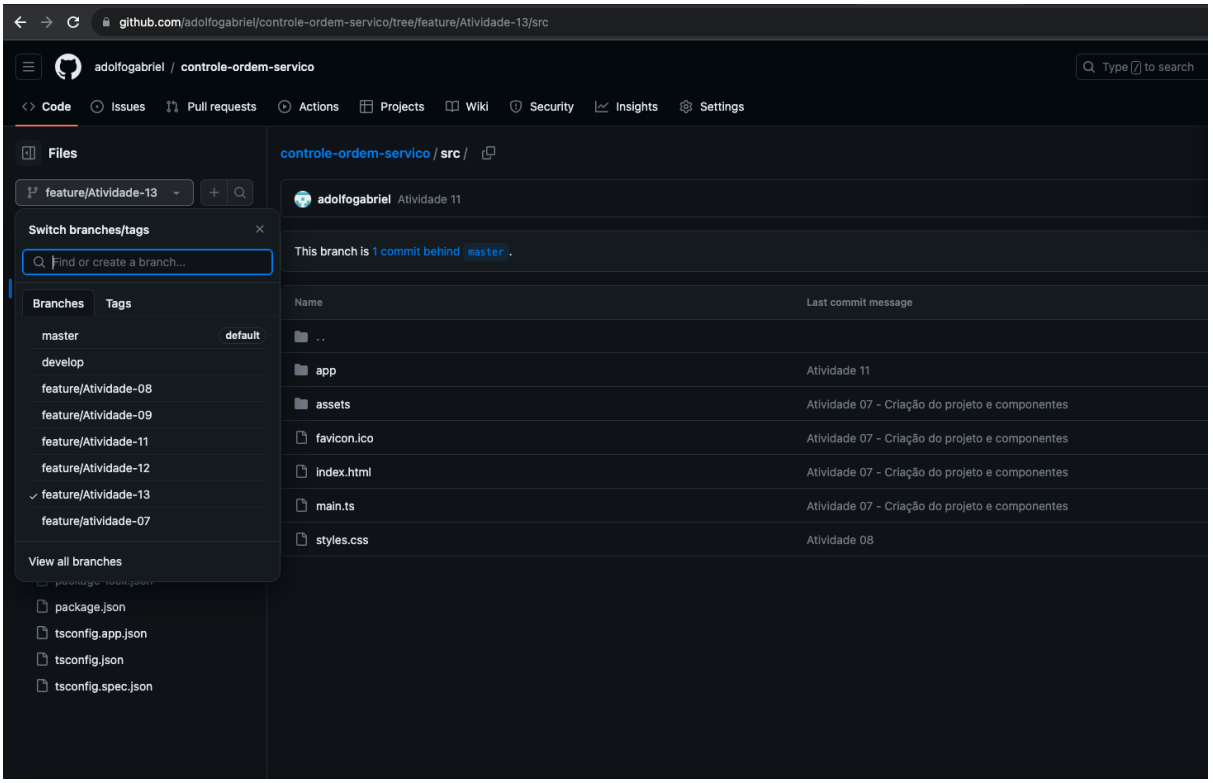
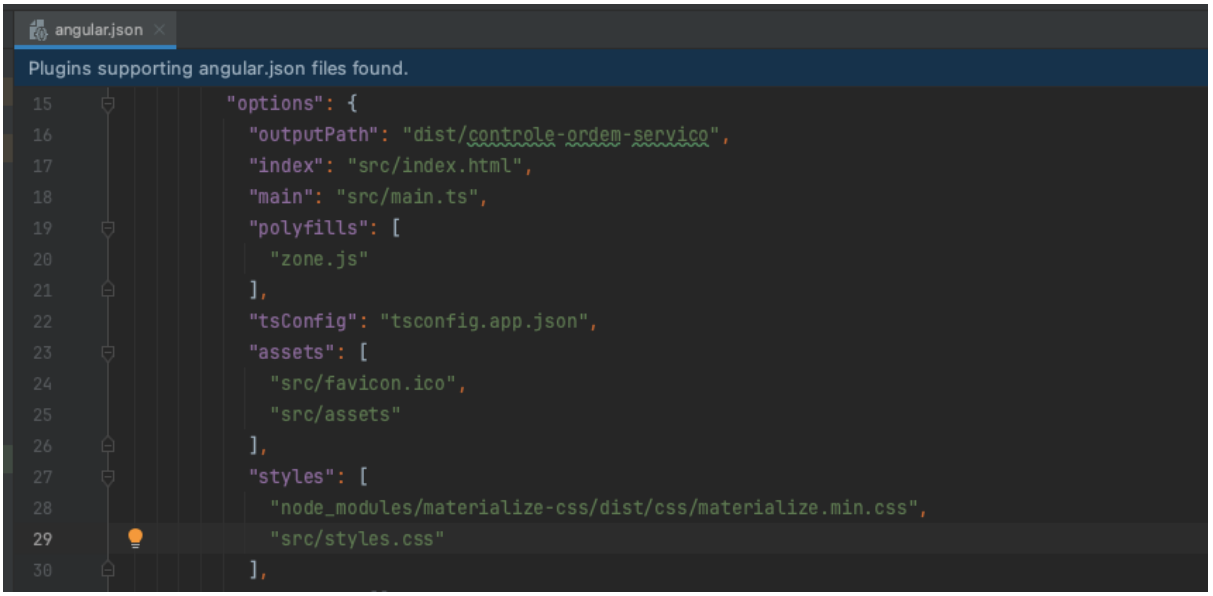
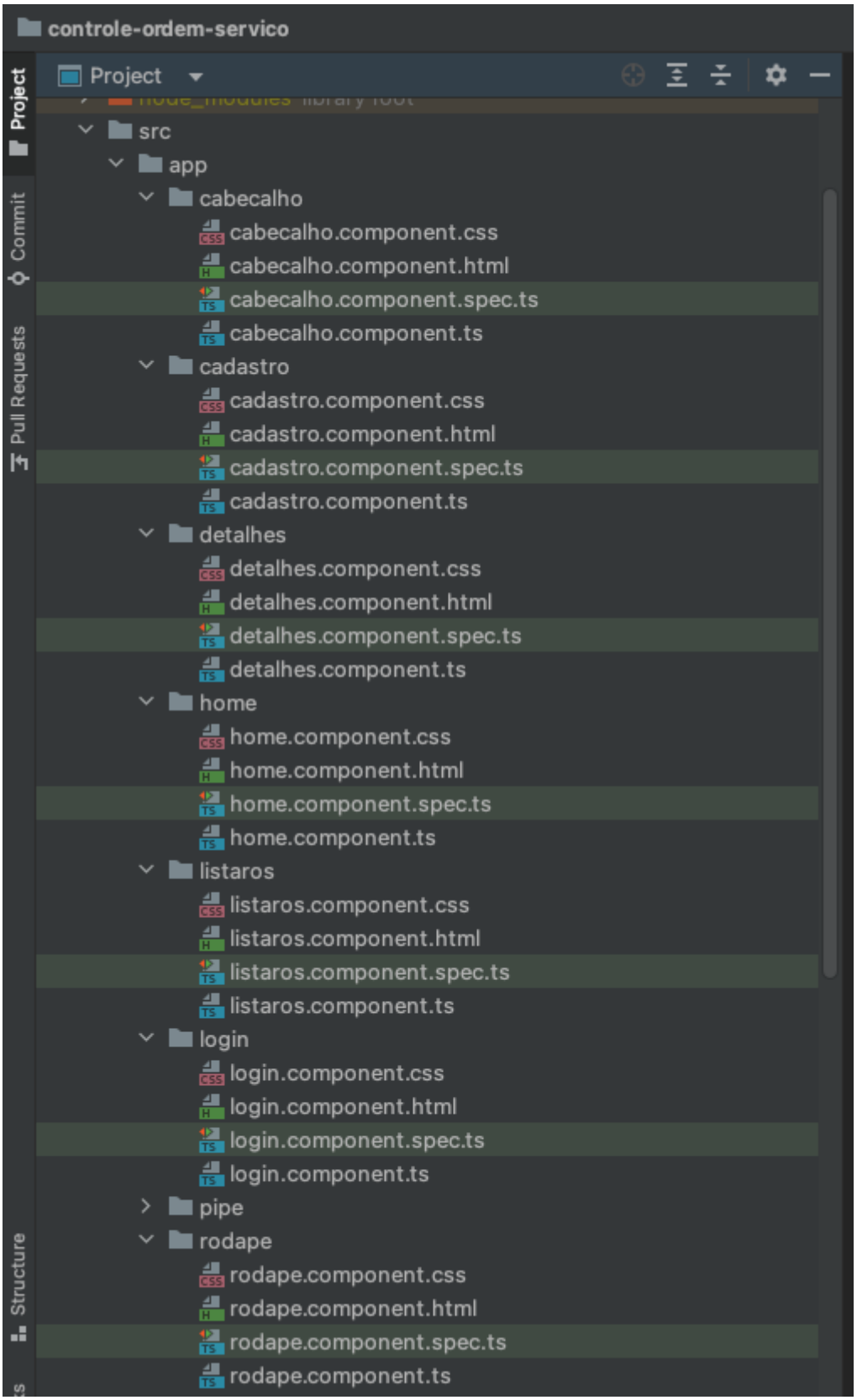
Criar o repositório no GitHub com a estrutura do Gitflow, ou seja, branches main e develop
https://github.com/adolfogabriel/controle-ordem-servico/



