Projeto Material Table

tabela de dados, paginação e classificação de dados usando a biblioteca angular material

traduzido e adaptado do original escrito por Digamber

Este projeto tem como objetivo criar um aplicativo Angular para implementar uma tabela de dados e entender como funciona a dependência Angular Material. O Angular Material é uma biblioteca de criação de componentes de interface do usuário e possui uma grande variedade de elementos. Para este projeto, será usado o recurso data table(tabelas de dados) para fazer a exibição de dados tabulados dentro da pagina que será construída.

A primeira etapa é o passo onde será executada a configuração do ambiente para que seja possível fazer uso da biblioteca material. Serão instalados e configurados: a aplicação Angular - com a mais recente CLI angular; na sequência, serão instaladas a biblioteca de interface do usuário do Angular material e, em seguida, será implementada a tabela de dados e a funcionalidade de paginação.

1 - Instalar e configurar o projeto Angular

Para executar e configurar o projeto Angular implemente o seguinte comando no prompt – dentro do local onde se deseja armazenar o projeto:

```
ng new material-data-table
```

As perguntas feitas – seqeuncialmente – serão respondidas como se seuge abaixo:

```
# ? Would you like to add Angular routing? = No

# ? Which stylesheet format would you like to use? = CSS
```

Navegue até a pasta do projeto recém criado

```
cd material-data-table
```

2 - Instalar e configurar a biblioteca Material

Para configurar a biblioteca de interface Material implemente o seguinte comando no prompt ou powershel do VS Code.

```
ng add @angular/material
```

Selecione o tema Angular Material entre as opções fornecidas. Observe as instruções abaixo:

? Choose a prebuilt theme name, or "custom" for a custom theme: Indigo/Pink❯ Indigo/Pink[Preview: https://material.angular.io?theme=indigo-pink]

Deep Purple/Amber[Preview: https://material.angular.io?theme=deeppurple-amber]

Pink/Blue Grey[Preview: https://material.angular.io?theme=pink-bluegrey]

Purple/Green[Preview: https://material.angular.io?theme=purple-green]

Para importar o tema do angular material, inclua o código abaixo no arquivo localizado em *src* > *index.html*. Observe – antes – se, automaticamente, já foi inserido.

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MaterialDataTable</title>
  <base href="/">
  <meta name="viewport" content="width=device-
width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Robo
to:wght@300;400;500&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Mate
rial+Icons" rel="stylesheet">
</head>
<body class="mat-typography">
  <app-root></app-root>
</body>
</html>
```

Navegue até o arquivo *src* > *styles.css* e adicione o seguinte código.

styles.css

```
/* You can add global styles to this file, and also import
other style files */
/*importanteo o material de forma golbal*/
@import "~@angular/material/prebuilt-themes/indigo-
pink.css";
```

Na sequência, execute o projeto seguindo a instrução indicada abaixo – o comando deve ser executada a partir da pasta do projeto:

ng serve --open

3 – Criar um centralizador de dependências para o módulo Angular material

Para melhor gerenciamento do projeto, pode ser criado um arquivo de módulo para centralizar todas as dependências do modulo angular material. Crie o novo arquivo dentro da pasta app:

new file ⟶ material.module.ts

Navegue até o arquivo *app>material.module.ts*. Aqui, serão centralizadas todas as dependências de componentes de interface Angular Material; esse arquivo será importado no arquivo *app.module.ts.*

material.module.ts

```
// este é nosso arquivo central de dependências
// da biblioteca Material

import { NgModule } from '@angular/core';
import { MatTableModule } from '@angular/material/table';

// vamos construir nossos arrays de registro

@NgModule({
    imports: [
        MatTableModule
    ],
    exports: [
        MatTableModule
    ]
})

export class AngularMaterialModule{}
```

As dependências foram importadas com êxito; o componente *MatTableModule* traz a possibilidade de se trabalhar com a propriedade *mat-table* da biblioteca da Angular Material.
Agora, navegue até o arquivo *app.module.ts* e importe a classe *AngularMaterialModule,* de seu arquivo de origem e, na sequencia, faça o registro do modulo no *array imports.* Observe o código abaixo:

app.module.ts

