1. Angular-Material (librairie de composants)

Angular-Material est une librairie de composants graphiques qui sont

- destinés à être intégrés au framework **angular**
- basés sur le look "**material**" (projet transversal mis en avant par "google" entre autres)

Angular-Material offre des composants intéressants tels que les onglets , les boîtes de dialogue , …

Ces composants sont pour certains agrémentés d'un redimensionnement automatique
(comportement "responsive").

Angular-material est un *concurrent direct* de "*ngx-bootstrap*" et "*primeNg*" .

NB : La plupart des composants de "angular-material" ne gèrent que très peu l'aspect "disposition / placement" . On a souvent besoin d'une technologie complémentaire pour cela .

Bien qu'étant facultatif , le complément angular "**flex-layout**" est souvent utilisé en accompagnement de "angular-material" .

Remarque : Bien que pas très conseillée pour éviter des juxtapositions de looks différents et hétérogènes , une utilisation conjointe/complémentaire de "angular-material" et d'une autre librairie de composants (telle que ngx-bootstrap) est techniquement possible . Cette idée a d'ailleurs été mise en oeuvre au sein d'un projet (existant mais peu utilisé) baptisé "....." .

1.1. intégration de "angular-material" au sein d'un projet angular

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

et facultativement :

```
npm install -s @angular/flex-layout
```

Effet dans **package.json** (exemple):

```
...
  "dependencies": {
    "@angular/animations": "^8.2.14",
    "@angular/cdk": "^8.2.3",
     ... ,
   "@angular/flex-layout": "^8.0.0-beta.27",
    "@angular/material": "^8.2.3",
  }
...
```

Effet ou paramétrages dans **angular.json** :

```
....
"styles": [
            "./node_modules/@angular/material/prebuilt-themes/indigo-pink.css",
            "src/styles.scss"
          ],
....
```

Type d'importations techniques à ajouter directement ou indirectement dans **app.module.ts** :

```
import { ImportMaterialModule } from './common/imports/import-material.module';
import { FlexLayoutModule } from "@angular/flex-layout";
...
@NgModule({
...
imports: [
   BrowserModule,    AppRoutingModule,    BrowserAnimationsModule,
   FlexLayoutModule,    ImportMaterialModule ,
   FormsModule,    ReactiveFormsModule
  ],
...
```

src/app/common/imports/**import-material.module.ts**

```
import { NgModule } from '@angular/core';
import { MatTabsModule} from '@angular/material/tabs';
import { MatInputModule } from '@angular/material/input';
import { MatSelectModule } from '@angular/material/select';
import { MatIconModule } from '@angular/material/icon';
import { MatMenuModule } from '@angular/material/menu';
import { MatButtonModule } from '@angular/material/button';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatRadioModule } from '@angular/material/radio';
import { MatCardModule } from '@angular/material/card';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatFormFieldModule } from '@angular/material/form-field';
import {MatSidenavModule} from '@angular/material/sidenav';
import {MatListModule} from '@angular/material/list';
import {MatDatepickerModule} from '@angular/material/datepicker';
import {MatNativeDateModule} from '@angular/material/core';
import {MatAutocompleteModule} from '@angular/material/autocomplete';
import {MatSlideToggleModule} from '@angular/material/slide-toggle';
import {MatExpansionModule} from '@angular/material/expansion';
import {MatBadgeModule} from '@angular/material/badge';
import {MatProgressSpinnerModule} from '@angular/material/progress-spinner';
import {MatTooltipModule} from '@angular/material/tooltip';
import {MatDialogModule} from '@angular/material/dialog';
import {MatTableModule} from '@angular/material/table';
import {MatTreeModule} from '@angular/material/tree';
import {MatStepperModule} from '@angular/material/stepper';
import {MatSortModule} from '@angular/material/sort';
import {MatPaginatorModule} from '@angular/material/paginator';

@NgModule({
```

```
  imports: [
     MatTabsModule,       MatCardModule,MatExpansionModule,
     MatIconModule,       MatFormFieldModule,       MatInputModule,       MatSelectModule,
     MatButtonModule,       MatListModule,       MatCheckboxModule,MatSlideToggleModule,
     MatRadioModule,       MatMenuModule,       MatToolbarModule,       MatSidenavModule,
     MatDatepickerModule,MatNativeDateModule,       MatAutocompleteModule,
     MatBadgeModule,MatProgressSpinnerModule,       MatTooltipModule,       MatDialogModule,
     MatTableModule,       MatTreeModule,       MatStepperModule,       MatSortModule,       MatPaginatorModule
     ],
  exports:[
     MatTabsModule,       MatCardModule,MatExpansionModule,       MatIconModule,       MatFormFieldModule,
     MatInputModule,       MatSelectModule,       MatButtonModule,       MatListModule,
     MatCheckboxModule,MatSlideToggleModule,       MatRadioModule,       MatMenuModule,
     MatToolbarModule,       MatSidenavModule,       MatDatepickerModule,MatNativeDateModule,
     MatAutocompleteModule,       MatBadgeModule,MatProgressSpinnerModule,       MatTooltipModule,       MatDialogModule,
     MatTableModule,       MatTreeModule,       MatStepperModule,       MatSortModule,       MatPaginatorModule
  ]
})
export class ImportMaterialModule { }
```

