



TP6 : LES FORMULAIRES

Matière : Atelier Framework Coté Client

Enseignante : Sonia Guerbouj

Classes : DSI2, MDW2

Durée : 3h

Objectif

Le but de ce TP est de maîtriser la création et manipulation des formulaires dans une application Angular.

1- Les formulaires

Les formulaires sont la base de nombreuses applications de gestion. Ils permettent aux utilisateurs de se connecter, de mettre à jour un profil, de saisir des informations sensibles et d'effectuer de nombreuses autres tâches de saisie de données.

Angular propose deux approches différentes pour gérer les entrées des utilisateurs via des formulaires : réactives et basées sur des modèles.

- Les formulaires basés sur des modèles (ngModels, ou template-driven forms) : Très facile à mettre en place, ils sont bien adaptés à des formulaires simples, n'ayant pas besoin de trop de validation ou de gestion d'erreur.
- Les formulaires réactifs (reactive forms) : Plus complexes à mettre en place, ils sont adaptés pour les saisies qui comportent de nombreux champs, des validations spécifiques ou réutilisables.

Dans notre projet, nous allons utiliser le premier type : Template-driven form qui utilisent la directive ngModel pour lier une variable à un champ du formulaire et ainsi maintenir les objets à jour.

2- Transmission de paramètres avec ActivatedRoute

Le module ActivatedRoute est un service qui fournit l'accès aux informations sur un itinéraire associé à un composant chargé dans une sortie (outlet). Il est utilisé pour permettre la transmission de paramètres entre composants à partir de l'URL. Il sert à parcourir l'arborescence RouterState et extraire les informations sur les nœuds parcourus à l'aide de snapshot et le tableau params.

3- Travail à faire

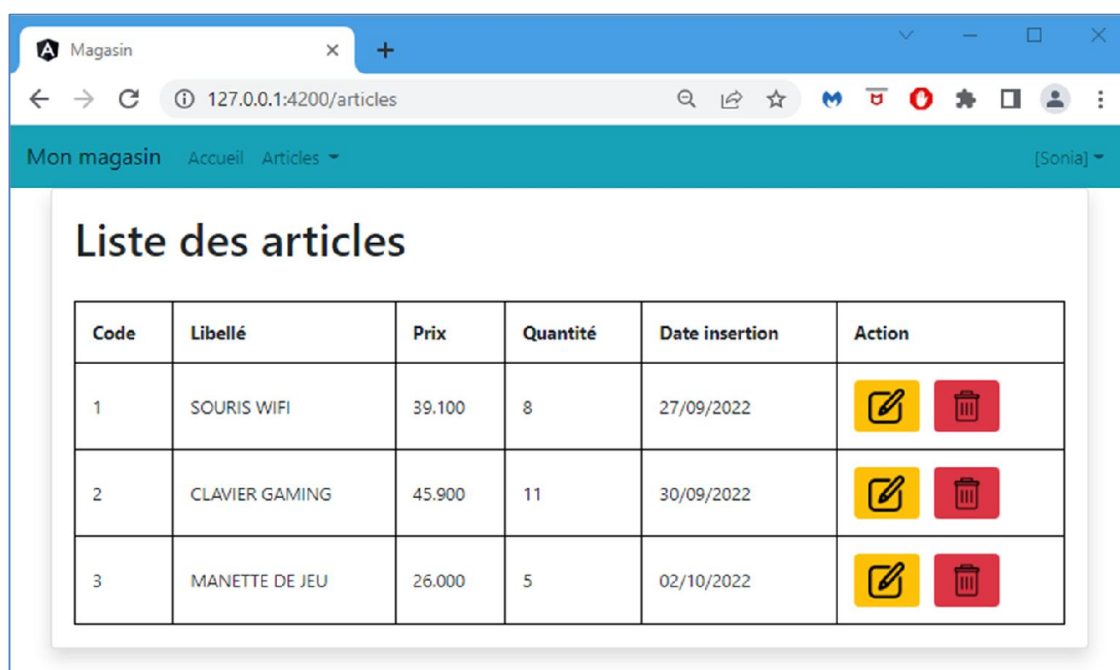
Ouvrez le projet « **magasin** » sous le dossier « DSI21 » à l'aide de VS Code.