```
// aqui vamos importar o CUSTOM SCHEMA
import { NgModule, CUSTOM_ELEMENTS_SCHEMA } from '@angular/
core';
import { BrowserModule } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
// aqui estamos importando o arquivo central de dependencia
s da lib Material
import { AngularMaterialModule } from './material.module';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    AngularMaterialModule
  ],
  providers: [],
  bootstrap: [AppComponent],
  schemas: [CUSTOM_ELEMENTS_SCHEMA]
})
export class AppModule { }
```

4 - utilizando MatTableModule

A propriedade mat-table fornece uma tabela de dados a partir do estilo proposto pelo modulo Material Design e pode ser usada para exibir dados tabulares.
Esta tabela tem como base da tabela de dados CDK e usa uma interface semelhante para modelo de dados; seus seletores de elemento e atributo serão prefixados com *mat-* e não com o prefixo *cdk* (referente ao *Component Dev Kit*)

Agora, para criar a tabela dentro do projeto, acesse o arquivo *app.component.html*. Observe o código abaixo e implemente-o como se segue:

app.component.html

```
<table mat-table [dataSource]="dataSource" class="mat-
elevation-z8">
  <!-- Coluna Posição -->
  <ng-container matColumnDef="position">
    <th mat-header-cell *matHeaderCellDef> Posição </th>
    <td mat-
cell *matCellDef="let element"> {{element.position}} </td>
  </ng-container>
  <!-- Coluna Nome -->
  <ng-container matColumnDef="name">
    <th mat-header-cell *matHeaderCellDef> Nome </th>
    <td mat-
cell *matCellDef="let element"> {{element.name}} </td>
```

```
  </ng-container>
  <!-- Coluna Weight (Peso)-->
  <ng-container matColumnDef="weight">
    <th mat-header-cell *matHeaderCellDef> Peso </th>
    <td mat-
cell *matCellDef="let element"> {{element.weight}} </td>
  </ng-container>
  <!-- Coluna Simbolo -->
  <ng-container matColumnDef="symbol">
    <th mat-header-cell *matHeaderCellDef> Simbolo </th>
    <td mat-
cell *matCellDef="let element"> {{element.symbol}} </td>
  </ng-container>
  <tr mat-header-
row *matHeaderRowDef="displayedColumns"></tr>
  <tr mat-
row *matRowDef="let row; columns: displayedColumns;"></tr>
</table>
```

Neste passo, navegue até o arquivo *src >app>app.component.css* e adicione o seguinte código:

app.component.css

```
/*configurando nossa table*/
table{
    max-width: 75%;
    width: 100%;
    margin: 2em auto;

}
```

Na sequencia, navegue até o arquivo *app.component.ts* e adicione o seguinte código:

app.component.ts

```
import { Component, ViewChild } from '@angular/core';
// vamos importar nossa dependência da lib material
import { MatPaginator } from '@angular/material/paginator';
import { MatTableDataSource } from '@angular/material/table
';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

```
export class AppComponent {
  title = 'material-data-table';

  // vamos implementar o array para exibirmos as colunas
  displayedColumns: string[] = ['position', 'name', 'weight
', 'symbol'];

  dataSource = ELEMENT_DATA

  // nossa fonte de dados
  dataSource = new MatTableDataSource<ElemPeriodicos> (ELEM
ENT_DATA);
  // vamos criar nossa ViewChild
  @ViewChild(MatPaginator, {static: true}) paginator: MatPa
ginator;

  ngOnInit(){
    this.dataSource.paginator = this.paginator;
  }

}

// ESTA É A NOSSA FONTE DE DADOS DEVIDAMENTE CONSTRUIDA
// estes dados são estáticos
// este é o nosso model data
export interface ElemPeriodicos{
  position: number;
  name: string;
  weight: number;
  symbol: string;
}

// este é nosso conjunto de dados
const ELEMENT_DATA: ElemPeriodicos[] = [
 { position: 1, name: 'Hidrogênio', weight: 1.0079, symbol:
 'H' },
 { position: 2, name: 'Hélio', weight: 4.0026, symbol: 'He'
 },
 { position: 3, name: 'Lítio', weight: 6.941, symbol: 'Li'
},
 { position: 4, name: 'Berílio', weight: 9.0122, symbol: 'B
e' },
 { position: 5, name: 'Boro', weight: 10.811, symbol: 'B' }
,
 { position: 6, name: 'Carbono', weight: 12.0107, symbol: '
C' },
 { position: 7, name: 'Nitrogênio', weight: 14.0067, symbol
: 'N' },
```

```
 { position: 8, name: 'Oxigênio', weight: 15.9994, symbol:
'O' },
 { position: 9, name: 'Fluor', weight: 18.9984, symbol: 'F'
 },
 { position: 10, name: 'Neônio', weight: 20.1797, symbol: '
Ne' },
];
```

No código acima, foi criado o model para estabelecer o “formato” que o conjunto de ados deve obedecer. Logo depois, foram declarados os dados fictícios – criando o conjunto de dados *ElemPeriodicos* (fora da classe ***AppComponent***) atibuidos a const. Dentro da classe principal é possível acessar os dados a partir da variável *dataSource*. Os nomes das colunas foram descritos dentro do array *displayedColumns – está descrição considera as propriedades implementadas no arquivo app.component.html.*

Execute o comando para iniciar a aplicação:

`ng serve`

Observe o resultado abaixo. A saída, a partir do browser, deve ser semelhante a indicada pela imagem abaixo:

Posição	Nome	Massa atômica	Simbolo
1	Hidrogênio	1.0079	H
2	Hélio	4.0026	He
3	Lítio	6.941	Li
4	Berílio	9.0122	Be
5	Boro	10.811	B
6	Carbono	12.0107	C
7	Nitrogênio	14.0067	N
8	Oxigênio	15.9994	O
9	Fluor	18.9984	F
10	Neônio	20.1797	Ne

5 - Angular pagination (paginação)

O primeiro passo para implementar a paginação dentro do projeto é solicitar a importação do módulo de dependência *MatPaginatorModule* no arquivo *material.module.ts.*

material.module.ts

```
// este é nosso arquivo central de dependências
// da biblioteca Material

import { NgModule } from '@angular/core';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/pagin
ator';

// vamos construir nossos arrays de registro

@NgModule({
    imports: [
        MatTableModule,
        MatPaginatorModule
    ],
    exports: [
        MatTableModule,
        MatPaginatorModule
    ]
})

export class AngularMaterialModule{}
```

Depois disso, navegue até o arquivo *app.component.ts* e importe os módulos ***MatPaginator*** e ***MatTableDataSource*** e o service *ViewChild.* Observe o código abaixo; implemente-o como se segue:

app.component.ts

```
import { Component, ViewChild } from '@angular/core';
// vamos importar nossa dependência da lib material
import { MatPaginator } from '@angular/material/paginator';
import { MatTableDataSource } from '@angular/material/table
';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
```

```
  title = 'material-data-table';

  // vamos implementar o array para exibirmos as colunas
  displayedColumns: string[] = ['position', 'name', 'weight
', 'symbol'];

  dataSource = ELEMENT_DATA

  // nossa fonte de dados
  dataSource = new MatTableDataSource<AlgunsElementosTab> (
ELEMENT_DATA);
  // vamos criar nossa ViewChild
  @ViewChild(MatPaginator, {static: true}) 'paginator': Mat
Paginator;

  ngOnInit(){
    this.dataSource.paginator = this.paginator;
  }

}

// ESTA É A NOSSA FONTE DE DADOS DEVIDAMENTE CONSTRUIDA
// estes dados são estáticos
// este é o nosso model data
export interface AlgunsElementosTab{
  position: number;
  name: string;
  weight: number;
  symbol: string;
}

// este é nosso conjunto de dados
const ELEMENT_DATA: AlgunsElementosTab[] = [
 { position: 1, name: 'Hidrogênio', weight: 1.0079, symbol:
 'H' },
 { position: 2, name: 'Hélio', weight: 4.0026, symbol: 'He'
 },
 { position: 3, name: 'Lítio', weight: 6.941, symbol: 'Li'
},
 { position: 4, name: 'Berílio', weight: 9.0122, symbol: 'B
e' },
 { position: 5, name: 'Boro', weight: 10.811, symbol: 'B' }
,
 { position: 6, name: 'Carbono', weight: 12.0107, symbol: '
C' },
 { position: 7, name: 'Nitrogênio', weight: 14.0067, symbol
: 'N' },
 { position: 8, name: 'Oxigênio', weight: 15.9994, symbol:
'O' },
```