2. Essentiel de "flex-layout" (en intégration angular)

```
npm install -s @angular/flex-layout
```

...html

```
empilement (si moins de 960px):
<div class="container" fxLayout.lt-md="column"
fxLayoutAlign="center" fxLayoutGap="10px" fxLayoutGap.lt-md="2px">
   <div class="a" fxFlex="25%">divA (25%)</div>
   <div class="b" fxFlex="50%">divB (50%)</div>
   <div class="c">divC</div>
</div>
```

empilement (si moins de 960px):
divA (25%) divB (50%) divC

empilement (si moins de 960px):
divA (25%)
divB (50%)
divC

breakpoints	size(px)
xs (extra small)	599 ou moins
sm (small or medium)	600 à 959
md (medium)	960 à 1279
lg (large)	1280 à 1919
xl (extra large)	1920 à 5000

breakpoints	size(px)
lt-sm (less than small)	moins que **600**
lt-md (less than md)	moins que **960**
lt-lg (less than large)	moins que **1280**
lt-xl (less than xl)	moins que **1920**

breakpoints	size(px)
gt-xs (greater than xs)	600 ou plus
gt-sm (greater than sm)	960 ou plus
gt-md (greater than md)	1280 ou plus
gt-lg (greater than large)	1920 ou plus

3. Quelques composants "angular-material"

Card (panneau d'encadrement) :


```
<mat-card class="my-card">
      <mat-card-header class="my-card-header">
        <mat-card-title>Basic</mat-card-title>
        <!-- <mat-card-subtitle>angular material</mat-card-subtitle> -->
      </mat-card-header>

      <mat-card-content class="my-card-content">
       ...
   </mat-card-content>
 </mat-card>
```

```
.basic { background: white;   }
.my-card { border: 1px; border-style:solid; border-color:blue ;   padding : 0 }

.my-card-header {
  background-color: rgb(23, 23, 112);   color : white;
  padding-left: 1em;   padding-top: 0.5em;
}

.my-card-content {    padding: 1em; }
```

Composants "onglets" (tab , tab-group , …)

```
<mat-tab-group>
            <mat-tab label="tva">
              <app-tva></app-tva>
            </mat-tab>
            <mat-tab label="titre onglet2">
                 ...contenu onglet 2...
            </mat-tab>
<mat-tab-group>
```

Composants élémentaires pour formulaires

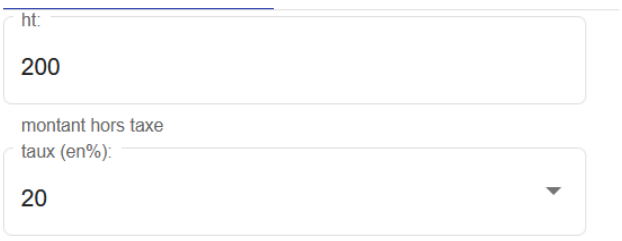
```
<div>
<form role="form"  class="form-container" >
<mat-form-field [appearance]="settingService.my_mat_appearence">
   <mat-label>ht:</mat-label>
   <input matInput placeholder="ht" name="ht" [(ngModel)]="ht" (input)="onCompute()"  />
   <!-- <mat-icon matSuffix>favorite</mat-icon> -->
   <mat-hint>montant hors taxe</mat-hint>
</mat-form-field>

<mat-form-field [appearance]="settingService.my_mat_appearence">
   <mat-label>taux (en%):</mat-label>
   <mat-select placeholder="taux" name="taux" [(ngModel)]="taux"
         (selectionChange)="onCompute()">
      <mat-option *ngFor="let t of listeTaux" [value]="t">{{t}}</mat-option>
   </mat-select>
</mat-form-field>
</form>
tva: <span>{{tva | currency:'EUR':'symbol':'1.0-2'}}</span> <br/>
ttc: <span>{{ttc | currency:'EUR':'symbol':'1.0-2'}}</span>
</div>
```