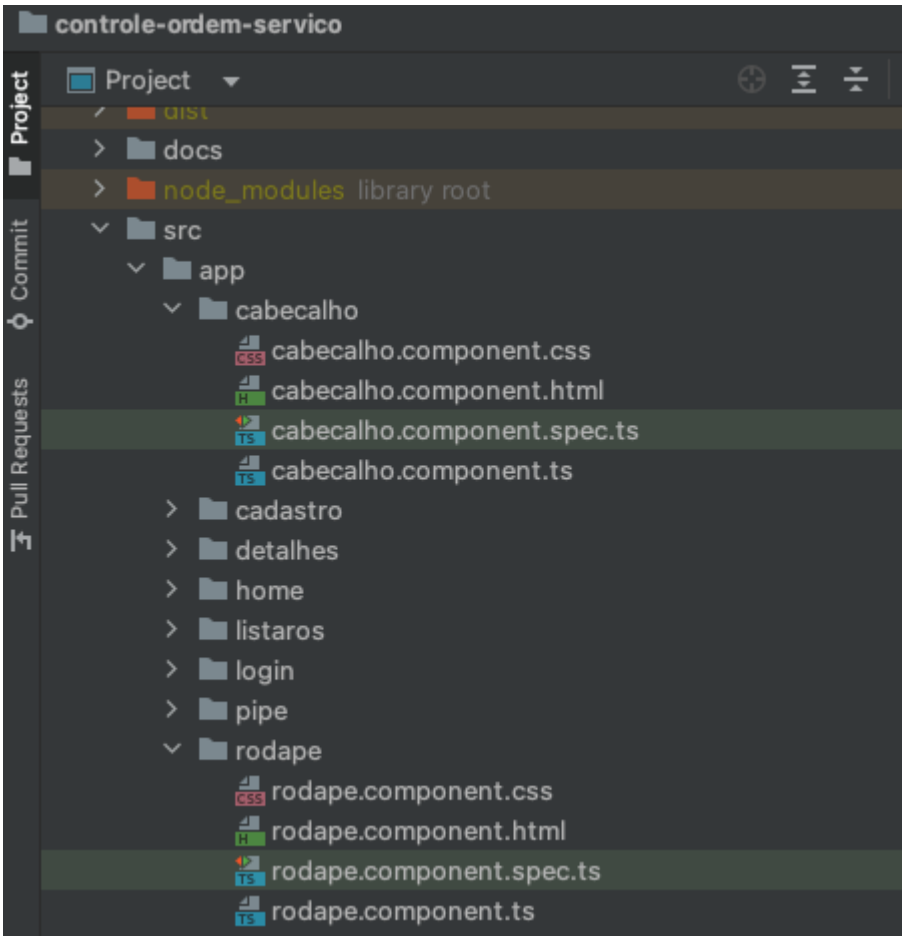
Usar componentes de algum framework CSS (Bootstrap, Materialize ou outro).



