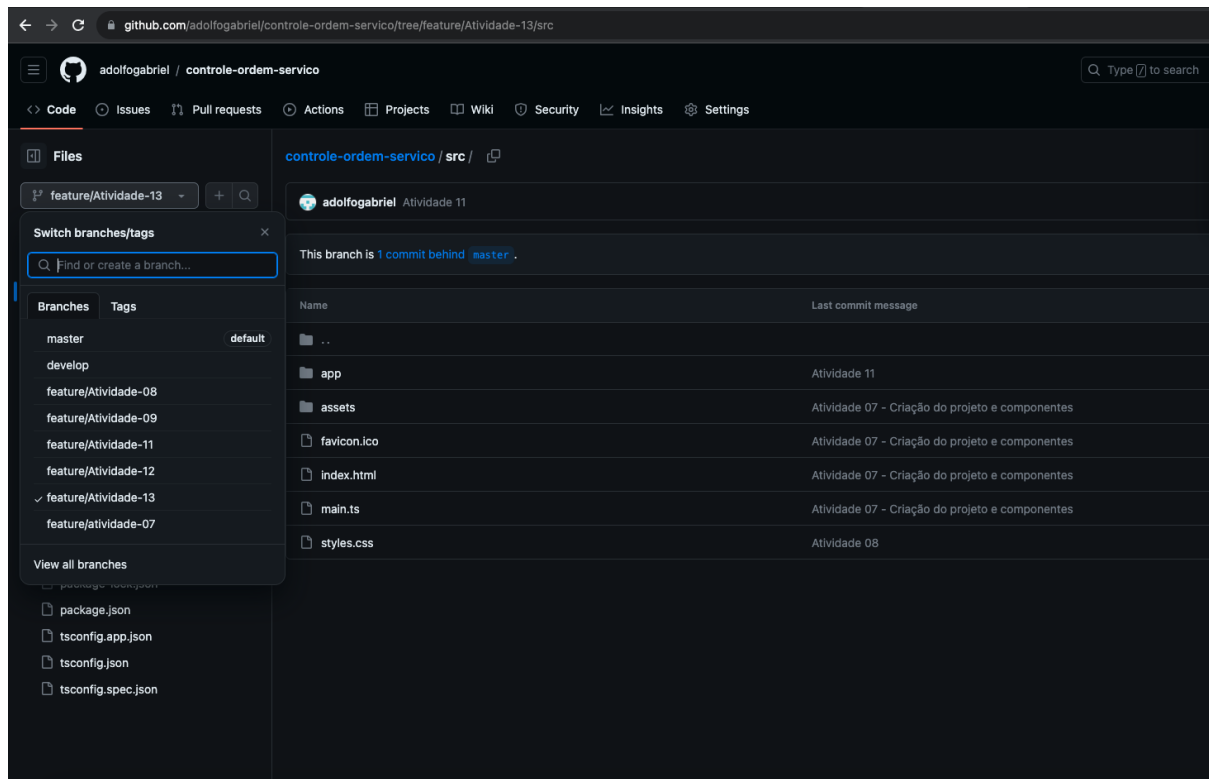
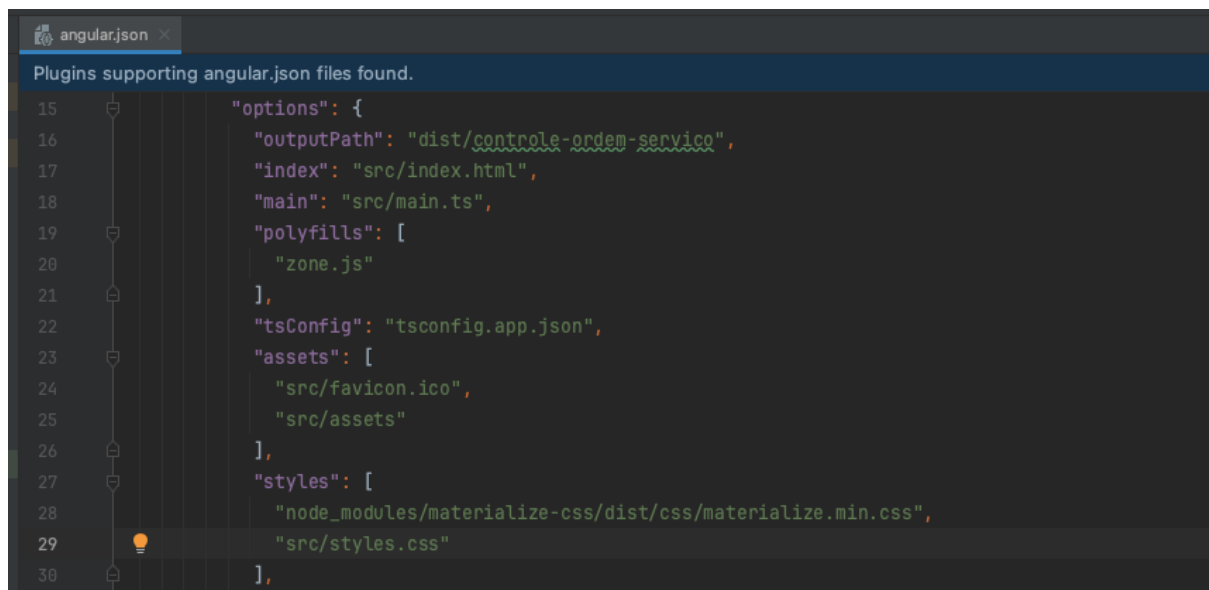