....css

```
.form-container {   display: flex;   flex-direction: column;  }
.form-container > * {   width: 30%;  }
```

Look avec le paramètre [*appearance*]=" 'standard' " :

Look avec le paramètre [*appearance*]=" 'outline' "

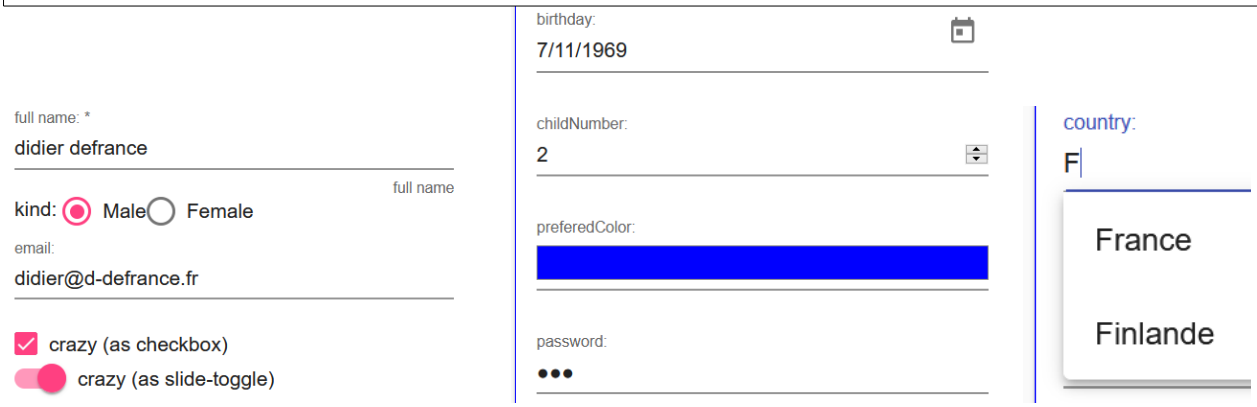


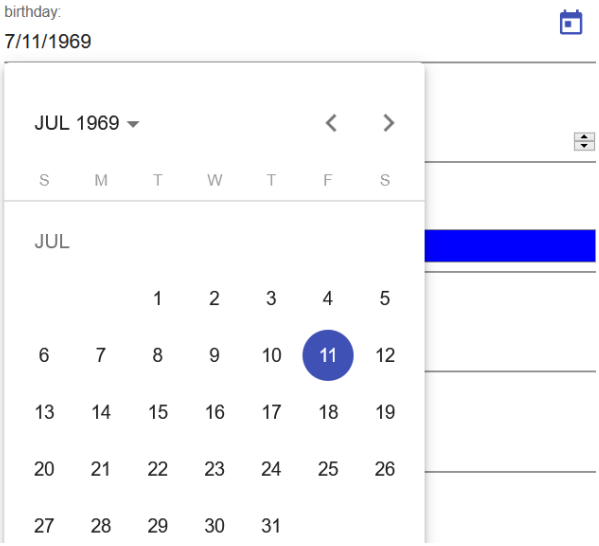
NB : les valeurs possibles de ***appearance*** sont "**standard**" , "**outline**" , "**fill**" et "**legacy**" .

