uniesp Centro Universitário	**Curso**: MBA em Full Stack Web Development
	Disciplina: Angular Bootcamp
	Docente: Nicoly Figueredo Pessoa de Almeida
	Assunto: Tutorial final do projeto Angular

Após ter concluído o tutorial 3, siga as próximas etapas para finalizar:

1. Vamos criar as interfaces necessárias para essa etapa:
   - **conta.ts**

```
export interface Conta {
    id: number,
    numero: string,
    agencia: string,
    saldo: number,
    cliente: number,
    nomeCliente: string
}
```

   - **saqueDeposito.ts**

```
export interface SaqueDeposito {
    valor: number,
    conta: number
}
```

   - **transferencia.ts**

```
export interface Transferencia {
    conta_origem: number,
    valor: number,
    conta_destino: number
}
```

2. Agora vamos criar o serviço de contas:
   - `ng generate service shared/services/conta/conta-service`

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import            {            environment            }            from
'../../../../environments/environment.development';
```

```
import { Conta } from '../../models/conta';
import { SaqueDeposito } from '../../models/saqueDeposito';
import { Transferencia } from '../../models/transferencia';

@Injectable({
  providedIn: 'root'
})
export class ContaService {
  api = `${environment.api}/contas/`;


  constructor(private clienteHttp: HttpClient) { }


  inserir(novaConta: Conta): Observable<Conta> {
    return this.clienteHttp.post<Conta>(
      this.api, novaConta);
  }


  listar(): Observable<Conta[]> {
    return this.clienteHttp.get<Conta[]>(this.api);
  }


   listar_paginado(page: number, pageSize: number): Observable<Conta[]>
{
                                                                   return
this.clienteHttp.get<Conta[]>(`${this.api}?page=${page}&pageSize=${page
Size}`);
  }


  deletar(idConta: number): Observable<object> {
    return this.clienteHttp.delete(`${this.api}${idConta}`);
  }


  pesquisarPorId(id: number): Observable<Conta> {
    return this.clienteHttp.get<Conta>(`${this.api}${id}`);
  }


  atualizar(conta: Conta): Observable<Conta> {
          return   this.clienteHttp.put<Conta>(`${this.api}${conta.id}`,
conta);
  }


  saque(saque: SaqueDeposito): Observable<SaqueDeposito>{
return
this.clienteHttp.post<SaqueDeposito>(`${this.api}${saque.conta}/saque/`
, saque);
  }
  deposito(deposito: SaqueDeposito): Observable<SaqueDeposito>{
                                                                  return
this.clienteHttp.post<SaqueDeposito>(`${this.api}${deposito.conta}/depo
sito/`, deposito);
  }


 tranferencia(transferencia: Transferencia): Observable<Transferencia>{
                                                                  return
this.clienteHttp.post<Transferencia>(`${this.api}${transferencia.conta_
origem}/transferencia/`, transferencia);
  }

}
```

3. Agora vamos gerar nosso componente de listagem de contas:
   `ng generate component pages/conta/listagem-conta`

4. Acesse **listagem-conta.component.html:**

```
    <div class="d-flex justify-content-between">
      <h1>Listagem de contas</h1>
      <button             mat-raised-button             color="primary"
routerLink="/conta/novo">Nova conta</button>
    </div>


    <table mat-table class="mt-5" [dataSource]="dataSource">
      <!-- Coluna ID -->
      <ng-container matColumnDef="id">
        <th mat-header-cell *matHeaderCellDef> No. </th>
        <td mat-cell *matCellDef="let element"> {{element.id}} </td>
      </ng-container>

      <!-- Coluna Nome -->
      <ng-container matColumnDef="numero">
       <th mat-header-cell *matHeaderCellDef> Número </th>
       <td mat-cell *matCellDef="let element"> {{element.numero}} </td>
      </ng-container>

      <!-- Coluna CPF -->
      <ng-container matColumnDef="agencia">
        <th mat-header-cell *matHeaderCellDef> Agência </th>
      <td mat-cell *matCellDef="let element"> {{element.agencia}} </td>
      </ng-container>

      <!-- Coluna Email -->
      <ng-container matColumnDef="saldo">
        <th mat-header-cell *matHeaderCellDef> Saldo </th>
        <td mat-cell *matCellDef="let element"> {{element.saldo}} </td>
      </ng-container>


      <!-- Coluna Status -->
      <ng-container matColumnDef="cliente">
        <th mat-header-cell *matHeaderCellDef> Cliente </th>
        <td mat-cell *matCellDef="let element"> {{element.nomeCliente}}
</td>
      </ng-container>


