Stack Angular

**Introdução**

Angular é uma plataforma de aplicações web de código-fonte aberto e front-end baseado em TypeScript liderado pela Equipe Angular do Google e por uma comunidade de indivíduos e corporações. Angular é uma reescrita completa do AngularJS, feito pela mesma equipe que o construiu. A grande proposta do Angular é o desenvolvimento de uma nova estrutura de aplicações web conhecida como SPA (sigla para ***Single Page Applications*** ou ***Aplicações de Página Única***).

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

​<https://angular.io/>​

Uma SPA é basicamente uma aplicação web construída em uma só página, na qual a interação e a navegação entre as sessões de uma página ocorrem de maneira a qual não é necessário recarregar a página em cada uma dessas mudanças.

A grande proposta é nos oferecer ferramentas para o desenvolvimento de aplicações SPA, com a perpectiva de desenvolvimento simples e otimizado. Com Angular também podemos desenvolver aplicações para plataformas desktop quanto mobile, tornando-as dinâmicas, modernas e escaláveis.

Angular possui um novo paradigma de desenvolvimento focado nos dados da aplicação. Ele não utiliza uma virtualização do DOM para manipulá-lo: ele utiliza mecanismos próprios de detecção de alterações na interface conhecida **Two-Way Data Binding**.

Mas antes de pensar em dominar Angular, é extremamente relevante compreender sobre JavaScript e TypeScript.

**JavaScript**

JavaScript é uma linguagem de programação de alto nível criada, a princípio, para ser executada em navegadores e manipular comportamentos de páginas web.

Logotipo

Descrição gerada automaticamente

Segundo a **Mozilla Foundation**, atual nome da antiga **Netscape Communications Corporations**, empresa responsável pela criação do JS, "*JavaScript é uma linguagem de programação, leve, interpretada, orientada a objetos, baseada em protótipos e em first-class functions (funções de primeira classe), mais conhecida como a linguagem de script da Internet.*"

Hoje através do projeto NodeJS somos capazes de também utilizar o JavaScript no lado do servidor como linguagem de Back-end.

**TypeScript**

Typescript é uma linguagem de código aberto desenvolvida pela Microsoft que foi construída em cima do Javascript, que é muito difundido atualmente. Então esse “superset” foi criado para adicionar recursos de tipagem estáticas à linguagem original.

Em outras palavras, temos todas as funcionalidades do Javascript no Typescript acrescidas de várias outras funcionalidades que caracterizam o Typescript.

Logotipo

Descrição gerada automaticamente

**Referências**

**Tecnologias**

Atualmente para se desenvolver aplicações web é necessário mais que HTML, CSS e JavaScript. Hoje estamos na era dos **Web Components** ou desenvolvimento baseado a componentes, e isto nos exige algumas outras tecnologias e ferramentas que tornarão nosso desenvolvimento mais organizado e reutilizável.

* **JavaScript**: Linguagem oficial para desenvolvimento web.
* **TypeScript**: Superset que adiciona recursos como tipagem de dados e outros recursos sobre JavaScript
* **NodeJS**: Plataforma que nos permite rodar JavaScript fora do navegador.
* **Angular**: Framework JavaScript que provê mecanismos de manipulação da DOM e o padrão SPA.
* **VS Code**: IDE para desenvolvimento Web desenvolvida pela Microsoft.

Desenho de placa de sinalização

Descrição gerada automaticamente com confiança média

​

**Configuração**

Antes de tudo precisaremos instalar o NodeJS em nossa máquina.

Link para download: <https://nodejs.org/en/download/>​

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Em seguida o Visual Studio Code: <https://code.visualstudio.com/download>​

Linha do tempo

Descrição gerada automaticamente

Vamos validar se instalamos os recursos citados acima de forma correta:

* No VSCode acesse a opção Terminal -> Novo Terminal
* Digite o comando **node --version**

**Projeto**

Esta etapa do projeto consiste em criar uma aplicação web que se comunicará via requisições https baseadas na arquitetura REST em nossa Brankline-API desenvolvida com Springboot e disponibilzada no Heroku.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Iremos explorar recursos essenciais em um desenvolvimento utilizando Angular conforme abaixo:

* Angular 13: <https://angular.io/>
* Angular HttpClient: <https://angular.io/api/common/http/HttpClient>
* Angular Router: <https://angular.io/api/router>
* Bootstrap 5: <https://getbootstrap.com/>

**Criando o projeto com angular-cli**

Vamos criar no diretório **c:\estudos** o projeto **bankline-app** utilizando a *interface de linha de comandos* o angular-cli.

* Primeiramente iremos confirmar se estamos com o node e npm instalados

1

node --version

2

​

3

e

4

​

5

npm --version

Copied!

* Depois precisaremos instalar o **angular-cli**

1

npm install -g @angular/cli

Copied!

* Em seguida será possível executar os comandos angular pelo terminal

1

// obtendo a versão do Angular CLI

2

​

3

ng --version

4

​

5

//gerando uma nova aplicação

6

//é importante que este comando segue executado em c:\estudos

7

//deste forma já teremos um recurso de teste e o TypeScript configurado

8

​

9

ng new bankline-app

10

​

11

//após abrir o projeto no VSCode

12

ng serve -o

Copied!

Conhecendo os arquivos do nosso projeto:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

* **node\_modules:** pasta contendo todas as dependências do projeto
* **src**: pasta que contém todo o código fonte do projeto
* **.editorconfig**: arquivo de configuração do VSCode para projeto Angular
* **.gitignore**: arquivo que configuramos para determinar pastas, extensões e arquivos que não devem ser sincronizados com github
* **angular.json**: Arquivo para as configurações do angular
* **package.json**: arquivo para configurar definições e dependências do projeto
* **README.md**: Arquivo que contém a descrição de nosso projeto.
* **tsconfig.json**: Arquivo de configuração do TypeScript, exemplo qual versão do JavaScript você deseja compilar
* **tsconfig.spec.json**: Arquivo de configuração para testes da aplicação.
* **src/app:** Pasta principal contendo todos arquivos para a criação das funcionalidades da aplicação.
* **src/assets:** Pasta que contém arquivos javascript, css, imagens adicionais.
* **src/environments:** Diretório pertinente aos ambientes da aplicação
* **src/favicon.ico:** Ícone da aplicação
* **src/index.html:** Página principal da aplicação
* **src/main.ts:** Arquivo principal da aplicação
* **src/polyfills.ts**: Arquivo responsável por carregar os módulos da aplicação.
* **src/styles.css**: Arquivo para estilização global da aplicação.
* **src/teste.ts**: Arquivo para carregar nossos testes

**Construção do app**

Após analisar a estrutura do projeto criado, iremos ajustes alguns arquivos e criar nossos components, services e demais recursos necessários.

* Criando o componente para exibir as movimentações

1

ng g c components/movimentacao-list

Copied!

* Criando o componente para adicionar uma nova movimentação

1

ng g c components/movimentacao-new

Copied!

* Configurando a rota para exibir nossos novos componentes

1

// localize o arquivo src/app/app-routing.module.ts

2

​

3

import { MovimentacaoNewComponent } from './components/movimentacao-new/movimentacao-new.component';

4

import { MovimentacaoListComponent } from './components/movimentacao-list/movimentacao-list.component';

5

​

6

const routes: Routes = [

7

​

8

{ path: 'movimentacoes-new', component: MovimentacaoNewComponent },

9

{ path: 'movimentacoes', component: MovimentacaoListComponent },

10

{ path: '', redirectTo: 'movimentacoes', pathMatch: 'full' },

11

​

12

];

Copied!

* Removendo conteúdo gerado pelo Angular e exibir nossos componentes:

1

<!-- Toolbar -->

2

<div class="toolbar" role="banner">

3

<img

4

width="40"

5

alt="Angular Logo"

6

src=""

7

/>

8

<span>{{ title }}</span>

9

<div class="spacer"></div>

10

<a aria-label="Angular on twitter" target="\_blank" rel="noopener" href="https://twitter.com/angular" title="Twitter">

11

<svg id="twitter-logo" height="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">

12

<rect width="400" height="400" fill="none"/>

13

<path d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,0,1-29.46,8.07,51.47,51.47,0,0,0,22.55-28.37,102.79,102.79,0,0,1-32.57,12.45,51.34,51.34,0,0,0-87.41,46.78A145.62,145.62,0,0,1,92.4,107.81a51.33,51.33,0,0,0,15.88,68.47A50.91,50.91,0,0,1,85,169.86c0,.21,0,.43,0,.65a51.31,51.31,0,0,0,41.15,50.28,51.21,51.21,0,0,1-23.16.88,51.35,51.35,0,0,0,47.92,35.62,102.92,102.92,0,0,1-63.7,22A104.41,104.41,0,0,1,75,278.55a145.21,145.21,0,0,0,78.62,23" fill="#fff"/>