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:

Event Binding:

Interpolation e Event Binding

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

5+ usages  Adolfo Gabriel *
@Component({
  selector: 'app-cabecalho',
  templateUrl: './cabecalho.component.html',
  styleUrls: ['./cabecalho.component.css']
})
export class CabecalhoComponent {
  @Input() dadosLogin: { usuario: string} | undefined;
  title : string  = 'Controle ordem servico';

  1 usage  new *
  onLinkClick() : void  {
    alert("Obrigado "+ this.dadosLogin?.usuario +"!");
  }
}
```

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

@Input

cabecalho.component.html | cabecalho.component.ts

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

5+ usages  Adolfo Gabriel *
@Component({
  selector: 'app-cabecalho',
  templateUrl: './cabecalho.component.html',
  styleUrls: ['./cabecalho.component.css']
})
export class CabecalhoComponent {
  @Input() dadosLogin: { usuario: string} | undefined;
  title : string  = 'Controle ordem servico';
```

@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

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './login/login.component';
import {HomeComponent} from "./home/home.component";
import {DetalhesComponent} from "./detalhes/detalhes.component";
import {CadastroComponent} from "./cadastro/cadastro.component";
import {ListarosComponent} from "./listaros/listaros.component";


const routes: Routes = [
  { path: 'login', component: LoginComponent },
  { path: 'home', component: HomeComponent },
  { path: 'detalhes/:id', component: DetalhesComponent},
  { path: 'cadastro', component: CadastroComponent },
  { path: 'listar', component: ListarosComponent },
  { path: '', redirectTo: '/login', pathMatch: 'full' },
];

2 usages  Adolfo Gabriel
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Criar navegação entre páginas por meio de rotas.