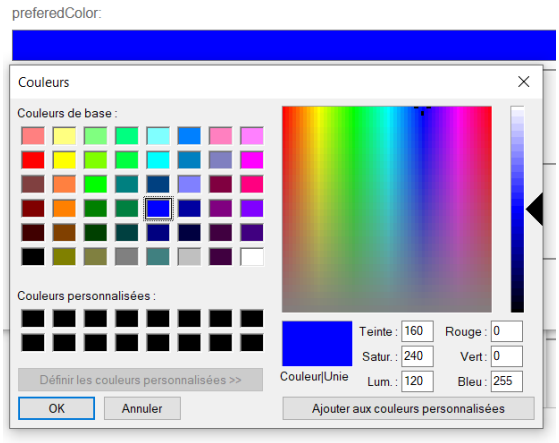
Autres composants de base (***exemples***):

```
<div>
  <form role="form"  class="form-container" >
  <mat-form-field [appearance]=" 'standard' ">
      <!-- [hideRequiredMarker]="false" [floatLabel]="'auto'" by default on mat-form-field -->
     <mat-label>full name:</mat-label>
     <input matInput placeholder="name" name="name"
          minLength="6" maxlength="20" required
         [(ngModel)]="person.name"  />
     <mat-hint align="end">full name</mat-hint>
  </mat-form-field>
  <div>
     <label>kind: </label>
     <mat-radio-group placeholder="kind" name="kind" [(ngModel)]="person.kind" >
       <mat-radio-button matInput *ngFor="let k of listeKind" [value]="k">{{k}}</mat-radio-button>
     </mat-radio-group>
      <!-- mat-radio-group cannot be put inside mat-form-field , ??? -->
  </div>

  <mat-form-field [appearance]=" 'standard' ">
        <mat-label>email:</mat-label>
        <input matInput name="email" type="email" placeholder="email" [(ngModel)]="person.email" />
  </mat-form-field>

  <mat-checkbox  name="crazy" [(ngModel)]="person.crazy" >crazy (as checkbox)</mat-checkbox>
  <mat-slide-toggle name="crazy" [(ngModel)]="person.crazy" >crazy (as slide-toggle)</mat-slide-toggle>
  <!-- mat-checkbox cannot be put inside mat-form-field , already has a label -->

  <mat-form-field [appearance]=" 'standard' ">
        <mat-label>birthday:</mat-label>
        <input matInput name="birthday" [matDatepicker]="myDatePicker"
              placeholder="birthday" [(ngModel)]="person.birthday" /> <!-- type="date" if no picker -->
        <mat-datepicker-toggle matSuffix [for]="myDatePicker"></mat-datepicker-toggle>
        <mat-datepicker #myDatePicker></mat-datepicker>
  </mat-form-field>

  <mat-form-field [appearance]=" 'standard' ">
        <mat-label>childNumber:</mat-label>
        <input matInput name="childNumber" type="number" placeholder="childNumber" [(ngModel)]="person.childNumber" />
  </mat-form-field>
  <mat-form-field [appearance]=" 'standard' ">
        <mat-label>preferedColor:</mat-label>
        <input matInput name="preferedColor" type="color" placeholder="preferedColor" [(ngModel)]="person.preferedColor" />
  </mat-form-field>
  <mat-form-field [appearance]=" 'standard' ">
        <mat-label>password:</mat-label>
        <input matInput name="password" type="password" placeholder="password" [(ngModel)]="person.password" />
  </mat-form-field>
  <mat-form-field [appearance]=" 'standard' ">
          <mat-label>country:</mat-label>
          <input matInput name="country" placeholder="country"
               (ngModelChange)="adjustfilteredCountries()"
               [(ngModel)]="person.country" [matAutocomplete]="autoCountry" />
          <!-- for reactiveForm , [formControl]="myControl" and no ngModel -->
          <mat-autocomplete #autoCountry="matAutocomplete">
                    <mat-option *ngFor="let c of filteredCountries | async" [value]="c">  {{c}}     </mat-option>
          </mat-autocomplete>
  </mat-form-field>
```

```
<mat-form-field [appearance]=" 'standard' ">
      <mat-label>comment:</mat-label>
      <textarea matInput name="comment" placeholder="comment" [(ngModel)]="person.comment"></textarea>
</mat-form-field>
</form>
person: <span>{{person | json}}</span> <br/>
</div>
```