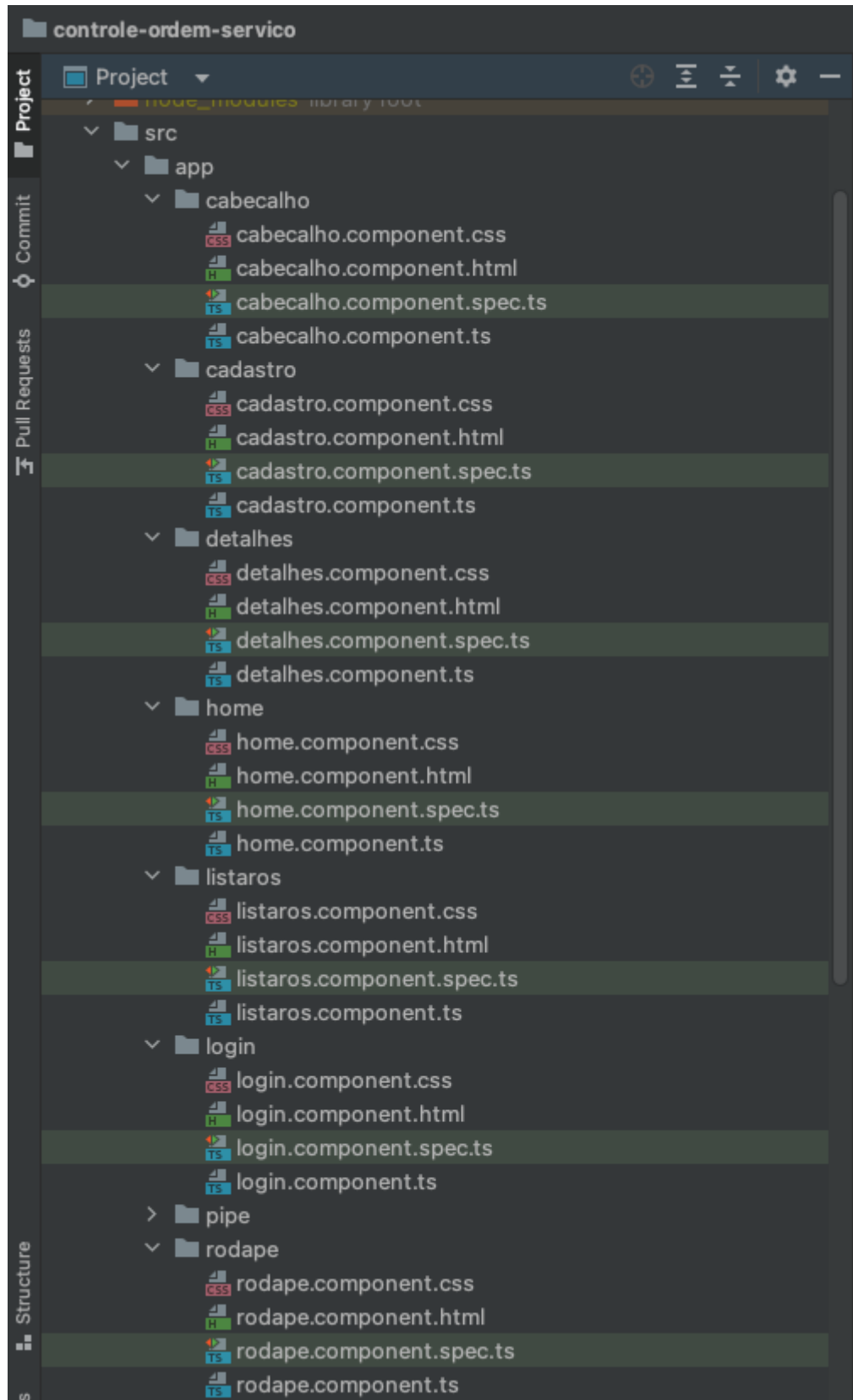
Criar o repositório no GitHub com a estrutura do Gitflow, ou seja, branches main e develop
<https://github.com/adolfogabriel/controle-ordem-servico/>