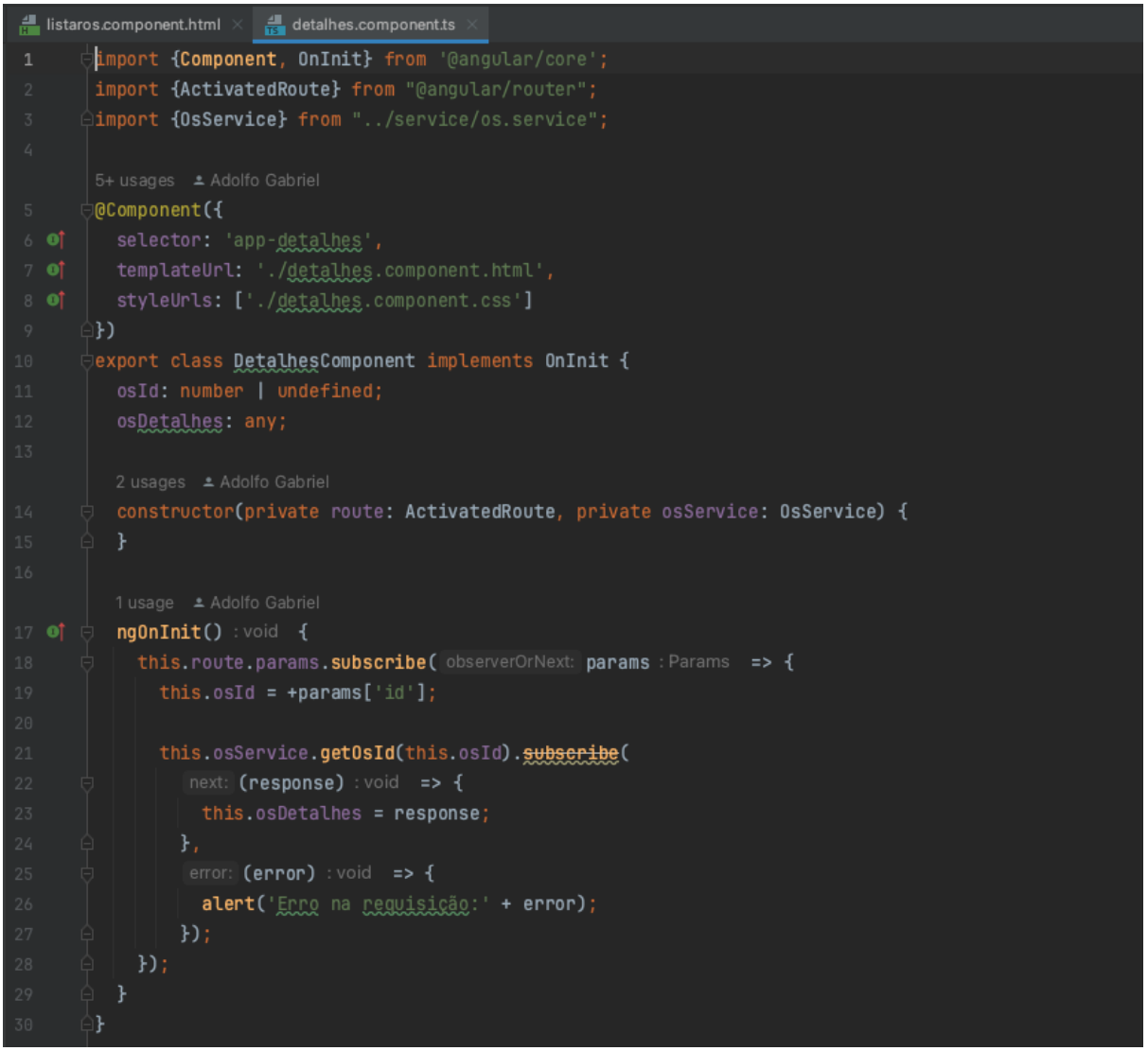
app-routing.module.ts | sidebar.component.html

```html
<div class="sidebar teal lighten-2">
  <ul>
    <li><a routerLink="/cadastro">Cadastro</a></li>
    <li><a routerLink="/listar">Listar</a></li>
  </ul>
</div>
```