Exercice 1

Dans cet exercice, vous allez modifier le projet « **magasin** » afin de permettre la modification d'un article.

- 1) Dans le fichier « **articles.component.ts** », ajoutez ce code dans la colonne « Action », à côté du bouton supprimer :

```
<a class="btn btn-warning" [routerLink]="['/updateArticle', article.codea]">
</a>
```



- 2) Sous le répertoire « assets », créez un dossier « img », dans lequel copiez les 2 images fournies par l'enseignante.
- 3) Dans le fichier « **articles.component.ts** », modifiez le lien de suppression par l'image correspondante.
- 4) Ouvrez un terminal et créez un composant « **update-article** » sous le dossier « **magasin** » : `ng g c update-article`
- 5) Dans le fichier « **app-routing.module.ts** », ajoutez la route vers ce composant :

```
...
import { UpdateArticleComponent } from '../update-article/update-
article.component';

const routes: Routes = [
  {path: "articles", component : ArticlesComponent},
  {path: "add-article", component : AddArticleComponent},
  {path: "updateArticle/:id", component: UpdateArticleComponent},
  {path: "", redirectTo: "articles", pathMatch: "full"}
];
...
```

- 6) Lancez le serveur et vérifiez que le lien du bouton « modifier » fonctionne.
- 7) Dans le fichier « **article.service.ts** », ajoutez les 2 fonctions suivantes à la classe « ArticleService » :

```
consulterArticle(id:number): Article{
  this.article = this.articles.find(art => art.codea == id!);
  return this.article;
}

modifierArticle(art: Article) {
  this.supprimerArticle(art);
  this.ajouterArticle(art);
}
```

- 8) Dans le fichier « **update-article.component.ts** », importez le modèle « Article », le service et le module « ActivatedRoute ». Ensuite, ajoutez un attribut intitulé « articleCourant » et la fonction « modifArticle. Enfin, modifiez le constructeur et la fonction « ngOnInit » comme suit :

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Article } from '../model/article.model';
import { ArticleService } from '../services/article.service';

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

export class UpdateArticleComponent implements OnInit {

  articleCourant = new Article();

  constructor(private activatedRoute: ActivatedRoute, private
  articleService: ArticleService) { }

  modifArticle() {
    this.articleService.modifierArticle(this.articleCourant);
    console.log("Article modifié avec succes : "+
this.articleCourant.libelle+": "+this.articleCourant.prix+"DT, qté="+
this.articleCourant.qte+"", ajoutée le "+this.articleCourant.dateAjout);
  }