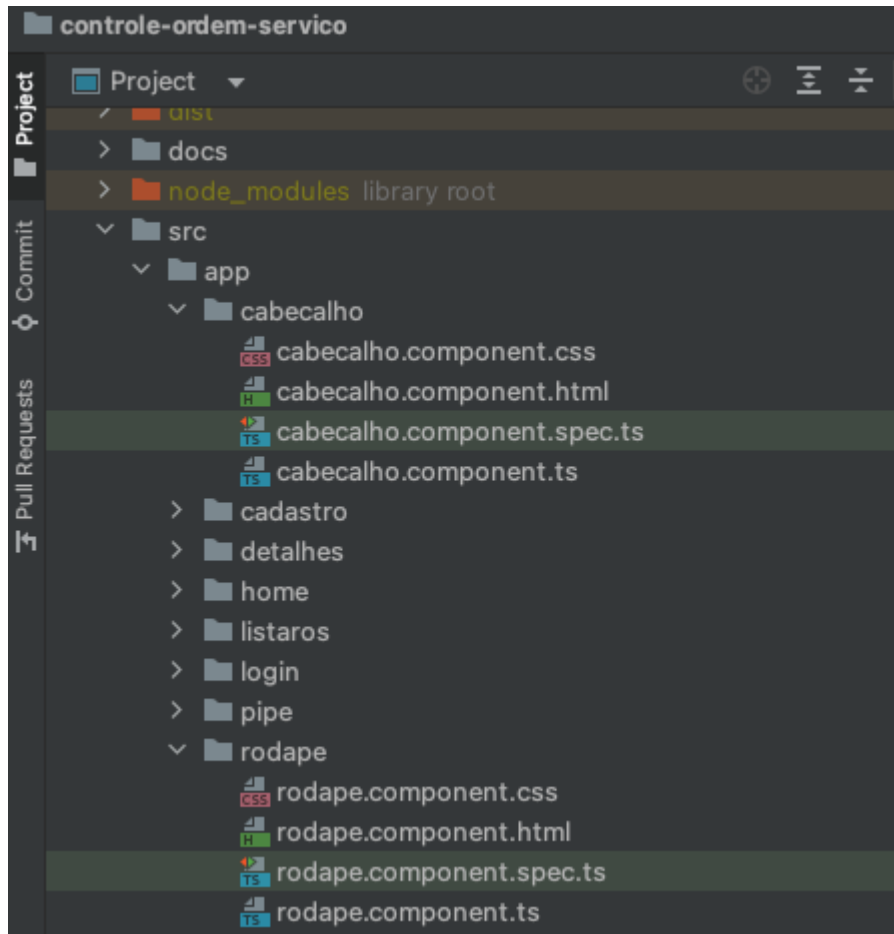
Usar componentes de algum framework CSS (Bootstrap, Materialize ou outro).



Construir páginas web com o conceito de componentes.



Criar o layout da aplicação com componentes, ou seja, o cabeçalho e rodapé precisam ser componentes.



Usar pelo menos dois tipos de data-binding (Interpolation, Property Binding, Event Binding e Two Way Data Binding).

Interpolation:

```
cabecalho.component.html x cabecalho.component.ts x
1 <nav>
2   <div class="nav-wrapper teal lighten-2">
3     <span>{{ title }}</span>
4     <span class="direction-right">Usuário: {{ dadosLogin?.usuario }}</span>
5     <ul id="nav-mobile" class="right hide-on-med-and-down">
6       <li><a routerLink="/home">Home</a></li>
7       <li><a (click)="onLinkClick()">Gostei</a></li>
8     </ul>
9   </div>
10 </nav>
11
```

Event Binding:

```
cabecalho.component.html x cabecalho.component.ts x
1 <nav>
2   <div class="nav-wrapper teal lighten-2">
3     <span>{{ title }}</span>
4     <span class="direction-right">Usuário: {{ dadosLogin?.usuario }}</span>
5     <ul id="nav-mobile" class="right hide-on-med-and-down">
6       <li><a routerLink="/home">Home</a></li>
7       <li><a (click)="onLinkClick()">Gostei</a></li>
8     </ul>
9   </div>
10 </nav>
11
```

Interpolation e Event Binding

```

1  import {Component, Input} from '@angular/core';
2
3  5+ usages  Adolfo Gabriel *
4  @Component({
5      selector: 'app-cabecalho',
6      templateUrl: './cabecalho.component.html',
7      styleUrls: ['./cabecalho.component.css']
8  })
9  export class CabecalhoComponent {
10     @Input() dadosLogin: { usuario: string } | undefined;
11     title :string = 'Controle ordem servico';
12
13     1 usage  new *
14     onLinkClick() : void {
15         alert("Obrigado " + this.dadosLogin?.usuario + "!");
16     }
17 }

```

Passar dados via hierarquia de componentes, ou seja, usando @Input ou @Output

@Input

```

1  import {Component, Input} from '@angular/core';
2
3  5+ usages  Adolfo Gabriel *
4  @Component({
5      selector: 'app-cabecalho',
6      templateUrl: './cabecalho.component.html',
7      styleUrls: ['./cabecalho.component.css']
8  })
9  export class CabecalhoComponent {
10     @Input() dadosLogin: { usuario: string } | undefined;
11     title :string = 'Controle ordem servico';
12 }

```

@Output

```
5+ usages  ▴ Adolfo Gabriel
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent {
  username: string = '';
  password: string = '';
  loginError: boolean = false;
  isVisible: boolean = true;

  @Output() loginEvent: EventEmitter<{usuario: string}>... = new EventEmitter<{ usuario: string }>();

2 usages  ▴ Adolfo Gabriel
  constructor( public authService: AuthService, public router: Router) {}

5+ usages  ▴ Adolfo Gabriel
  onSubmit(): void {
    const isAuthenticated: boolean = this.authService.login(this.username, this.password);

    if (isAuthenticated) {
      this.loginEvent.emit( value: {usuario: this.username});
      this.router.navigate( commands: ['/home']);
      this.loginError = false;
      this.isVisible = false;
      const objeto :{nome: string} = { nome: this.username};
      localStorage.setItem('usuario', JSON.stringify(objeto));
    } else {
      const usuarioString : string | null = localStorage.getItem( key: 'usuario');
      if (usuarioString != null) {
        this.loginError = true;
      } else {
        this.loginError = true;
      }
    }
  }
}
```

Mapear componentes à rotas no módulo de rotas

```
app-routing.module.ts
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { LoginComponent } from '../login/login.component';
4 import { HomeComponent } from '../home/home.component';
5 import { DetalhesComponent } from '../detalhes/detalhes.component';
6 import { CadastroComponent } from '../cadastro/cadastro.component';
7 import { ListarosComponent } from '../listaros/listaros.component';
8
9
10 const routes: Routes = [
11   { path: 'login', component: LoginComponent },
12   { path: 'home', component: HomeComponent },
13   { path: 'detalhes/:id', component: DetalhesComponent },
14   { path: 'cadastro', component: CadastroComponent },
15   { path: 'listar', component: ListarosComponent },
16   { path: '', redirectTo: '/login', pathMatch: 'full' },
17 ];
18
19 @NgModule({
20   imports: [RouterModule.forRoot(routes)],
21   exports: [RouterModule]
22 })
23 export class AppRoutingModule { }
```