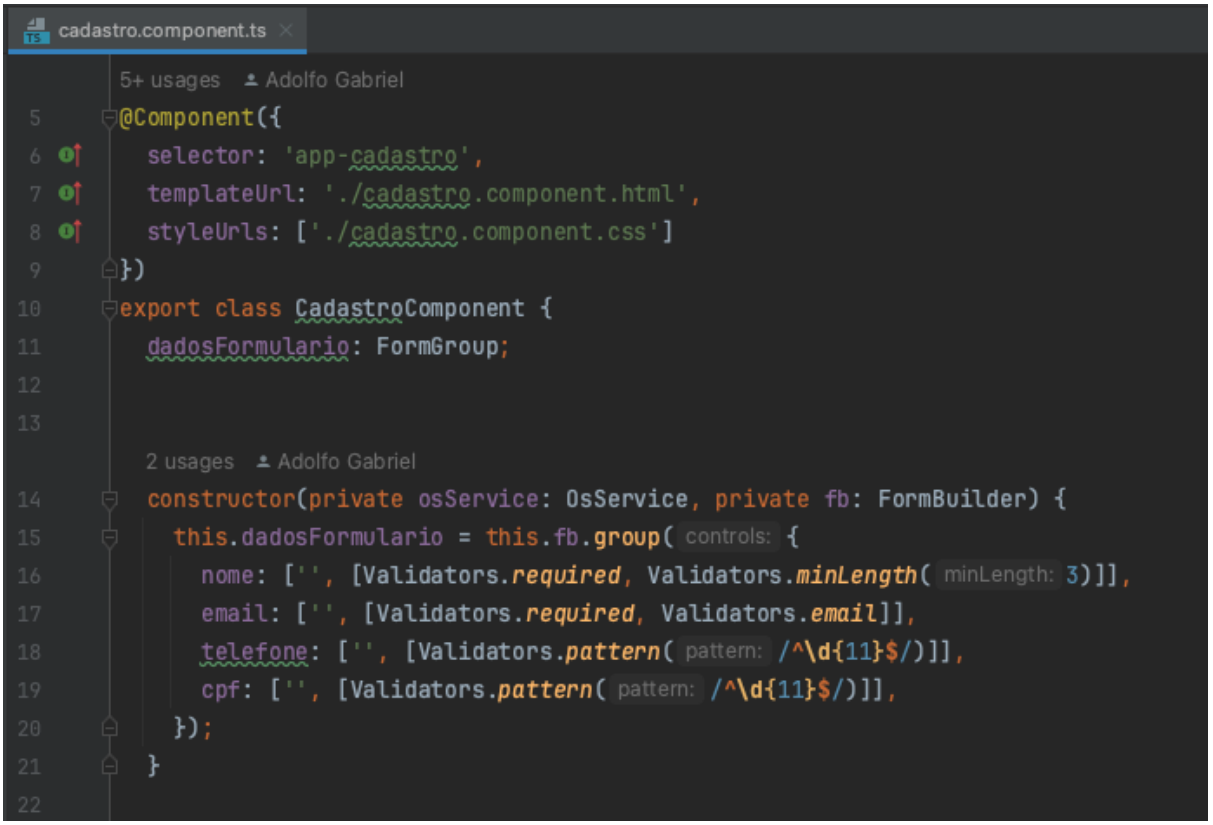
Passar dados entre componentes que representam diferentes telas via parâmetros de rotas

listaros.component.html

```
<h2>Lista de Os´s</h2>

<table>
  <thead>
  <tr>
    <th>ID</th>
    <th>Nome</th>
    <th>Email</th>
    <th>Telefone</th>
    <th>Cpf</th>
  </tr>
  </thead>
  <tbody>
  <tr *ngFor="let os of os">
    <td><a [routerLink]="['/detalhes', os.id]">{{ os.id }}</a></td>
    <td>{{ os.nome }}</td>
    <td>{{ os.email }}</td>
    <td>{{ os.telefone | telefone }}</td>
    <td>{{ os.cpf | cpf }}</td>
  </tr>
  </tbody>
</table>

```


listaros.component.html
detalhes.component.ts
import {Component, OnInit} from '@angular/core';
import {ActivatedRoute} from "@angular/router";
import {OsService} from "../service/os.service";
5+ usages Adolfo Gabriel
@Component({
selector: 'app-detalhes',
templateUrl: './detalhes.component.html',
styleUrls: ['./detalhes.component.css']
})
export class DetalhesComponent implements OnInit {
osId: number | undefined;
osDetalhes: any;
2 usages Adolfo Gabriel
constructor(private route: ActivatedRoute, private osService: OsService) {
}
1 usage Adolfo Gabriel
ngOnInit() : void {
this.route.params.subscribe(observerOrNext: params : Params => {
this.osId = +params['id'];
this.osService.getOsId(this.osId).subscribe(
next: (response) : void => {
this.osDetalhes = response;
},
error: (error) : void => {
alert('Erro na requisição:' + error);
});
});
}
}