      <!-- Coluna Funcoes -->
      <ng-container matColumnDef="funcoes">
        <th mat-header-cell *matHeaderCellDef> </th>
        <td mat-cell *matCellDef="let element">
             <mat-icon fontIcon="edit" [routerLink]="'/conta/editar/' +
element.id"  style="cursor:pointer"></mat-icon>
                                                  <mat-icon    fontIcon="delete"
(click)="deletarCLiente(element.id)"></mat-icon>
        </td>
      </ng-container>

      <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
                    <tr    mat-row    *matRowDef="let    row;    columns:
displayedColumns;"></tr>
    </table>

    <mat-paginator [pageSizeOptions]="[5, 10, 20, 30]"
      showFirstLastButtons
                          [length]="dataSource.data.length     +     1"
(page)="onPageChange($event)">
    </mat-paginator>
  </div>
```

nta.component.html

5. Depois acesse **listagem-conta.component.ts:**

```
import { Component, ViewChild } from '@angular/core';
import    {    MatPaginator,    MatPaginatorModule,    PageEvent    }    from
'@angular/material/paginator';
import      {      MatTableDataSource,      MatTableModule      }      from
'@angular/material/table';
import Swal from 'sweetalert2';
import { Conta } from '../../../shared/models/conta';
import            {            ClienteService            }            from
'../../../shared/services/cliente/cliente-service';
import             {             ContaService             }             from
'../../../shared/services/conta/conta-service';
import { CommonModule } from '@angular/common';
import { MatButton } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { RouterLink } from '@angular/router';

@Component({
  selector: 'app-listagem-conta',
  imports: [
    CommonModule,
    RouterLink,
    MatTableModule,
    MatPaginatorModule,
    MatIconModule,
    MatButton,
  ],
  templateUrl: './listagem-conta.html',
  styleUrl: './listagem-conta.scss'
})
export class ListagemConta {
   displayedColumns:  string[]  =  ['id',  'numero',  'agencia',  'saldo',
'cliente', 'funcoes'];
  dataSource = new MatTableDataSource<Conta>([]);

  @ViewChild(MatPaginator) paginator!: MatPaginator;


        constructor(private    contaService:    ContaService,    private
clienteService: ClienteService){
  }


  ngAfterViewInit() {
    this.listarContas(1, 5)
  }


  listarContas(page: number, pageSize: number) {
      this.contaService.listar_paginado(page,  pageSize).subscribe(contas
=> {
      this.clienteService.listar().subscribe(clientes => {
        const contasComNomesDeClientes = contas.map(conta => {
              const  cliente  =  clientes.find(cliente  =>  cliente.id  ===
conta.cliente);
          if (cliente) {
            conta.nomeCliente = cliente.nome;
          }
          return conta;
        });
        this.dataSource.data = contasComNomesDeClientes;
      });
    });
  }




  onPageChange(event: PageEvent) {
    const pageIndex = event.pageIndex + 1;
    const pageSize = event.pageSize;
    this.listarContas(pageIndex, pageSize);
  }


  deletarCliente(id: number){
    Swal.fire({
      title: 'Você tem certeza que deseja deletar?',
      text: "Não tem como reverter essa ação",
      icon: 'warning',
      showCancelButton: true,
      confirmButtonColor: 'red',
      cancelButtonColor: 'grey',
      confirmButtonText: 'Deletar'
    }).then((result) => {
      if (result.isConfirmed) {
        this.contaService.deletar(id).subscribe({
          next: () => {
            Swal.fire({
              icon: 'success',
              title: 'Sucesso',
              text: 'Conta deletada com sucesso!',
              showConfirmButton: false,
              timer: 1500
            })
            this.listarContas(1,5)
          },
          error: (error) => {
            console.error(error)
            Swal.fire({
              icon: 'error',
              title: 'Oops...',
              text: 'Erro ao deletar conta!',
            })
          }})
      }})}

}
```

6. Agora vamos criar o componente de cadastro de conta:

```
ng generate component pages/conta/cadastro-conta
```

7. Agora o componente de cadastro de conta:
   - **cadastro-conta.component.html**

```
<div class="container mt-5 d-flex justify-content-center" >
    <div>
        <h1>{{ editar ? 'Editar conta' :'Nova conta'}} </h1>
        <form [formGroup]="formGroup" class="example-form mt-4"
(ngSubmit)="cadastrar()">
            <small class="text-danger mb-2"
*ngIf="formGroup.get('numero')?.errors &&
            formGroup.get('numero')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Número</mat-label>
                <input [readonly]="editar" matInput
formControlName="numero">
            </mat-form-field>
            <small class="text-danger mb-2"
*ngIf="formGroup.get('agencia')?.errors &&
            formGroup.get('agencia')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Agência</mat-label>
                <input [readonly]="editar" matInput
formControlName="agencia">
```

