   <div *ngIf="f['title'].touched && f['title'].invalid" class="alert alert-danger">
    <div *ngIf="f['title'].errors && f['title'].errors['required']">
     O título é obrigatório.
    </div>
   </div>
  </div>
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/edit/edit.component.html

```
  <div class="form-group">
    <label for="body">Conteúdo</label>
    <textarea
      formControlName="body"
      id="body"
      type="text"
      [(ngModel)]="post.body"
      class="form-control">
    </textarea>
    <div *ngIf="f['body'].touched && f['body'].invalid" class="alert alert-danger">
      <div *ngIf="f['body'].errors && f['body'].errors['required']">
        O conteúdo é obrigatório.
      </div>
    </div>
  </div>

  <button class="btn btn-primary" type="submit" [disabled]="!form.valid">Atualizar</button>
 </form>
</div>
```

Atualizar Lógica e Template do Componente

Detalhar Template e Componente de Página
Aqui, exibiremos os dados da postagem armazenados no servidor.

Arquivo: src/app/post/view/view.component.ts
Arquivo: src/app/post/view/view.component.html

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/view/view.component.ts Código completo

```
import { Component } from '@angular/core';
import { PostService } from '../post.service';
import { ActivatedRoute, Router } from '@angular/router';
import { Post } from '../post';

@Component({
  selector: 'app-view',
  standalone: true,
  imports: [],
  templateUrl: './view.component.html',
  styleUrl: './view.component.css'
})
export class ViewComponent {

  id!: number;
  post!: Post;

  constructor(
    public postService: PostService,
    private route: ActivatedRoute,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.id = this.route.snapshot.params['postId'];
    this.postService.find(this.id).subscribe((data: Post) => {
      this.post = data;
    });
  }
}
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/view/view.component.ts

```
import { Component } from '@angular/core';
import { PostService } from '../post.service';
import { ActivatedRoute, Router } from '@angular/router';
import { Post } from '../post';

@Component({
  selector: 'app-view',
  standalone: true,
  imports: [],
  templateUrl: './view.component.html',
  styleUrl: './view.component.css'
})
export class ViewComponent {

  id!: number;
  post!: Post;
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/view/view.component.ts

```
  constructor(
    public postService: PostService,
    private route: ActivatedRoute,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.id = this.route.snapshot.params['postId'];
    this.postService.find(this.id).subscribe((data: Post) => {
      this.post = data;
    });
  }
}
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/view/view.component.html Código completo

```
<div class="container">
 <h1>Visualizar Postagem</h1>

 <a href="#" routerLink="/post/index" class="btn btn-primary">Voltar</a>

 <div>
  <strong>ID:</strong>
  <p>{{ post.id }}</p>
 </div>

 <div>
  <strong>Título:</strong>
  <p>{{ post.title }}</p>
 </div>

 <div>
  <strong>Conteúdo:</strong>
  <p>{{ post.body }}</p>
 </div>
</div>
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/view/view.component.html

```
<div class="container">
  <h1>Visualizar Postagem</h1>

  <a href="#" routerLink="/post/index" class="btn btn-primary">Voltar</a>

  <div>
    <strong>ID:</strong>
    <p>{{ post.id }}</p>
  </div>

  <div>
    <strong>Título:</strong>
    <p>{{ post.title }}</p>
  </div>
```

Atualizar Lógica e Template do Componente

Arquivo: src/app/post/view/view.component.html

```
 <div>
  <strong>Conteúdo:</strong>
  <p>{{ post.body }}</p>
 </div>
</div>
```

Atualizar Lógica e Template do Componente

Atualizar Visualização no app.component.html

<router-outlet></router-outlet>

Executar Aplicativo Angular

Execute o aplicativo com o comando:

ng serve

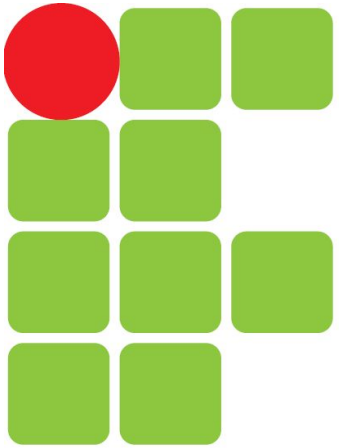
Abra no navegador: http://localhost:4200/post

Referências

- Site oficial do angular: https://angular.dev/
- Introdução ao Angular https://codelabs.developers.google.com/introduction-to-angular?hl=pt-br#0
- Angular Documentation https://devdocs.io/angular/
- Guide to AngularJS Documentation https://docs.angularjs.org/guide

Referências

- Projeto Angular: https://github.com/savanihd/Angular-18-CRUD-Application-Tutorial-Example
- Aplicação CRUD Angular 18 - Tutorial Exemplo https://github.com/daniel-abella/angular18-crud//?tab=readme-ov-file#arquivo-srcapppostedite ditcomponenthtml
- Angular 18 Básico https://github.com/daniel-abella/angular1 co



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO