

Formation Angular

TP N°9: Reactive Form

SOMMAIRE

1.	Pré-requis :	3
2.	Objectifs	3
3.	Développement de l'application	3
a.	Création de l'application.....	3
b.	Installation de bootstrap	3
c.	Installation de jquery	3
d.	Intégrer les feuilles de style et les scripts au niveau du fichier angular.json	3
e.	Démarrer ensuite l'application :.....	3
f.	Créer le module users :	3
g.	Ajouter les modules FormsModule et ReactiveFormsModule au module stock	3
h.	Créer les modèles Article et Propriete au niveau du module « Sock ».....	4
i.	Créer le composant exemple1 au niveau du module « Stock »	4
j.	Créer le composant exemple2 au niveau du module « Stock »	6
k.	Créer le composant exemple3 au niveau du module « Stock »	9
l.	Créer le composant exemple4 au niveau du module « Stock »	13
m.	Créer le Guard « Check »	15
n.	Configurer les routes au niveau du fichier app-routing.module.ts.....	16
o.	Configurer les routes au niveau du fichier stock-routing.module.ts	16
p.	Modifier la feuille de style styles.css.....	17
q.	Modifier le template du composant principal	17
4.	Tests	18
a.	Tester le formulaire Exemple1	18
b.	Tester le formulaire Exemple2	19
c.	Tester le formulaire Exemple3	20
d.	Tester le formulaire Exemple4	21
5.	Le Packaging	22

1. Pré-requis :

- Réaliser les TPs 1-8 en premier.

2. Objectifs

- ✓ Utiliser les classes FormBuilder, FormControl, FormGroup et FormArray.
- ✓ Créer des formulaires dynamiques.
- ✓ Créer un module et utiliser le mode lazy.
- ✓ Créer un Guard pour autoriser ou interdire le chargement d'un module.

3. Développement de l'application

a. Création de l'application

- Créer l'application tp8bis avec la commande : `ng new tpReactiveFormSuite`

b. Installation de bootstrap

- Lancer la commande : `npm install bootstrap`

c. Installation de jquery

- Lancer la commande : `npm install jquery`

d. Intégrer les feuilles de style et les scripts au niveau du fichier angular.json

Au niveau du fichier angular.json, modifier les valeurs des clés « styles » et « scripts » comme suit :

```
"styles": [
  "src/styles.css",
  "./node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "./node_modules/jquery/dist/jquery.min.js",
  "./node_modules/bootstrap/dist/js/bootstrap.bundle.min.js",
  "./node_modules/bootstrap/dist/js/bootstrap.min.js"
]
```

e. Démarrer ensuite l'application :

- `cd tpReactiveFormSuite`
- `ng serve.`

f. Créer le module users :

- `ng g m modules/stock --routing`

g. Ajouter les modules FormsModule et ReactiveFormsModule au module stock

- Ajouter les modules FormsModule et ReactiveFormsModule au niveau de la classe StockModule :

```
import { NgModule } from '@angular/core';
```

```
import { CommonModule } from '@angular/common';

import { StockRoutingModule } from '../stock-routing.module';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({
  declarations: [
  ],
  imports: [
    CommonModule,
    StockRoutingModule,
    FormsModule,
    ReactiveFormsModule
  ]
})
export class StockModule { }
```

h. Créer les modèles Article et Propriete au niveau du module « Sock »

- Créer le modèle Propriete : **ng g class modules/stock/propriete**

```
export class Propriete {

  constructor(
    public detail:string,
    public performance:string) {

  }

}
```

- Créer le modèle Article : **ng g class modules/stock/article**

```
import { Propriete } from "../propriete";

export class Article {

  constructor(
    public id:number,
    public description:string,
    public prix:number,
    public quantite:number,
    public propriete:Propriete) {}

}
```

i. Créer le composant exemple1 au niveau du module « Stock »

- Lancer la commande **ng g c modules/stock/exemple1**

- Modifier la classe comme suit :

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormControl, Validators } from '@angular/forms';