14

</svg>

15

</a>

16

<a aria-label="Angular on YouTube" target="\_blank" rel="noopener" href="https://youtube.com/angular" title="YouTube">

17

<svg id="youtube-logo" height="24" width="24" data-name="Logo" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="#fff">

18

<path d="M0 0h24v24H0V0z" fill="none"/>

19

<path d="M21.58 7.19c-.23-.86-.91-1.54-1.77-1.77C18.25 5 12 5 12 5s-6.25 0-7.81.42c-.86.23-1.54.91-1.77 1.77C2 8.75 2 12 2 12s0 3.25.42 4.81c.23.86.91 1.54 1.77 1.77C5.75 19 12 19 12 19s6.25 0 7.81-.42c.86-.23 1.54-.91 1.77-1.77C22 15.25 22 12 22 12s0-3.25-.42-4.81zM10 15V9l5.2 3-5.2 3z"/>

20

</svg>

21

</a>

22

</div>

23

​

24

<div class="content" role="main">

25

<router-outlet></router-outlet>

26

</div>

27

​

28

​

29

​

Copied!

* Adicionando a biblioteca do Bootstrap pelo angular-cli

1

1. no terminal, pare a aplicação com CTRL+C

2

​

3

2. ng add @ng-bootstrap/ng-bootstrap

Copied!

Se ao tentar instalar o bootstrap apresentar algum erro, execute os comandos abaixo e repita o processo acima.

npm config set legacy-peer-deps true

* Criando uma variável que representará as nossas movimentações

1

// movimentacao-list.component.ts

2

​

3

movimentacoes:any;

Copied!

* Estruturando a tabela que exibibirá as movimentações

1

// movimentacao-list.component.html

2

​

3

<table class="table table-striped">

4

<thead>

5

<tr>

6

<th scope="col">#</th>

7

<th scope="col">Data Hora</th>

8

<th scope="col">Descrição</th>

9

<th scope="col">Valor</th>

10

<th scope="col">Tipo</th>

11

</tr>

12

</thead>

13

<tbody>

14

<tr \*ngFor="let item of movimentacoes; index as i">

15

<th scope="row">{{ i + 1 }}</th>

16

<td>{{ item.dataHora }}</td>

17

<td>{{ item.descricao }}</td>

18

<td>{{ item.valor }}</td>

19

<td>{{ item.tipo }}</td>

20

</tr>

21

</tbody>

22

</table>

Copied!

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

**Back-end CORS**

Antes de pensar em utilizar a API, verifique se foi habilitado o recurso de **CORS.**

*Cross-Origin Resource Sharing* ou *CORS* é um mecanismo que permite que recursos restritos em uma página da web sejam recuperados por outro domínio fora do domínio ao qual pertence o recurso que será recuperado

1

// WebConfig.java

2

​

3

package com.dio.santander.bankline.api.config;

4

​

5

import org.springframework.context.annotation.Configuration;

6

import org.springframework.web.servlet.config.annotation.CorsRegistry;

7

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

8

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

9

​

10

@Configuration

11

@EnableWebMvc

12

public class WebConfig implements WebMvcConfigurer {

13

​

14

@Override

15

public void addCorsMappings(CorsRegistry registry) {

16

registry.addMapping("/\*\*");

17

}

18

}

Copied!

**​**

**Integrando com nossa API**

Iremos agora implementar mais uma camada da aplicação, os serviços que estarão interagindo com a nossa api REST.

* Criando o serviço de movimentações

1

ng g s services/movimentacao

Copied!

* Como iremos trabalhar com requisições HTTPs e formulário, já vamos adicionando duas dependências no projeto.

1

// app.module.ts

2

​

3

import { FormsModule } from '@angular/forms';

4

import { HttpClientModule } from '@angular/common/http';

5

​

6

@NgModule({

7

imports: [

8

FormsModule,

9

HttpClientModule

10

],

11

12

})

Copied!

* Criando nosso primeiro recurso para obter as movimentações já realizadas

1

// movimentacao.service.ts

2

​

3

import { Injectable } from '@angular/core';

4

import { HttpClient } from '@angular/common/http';

5

import { Observable } from 'rxjs';

6

​

7

//mude para sua aplicação

8

const baseUrl = 'http://localhost:8080';

9

​

10

@Injectable({

11

providedIn: 'root'

12

})

13

export class MovimentacaoService {

14

constructor(private http: HttpClient) { }

15

16

list(): Observable<any> {

17

return this.http.get(`${baseUrl}/movimentacoes`);

18

}

19

20

}

Copied!

* Agora será necessário determinar que quando o componente de listagem de movimentações for carregado seja realizada uma requisição à nossa API.

1

// movimentacao-list.component.ts

2

​

3

import { Component, OnInit } from '@angular/core';

4

import { MovimentacaoService } from 'src/app/services/movimentacao.service';

5

​

6

@Component({

7

selector: 'app-movimentacao-list',

8

templateUrl: './movimentacao-list.component.html',

9

styleUrls: ['./movimentacao-list.component.css']

10

})

11

​

12

​

13

export class MovimentacaoListComponent implements OnInit {

14

movimentacoes:any;

15

​

16

constructor(private movimentacaoService: MovimentacaoService) { }

17

​

18

ngOnInit(): void {

19

this.listMovimentacoes();

20

}

21

listMovimentacoes(): void {

22

this.movimentacaoService.list()

23

.subscribe(

24

data => {

25

this.movimentacoes = data;

26

console.log(data);

27

},

28

error => {

29

console.log(error);

30

});

31

}

32

​

33

}

34

​

35

​

Copied!

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Formatando campos na tela**

Para formatação de valores do tipo data e monetário por exemplo, o Angular dispõe de um recurso denominado de Pipe.

Vamos ajustar alguns arquivos para formatar os campos Data Hora e Valor.

* Adicionado algumas configurações no módulo do projeto

1

// app.module.ts

2

​

3

import { LOCALE\_ID } from '@angular/core';

4

import { registerLocaleData } from '@angular/common';

5

import localePt from '@angular/common/locales/pt';

6

​

7

registerLocaleData(localePt);

8

...

9

​

10

providers: [{provide: LOCALE\_ID, useValue: 'pt-BR'}],

Copied!

* Adicionando as formatações nos campos Data Hora e Valor

1

// movimentacao-list.component.html

2

​

3

<td>{{ item.dataHora | date: 'dd/MM/yyyy HH:mm:ss' }}</td>

4

<td>{{ item.valor | currency:'BRL' }}</td>

Copied!

* Aplicando estilo com base em uma condição em nossos registros.

1

// movimentacao-list.component.html

2

​

3

<td [ngStyle]="{'color':item.tipo === 'RECEITA' ? 'blue' : 'red'}">{{ item.valor | currency:'BRL' }}</td

Copied!

* Adicionando o botão Novo na página de movimentação.

1

// movimentacao-list.component.html

2

​

3

//link tem o mesmo estilo que um button

4

​

5

<a class="btn btn-outline-primary btn-sm">Nova Movimentação</a>

6

​

Copied!

* Centralizando o conteúdo da página

1

// movimentacao-list.component.html

2

​

3

<div class="container">

4

​

5

//conteúdo já existente

6

​

7

</div>

Copied!

* Configurando a rota para a página para cadastrar uma nova movimentação.

1

// movimentacao-list.component.html

2

​

3

<a routerLink="/movimentacoes-new">Novo</a>

4

​

Copied!

**Nova Movimentação**

O primeiro passo é exibir os campos de entrada para gerar uma nova movimentação conforme conteúdo abaixo:

1

// movimentacao-new.component.html

2

<div class="container">

3

<div class="jumbotron">

4

<h1 class="display-6">Movimentação</h1>

5

<p class="lead-1">Inclusão de uma nova movimentação</p>

6

<hr class="my-2">

7

8

9

<div class="col-md-3">

10

<label for="iData" class="form-label">Data</label>

11

<input type="date" class="form-control" id="iData">

12

</div>

13

<div class="col-md-9">

14

<label for="iCorrentista" class="form-label">Correntista</label>

15

<select id="iCorrentista" class="form-select">

16

<option selected>Choose...</option>

17

<option>...</option>

18

</select>

19

</div>

20

<div class="col-12">

21

<label for="iDescricao" class="form-label">Descrição</label>

22

<input type="text" class="form-control" id="iDescricao" placeholder="Informe uma descrição">

23

</div>

24

<div class="col-md-3">

25

<label for="iTipo" class="form-label">Tipo Movimentação</label>

26

<select id="iTipo" class="form-select">

27

<option selected>Tipos</option>

28

<option>...</option>

29

</select>

30

</div>

31

<div class="col-md-2">

32

<label for="iValor" class="form-label">R$ Valor</label>

33

<input type="text" class="form-control" id="iValor">

34

</div>

35

<div class="d-grid gap-2 d-md-flex ">

36

<button class="btn btn-outline-primary me-md-2" type="button">Confirmar</button>

37

<a routerLink="/movimentacoes" class="btn btn-outline-secondary">Cancelar</a>

38

</div>

39

40

</div>

41

​

42

</div>

43

​

44

​

Copied!