  ngOnInit(): void {
    this.articleCourant = this.articleService.consulterArticle
(this.activatedRoute.snapshot.params['id']);
    console.log("Code article à modifier "+this.articleCourant.codea);
  }
}
```

- 9) Remplacez le contenu du fichier « **update-article.component.ts** » par ce code :

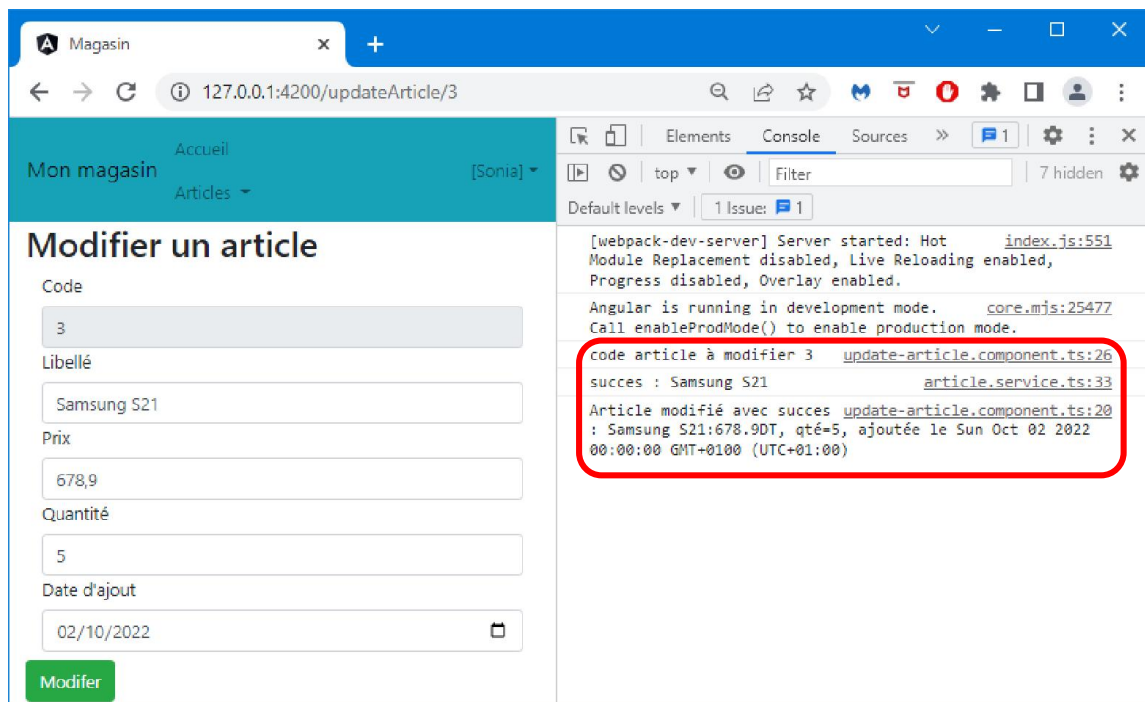
```
<div class="container">
  <div>
    <h2>Modifier un article</h2>
  </div>
  <form >
    <div class="col-sm-2 col-md-2 col-lg-2">
      <label >Code</label>
```

```

        <input readonly type="text" [(ngModel)]="articleCourant.codea"
name="codea" class="form-control">
    </div>
    <div class="col-sm-4 col-md-4 col-lg-4" >
        <label >Libellé</label>
        <input type="text" [(ngModel)]=" articleCourant.libelle"
name="libelle" class="form-control">
    </div>
    <div class="col-sm-2 col-md-2 col-lg-2">
        <label >Prix</label>
        <input type="number" [(ngModel)]=" articleCourant.prix"
name="prix" class="form-control">
    </div>
    <div class="col-sm-2 col-md-2 col-lg-2">
        <label >Quantité</label>
        <input type="number" [(ngModel)]=" articleCourant.qte" name="qte"
class="form-control">
    </div>
    <div class="col-sm-4 col-md-4 col-lg-4">
        <label >Date d'ajout</label>
        <input type="date" [(ngModel)]="articleCourant.dateAjout | date:
'yyyy-MM-dd'" (ngModelChange)=" articleCourant.dateAjout= $event"
name="dateAjout" class="form-control">
    </div>
    <div class="mt-2">
        <button type="submit" (click)="modifArticle()" class="btn btn-
success">Modifier</button>
    </div>
</form>
</div>

```

10) Testez le bouton du formulaire de modification et vérifiez dans la console du navigateur que le code fonctionne.



The screenshot shows a web application interface for modifying an article. The form includes fields for Code (3), Libellé (Samsung S21), Prix (678,9), Quantité (5), and Date d'ajout (02/10/2022). A green 'Modifier' button is at the bottom. The browser's developer console is open, showing the following messages:

```

[webpack-dev-server] Server started: Hot    index.js:551
Module Replacement disabled, Live Reloading enabled,
Progress disabled, Overlay enabled.