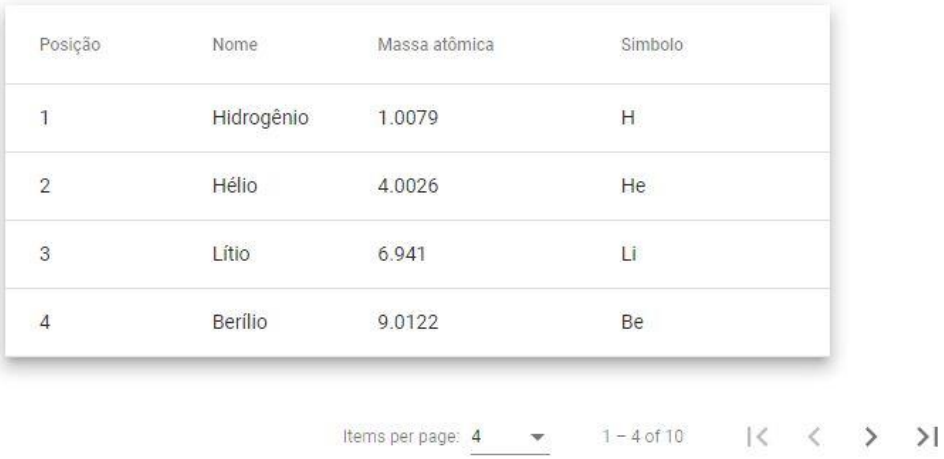
```
  { position: 9, name: 'Fluor', weight: 18.9984, symbol: 'F'
  },
  { position: 10, name: 'Neônio', weight: 20.1797, symbol: '
Ne' },
];
```

No próximo passo, navegue até o arquivo *app.component.html* e coloque essa diretiva de paginação logo após o fechamento da *tag </table>.* Observe o código abaixo e implemente-o como se segue:

app.component.html

```
 <!-- aqui, vamos implementar nosso paginador / paginator -
->
  <mat-
paginator [pageSizeOptions]="[5, 10, 15]" showFirstLastButt
ons>
  </mat-paginator>
```

Se, até aqui, tudo estiver correto, você obterá o resultado indicado abaixo.



6 - Classificação de dados

Para classificar os dados dentro da tabela, é necessário importar o servisse *MatSortModule* da dependência material dentro do arquivo *material.module.ts. Observe o código abaixo e implemente-o como se segue:*

material.module.ts

```
// este é nosso arquivo central de dependencias
// da biblioteca Material

import { NgModule } from '@angular/core';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/pagin
ator';
import {MatSortModule} from '@angular/material/sort';


// vamos construir nossos arrays de registro

@NgModule({
    imports: [
        MatTableModule,
        MatPaginatorModule,
        MatSortModule
    ],
    exports: [
        MatTableModule,
        MatPaginatorModule,
        MatSortModule
    ]
})

export class AngularMaterialModule{}
```

Em seguida, navegue até o arquivo *app.component.ts* e importe o módulo *MatSort*. Observe o código abaixo e implemente-o como se segue:

app.component.ts

```
import { Component, ViewChild } from '@angular/core';
// vamos importar nossa dependencia da lib material
import { MatPaginator } from '@angular/material/paginator';
import { MatTableDataSource } from '@angular/material/table
';
import {MatSort} from '@angular/material/sort';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
```

```
  title = 'material-data-table';

  // vamos implementar o array para exibirmos as colunas
  displayedColumns: string[] = ['position', 'name', 'weight
', 'symbol'];

  // nossa fonte de dados
  dataSource = new MatTableDataSource<ElemPeriodicos> (ELEM
ENT_DATA);
  // vamos criar nossa ViewChild
  @ViewChild(MatPaginator, {static: true}) 'paginator': Mat
Paginator;
  @ViewChild(MatSort, {static: true}) 'sort': MatSort;

  ngOnInit(){
    this.dataSource.paginator = this.paginator;
    this.dataSource.sort = this.sort;
  }

}

// ESTA É A NOSSA FONTE DE DADOS DEVIDAMENTE CONSTRUIDA
// estes dados são estaticos
// este é o nosso model data
export interface ElemPeriodicos{
  position: number;
  name: string;
  weight: number;
  symbol: string;
}

// este é nosso conjunto de dados
const ELEMENT_DATA: ElemPeriodicos[] = [
 { position: 1, name: 'Hidrogênio', weight: 1.0079, symbol:
 'H' },
 { position: 2, name: 'Hélio', weight: 4.0026, symbol: 'He'
 },
 { position: 3, name: 'Lítio', weight: 6.941, symbol: 'Li'
},
 { position: 4, name: 'Berílio', weight: 9.0122, symbol: 'B
e' },
 { position: 5, name: 'Boro', weight: 10.811, symbol: 'B' }
,
 { position: 6, name: 'Carbono', weight: 12.0107, symbol: '
C' },
 { position: 7, name: 'Nitrogênio', weight: 14.0067, symbol
: 'N' },
 { position: 8, name: 'Oxigênio', weight: 15.9994, symbol:
'O' },
```

```
 { position: 9, name: 'Fluor', weight: 18.9984, symbol: 'F'
 },
 { position: 10, name: 'Neônio', weight: 20.1797, symbol: '
Ne' },
];
```