```
      </mat-form-field>
      <small class="text-danger mb-2"
*ngIf="formGroup.get('saldo')?.errors &&
      formGroup.get('saldo')?.hasError('required')">Campo
obrigatório</small>
      <mat-form-field class="example-full-width">
        <mat-label>Saldo</mat-label>
        <input matInput formControlName="saldo">
      </mat-form-field>
      <small class="text-danger mb-2"
*ngIf="formGroup.get('cliente')?.errors &&
      formGroup.get('cliente')?.hasError('required')">Campo
obrigatório</small>
      <mat-form-field class="example-full-width">
        <mat-label>Cliente</mat-label>
        <mat-select formControlName="cliente">
         <mat-option *ngFor="let cliente of clientes"
[value]="cliente.id">
          {{cliente.nome}}
         </mat-option>
        </mat-select>
      </mat-form-field>
      <!--Botao de submissao do formulario-->
      <div class="d-flex justify-content-center">
        <button [disabled]="!formGroup.valid" type="submit"
mat-raised-button color="primary">
          {{editar ? 'Editar' : 'Cadastrar'}}
        </button>
      </div>
    </form>
  </div>
</div>
```

- **cadastro-conta.component.ts**

```
      import { Component } from '@angular/core';
import { FormGroup, FormControl, Validators, FormsModule, ReactiveFormsModule
} from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import Swal from 'sweetalert2';
import { Cliente } from '../../../shared/models/cliente';
import { Conta } from '../../../shared/models/conta';
import { ClienteService } from
'../../../shared/services/cliente/cliente-service';
import { ContaService } from '../../../shared/services/conta/conta-service';
import { CommonModule } from '@angular/common';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatRadioModule } from '@angular/material/radio';
import {MatSelectModule} from '@angular/material/select';

@Component({
  selector: 'app-cadastro-conta',
  imports: [MatInputModule, MatFormFieldModule, MatRadioModule,
MatSelectModule, CommonModule, FormsModule, ReactiveFormsModule],
  templateUrl: './cadastro-conta.html',
  styleUrl: './cadastro-conta.scss'
```

```
})
export class CadastroConta {
   editar;
  formGroup: FormGroup;
  clientes: Cliente[]


  constructor(private clienteService: ClienteService, private contaService:
ContaService, private router: Router, private route: ActivatedRoute){
    this.editar = false
    this.formGroup = new FormGroup({
      id: new FormControl(null),
      numero: new FormControl('', Validators.required),
      agencia: new FormControl('', Validators.required),
      saldo: new FormControl('', Validators.required),
      cliente: new FormControl('', Validators.required)
    });
    this.clientes = []
  }


  ngOnInit(): void {
    if (this.route.snapshot.params["id"]){
      this.editar = true

this.contaService.pesquisarPorId(this.route.snapshot.params["id"]).subscribe(
        cliente => {
          this.formGroup.patchValue(cliente)
        }
      )
    }
    this.listarClientes()
  }


  listarClientes(): void{
    this.clienteService.listar().subscribe(values => {
      this.clientes = values
    })
  }




  cadastrar() {
    const conta: Conta = this.formGroup.value;
    if (this.editar) {
      this.contaService.atualizar(conta).subscribe({
        next: () => {
          Swal.fire({
            icon: 'success',
            title: 'Sucesso',
            text: 'Conta atualizada com sucesso!',
            showConfirmButton: false,
            timer: 1500
          })
          this.router.navigate(['/conta']);
        },
        error: (error) => {
          console.error(error);
          Swal.fire({
            icon: 'error',
            title: 'Oops...',
            text: 'Erro ao atualizar conta!',
          });
        }
      });
    } else {
      // Modo de criação
      this.contaService.inserir(conta).subscribe({
        next: () => {
          Swal.fire({
            icon: 'success',
            title: 'Sucesso',
            text: 'Conta cadastrada com sucesso!',
            showConfirmButton: false,
            timer: 1500
          })
          this.router.navigate(['/conta']);
        },
        error: (error) => {
          console.error(error);
          Swal.fire({
            icon: 'error',
            title: 'Oops...',
            text: 'Erro ao cadastrar conta!',
          });
        }
      });
    }
  }

}
```

8. **Hora de fazer os componentes de depósito, transferência e saque**
9. **Primeiro vamos criar os componentes**

```
ng generate component pages/conta/components/deposito
ng generate component pages/conta/components/saque
ng generate component pages/conta/components/transferencia
```

10.