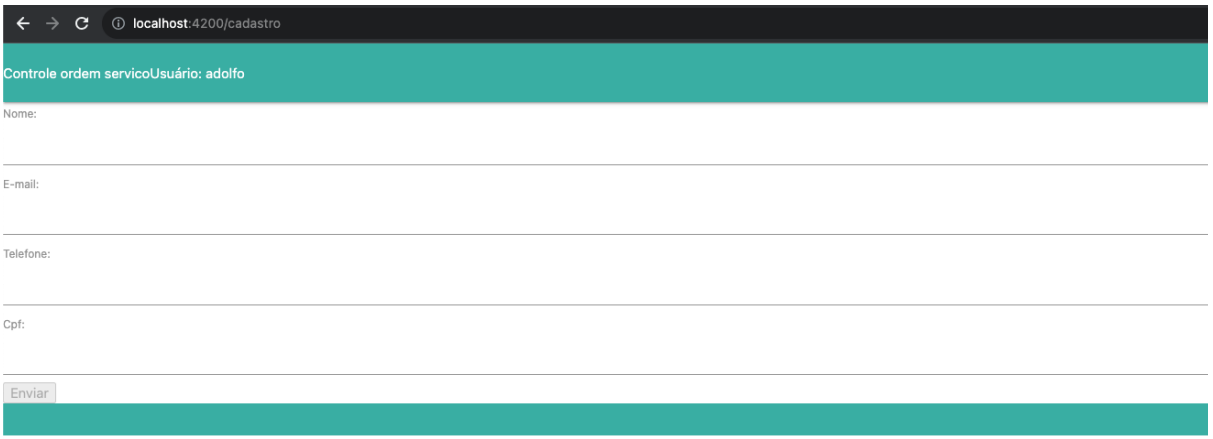
Validar campos do formulário com REGEX e apresentar os erros.

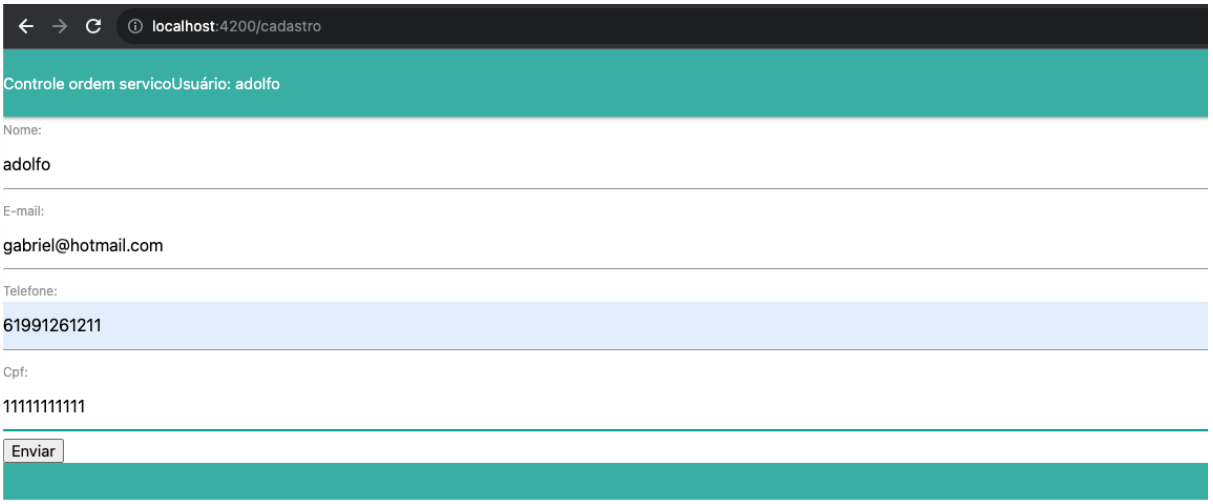


Desabilitar o botão de submit enquanto o formulário está inválido.