Observe que alguns campos são opções já definidas como **Correntista** e **Tipo Movimentação**, estes dois são campos de seleção em nossa página.

* Preenchendo as opções do **Tipo Movimentação**

1

<select id="iTipo" class="form-select">

2

<option selected>RECEITA</option>

3

<option>DESPESA</option>

4

</select>

Copied!

* Para preencher o campo seleção do campo Correntista precisamos trazer os dados da nossa API, sendo assim, iremos realizar alguns procedimentos adicionais:

1

// criando o serviço para acessar os recursos de Correntista

2

​

3

ng g s services/correntista

Copied!

A estrutura dos services são muito semelhantes, e para criar um componente no Angular você precisar para a aplicar a aplicação e utilizar o terminal.

1

// services/correntista.service.ts

2

​

3

import { Injectable } from '@angular/core';

4

import { HttpClient } from '@angular/common/http';

5

import { Observable } from 'rxjs';

6

const baseUrl = 'http://localhost:8080';

7

@Injectable({

8

providedIn: 'root'

9

})

10

export class CorrentistaService {

11

constructor(private http: HttpClient) { }

12

13

list(): Observable<any> {

14

return this.http.get(`${baseUrl}/correntistas`);

15

}

16

Copied!

* Vamos solicitar para o nosso componente movimentacao-new.component.ts conectar a nossa API para buscar os **correntistas** do banco de dados.

1

// compoenents/movimentacao-new.component.ts

2

​

3

import { Component, OnInit } from '@angular/core';

4

import { CorrentistaService } from 'src/app/services/correntista.service';

5

import { MovimentacaoService } from 'src/app/services/movimentacao.service';

6

​

7

@Component({

8

selector: 'app-movimentacao-new',

9

templateUrl: './movimentacao-new.component.html',

10

styleUrls: ['./movimentacao-new.component.css']

11

})

12

export class MovimentacaoNewComponent implements OnInit {

13

correntistas:any;

14

​

15

constructor(

16

private movimentacaoService: MovimentacaoService,

17

private correntistaService: CorrentistaService,

18

) { }

19

​

20

ngOnInit(): void {

21

this.exibirCorrentistas();

22

}

23

exibirCorrentistas(): void {

24

this.correntistaService.list()

25

.subscribe(

26

data => {

27

this.correntistas = data;

28

console.log(data);

29

},

30

error => {

31

console.log(error);

32

});

33

}

34

​

35

}

36

​

Copied!

* Agora iremos exibir os nossos correntistas na tela de nova movimentação

1

// movimentacao-new.component.html

2

<select id="iCorrentista" [(ngModel)]="correntista" class="form-select">

3

<option \*ngFor="let c of correntistas" [ngValue]="c">{{c.cpf}}-{{c.nome}}</option>

4

</select>

Copied!

* Para poder identificar qual foi o correntista selecionado, agora iremos criar uma variável correntista em nosso componente **movimentacao-new.component.ts**.

1

correntista:any;

Copied!

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Para finalizar a nossa tela da movimentação, precisaremos agora somente definir uma máscara no campos **R$ Valor**, iremos instalar o componente ng2-currency-mask conforme procedimento no próprio site do **npm**.

Link: <https://www.npmjs.com/package/ng2-currency-mask>​

**Incluindo uma movimentação**

Com a tela devidamente estruturada é hora de incluir nossa primeira movimentação pela a aplicação.

1

// services/movimentacao.service.ts

2

​

3

// adicionando o método de inclusão (POST) via API

4

create(movimentacao:any): Observable<any> {

5

return this.http.post(`${baseUrl}/movimentacoes`,movimentacao);

6

}

Copied!