Angular is running in development mode.    core.mjs:25477
Call enableProdMode() to enable production mode.

code article à modifier 3    update-article.component.ts:26
succes : Samsung S21    article.service.ts:33
Article modifié avec succes    update-article.component.ts:20
: Samsung S21:678.9DT, qté=5, ajoutée le Sun Oct 02 2022
00:00:00 GMT+0100 (UTC+01:00)

```

The last three lines of the console output are highlighted with a red rectangle.

Exercice 2

Dans le projet **magasin** de l'exercice 1, on modifie les fichiers du composant update-article pour permettre le retour vers la liste des articles après modification.

- 1) Dans le fichier « **update-article.component.ts** », ajoutez l'import du module « Router » et son appel en tant que paramètre dans le constructeur de la classe « UpdateArticleComponent ». Enfin, modifiez la méthode « **modifArticle** » ainsi :

```
...
import { ActivatedRoute, Router } from '@angular/router';
...
constructor(private activatedRoute: ActivatedRoute, private router
:Router, private articleService: ArticleService) { }
modifArticle() {
  this.articleService.modifierArticle(this.articleCourant);
  console.log("Article modifié avec succes : "+this.articleCourant.libelle
+": "+this.articleCourant.prix+"DT, qté="+this.articleCourant.qte+",
ajoutée le "+this.articleCourant.dateAjout);
  this.router.navigate(['articles']);
}
...
```

- 2) Modifiez le fichier « **add-article.component.ts** » pour permettre le retour vers la liste des articles après l'ajout aussi.

Exercice 3

Dans le projet **magasin** de l'exercice 2, on enrichie le modèle de données.

- 1) Sous le dossier « **model** », créez un fichier « **categorie.model.ts** » à l'aide de VSCode. Définissez dans ce fichier une classe « **Categorie** » qui comporte les attributs codec, nomCat et description.
- 2) Dans le fichier « **article.model.ts** », importez la classe « **Categorie** » à partir du fichier « **categorie.model.ts** » :

```
import { Categorie } from "../categorie.model";
```

 Puis, ajoutez un attribut « **categ** » de type objet de la classe « **Categorie** ».
- 3) Dans le fichier « **article.service.ts** », effectuez les opérations suivantes :
 - a) Importez la classe « **Categorie** » à partir du fichier « **categorie.model.ts** »
 - b) Ajoutez dans la classe « **ArticleService** », un attribut « **categories** » qui est un tableau d'objets **Categorie**, et remplacez le code dans le constructeur de la classe comme suit :

```
constructor() {
  this.categories = [
    {codec : 1, nomCat : "Périphérique PC", description:"Périphériques
d'entrée/sortie pour les PCs"},
    {codec : 2, nomCat : "PC", description:"Toutes les marques de
Laptops"},
    {codec : 3, nomCat : "Smartphone", description:"Toutes les marques
de téléphones"}
  ];
}
```