@Component({
  selector: 'app-exemple1',
  templateUrl: './exemple1.component.html',
  styleUrls: ['./exemple1.component.css']
})
export class Exemple1Component implements OnInit {

  adresse = new FormControl('', [Validators.required]);
  comment = this.fb.control('', [Validators.required]);

  constructor(private fb: FormBuilder) { }

  ngOnInit(): void {

  }

  update() {
    this.adresse.setValue('TEST123');
  }

  updateControles() {
    this.adresse.setValidators([Validators.required, Validators.maxLength(10)])
    this.adresse.updateValueAndValidity();
  }
}
```

- Modifier le Template comme suit :

```
<div class="container">
  <label>enter email :
    <input type="text" [formControl]='adresse'>

    <div *ngIf="adresse.touched && adresse.errors != null &&
adresse.errors['required']" class="alert alert-danger">
      <p>mail obligatoire</p>
    </div>

    <div *ngIf="adresse.touched && adresse.errors != null &&
adresse.errors['maxlength']" class="alert alert-danger">
      <p>mail doit être composé de 10 maximum</p>
    </div>
  </div>
```

```

    </div>

    <p>Donnée saisie : {{adresse.value}}</p>
  </label>

  <br>

  <label>enter comment :
    <input type="text" [formControl]='comment'>

    <div *ngIf="comment.touched && comment.errors != null &&
comment.errors['required']" class="alert alert-danger">
      <p>commentaire obligatoire</p>

    </div>
    <p>Donnée saisie : {{comment.value}}</p>
  </label>

  <br>
  <button (click)="update()">Modifier</button> <button
(click)="updateContrôles()">Modifier les contrôles</button>
</div>

```

j. Créer le composant exemple2 au niveau du module « Stock »

- Lancer la commande **ng g c modules/stock/exemple2**
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';
import { FormBuilder, Validators } from '@angular/forms';
import { Article } from '../article';

@Component({
  selector: 'app-exemple2',
  templateUrl: './exemple2.component.html',
  styleUrls: ['./exemple2.component.css']
})
export class Exemple2Component implements OnInit {

  article!: Article;
  form = this.fb.group({
    id: ['', [Validators.required, Validators.minLength(2)]],
    description: ['', [Validators.required]],
    prix: ['', [Validators.required, Validators.min(0), Validators.max(10000)]],
    quantite: ['', [Validators.required, Validators.minLength(1)]],

```

```

    })

    get articles() {
        return this.form.controls;
    }
    constructor(private fb: FormBuilder) { }

    ngOnInit(): void {
    }

    save() {
        let json=JSON.stringify(this.form.value);
        this.article=JSON.parse(json);
    }
}

```

- Modifier le Template comme suit :

```

<form [formGroup]="form" (ngSubmit)="save()">
    <label>Entrer l'Id :
        <input type="text" formControlName="id">

        <div *ngIf="articles['id'].touched && articles['id'].invalid"
class="alert alert-danger">
            <div *ngIf="articles['id'].errors != null &&
articles['id'].errors['required']">
                <p>Id obligatoire</p>
            </div>

            <div *ngIf="articles['id'].errors != null &&
articles['id'].errors['minlength']">
                <p>Id doit être composé de 02 caractères</p>
            </div>
        </div>

    </label>
    <br>

    <label>Entrer la description :
        <input type="text" formControlName="description">

        <div *ngIf="articles['description'].touched &&
articles['description'].invalid" class="alert alert-danger">
            <div *ngIf="articles['description'].errors != null &&
articles['description'].errors['required']">
                <p>description obligatoire</p>
            </div>
        </div>
    </div>

```

```

</label>
<br>
<label>Entrer le prix :
    <input type="text" formControlName="prix">

    <div *ngIf="articles['prix'].touched && articles['prix'].invalid"
class="alert alert-danger">

        <div *ngIf="articles['prix'].errors != null &&
articles['prix'].errors['required']">
            <p>Age obligatoire</p>
        </div>

        <div *ngIf="articles['prix'].errors != null &&
articles['prix'].errors['min']">
            <p>L'age minimum est 0</p>
        </div>
        <div *ngIf="articles['prix'].errors != null &&
articles['prix'].errors['max']">
            <p>L'age maximum est 10000</p>
        </div>
    </div>

</label>

<br>
<label>Entrer la quantité :
    <input type="text" formControlName="quantite">

    <div *ngIf="articles['quantite'].touched && articles['quantite'].invalid"
class="alert alert-danger">

        <div *ngIf="articles['quantite'].errors != null &&
articles['quantite'].errors['required']">
            <p>Quantité obligatoire</p>
        </div>

        <div *ngIf="articles['quantite'].errors != null &&
articles['quantite'].errors['min']">
            <p>La quantité minimum est 1</p>
        </div>

    </div>

</label>
<button type="submit">save</button>
<p >
    {{form.value | json}}