Criar navegação entre páginas por meio de rotas.

```
app-routing.module.ts sidebar.component.html
1 <div class="sidebar teal lighten-2">
2   <ul>
3     <li><a routerLink="/cadastro">Cadastro</a></li>
4     <li><a routerLink="/listar">Listar</a></li>
5   </ul>
6 </div>
```

Passar dados entre componentes que representam diferentes telas via parâmetros de rotas

```
listaros.component.html x
1      <h2>Lista de Os's</h2>
2
3      <table>
4      <thead>
5      <tr>
6          <th>ID</th>
7          <th>Nome</th>
8          <th>Email</th>
9          <th>Telefone</th>
10         <th>Cpf</th>
11     </tr>
12 </thead>
13 <tbody>
14 <tr *ngFor="let os of os">
15     <td><a [routerLink]="['/detalhes', os.id]">{{ os.id }}</a></td>
16     <td>{{ os.nome }}</td>
17     <td>{{ os.email }}</td>
18     <td>{{ os.telefone | telefone }}</td>
19     <td>{{ os.cpf | cpf }}</td>
20 </tr>
21 </tbody>
22 </table>
23
```



```

1  import {Component, OnInit} from '@angular/core';
2  import {ActivatedRoute} from '@angular/router';
3  import {OsService} from "../service/os.service";
4
5  5+ usages  Adolfo Gabriel
6  @Component({
7      selector: 'app-detalhes',
8      templateUrl: './detalhes.component.html',
9      styleUrls: ['./detalhes.component.css']
10 })
11 export class DetalhesComponent implements OnInit {
12     osId: number | undefined;
13     osDetalhes: any;
14
15     2 usages  Adolfo Gabriel
16     constructor(private route: ActivatedRoute, private osService: OsService) {
17
18     1 usage  Adolfo Gabriel
19     ngOnInit(): void {
20         this.route.params.subscribe( observerOrNext: params : Params => {
21             this.osId = +params['id'];
22
23             this.osService.getOsId(this.osId).subscribe(
24                 next: (response) : void => {
25                     this.osDetalhes = response;
26                 },
27                 error: (error) : void => {
28                     alert('Erro na requisição:' + error);
29                 });
30     });
31 }
32 }

```

Validar campos do formulário com REGEX e apresentar os erros.

```
cadastro.component.ts x
5  @Component({
6  selector: 'app-cadastro',
7  templateUrl: './cadastro.component.html',
8  styleUrls: ['./cadastro.component.css']
9  })
10 export class CadastroComponent {
11     dadosFormulario: FormGroup;
12
13
14     2 usages  Adolfo Gabriel
15     constructor(private osService: OsService, private fb: FormBuilder) {
16         this.dadosFormulario = this.fb.group(controls: {
17             nome: ['', [Validators.required, Validators.minLength(3)]],
18             email: ['', [Validators.required, Validators.email]],
19             telefone: ['', [Validators.pattern(pattern: /\d{11}$/)]],
20             cpf: ['', [Validators.pattern(pattern: /\d{11}$/)]],
21         });
22     }
```

Desabilitar o botão de submit enquanto o formulário está inválido.

```
1 <form [formGroup]="dadosFormulario" (ngSubmit)="onSubmit()">
2   <label for="nome">Nome:</label>
3   <input type="text" id="nome" formControlName="nome">
4   <div *ngIf="dadosFormulario.get('nome')?.hasError( errorCode: 'required') && dadosFormulario.get('nome')?.touched">
5     0 campo nome é obrigatório.
6   </div>
7
8   <label for="email">E-mail:</label>
9   <input type="email" id="email" formControlName="email">
10  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'required') && dadosFormulario.get('email')?.touched">
11    0 campo e-mail é obrigatório.
12  </div>
13  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'email') && dadosFormulario.get('email')?.touched">
14    E-mail inválido.
15  </div>
16
17  <label for="telefone">Telefone:</label>
18  <input type="text" id="telefone" formControlName="telefone">
19  <div *ngIf="dadosFormulario.get('telefone')?.hasError( errorCode: 'pattern') && dadosFormulario.get('telefone')?.touched">
20    Telefone inválido (deve ter 10 dígitos).
21  </div>
22
23  <label for="cpf">Cpf:</label>
24  <input type="text" id="cpf" formControlName="cpf">
25  <div *ngIf="dadosFormulario.get('cpf')?.hasError( errorCode: 'pattern') && dadosFormulario.get('cpf')?.touched">
26    cpf inválido (deve ter 11 dígitos).
27  </div>
28
29  <button type="submit" [disabled]="!dadosFormulario.valid">Enviar</button>
30 </form>
31
```

