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 seguido o tutorial inicial do projeto, siga as próximas etapas para finalizar:

1. Igual fizemos na parte de clientes vamos criar o módulo de contas em **src/pages/conta/conta.module**:
    - O arquivo ficará como no exemplo a seguir:

```
import { CUSTOM_ELEMENTS_SCHEMA, NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ListagemContaComponent } from
'./listagem-conta/listagem-conta.component';
import { CadastroContaComponent } from
'./cadastro-conta/cadastro-conta.component';
import { DepositoContaComponent } from
'./deposito-conta/deposito-conta.component';
import { TransferenciaContaComponent } from
'./transferencia-conta/transferencia-conta.component';
import { SaqueContaComponent } from
'./saque-conta/saque-conta.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { BrowserAnimationsModule } from
'@angular/platform-browser/animations';
import { RouterModule } from '@angular/router';
import { HttpClientModule } from '@angular/common/http';
import { MaterialModule } from
'src/app/shared/material/material.module';

@NgModule({
  declarations: [
    ListagemContaComponent,
    CadastroContaComponent,
    DepositoContaComponent,
    TransferenciaContaComponent,
    SaqueContaComponent
  ],
  imports: [
    CommonModule,
    FormsModule,
    BrowserAnimationsModule,
    RouterModule,
    ReactiveFormsModule,
```

```
    HttpClientModule,
    MaterialModule,
  ],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
  providers: [],
  exports: [
    ListagemContaComponent,
    CadastroContaComponent,
    DepositoContaComponent,
    TransferenciaContaComponent,
    SaqueContaComponent
  ]
})
export class ContaModule { }
```

2. Ficará com alguns erros porque ainda não criamos os componentes que estão declarados, crie os componentes:
    - `ng generate component pages/conta/cadastro-conta`
    - `ng generate component pages/conta/listagem-conta`
    - `ng generate component pages/conta/deposito-conta`
    - `ng generate component pages/conta/saque-conta`
    - `ng generate component pages/conta/transferencia-conta`
3. Agora 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
}
```

    - **Transferência.ts**

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

4. Agora vamos criar o serviço de contas:

   - `ng generate service shared/services/conta`

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { environment } from 'src/environments/environment.development';
import { Observable } from 'rxjs';
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=${pageS
ize}`);
  }

  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}/depos
ito/`, deposito);
  }

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

5. Agora voltando aos componentes vamos configurar o componente de listagem-contas:
    - **listagem-conta.component.html**

```
<div class="container mt-5">
    <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>
```

- **listagem-conta.component.ts**

```
import { AfterViewInit, Component, ViewChild } from '@angular/core';
import { MatPaginator, PageEvent } from '@angular/material/paginator';
import { MatTableDataSource } from '@angular/material/table';
import { Conta } from 'src/app/shared/models/Conta';
```

```
import { ClienteService } from
'src/app/shared/services/cliente.service';
import { ContaService } from 'src/app/shared/services/conta.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-listagem-conta',
  templateUrl: './listagem-conta.component.html',
  styleUrls: ['./listagem-conta.component.scss']
})
export class ListagemContaComponent implements AfterViewInit{
  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 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, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { Cliente } from 'src/app/shared/models/Cliente';
import { Conta } from 'src/app/shared/models/Conta';
import { ClienteService } from
'src/app/shared/services/cliente.service';
import { ContaService } from 'src/app/shared/services/conta.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-cadastro-conta',
  templateUrl: './cadastro-conta.component.html',
  styleUrls: ['./cadastro-conta.component.scss']
})
export class CadastroContaComponent implements OnInit{
  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"]).subsc
ribe(
        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!',
          });
        }
      });
    }
  }
}
```

7. Vamos seguir para os componentes de saque, depósito e transferência:
    - **saque.component.html**

```
<div class="container mt-5 d-flex justify-content-center" >
    <div>
        <h1>Saque</h1>
        <form [formGroup]="formGroup" class="example-form mt-4"
(ngSubmit)="cadastrar()">
            <small class="text-danger mb-2"
*ngIf="formGroup.get('valor')?.errors &&
            formGroup.get('valor')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Valor</mat-label>
                <input matInput formControlName="valor">
            </mat-form-field>
            <small class="text-danger mb-2"
*ngIf="formGroup.get('conta')?.errors &&
            formGroup.get('conta')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Conta</mat-label>
                <mat-select formControlName="conta">
                  <mat-option *ngFor="let conta of contas"
[value]="conta.id">
                    {{conta.numero}} - {{conta.agencia}} -
{{conta.nomeCliente}}
                  </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">
                    Cadastrar
                </button>
            </div>
        </form>
    </div>
</div>
```

- **saque.component.ts**