```



```

<p>

<p *ngIf="article && article.id">
  Id : {{article.id}}
</p>

<p *ngIf="article && article.description">
  Id : {{article.description}}
</p>

<p *ngIf="article && article.prix">
  Id : {{article.prix}}
</p>

<p *ngIf="article && article.quantite">
  Id : {{article.quantite}}
</p>

</form>

```

k. Créer le composant exemple3 au niveau du module « Stock »

- Lancer la commande `ng g c modules/stock/exemple3`
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Article } from '../article';

@Component({
  selector: 'app-exemple3',
  templateUrl: './exemple3.component.html',
  styleUrls: ['./exemple3.component.css']
})
export class Exemple3Component implements OnInit {

  article!: Article;
  form = this.fb.group({
    id: ['', [Validators.required, Validators.minLength(2)]],
    description: ['', [Validators.required]],
    prix: ['', [Validators.required, Validators.min(0), Validators.max(10000)]],
    quantite: ['', [Validators.required, Validators.minLength(1)]],
    proprietes : this.fb.group({
      detail: ['', [Validators.required, Validators.minLength(2)]],
      performance: ['', [Validators.required]]
    }))
}

```

```

get articles() {
    return this.form.controls;
}

get proprietes() {
    return (this.form.get('proprietes') as FormGroup).controls;
}
constructor(private fb: FormBuilder) { }

ngOnInit(): void {
}

save() {
    let json=JSON.stringify(this.form.value);
    this.article=JSON.parse(json);
}
}

```

- Modifier le Template comme suit :

```

<form [formGroup]="form" (ngSubmit)="save()">
    <label>Entrer l'Id :
        <input type="text" formControlName="id">

        <div *ngIf="articles['id'].touched && articles['id'].invalid"
class="alert alert-danger">
            <div *ngIf="articles['id'].errors != null &&
articles['id'].errors['required']">
                <p>Id obligatoire</p>
            </div>

            <div *ngIf="articles['id'].errors != null &&
articles['id'].errors['minlength']">
                <p>Id doit être composé de 02 caractères</p>
            </div>
        </div>

    </label>
    <br>

    <label>Entrer la description :
        <input type="text" formControlName="description">

        <div *ngIf="articles['description'].touched &&
articles['description'].invalid" class="alert alert-danger">
            <div *ngIf="articles['description'].errors != null &&
articles['description'].errors['required']">
                <p>description obligatoire</p>
            </div>
        </div>
    </label>

```

```

    </div>

    </label>
    <br>
    <label>Entrer le prix :
      <input type="text" formControlName="prix">

      <div *ngIf="articles['prix'].touched && articles['prix'].invalid"
class="alert alert-danger">

        <div *ngIf="articles['prix'].errors != null &&
articles['prix'].errors['required']">
          <p>Age obligatoire</p>
        </div>

        <div *ngIf="articles['prix'].errors != null &&
articles['prix'].errors['min']">
          <p>L'age minimum est 0</p>
        </div>
        <div *ngIf="articles['prix'].errors != null &&
articles['prix'].errors['max']">
          <p>L'age maximum est 10000</p>
        </div>
      </div>

    </label>

    <br>
    <label>Entrer la quantité :
      <input type="text" formControlName="quantite">

      <div *ngIf="articles['quantite'].touched && articles['quantite'].invalid"
class="alert alert-danger">

        <div *ngIf="articles['quantite'].errors != null &&
articles['quantite'].errors['required']">
          <p>Quantité obligatoire</p>
        </div>

        <div *ngIf="articles['quantite'].errors != null &&
articles['quantite'].errors['min']">
          <p>La quantité minimum est 1</p>
        </div>

      </div>

    </label>

```

```

<div formGroupName="proprietes">
  <label>Entrer la propriété :
    <input type="text" formControlName="detail">
    <input type="text" formControlName="performance">
    <div *ngIf="proprietes['detail'].touched &&
proprietes['detail'].invalid" class="alert alert-danger">
      <div *ngIf="proprietes['detail'].errors != null &&
proprietes['detail'].errors['required']">
        <p>Detail obligatoire</p>
      </div>
    </div>
  </div>

  <div *ngIf="proprietes['performance'].touched &&
proprietes['performance'].invalid" class="alert alert-danger">
    <div *ngIf="proprietes['performance'].errors != null &&
proprietes['performance'].errors['required']">
      <p>Performance obligatoire</p>
    </div>
  </div>

</label>
</div>

<button type="submit">save</button>
<p >
  {{form.value | json}}
<p>

<p *ngIf="article && article.id">
  Id : {{article.id}}
</p>

<p *ngIf="article && article.description">
  Description : {{article.description}}
</p>

<p *ngIf="article && article.prix">
  Prix : {{article.prix}}
</p>

<p *ngIf="article && article.quantite">
  Quantité : {{article.quantite}}
</p>

<p *ngIf="article && article.propriete && article.propriete.detail">
  Détail : {{article.propriete.detail}}

```

```

    </p>

    <p *ngIf="article && article.propriete && article.propriete.performance">
        Détail : {{article.propriete.performance}}
    </p>

</form>

```

I. Créer le composant exemple4 au niveau du module « Stock »

- Lancer la commande `ng g c modules/stock/exemple4`
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';
import { FormArray, FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Article } from '../article';

@Component({
  selector: 'app-exemple4',
  templateUrl: './exemple4.component.html',
  styleUrls: ['./exemple4.component.css']
})
export class Exemple4Component implements OnInit {

  mesarticles: Article[] = [];
  isSaved = false
  form = this.fb.group({
    articles: this.fb.array([this.fb.group({
      id: ['', [Validators.required]],
      description: ['', [Validators.required]],
      prix: ['', [Validators.required]],
      quantite: ['', [Validators.required]]
    })])
  });

  constructor(private fb: FormBuilder) { }

  get articles() {
    return this.form.get("articles") as FormArray;
  }

  ngOnInit(): void {
  }

  addArticle() {
    const f = this.fb.group({
      id: ['', [Validators.required]],

```

```

        description: ['', [Validators.required]],
        prix: ['', [Validators.required]],
        quantite: ['', [Validators.required]],
    });
    this.articles.push(f);
}

removeArticle(i: number) {
    this.articles.removeAt(i)
}

save() {
    this.isSaved = true;
    console.log('a', this.form.value.articles);
    const json = JSON.stringify(this.form.value.articles);
    console.log('json', json);
    this.mesarticles = JSON.parse(json);
}

addValidatorRule(i: number) {
    (this.articles.at(i) as
FormGroup).get('prix')?.setValidators([Validators.required,Validators.max(15000)])
;
    (this.articles.at(i) as FormGroup).get('prix')?.updateValueAndValidity();
}

prix(i:number) {
    return (this.articles.at(i) as FormGroup).controls['prix'];
}

malist(i:number) {
    return (this.articles.at(i) as FormGroup).controls;
}
}

```

- Modifier le Template comme suit :

```

<form [formGroup]="form">
    <div formArrayName="articles">
        <div *ngFor="let aa of articles.controls; let i=index"
[formGroupName]="i">

            <label>
                ID : <input type="text" formControlName="id">
                Description : <input type="text" formControlName="description">
                Prix : <input type="text" formControlName="prix">

```

```

        Quantité : <input type="text" formControlName="quantite">

        <div *ngIf="prix(i).touched && prix(i).invalid" class="alert
alert-danger">
            <div *ngIf="prix(i) != null && prix(i).errors != null &&
prix(i).errors?.['required']">
                <p>Prix obligatoire</p>
            </div>
        </div>

        <button type="submit"
(click)='removeArticle(i) '>Supprimer</button>
        <button type="submit" (click)='addValidatorRule(i) '>Add
Rule</button>
        </label>

    </div>

    <br>
    <button (click)='addArticle()'>Ajouter</button>

    <button (click)="save()">save</button>
    {{form.value | json}}
    <div *ngIf="!!mesarticles">Liste des articles :
    <div *ngFor="let article of mesarticles">

        <p>{{article.id}} - {{article.description}} - {{article.prix}} -
{{article.quantite}}</p>
    </div>
    </div>
    </div>

    <div>
        <li class="nav-item"><a class="nav-link" routerLink='/test'>Visiter ce
lien</a></li>
    </div>
</form>