localhost:4200/cadastro

Controle ordem serviçoUsuário: adolfo

Nome:

E-mail:

Telefone:

Cpf:

Enviar

localhost:4200/cadastro

Controle ordem serviçoUsuário: adolfo

Nome:

adolfo

E-mail:

gabriel@hotmail.com

Telefone:

61991261211

Cpf:

11111111111

Enviar

Fazer requisições a API com tratamento da resposta com Promises ou Observables

Promises

```
cadastro.component.ts x
5  @Component({
6    selector: 'app-cadastro',
7    templateUrl: './cadastro.component.html',
8    styleUrls: ['./cadastro.component.css']
9  })
10 export class CadastroComponent {
11   dadosFormulario: FormGroup;
12
13   2 usages  ⚙ Adolfo Gabriel
14   constructor(private osService: OsService, private fb: FormBuilder) {
15     this.dadosFormulario = this.fb.group( controls: {
16       nome: ['', [Validators.required, Validators.minLength( minLength: 3)]],
17       email: ['', [Validators.required, Validators.email]],
18       telefone: ['', [Validators.pattern( pattern: /\d{11}$/)]],
19       cpf: ['', [Validators.pattern( pattern: /\d{11}$/)]],
20     });
21   }
22
23   5+ usages  ⚙ Adolfo Gabriel
24   onSubmit() : void {
25     if (this.dadosFormulario.valid) {
26       this.osService.postOs(this.dadosFormulario.value)
27         .then(response => {
28           console.log('Dados salvos no servidor:', response);
29           this.dadosFormulario.reset();
30         }).catch((error) : void => {
31           alert('Erro na requisição:' + error);
32         });
33     } else {
34       alert("Campos incorretos, favor verificar!")
35     }
36   }
37 }
```

```
cadastro.component.ts x os.service.ts x
1 import ...
4
5+ usages Adolfo Gabriel
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class OsService {
9
10   private Url: string = "http://localhost:3000/os"
11
12   2 usages Adolfo Gabriel
12   constructor(private http: HttpClient) {
13   }
14
15   2 usages Adolfo Gabriel
15   getOs(): Promise<any> {
16     return this.http.get(this.Url).toPromise();
17   }
18
19   1 usage Adolfo Gabriel
19   postOs(data: any): Promise<any> {
20     return this.http.post(this.Url, data).toPromise();
21   }
22
23   2 usages Adolfo Gabriel
23   getOsId(id: number): Observable<any> {
24     return this.http.get(url: `${this.Url}/${id}`);
25   }
26 }
```

```
cadastro.component.ts x os.service.ts x detalhes.component.ts x
3 import {OsService} from "../service/os.service";
4
5 5+ usages Adolfo Gabriel
6 @Component({
7   selector: 'app-detalhes',
8   templateUrl: './detalhes.component.html',
9   styleUrls: ['./detalhes.component.css']
10 })
11 export class DetalhesComponent implements OnInit {
12   osId: number | undefined;
13   osDetalhes: any;
14
15   2 usages Adolfo Gabriel
16   constructor(private route: ActivatedRoute, private osService: OsService) {
17   }
18
19   1 usage Adolfo Gabriel
20   ngOnInit() : void {
21     this.route.params.subscribe( observerOrNext: params : Params => {
22       this.osId = +params['id'];
23
24       this.osService.getOsId(this.osId).subscribe(
25         next: (response) : void => {
26           this.osDetalhes = response;
27         },
28         error: (error) : void => {
29           alert('Erro na requisição:' + error);
30         }
31       );
32     });
33   }
34 }
```