```
import { Component } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { Conta } from 'src/app/shared/models/Conta';
import { SaqueDeposito } from 'src/app/shared/models/SaqueDeposito';
import { ClienteService } from
'src/app/shared/services/cliente.service';
import { ContaService } from 'src/app/shared/services/conta.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-saque-conta',
  templateUrl: './saque-conta.component.html',
  styleUrls: ['./saque-conta.component.scss']
})
export class SaqueContaComponent {
  formGroup: FormGroup;
  contas: Conta[]

  constructor(private contaService: ContaService, private router:
Router, private clienteService: ClienteService){

    this.formGroup = new FormGroup({
      valor: new FormControl('', Validators.required),
      conta: new FormControl('', Validators.required)
    });
    this.contas = []
  }
  ngOnInit(): void {
    this.listarContas()
  }

  listarContas(): void{
    this.contaService.listar().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.contas = contasComNomesDeClientes;
      });
    })
  }

  cadastrar() {
    const saque: SaqueDeposito = this.formGroup.value;
      // Modo de criação
      this.contaService.saque(saque).subscribe({
        next: () => {
          Swal.fire({
            icon: 'success',
            title: 'Sucesso',
            text: 'Saque registrado com sucesso!',
            showConfirmButton: false,
            timer: 1500
          })
          this.router.navigate(['/conta']);
        },
        error: (error) => {
          console.error(error);
          Swal.fire({
            icon: 'error',
            title: 'Oops...',
            text: 'Erro ao registrar saque!',
          });
        }
      });
    }
}
```

- **deposito.component.html**

```
<div class="container mt-5 d-flex justify-content-center" >
    <div>
        <h1>Depósito</h1>

        <form [formGroup]="formGroup" class="example-form mt-4"
(ngSubmit)="cadastrar()">
```

```
            <!--Input de nome do cliente-->
            <small class="text-danger mb-2"
*ngIf="formGroup.get('valor')?.errors &&
            formGroup.get('valor')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Valor</mat-label>
                <input matInput formControlName="valor">
            </mat-form-field>

            <!--Input de observacoes do cliente-->
            <small class="text-danger mb-2"
*ngIf="formGroup.get('conta')?.errors &&
            formGroup.get('conta')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Conta</mat-label>
                <mat-select formControlName="conta">
                  <mat-option *ngFor="let conta of contas"
[value]="conta.id">
                    {{conta.numero}} - {{conta.agencia}} -
{{conta.nomeCliente}}
                  </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">
                    Cadastrar
                </button>
            </div>
        </form>
    </div>
</div>
```

- **deposito.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { Conta } from 'src/app/shared/models/Conta';
import { SaqueDeposito } from 'src/app/shared/models/SaqueDeposito';
import { ClienteService } from
'src/app/shared/services/cliente.service';
import { ContaService } from 'src/app/shared/services/conta.service';
```

```
import Swal from 'sweetalert2';

@Component({
  selector: 'app-deposito-conta',
  templateUrl: './deposito-conta.component.html',
  styleUrls: ['./deposito-conta.component.scss']
})
export class DepositoContaComponent implements OnInit{
  formGroup: FormGroup;
  contas: Conta[]

  constructor(private contaService: ContaService, private router:
Router, private clienteService: ClienteService){

    this.formGroup = new FormGroup({
      valor: new FormControl('', Validators.required),
      conta: new FormControl('', Validators.required)
    });
    this.contas = []
  }
  ngOnInit(): void {
    this.listarContas()
  }

  listarContas(): void{
    this.contaService.listar().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.contas = contasComNomesDeClientes;
      });
    })
  }

  cadastrar() {
    const deposito: SaqueDeposito = this.formGroup.value;
      // Modo de criação
      this.contaService.deposito(deposito).subscribe({
        next: () => {
          Swal.fire({
```

```
            icon: 'success',
            title: 'Sucesso',
            text: 'Depósito registrado com sucesso!',
            showConfirmButton: false,
            timer: 1500
          })
          this.router.navigate(['/conta']);
        },
        error: (error) => {
          console.error(error);
          Swal.fire({
            icon: 'error',
            title: 'Oops...',
            text: 'Erro ao registrar depósito!',
          });
        }
      });
    }
}
```