1

// components/movimentacao-new.components.ts

2

​

3

// criando o método para ser chamado na tela e assim incluir a movimentação

4

​

5

save(): void {

6

console.log(this.correntista)

7

const movimentacao = {

8

valor:this.valor,

9

descricao:this.descricao,

10

tipo:this.tipo,

11

idConta:this.correntista.id,

12

dataHora:this.dataHora

13

​

14

};

15

console.log(movimentacao);

16

this.movimentacaoService.create(movimentacao)

17

.subscribe(

18

response => {

19

console.log(response);

20

},

21

error => {

22

console.log(error);

23

});

24

}

Copied!

Para finalizar, agora só é preciso executar o método save no click do botão **Confirmar**.

1

// components/movimentacao-new.components.html

2

​

3

<button class="btn btn-outline-primary me-md-2 btn-sm" type="button" (click)="save()">Confirmar</button>

4

​

5

//(click)="save()"

Copied!

**Movimentações por Correntista**

Precisamos ter uma interação constante com a API Back-end, e isso será via swagger. Verifique se temos um serviço para listar as movimentações por correntistas, caso não existe, é necessário criar uma nova funcionalidade no back-end.

1

// MovimentacaoRepository.java

2

​

3

public MovimentacaoRepository{

4

...

5

public List<Movimentacao>findByIdConta(Integer idConta);

6

}

Copied!

1

// MovimentacaoController.java

2

​

3

public class MovimentacaoController {

4

...

5

@GetMapping("/{idConta}")

6

public List<Movimentacao> findAll(@PathVariable("idConta") Integer idConta){

7

return repository.findByIdConta(idConta);

8

}

9

}

Copied!

* Refinando nossa tela de consulta de movimentações

1

<div class="container">

2

<div class="jumbotron">

3

<h1 class="display-6">Movimentações</h1>

4

<p class="lead-1">Listagem das movimentações por correntista</p>

5

<hr class="my-2">

6

<div class="input-group">

7

<select id="iCorrentista" [(ngModel)]="correntista" class="form-select">

8

<option \*ngFor="let c of correntistas" [ngValue]="c" >{{c.cpf}}-{{c.nome}}</option>

9

</select>

10

<button class="btn btn-outline-secondary me-md-2 btn-sm" type="button" (click)="listMovimentacoes()">Buscar</button>

11

<a class="btn btn-outline-primary btn-sm" routerLink="/movimentacoes-new">Nova Movimentação</a>

12

<a class="btn btn-outline-primary btn-sm" routerLink="/correntistas">Correntistas</a>

13

</div>

14

<table class="table table-striped">

15

<thead>

16

<tr>

17

<th scope="col">#</th>

18

<th scope="col">Data Hora</th>

19

<th scope="col">Descrição</th>

20

<th scope="col">Valor</th>

21

<th scope="col">Tipo</th>

22

</tr>

23

</thead>

24

<tbody>

25

<tr \*ngFor="let item of movimentacoes; index as i">

26

<th scope="row">{{ i + 1 }}</th>

27

<td>{{ item.dataHora | date: 'dd/MM/yyyy HH:mm:ss' }}</td>

28

<td>{{ item.descricao }}</td>

29

<td [ngStyle]="{'color':item.tipo === 'RECEITA' ? 'blue' : 'red'}">{{ item.valor | currency:'BRL' }}</td>

30

<td>{{ item.tipo }}</td>

31

</tr>

32

</tbody>

33

</table>

34

</div>

35

</div>htm

Copied!

* Criando o serviço para listar as movimentações por correntista

1

// services/movimentacao.service.ts

2

​

3

findByIdConta(idConta:any): Observable<any> {

4

return this.http.get(`${baseUrl}/movimentacoes/${idConta}`);

5

}

Copied!

1

// components/movimentacao-list.component.ts

2

​

3

import { Component, OnInit } from '@angular/core';

4

import { CorrentistaService } from 'src/app/services/correntista.service';

5

import { MovimentacaoService } from 'src/app/services/movimentacao.service';

6

​

7

@Component({

8

selector: 'app-movimentacao-list',

9

templateUrl: './movimentacao-list.component.html',

10

styleUrls: ['./movimentacao-list.component.css']

11

})

12

​

13

​

14

export class MovimentacaoListComponent implements OnInit {

15

movimentacoes:any;

16

correntistas:any;

17

correntista:any={};

18

constructor(

19

private movimentacaoService: MovimentacaoService,

20

private correntistaService: CorrentistaService,

21

) { }

22

ngOnInit(): void {

23

this.exibirCorrentistas();

24

}

25

26

listMovimentacoes(): void {

27

this.movimentacaoService.findByIdConta(this.correntista.id)

28

.subscribe(

29

data => {

30

this.movimentacoes = data;

31

console.log(data);

32

},

33

error => {

34

console.log(error);

35

});

36

}

37

exibirCorrentistas(): void {

38

this.correntistaService.list()

39

.subscribe(

40

data => {

41

this.correntistas = data;

42

console.log(data);

43

},

44

error => {

45

console.log(error);

46

});

47

}

48

49

}

50

​

Copied!