....ts

```
....
import { Observable , of} from 'rxjs';
import { map , startWith} from 'rxjs/operators';

@Component({  ....})
export class VariousFormComponent implements OnInit {
  listeKind = [ "Male" , "Female"];
  //myControl = new FormControl();//if reactiveForm and autocomplete
  countries = [ "France" , "Finlande" , "Allemagne" , "Autriche" , "Italie" , "Espagne" , "..."];
  filteredCountries: Observable<string[]>;

  person : Person = new Person();
  constructor() { }
  ngOnInit(){  }

  adjustfilteredCountries() {
    this.filteredCountries = of(this.countries)
```

```
    .pipe(
      map(listCountries => this._filterCountriesIgnoreCase(listCountries,this.person.country))
    );
  }

  private _filterCountriesIgnoreCase(listCountries:string[],value: string): string[] {
    const filterValue = value.toLowerCase();
    return listCountries.filter(c => c.toLowerCase().includes(filterValue));
  }
}
```

Composants "boite de dialogue" (exemple , autres variantes possibles)

example-dialog.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { MatDialogRef, MAT_DIALOG_DATA } from '@angular/material/dialog';
import { MyDialogData } from './myDialogData';

@Component({
  selector: 'app-example-dialog',
  templateUrl: './example-dialog.component.html',
  styleUrls: ['./example-dialog.component.scss']
})
export class ExampleDialogComponent implements OnInit {

/*
 entryComponents: [ExampleDialogComponent],
 must be added in @NgModule()
*/

  constructor(
    public dialogRef: MatDialogRef<ExampleDialogComponent>,
    @Inject(MAT_DIALOG_DATA) public data: MyDialogData) {}

  onNoClick(): void {    this.dialogRef.close();  }

  ngOnInit() {  }
}
```

```
export class MyDialogData{
    name:string;
    animal:string;
}
```

example-dialog.component.html

```
<h1 mat-dialog-title>Hi {{data.name}}</h1>
<div mat-dialog-content>
  <p>What's your favorite animal?</p>
  <mat-form-field>
    <mat-label>Favorite Animal</mat-label>
    <input matInput  [(ngModel)]="data.animal">
```

```
  </mat-form-field>
</div>
<div mat-dialog-actions>
  <button mat-button (click)="onNoClick()">No Thanks</button>
  <button mat-button  [mat-dialog-close]="data.animal" cdkFocusInitial>Ok</button>
</div>
```

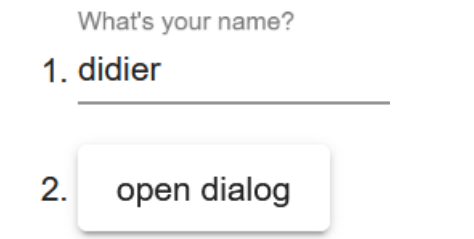
Exemple d'utilisation :

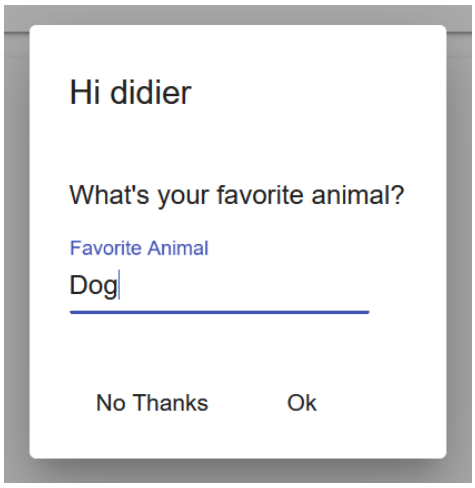
.....ts

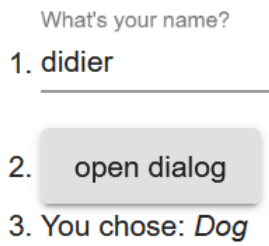
```
import { Component, OnInit } from '@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { ExampleDialogComponent } from './example-dialog/example-dialog.component';

@Component({  selector: 'app-divers',  templateUrl: './divers.component.html',
  styleUrls: ['./divers.component.scss']
})
export class DiversComponent implements OnInit {
  animal: string;
  name: string;

  constructor(public dialog: MatDialog) {}

  openDialog(): void {
    /*
 entryComponents: [ExampleDialogComponent],
 must be added in @NgModule()
*/
    const dialogRef = this.dialog.open(ExampleDialogComponent, {
      width: '250px',
      data: {name: this.name, animal: this.animal}
    });

    dialogRef.afterClosed().subscribe(result => {
      console.log('The dialog was closed');
      this.animal = result;
    });
  }
ngOnInit() {}
}
```