cadastro.component.html

```
<form [formGroup]="dadosFormulario" (ngSubmit)="onSubmit()">
  <label for="nome">Nome:</label>
  <input type="text" id="nome" formControlName="nome">
  <div *ngIf="dadosFormulario.get('nome')?.hasError( errorCode: 'required') && dadosFormulario.get('nome')?.touched">
    O campo nome é obrigatório.
  </div>

  <label for="email">E-mail:</label>
  <input type="email" id="email" formControlName="email">
  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'required') && dadosFormulario.get('email')?.touched">
    O campo e-mail é obrigatório.
  </div>
  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'email') && dadosFormulario.get('email')?.touched">
    E-mail inválido.
  </div>

  <label for="telefone">Telefone:</label>
  <input type="text" id="telefone" formControlName="telefone">
  <div *ngIf="dadosFormulario.get('telefone')?.hasError( errorCode: 'pattern') && dadosFormulario.get('telefone')?.touched">
    Telefone inválido (deve ter 10 dígitos).
  </div>

  <label for="cpf">Cpf:</label>
  <input type="text" id="cpf" formControlName="cpf">
  <div *ngIf="dadosFormulario.get('cpf')?.hasError( errorCode: 'pattern') && dadosFormulario.get('cpf')?.touched">
    cpf inválido (deve ter 11 dígitos).
  </div>

  <button type="submit" [disabled]="!dadosFormulario.valid">Enviar</button>
</form>
```

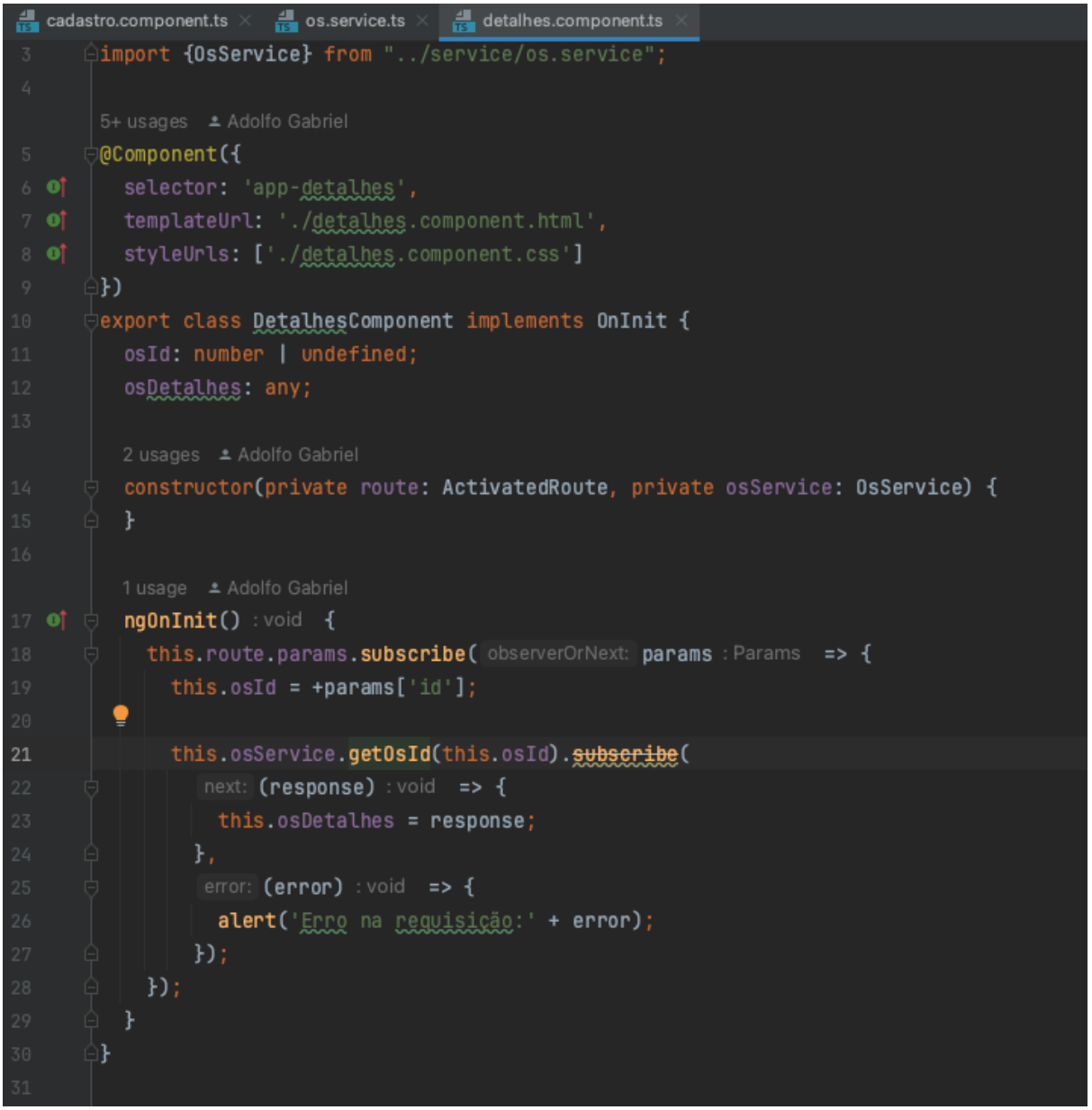



localhost:4200/cadastro
Controle ordem servicoUsuário: adolfo
Nome:
adolfo
E-mail:
gabriel@hotmail.com
Telefone:
61991261211
Cpf:
11111111111
Enviar