```

    this.articles = [
      {codea : 1, libelle : "Souris Wifi", prix : 39.100, qte : 8,
dateAjout : new Date("09/27/2022"), categ:{codec : 1, nomCat :
"Périphérique PC", description:"Périphériques d'entrée/sortie pour les
PCs"}},
      {codea : 2, libelle : "Clavier Gaming", prix : 45.900, qte : 11,
dateAjout : new Date("09/30/2022"), categ:{codec : 1, nomCat :
"Périphérique PC", description:"Périphériques d'entrée/sortie pour les
PCs"}},
      {codea : 3, libelle : "Samsung S21", prix : 678.9, qte : 5,
dateAjout : new Date("10/02/2022"), categ:{codec : 3, nomCat :
"Smartphone", description:"Toutes les marques de téléphones"}}
    ];
  }

```

- c) Ajoutez dans la classe, les méthodes « `listCategories` » et « `consulterCategorie` » définies comme suit :

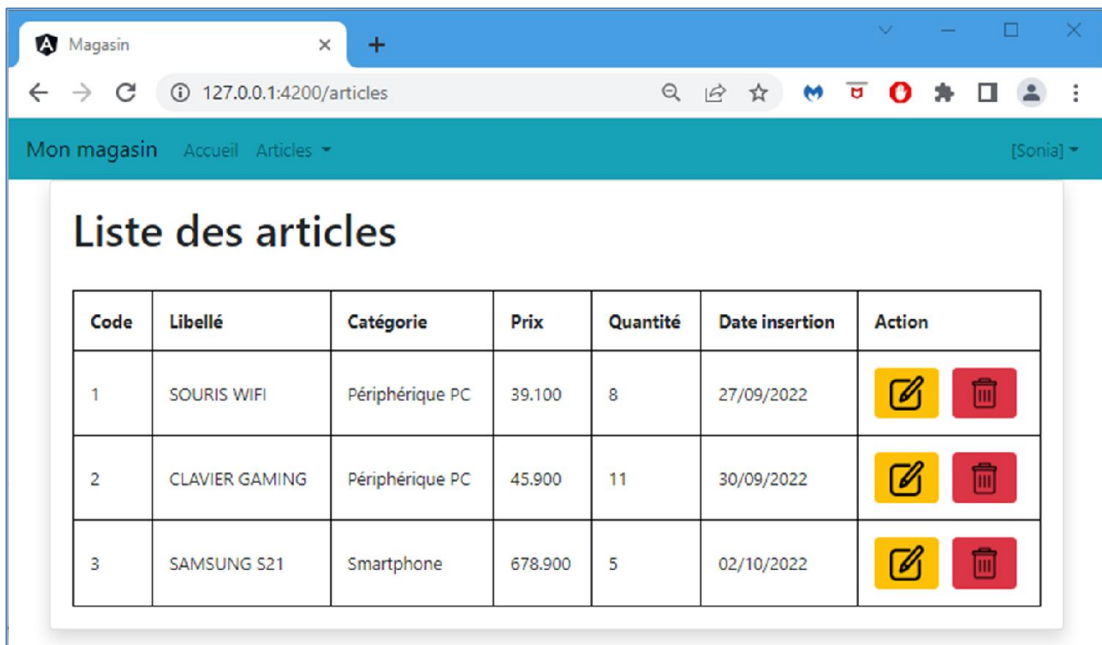
```







listeCategories():Categorie[] {
  return this.categories;
}
consulterCategorie(id:number): Categorie{
  return this.categories.find(cat => cat.codec == id)!;
}

```

- 4) Dans le fichier « `articles.component.html` », ajoutez une colonne « Catégorie » dans le tableau et affichez la catégorie de chaque article à l'aide du code :