```
            <ol>
                <li>
                  <mat-form-field>
                    <mat-label>What's your name?</mat-label>
                    <input matInput [(ngModel)]="name">
```

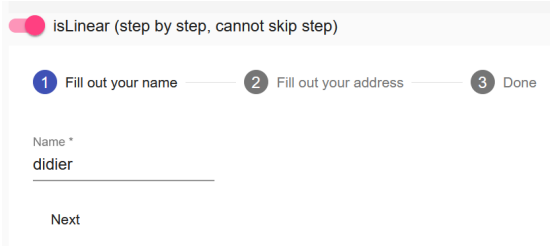
```
            </mat-form-field>
          </li>
          <li>
            <button mat-raised-button (click)="openDialog()">open dialog</button>
          </li>
          <li *ngIf="animal">
            You chose: <i>{{animal}}</i>
          </li>
        </ol>
```



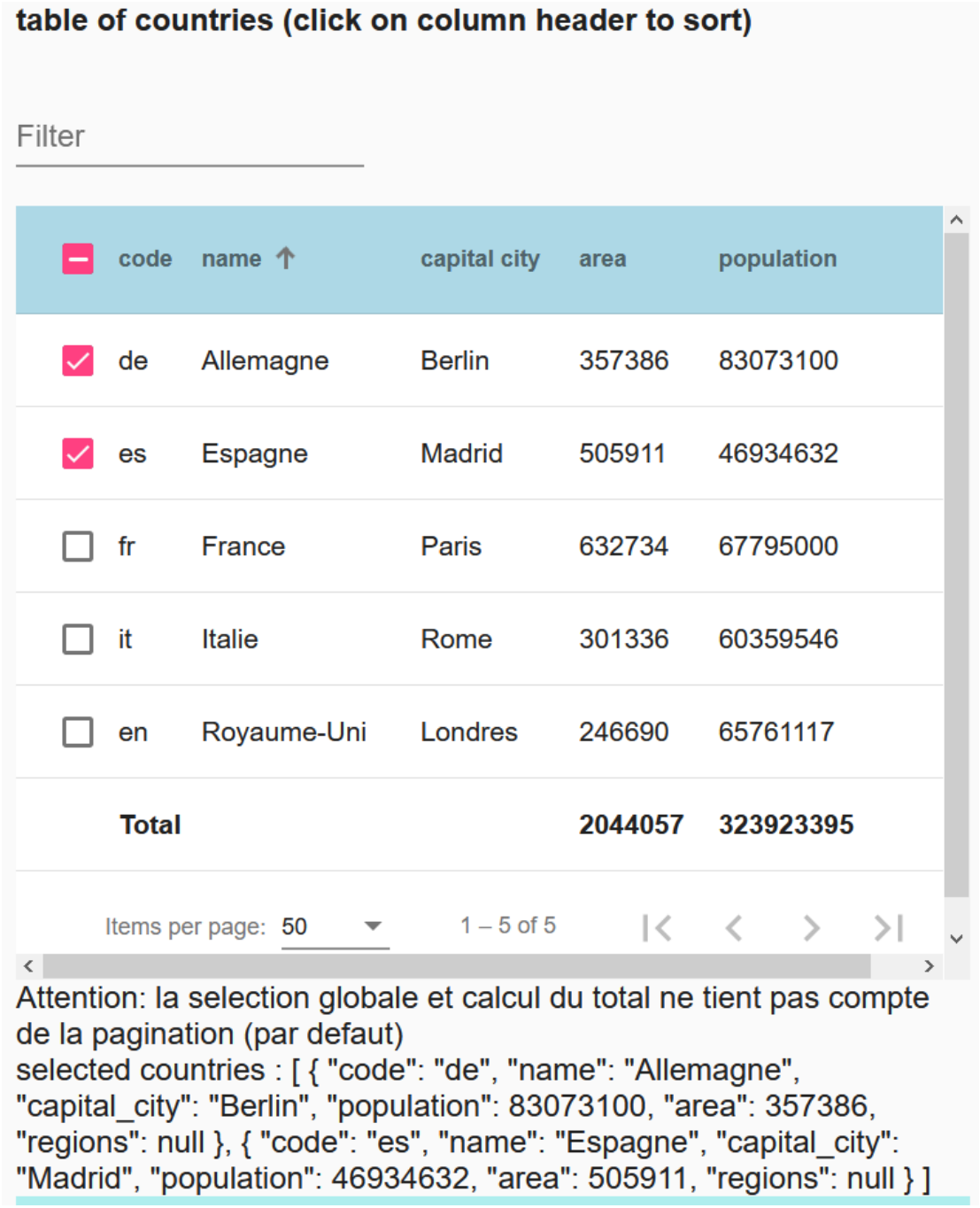




Composants "step" , ...



Composants "mat-table" avec dataSource



NB : bien que l'exemple suivant combine "selection, filtrage , tri , totaux, pagination, ..." , chacun de ces aspects est facultatif et l'on peut dans beaucoup de cas effectuer une mise en oeuvre bien plus simple .

Dans la logique complexe/élaborée de construction/fonctionnement des tableaux "***mat-table***" , **les lignes d'entête , de données et de "totaux/pied de tableau" seront automatiquement générés à partir des éléments complémentaires suivants** :

- **liste ordonnées des noms de colonnes à afficher** (ex : *displayedColumns coté .ts*)
- définition abstraite de chaque colonnes (<ng-container ***matColumnDef***="**colNamexy**">)
- **source de données** (intermédiaire "***dataSource***" entre données et vue , prenant en compte les **tris** et éventuels filtrages, ...)

with-table.component.ts

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { GeoService } from '../common/service/geo.service';
import { MatTableDataSource } from '@angular/material/table';
import { MatSort } from '@angular/material/sort';
import { Country } from '../common/data/country';
import { SelectionModel } from '@angular/cdk/collections';
import { MatPaginator } from '@angular/material/paginator';

@Component({
  selector: 'app-with-table',
  templateUrl: './with-table.component.html',
  styleUrls: ['./with-table.component.scss']
})
export class WithTableComponent implements OnInit {

  displayedColumns: string[] = ['select','code', 'name', 'capital_city', 'area', 'population'];
  countries : Country[] = [];
  dataSource  = new MatTableDataSource(this.countries);//this.countries; if no sort