Para classificar os dados em tabelas utilizando material, vinculamos o objeto de classificação ao array *dataSource*. Basta ir ao arquivo *app.component.html* e implementar código abaixo.

app.component.html

```
<div class="wrapper">
  <table mat-
table [dataSource]="dataSource" matSort class="mat-
elevation-z8">
    <ng-container matColumnDef="position">
      <th mat-header-cell *matHeaderCellDef mat-sort-
header> Posição </th>
      <td mat-
cell *matCellDef="let element"> {{element.position}} </td>
    </ng-container>

    <ng-container matColumnDef="name">
      <th mat-header-cell *matHeaderCellDef mat-sort-
header> Nome </th>
      <td mat-
cell *matCellDef="let element"> {{element.name}} </td>
    </ng-container>

    <ng-container matColumnDef="weight">
      <th mat-header-cell *matHeaderCellDef mat-sort-
header> Massa atômica </th>
      <td mat-
cell *matCellDef="let element"> {{element.weight}} </td>
    </ng-container>

    <ng-container matColumnDef="symbol">
      <th mat-header-cell *matHeaderCellDef mat-sort-
header> Simbolo </th>
      <td mat-
cell *matCellDef="let element"> {{element.symbol}} </td>
    </ng-container>

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

```
  <!-
- aqui, vamos implementar nosso paginador / paginator -->
  <mat-
paginator [pageSizeOptions]="[5, 10, 15]" showFirstLastButt
ons></mat-paginator>
</div>
```

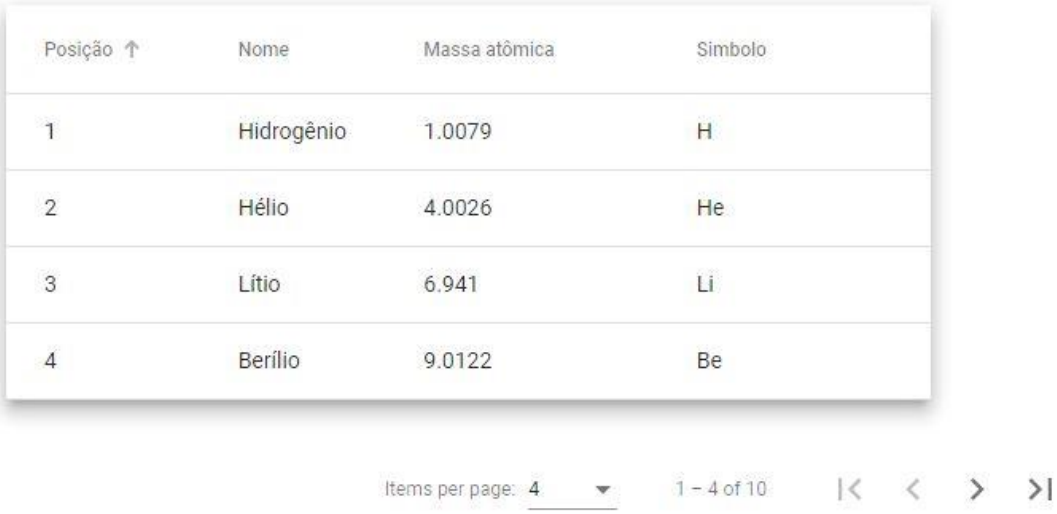
Por fim, navegue até o arquivo *app.component.css* e implemente o seguinte código.

app.component.css

```
/*configurando nossa table*/
table{
    max-width: 75%;
    width: 100%;
    margin: 2em auto;

}
th.mat-sort-header-sorted{
    color: black;
}
```

Observe o resultado final no browser. Deve ser semelhante ao indicado abaixo:



Referencia: *https://www.positronx.io/angular-material-8-data-table-pagination-sorting-tutorial/*