```
<td>{{a.categ.nomCat}}</td>
```



Code	Libellé	Catégorie	Prix	Quantité	Date insertion	Action
1	SOURIS WIFI	Périphérique PC	39.100	8	27/09/2022	 
2	CLAVIER GAMING	Périphérique PC	45.900	11	30/09/2022	 
3	SAMSUNG S21	Smartphone	678.900	5	02/10/2022	 

Il faut à présent, ajouter le champ de la catégorie à tous les formulaires (ajout et modification) des articles.

Exercice 4

Dans le projet **magasin** de l'exercice 2, on applique les modifications du modèle de données aux formulaires, d'ajout et modification.

1) On commence les modifications au niveau du composant « **add-article** » :

a) Dans le fichier « **add-article.component.ts** », importez la classe « **Categorie** » du modèle et modifiez la classe « **AddArticleComponent** » comme suit :

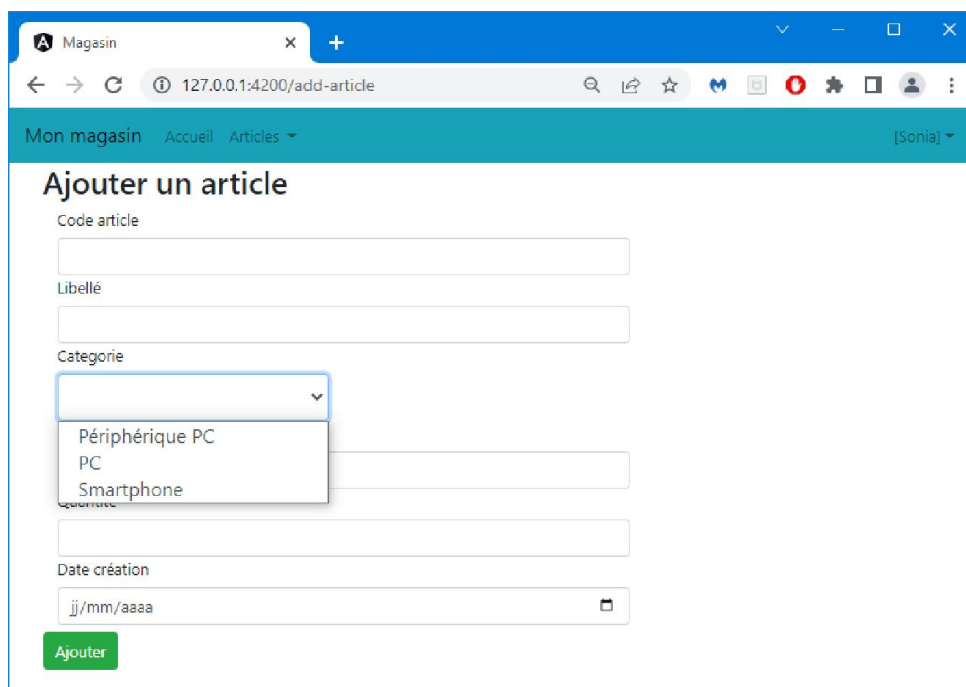
```
import { Categorie } from '../model/categorie.model';
...
export class AddArticleComponent implements OnInit {
  newArticle = new Article();
  categories! : Categorie[];
  newCodec! : number;
  newCateg! : Categorie;

  constructor(private router :Router,
               private articleService: ArticleService ) { }

  addArticle() {
    this.newCateg = this.articleService.consulterCategorie(this.newCodec);
    this.newArticle.categ = this.newCateg;
    this.articleService.ajouterArticle(this.newArticle);
    console.log(this.newArticle);
    this.router.navigate(['articles']);
  }

  ngOnInit(): void {
    this.categories = this.articleService.listeCategories();
  }
}
```

b) Ajoutez une liste déroulante contenant toutes les catégories dans le formulaire du fichier « **add-article.component.html** » comme suit :



```
<div class="col-sm-4 col-md-4 col-lg-4">
  <label>Categorie</label>
  <select class="form-control form-control-lg" id="codec" name="codec"
  [(ngModel)]="newCodec">
    <option *ngFor="let cat of categories" [value]="cat.codec">
    {{cat.nomCat}}</option>
  </select>
</div>
```

2) Maintenant, on effectue les modifications nécessaires au niveau du composant « update-article »

- a) Dans le fichier « **update-article.component.ts** », importez la classe « **Categorie** » du modèle et modifiez la classe « **UpdateArticleComponent** » comme suit :

```
import { Categorie } from '../model/categorie.model';
...
export class UpdateArticleComponent implements OnInit {

  articleCourant = new Article();
  categories! : Categorie[];
  codecModifie! : any;

  constructor(private activatedRoute: ActivatedRoute,
    private router :Router, private articleService: ArticleService) { }