  selection = new SelectionModel<Country>(true /*allowMultiSelect*/, [] /*initialSelection*/);

  constructor(private geoService: GeoService) { }

  //sort refer to table with matSort directive in .html
  //useful for get order choice (by name , by population, ...)
  @ViewChild(MatSort, {static: true}) sort: MatSort;

  @ViewChild(MatPaginator, {static: true}) paginator: MatPaginator;

  ngOnInit() {
    this.geoService.getCountries().subscribe(
      (countries)=>{ /*this.dataSource=countries; if no sort*/
        this.countries=countries;
        this.dataSource= new MatTableDataSource(countries);
        this.dataSource.sort=this.sort; //by name or by ...
        this.dataSource.paginator = this.paginator;
        this.dataSource.filterPredicate =
          (data: Country, filter: string) => !filter || (data.name.toLowerCase()).includes(filter);
      }
    );
  }

  applyFilter(event: Event) {
    const filterValue = (event.target as HTMLInputElement).value;
```

```
    this.dataSource.filter = filterValue.toLowerCase();
  }

  /** Gets the total population and total area of all countries. */
  getTotalPopulation() {
    //this.countries.map(..) or this.dataSource.filteredData.map(...)
    return this.dataSource.filteredData.map(c => c.population)
                                       .reduce((acc, value) => acc + value, 0);
  }

  getTotalArea() {
    return this.dataSource.filteredData.map(c => c.area)
                                       .reduce((acc, value) => acc + value, 0);
  }

  /** Whether the number of selected elements matches the total number of rows. */
  isAllSelected() {
    const numSelected = this.selection.selected.length;
    const numRows = this.dataSource.data.length;
    return numSelected === numRows;
  }

  /** Selects all rows if they are not all selected; otherwise clear selection. */
  masterToggle() {
    this.isAllSelected() ?
        this.selection.clear() :
        this.dataSource.data.forEach(row => this.selection.select(row));
  }

  /** The label for the checkbox on the passed row */
  checkboxLabel(row?: Country): string {
    if (!row) {
      return `${this.isAllSelected() ? 'select' : 'deselect'} all`;
    }
    return `${this.selection.isSelected(row) ? 'deselect' : 'select'} row ${row.code}`;
  }
}
```