```

m. Créer le Guard « Check »

- Lancer la commande **ng g guard guards/check**
- Modifier la classe CheckGuard comme suit :

```

import { Injectable } from '@angular/core';
import { CanLoad, Route, UrlSegment, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';
import { StockService } from '../services/stock.service';

```

```

@Inject({
  providedIn: 'root'
})
export class CheckGuard implements CanLoad {
  canLoad(
    route: Route,
    segments: UrlSegment[]): Observable<boolean | UrlTree> | Promise<boolean |
UrlTree> | boolean | UrlTree {
    //vous implémenter votre logique ici pour autoriser au
    //interdire le chargement du module
    return true;
  }
}

```

n. Configurer les routes au niveau du fichier app-routing.module.ts

Modifier le fichier app-routing.module.ts comme suit :

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CheckGuard } from './guards/check.guard';

const routes: Routes = [
{
  path: 'stock',
  loadChildren: () => import('./modules/stock/stock-routing.module')
    .then(mod => mod.StockRoutingModule), canLoad: [CheckGuard]}
];

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

```

o. Configurer les routes au niveau du fichier stock-routing.module.ts

Modifier le fichier stock-routing.module.ts comme suit :

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { Exemple1Component } from './exemple1/exemple1.component';
import { Exemple2Component } from './exemple2/exemple2.component';
import { Exemple3Component } from './exemple3/exemple3.component';
import { Exemple4Component } from './exemple4/exemple4.component';

```



```
const routes: Routes = [
  {path: 'exemple1', component: Exemple1Component},
  {path: 'exemple2', component: Exemple2Component},
  {path: 'exemple3', component: Exemple3Component},
  {path: 'exemple4', component: Exemple4Component}
];

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

p. Modifier la feuille de style styles.css

Modifier le fichier styles.css comme suit :

```
.container {
  text-align: center;
}
label {
  text-align: left;
}
h1 {
  text-align: center;
}

.alert{
  color: white;
  margin-bottom: 15px;
  text-align: left;
}
.alert-danger {
  background-color: red;
}
```

q. Modifier le template du composant principal

Modifier le fichier app.component.html comme suit :

```
<router-outlet></router-outlet>
```

4. Tests

a. Tester le formulaire Exemple1

- Lancer le lien <http://localhost:4200/stock/exemple1>. Le formulaire suivant est affiché :

enter email :

Donnée saisie :

enter comment :

Donnée saisie :

- N'entrer rien dans les deux champs. Vérifier que les messages suivants sont affichés :

enter email :

mail obligatoire

Donnée saisie :

enter comment :

commentaire obligatoire

Donnée saisie :

- Au niveau du champ « Adresse mail », entrer une valeur dépassant 10 chiffres au niveau de sa longueur et vérifier que l'application l'accepte :

enter email :

Donnée saisie : |||||

enter comment :

Donnée saisie :

- Maintenant cliquer sur « Modifier les contrôles » et vérifier que le message suivant est affiché :

enter email :

mail doit être composé de 10 maximum

Donnée saisie :

enter comment :

Donnée saisie :

-

b. Tester le formulaire Exemple2

- Lancer le lien <http://localhost:4200/stock/exemple2>. Le formulaire suivant est affiché :

Entrer l'id :

Entrer la description :

Entrer le prix :

Entrer la quantité :

{ "id": "", "description": "", "prix": "", "quantite": "" }

- Entrer les valeurs suivantes et vérifier les règles de gestion sur les champs :

Entrer l'id :

Id doit être composé de 02 caractères

Entrer la description :

description obligatoire

Entrer le prix :

L'age maximum est 10000

Entrer la quantité :

Quantité obligatoire

```
{ "id": "1", "description": "", "prix": "20000", "quantite": "" }
```

c. Tester le formulaire Exemple3

- Lancer le lien <http://localhost:4200/stock/exemple3>. Le formulaire suivant est affiché :

Entrer l'id :

Entrer la description :

Entrer le prix :

Entrer la quantité :

Entrer la propriété :