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

Promises


cadastro.component.ts
os.service.ts
import ...
5+ usages Adolfo Gabriel
@Injectable({
providedIn: 'root'
})
export class OsService {
private Url: string = "http://localhost:3000/os"
2 usages Adolfo Gabriel
constructor(private http: HttpClient) {
}
2 usages Adolfo Gabriel
getOs(): Promise<any> {
return this.http.get(this.Url).toPromise();
}
2 usages Adolfo Gabriel
postOs(data: any): Promise<any> {
return this.http.post(this.Url, data).toPromise();
}
2 usages Adolfo Gabriel
getOsId(id: number): Observable<any> {
return this.http.get(url: `${this.Url}/${id}`);
}
}



Cadastrar uma entidade no JSON Server.

db.json

```json
{
  "os": [
    {
      "nome": "adolfo",
      "email": "adolfo@adolfo.com",
      "telefone": "11111111111",
      "cpf": "11111111111",
      "id": 1
    }
  ]
}
```

Apresentar uma lista de dados com a diretiva estrutural ngFor.

listaros.component.html

```html
<h2>Lista de Os´s</h2>

<table>
  <thead>
  <tr>
    <th>ID</th>
    <th>Nome</th>
    <th>Email</th>
    <th>Telefone</th>
    <th>Cpf</th>
  </tr>
  </thead>
  <tbody>
  <tr *ngFor="let os of os">
    <td><a [routerLink]="['/detalhes', os.id]">{{ os.id }}</a></td>
    <td>{{ os.nome }}</td>
    <td>{{ os.email }}</td>
    <td>{{ os.telefone | telefone }}</td>
    <td>{{ os.cpf | cpf }}</td>
  </tr>
  </tbody>
</table>
```

Usar a diretiva ngIf

cadastro.component.html

```html
<form [formGroup]="dadosFormulario" (ngSubmit)="onSubmit()">
  <label for="nome">Nome:</label>
  <input type="text" id="nome" formControlName="nome">
  <div *ngIf="dadosFormulario.get('nome')?.hasError( errorCode: 'required') && dadosFormulario.get('nome')?.touched">
    O campo nome é obrigatório.
  </div>

  <label for="email">E-mail:</label>
  <input type="email" id="email" formControlName="email">
  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'required') && dadosFormulario.get('email')?.touched">
    O campo e-mail é obrigatório.
  </div>
  <div *ngIf="dadosFormulario.get('email')?.hasError( errorCode: 'email') && dadosFormulario.get('email')?.touched">
    E-mail inválido.
  </div>

  <label for="telefone">Telefone:</label>
  <input type="text" id="telefone" formControlName="telefone">
  <div *ngIf="dadosFormulario.get('telefone')?.hasError( errorCode: 'pattern') && dadosFormulario.get('telefone')?.touched">
    Telefone inválido (deve ter 10 dígitos).
  </div>

  <label for="cpf">Cpf:</label>
  <input type="text" id="cpf" formControlName="cpf">
  <div *ngIf="dadosFormulario.get('cpf')?.hasError( errorCode: 'pattern') && dadosFormulario.get('cpf')?.touched">
    cpf inválido (deve ter 11 dígitos).
  </div>

  <button type="submit" [disabled]="!dadosFormulario.valid">Enviar</button>
</form>
```

Formatar a apresentação de dados com Pipes

cpf.pipe.ts

```ts
import { Pipe, PipeTransform } from '@angular/core';

5+ usages  Adolfo Gabriel
@Pipe({
  name: 'cpf'
})
export class CpfPipe implements PipeTransform {
  4 usages  Adolfo Gabriel
  transform(cpf: string): string {
    if (cpf && cpf.length === 11) {
      return cpf.replace( searchValue: /(\d{3})(\d{3})(\d{3})(\d{2})/,  replaceValue: '$1.$2.$3-$4');
    }
    return cpf;
  }
}
```

cpf.pipe.ts telefone.pipe.ts

```ts
import { Pipe, PipeTransform } from '@angular/core';

5+ usages  Adolfo Gabriel
@Pipe({
  name: 'telefone'
})
export class TelefonePipe implements PipeTransform {
  4 usages  Adolfo Gabriel
  transform(telefone: string): string {
    if (telefone && telefone.length === 11) {
      return `${telefone.substr( from: 0,  length: 2)}-${telefone.substr( from: 2,  length: 6)}-${telefone.substr( from: 7)}`;
    }
    return telefone;
  }
}
```

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/