with-table.component.css

```
.selected {
  background-color: red;
}

.mat-row.highlighted {
  background: lightblue;
}

table {
  width: 100%;
  overflow-x: auto;
  overflow-y: hidden;
  min-width: 500px;
}
```

```
 .mat-header-cell, .mat-sort-header {
   background-color: lightblue;
   font-weight: bold;
 }

.my-table-container-for-scroll{
  height: 400px;
  overflow: auto;
}

tr.mat-footer-row {
  font-weight: bold;
}
```

with-table.component.html

```
<h4>table of countries (click on column header to sort)</h4>

<mat-form-field>
   <mat-label>Filter</mat-label>
   <input matInput (keyup)="applyFilter($event)"
        placeholder="Ex. i">
</mat-form-field>
<div class="my-table-container-for-scroll">
<table mat-table [dataSource]="dataSource" matSort>

 <!--- Note that these columns can be defined in any order.
       The actual rendered columns are set as a property on the row definition" -->

       <!-- code Column -->
       <ng-container matColumnDef="code">
         <th mat-header-cell *matHeaderCellDef> code </th>
         <td mat-cell *matCellDef="let c"> {{c.code}} </td>
         <td mat-footer-cell *matFooterCellDef>Total</td>
       </ng-container>

       <!-- Name Column -->
       <ng-container matColumnDef="name">
         <th mat-header-cell *matHeaderCellDef mat-sort-header> name </th>
         <td mat-cell *matCellDef="let c"> {{c.name}} </td>
         <td mat-footer-cell *matFooterCellDef></td>
       </ng-container>

       <!-- Capital-city Column -->
       <ng-container matColumnDef="capital_city">
         <th mat-header-cell *matHeaderCellDef> capital city </th>
         <td mat-cell *matCellDef="let c"> {{c.capital_city}} </td>
         <td mat-footer-cell *matFooterCellDef></td>
       </ng-container>
```

```
    <!-- Population Column -->
    <ng-container matColumnDef="population">
      <!-- mat-sort-header only ok if matSort in table -->
      <th mat-header-cell  *matHeaderCellDef mat-sort-header> population </th>
      <td mat-cell *matCellDef="let c"> {{c.population}} </td>
      <td mat-footer-cell *matFooterCellDef> {{getTotalPopulation()}}</td>
    </ng-container>

    <!-- Area Column -->
    <ng-container matColumnDef="area">
          <th mat-header-cell *matHeaderCellDef mat-sort-header> area </th>
          <td mat-cell *matCellDef="let c"> {{c.area}} </td>
          <td mat-footer-cell *matFooterCellDef> {{getTotalArea()}}</td>
    </ng-container>

    <!-- Checkbox selection Column -->
    <ng-container matColumnDef="select">
    <th mat-header-cell *matHeaderCellDef>
      <mat-checkbox (change)="$event ? masterToggle() : null"
                    [checked]="selection.hasValue() && isAllSelected()"
                    [indeterminate]="selection.hasValue() && !isAllSelected()"
                    [aria-label]="checkboxLabel()">
      </mat-checkbox>
    </th>
    <td mat-cell *matCellDef="let row">
      <mat-checkbox (click)="$event.stopPropagation()"
                    (change)="$event ? selection.toggle(row) : null"
                    [checked]="selection.isSelected(row)"
                    [aria-label]="checkboxLabel(row)">
      </mat-checkbox>
    </td>
    <td mat-footer-cell *matFooterCellDef></td>
   </ng-container>

    <tr mat-header-row *matHeaderRowDef="displayedColumns; sticky: true"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"
      > <!-- (click)="selection.toggle(row)" --> </tr>
    <!-- if mat-footer-row , mat-footer-cell must be defined for each displayedColumns -->
    <tr mat-footer-row *matFooterRowDef="displayedColumns;"></tr>
</table>
<mat-paginator [pageSizeOptions]="[50, 20, 12 , 6 , 3]" showFirstLastButtons></mat-paginator>
</div>
Attention: la selection globale et calcul du total ne tient pas compte de la pagination (par defaut)<br/>
selected countries : {{selection._selected | json}}
```