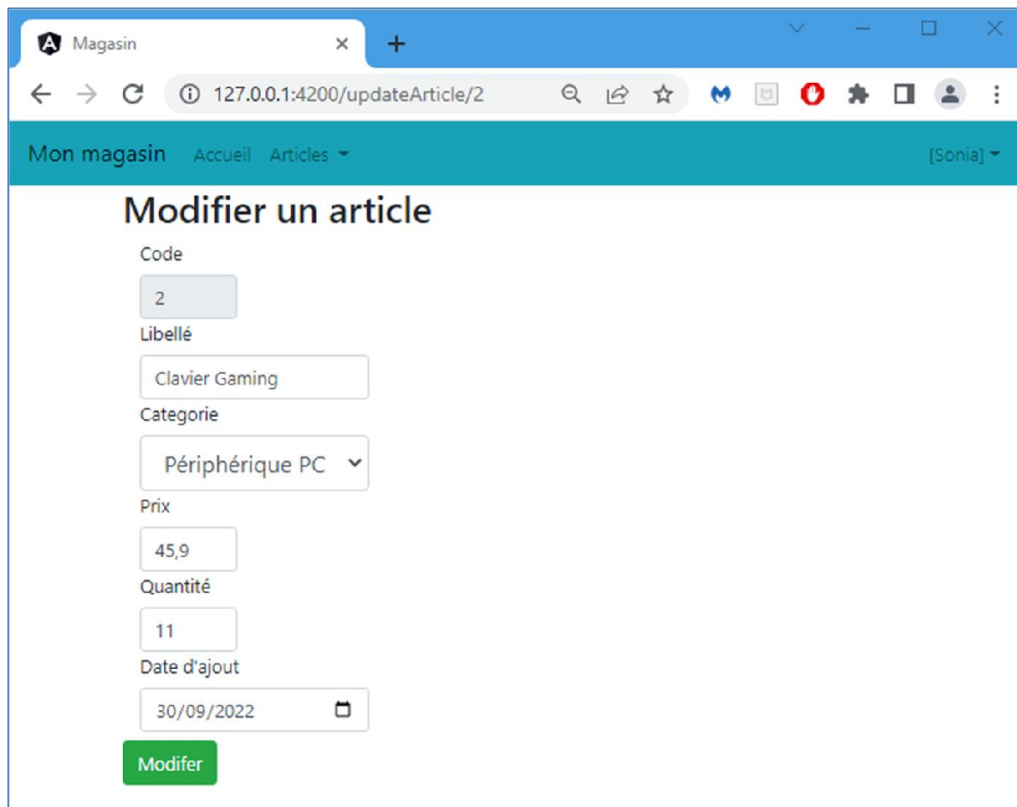
  modifArticle(){
    this.articleCourant.categ = this.articleService.consulterCategorie
                                (this.codecModifie);

    this.articleService.modifierArticle(this.articleCourant);
    console.log("Article modifié avec succes : "+
this.articleCourant.libelle+": "+this.articleCourant.prix+"DT, qté="+
this.articleCourant.qte+", ajoutée le "+this.articleCourant.dateAjout);
    this.router.navigate(['articles']);
  }

  ngOnInit(): void {
    this.categories = this.articleService.listeCategories();
    this.articleCourant = this.articleService.consulterArticle
                                (this.activatedRoute.snapshot.params['id']);
    console.log("code article à modifier "+this.articleCourant.codea);
    this.codecModifie = this.articleCourant.categ.codec;
  }
}
```

- b) Ajoutez la liste déroulante contenant toutes les catégories dans le formulaire du fichier « **update-article.component.html** » comme suit

```
<div class="col-sm-4 col-md-4 col-lg-4">
  <label>Categorie</label>
  <select class="form-control form-control-lg" id="codec" name="codec"
  [(ngModel)]="codecModifie">
    <option *ngFor="let category of categories" [value]=" category.codec">
    {{ category.nomCat}}</option>
  </select>
</div>
```

Magasin

127.0.0.1:4200/updateArticle/2

Mon magasin Accueil Articles [Sonia]

Modifier un article

Code
2

Libellé
Clavier Gaming

Catégorie
Périphérique PC

Prix
45,9

Quantité
11

Date d'ajout
30/09/2022

Modifier

3) Enfin, testez les fonctionnalités ajout et modification après intégration de la catégorie d'article.