```
{ "id": "", "description": "", "prix": "", "quantite": "", "proprietes": { "detail": "", "performance": "" } }
```

- Entrer les valeurs suivantes et vérifier que les règles de contrôles sur les champs ont été bien validées :

Entrer l'id :

Id obligatoire

Entrer la description :

description obligatoire

Entrer le prix :

Entrer la quantité :

Quantité obligatoire

Entrer la propriété :

Detail obligatoire

Performance obligatoire

save

```
{ "id": "", "description": "", "prix": "12", "quantite": "", "proprietes": { "detail": "", "performance": "" } }
```

d. Tester le formulaire Exemple4

- Lancer le lien <http://localhost:4200/stock/exemple4>. Le formulaire suivant est affiché :

D : Description : Prix : Quantité :

{ "articles": [{ "id": "", "description": "", "prix": "", "quantite": "" }] }

Liste des articles :

[Visiter ce lien](#)

- Entrer des valeurs et ensuite cliquer sur le bouton Ajouter pour ajouter une ligne :

ID :	<input type="text" value="1"/>	Description :	<input type="text" value="PC HP"/>	Prix :	<input type="text" value="20000"/>	Quantité :	<input type="text" value="20"/>	<input type="button" value="Supprimer"/>	<input type="button" value="Add Rule"/>
ID :	<input type="text" value="2"/>	Description :	<input type="text" value="pc dell"/>	Prix :	<input type="text" value="30000"/>	Quantité :	<input type="text" value="30"/>	<input type="button" value="Supprimer"/>	<input type="button" value="Add Rule"/>

{ "articles": [{ "id": "1", "description": "PC HP", "prix": "20000", "quantite": "20" }, { "id": "2", "description": "pc dell", "prix": "30000", "quantite": "30" }] }

Liste des articles :

[Visiter ce lien](#)

- Cliquer ensuite sur le bouton « Add Rule » pour chaque ligne et vérifier si la règle concernant le prix a été bien validée :

ID :	1	Description :	PC HP	Prix :	20000	Quantité :	20
------	---	---------------	-------	--------	-------	------------	----

Prix maximum est 15000 !!

Supprimer Add Rule

ID :	2	Description :	pc dell	Prix :	30000	Quantité :	30
------	---	---------------	---------	--------	-------	------------	----

Prix maximum est 15000 !!

Supprimer Add Rule

Ajouter save { "articles": [{ "id": "1", "description": "PC HP", "prix": "20000", "quantite": "20" }, { "id": "2", "description": "pc dell", "prix": "30000", "quantite": "30" }] }

Liste des articles :

•

[Visiter ce lien](#)

5. Le Packaging

- Lancer la commande suivante : **ng build --base-href /tp** et vérifier que le dossier suivant a été bien généré :

TI31320200A (C:) > workspace_angular_forGL > tpReactiveFormSuite > dist > tp

Nom	Modifié le	Type	Taille
3rdpartylicenses.txt	09/05/2022 22:32	Document texte	14 Ko
106.9815a6e5fde3927c.js	09/05/2022 22:32	Fichier source Jav...	18 Ko
favicon.ico	08/05/2022 09:45	Icône	1 Ko
index.html	09/05/2022 22:32	Microsoft Edge H...	3 Ko
main.934468b78eff92ad.js	09/05/2022 22:32	Fichier source Jav...	243 Ko
polyfills.07f28aca626ecbe0.js	09/05/2022 22:32	Fichier source Jav...	33 Ko
runtime.157d8c4d65f53eb8.js	09/05/2022 22:32	Fichier source Jav...	3 Ko
scripts.f4aaaa02019f154c.js	09/05/2022 22:32	Fichier source Jav...	222 Ko
styles.140d96edcacd178c.css	09/05/2022 22:32	Fichier source CSS	159 Ko

Correspond au module Stock

- Remarquer que les composants correspondants au module « Stock » ont été mis dans un nouveau fichier js et non pas au niveau du fichier main.js. Ce fichier sera chargé une fois un composant parmi les 04 composants du module Stock a été consulté par l'utilisateur. C'est le mode Lazy implémenté par Angular.

Vous venez de développer un formulaire Reactive !

Vous pouvez passer au TP n° 10.

Fin du TP 9.