- **transferencia.component.html**

```
<div class="container mt-5 d-flex justify-content-center" >
    <div>
        <h1>Transferência</h1>
        <form [formGroup]="formGroup" class="example-form mt-4"
(ngSubmit)="cadastrar()">
            <small class="text-danger mb-2"
*ngIf="formGroup.get('conta_origem')?.errors &&
            formGroup.get('conta_origem')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Conta</mat-label>
                <mat-select formControlName="conta_origem">
                  <mat-option *ngFor="let conta of contas"
[value]="conta.id">
                    {{conta.numero}} - {{conta.agencia}} -
{{conta.nomeCliente}}
                  </mat-option>
                </mat-select>
            </mat-form-field>
            <small class="text-danger mb-2"
*ngIf="formGroup.get('valor')?.errors &&
            formGroup.get('valor')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Valor</mat-label>
```

```
                <input matInput formControlName="valor">
            </mat-form-field>
            <small class="text-danger mb-2"
*ngIf="formGroup.get('conta_destino')?.errors &&
            formGroup.get('conta_destino')?.hasError('required')">Campo
obrigatório</small>
            <mat-form-field class="example-full-width">
                <mat-label>Conta</mat-label>
                <mat-select formControlName="conta_destino">
                  <mat-option *ngFor="let conta of contas"
[value]="conta.id">
                    {{conta.numero}} - {{conta.agencia}} -
{{conta.nomeCliente}}
                  </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">
                    Cadastrar
                </button>
            </div>
        </form>
    </div>
</div>
```

- **transferencia.component.ts**

```
import { Component } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { Conta } from 'src/app/shared/models/Conta';
import { Transferencia } from 'src/app/shared/models/Transferencia';
import { ClienteService } from
'src/app/shared/services/cliente.service';
import { ContaService } from 'src/app/shared/services/conta.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-transferencia-conta',
  templateUrl: './transferencia-conta.component.html',
  styleUrls: ['./transferencia-conta.component.scss']
})
export class TransferenciaContaComponent {
  formGroup: FormGroup;
  contas: Conta[]
```

```
  constructor(private contaService: ContaService, private router:
Router, private clienteService: ClienteService){

    this.formGroup = new FormGroup({
      valor: new FormControl('', Validators.required),
      conta_destino: new FormControl('', Validators.required),
      conta_origem: new FormControl('', Validators.required)
    });
    this.contas = []
  }
  ngOnInit(): void {
    this.listarContas()
  }

  listarContas(): void{
    this.contaService.listar().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.contas = contasComNomesDeClientes;
      });
    })
  }

  cadastrar() {
    const tranferencia: Transferencia = this.formGroup.value;
      this.contaService.tranferencia(tranferencia).subscribe({
        next: () => {
          Swal.fire({
            icon: 'success',
            title: 'Sucesso',
            text: 'Transferência registrada com sucesso!',
            showConfirmButton: false,
            timer: 1500
          })
          this.router.navigate(['/conta']);
        },
        error: (error) => {
          console.error(error);
```

```
          Swal.fire({
            icon: 'error',
            title: 'Oops...',
            text: 'Erro ao registrar transferência!',
          });
        }
      });
    }
}
```

8. O arquivo de rotas precisará ser alterado para refletir os novos componentes:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CadastroClienteComponent } from
'./pages/cliente/cadastro-cliente/cadastro-cliente.component';
import { ListagemClienteComponent } from
'./pages/cliente/listagem-cliente/listagem-cliente.component';
import { CadastroContaComponent } from
'./pages/conta/cadastro-conta/cadastro-conta.component';
import { ListagemContaComponent } from
'./pages/conta/listagem-conta/listagem-conta.component';
import { DepositoContaComponent } from
'./pages/conta/deposito-conta/deposito-conta.component';
import { SaqueContaComponent } from
'./pages/conta/saque-conta/saque-conta.component';
import { TransferenciaContaComponent } from
'./pages/conta/transferencia-conta/transferencia-conta.component';

const routes: Routes = [
  {
    path: 'cliente',
    children: [
      {
        path: 'novo',
        component: CadastroClienteComponent
      },
      {
        path: 'editar/:id',
        component: CadastroClienteComponent
      },
      {
        path: '',
        component: ListagemClienteComponent,
      },
    ]
  },
```

```
  {
    path: 'conta',
    children: [
      {
        path: 'novo',
        component: CadastroContaComponent
      },
      {
        path: 'editar/:id',
        component: CadastroContaComponent
      },
      {
        path: 'deposito',
        component: DepositoContaComponent
      },
      {
        path: 'saque',
        component: SaqueContaComponent
      },
      {
        path: 'transferencia',
        component: TransferenciaContaComponent
      },
      {
        path: '',
        component: ListagemContaComponent,
      },
    ]
  },
  {
    path: '',
    component: ListagemClienteComponent,
  },

];

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

Para mais detalhes do projeto como navbar, importações e css dos componentes (para deixar os formulários bem bonitos e organizados) segue o link do github:

https://github.com/Nicoly-Almeida/sistema_bancario/