**Correntistas**

Após a finalização da funcionalidade de movimentação, agora é necessário também gerenciarmos os correntistas pela aplicação. O processo é muito semelhante a etapa anterior.

1

//terminal

2

​

3

ng g c components/correntista

Copied!

1

// app-routing.module.ts

2

​

3

import { CorrentistaComponent } from './components/correntista/correntista.component';

4

const routes: Routes = [

5

...

6

{ path: 'correntistas', component: CorrentistaComponent },

7

]

Copied!

Navegação simples para a rota correntistas e **CorrentistaComponent**

1

// movimentacao-list.component.html

2

​

3

<a class="btn btn-outline-primary btn-sm" routerLink="/correntistas">Correntistas</a>

Copied!

Tela para inclusão e listagem de correntistas

1

// components/correntista.component.html

2

​

3

<div class="container">

4

<div class="jumbotron">

5

<h1 class="display-6">Correntistas</h1>

6

<p class="lead-1">Cadastro de correntista</p>

7

<hr class="my-2">

8

</div>

9

<div class="row g-2">

10

<div class="col-md-3">

11

<input class="form-control" id="iCpf" [(ngModel)]="cpf" placeholder="CPF">

12

</div>

13

<div class="col-5">

14

<input type="text" class="form-control" id="iNome" [(ngModel)]="nome" placeholder="Nome do Correntista">

15

</div>

16

<div class="col-3">

17

<div class="d-grid gap-2 d-md-flex ">

18

<button class="btn btn-outline-primary me-md-2" type="button" (click)="save()" >Gravar</button>

19

<a routerLink="/movimentacoes" class="btn btn-outline-danger">Voltar</a>

20

</div>

21

</div>

22

</div>

23

<p></p>

24

<div \*ngFor="let item of correntistas; index as i" class="card" style="width: 18rem;">

25

<div class="card-body">

26

<h5 class="card-title">{{item.nome}}</h5>

27

<p class="card-text">Conta N° {{item.conta.numero}}</p>

28

<p class="card-text">Saldo {{item.conta.saldo | currency:'BRL'}}</p>

29

</div>

30

</div>

31

</div>

Copied!

Regra de negócio para inclusão e listagem dos correntistas

1

// components/correntista.component.ts

2

​

3

import { Component, OnInit } from '@angular/core';

4

import { CorrentistaService } from 'src/app/services/correntista.service';

5

​

6

​

7

@Component({

8

selector: 'app-correntista',

9

templateUrl: './correntista.component.html',

10

styleUrls: ['./correntista.component.css']

11

})

12

export class CorrentistaComponent implements OnInit {

13

correntistas:any;

14

cpf:any;

15

nome:any;

16

constructor(

17

private correntistaService: CorrentistaService,

18

) { }

19

ngOnInit(): void {

20

this.exibirCorrentistas();

21

}

22

exibirCorrentistas(): void {

23

this.correntistaService.list()

24

.subscribe(

25

data => {

26

this.correntistas = data;

27

console.log(data);

28

},

29

error => {

30

console.log(error);

31

});

32

}

33

save(): void {

34

const correntista = {

35

cpf:this.cpf,

36

nome:this.nome

37

};

38

console.log(correntista);

39

this.correntistaService.create(correntista)

40

.subscribe(

41

response => {

42

console.log(response);

43

this.exibirCorrentistas();

44

},

45

error => {

46

console.log(error);

47

});

48

}

49

}

50

​

Copied!

Criando mais uma funcionalidade no serviço de correntistas

1

// services/correntista.service.ts

2

​

3

create(correntista:any): Observable<any> {

4

return this.http.post(`${baseUrl}/correntistas`,correntista);

5

}

Copied!

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Referências**

Angular

Events Binding

Desenho de placa de sinalização de trânsito

Descrição gerada automaticamente

Angular

Angular Cli

Angular

Angular Generate