Cadastrar uma entidade no JSON Server.

```

1  {
2    "os": [
3      {
4        "nome": "adolfo",
5        "email": "adolfo@adolfo.com",
6        "telefone": "1111111111",
7        "cpf": "1111111111",
8        "id": 1
9      }
10   ]
11 }

```

Apresentar uma lista de dados com a diretiva estrutural ngFor.

```

1  <h2>Lista de Os's</h2>
2
3  <table>
4    <thead>
5      <tr>
6        <th>ID</th>
7        <th>Nome</th>
8        <th>Email</th>
9        <th>Telefone</th>
10       <th>Cpf</th>
11     </tr>
12   </thead>
13   <tbody>
14     <tr *ngFor="let os of os">
15       <td><a [routerLink]="['/detalhes', os.id]">{{ os.id }}</a></td>
16       <td>{{ os.nome }}</td>
17       <td>{{ os.email }}</td>
18       <td>{{ os.telefone | telefone }}</td>
19       <td>{{ os.cpf | cpf }}</td>
20     </tr>
21   </tbody>
22 </table>
23

```


Usar a diretiva ngIf

```
cadastro.component.html
1 <form [formGroup]="dadosFormulario" (ngSubmit)="onSubmit()">
2   <label for="nome">Nome:</label>
3   <input type="text" id="nome" formControlName="nome">
4   <div *ngIf="dadosFormulario.get('nome')?.hasError( errorCode: 'required') && dadosFormulario.get('nome')?.touched">
5     0 campo nome é obrigatório.
6   </div>
7
8   <label for="email">E-mail:</label>
9   <input type="email" id="email" formControlName="email">
10  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'required') && dadosFormulario.get('email')?.touched">
11    0 campo e-mail é obrigatório.
12  </div>
13  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'email') && dadosFormulario.get('email')?.touched">
14    E-mail inválido.
15  </div>
16
17  <label for="telefone">Telefone:</label>
18  <input type="text" id="telefone" formControlName="telefone">
19  <div *ngIf="dadosFormulario.get('telefone')?.hasError( errorCode: 'pattern') && dadosFormulario.get('telefone')?.touched">
20    Telefone inválido (deve ter 10 dígitos).
21  </div>
22
23  <label for="cpf">Cpf:</label>
24  <input type="text" id="cpf" formControlName="cpf">
25  <div *ngIf="dadosFormulario.get('cpf')?.hasError( errorCode: 'pattern') && dadosFormulario.get('cpf')?.touched">
26    cpf inválido (deve ter 11 dígitos).
27  </div>
28
29  <button type="submit" [disabled]="!dadosFormulario.valid">Enviar</button>
30 </form>
31
```

Formatar a apresentação de dados com Pipes

```
cpf.pipe.ts
1 import { Pipe, PipeTransform } from '@angular/core';
2
3 5+ usages  Adolfo Gabriel
4 @Pipe({
5   name: 'cpf'
6 })
7 export class CpfPipe implements PipeTransform {
8   4 usages  Adolfo Gabriel
9   transform(cpf: string): string {
10     if (cpf && cpf.length === 11) {
11       return cpf.replace( searchValue: /(\\d{3})(\\d{3})(\\d{3})(\\d{2})/, replaceValue: '$1.$2.$3-$4');
12     }
13     return cpf;
14   }
15 }
```

```
1 | import { Pipe, PipeTransform } from '@angular/core';
2 |
3 | @Pipe({
4 |   name: 'telefone'
5 | })
6 | export class TelefonePipe implements PipeTransform {
7 |   transform(telefone: string): string {
8 |     if (telefone && telefone.length === 11) {
9 |       return `${telefone.substr(0, 2)}-${telefone.substr(2, 6)}-${telefone.substr(7)} `;
10 |     }
11 |     return telefone;
12 |   }
13 | }
14 |
```

Controle ordem serviçoUsuário: adolfo

Lista de Os´s

ID	Nome	Email	Telefone	Cpf
1	adolfo	adolfo@adolfo.com	11-111111-1111	111.111.111-11
2	adolfo	gabriel@hotmail.com	61-991261-1211	111.111.111-11

Build e deploy da aplicação.

<https://adolfogabriel.github.io/controle-ordem-servico/>