# Cours Angular

2 | TD MyBooks

FormReactive – Validators – CRUD

## 1] Compléments pour le template de l'application

A la création de l'application, la mention de l'option pour le routage a permis de créer entre autres le fichier `app-routing.module.ts`. C'est dans ce fichier que nous indiquerons dans un tableau les routes possibles.

- ***Construction du routage***

**T1.1** Changez le contenu du fichier dédié au routage `app-routing.module.ts` en y ajoutant/modifiant les lignes suivantes :

```
…
import { HomeComponent } from "./components/home/home.component";
import { SearchComponent } from "./components/search/search.component";

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'search', component: SearchComponent },
];
…
```

Notez l'**absence de "/" au niveau du path** alors qu'il sera présent dans la directive **RouterLink** pour la page principale (voir plus bas).

- ***Construction du template de la page principale***

La page principale (home) aura deux composants : celui pour l'ajout et celui pour la liste.

**T1.2** Mettez le code suivant pour le template du `home` :

```
<div class="container">
    <div class="row">
        <div class="col-md-4">
            <app-book></app-book>
        </div>
        <div class="col-md-7">
            <app-books></app-books>
        </div>
    </div>
</div>
```

**T1.3** Et celui-ci pour le template du composant principal `app.component.html` :

```
<div class="jumbotron">
  <h2 style="text-align:center;">
    Application MyBooks
  </h2>
  <button type="button" routerLink="" class="btn btn-link">
    Home
   </button>
  <button type="button" routerLink="search" class="btn btn-link">
```

```
      Search
    </button>
</div>

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

On voit bien dans ces lignes la liaison entre le **composant** et le **chemin** servant de descripteur dans le menu. Notez aussi la directive : **RouterOutlet** (https://angular.io/api/router/RouterOutlet).

Tout fonctionne à priori pour obtenir ceci (vérifiez les deux liens) :


## 2] Gestion de l'ajout d'un livre

Pour implémenter des opérations de type CRUD (Create, Read, Update et Delete), nous aurons bien sûr besoin d'un formulaire pour le premier besoin et, avec les autres nous aurons besoin d'un service qui sera utilisé par tous les composants.

- ***Modification du service***

Ce service s'appuie sur le modèle créé précédemment (comprenez ici l'interface) de livre. Il est possible aussi de créer une classe mais dans notre application, le principe de l'interface suffit car nous n'avons pas besoin de constructeur ni de méthodes.

**T2.1** Intégrez dans le service le code nécessaire à l'utilisation de Firebase et sa première méthode, concernant l'ajout d'un livre :

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

import {
  Firestore, addDoc, collection, collectionData,
  doc, docData, deleteDoc, updateDoc, DocumentReference, setDoc
} from '@angular/fire/firestore';
import { Observable } from 'rxjs';
import { IBook } from '../models/book.model';

@Injectable({
  providedIn: 'root'
})

export class LivreService {

  constructor(private firestore: Firestore) { }
```

```
  addBook(book: IBook) {
    const livresRef = collection(this.firestore, 'livres');
    return addDoc(livresRef, book);
  }
}
```

Remarquez que nous avons besoin d'importer le module Firestore pour l'accès à Firebase et de le déclarer dans le constructeur du service.

Vous voyez aussi que nous avons besoin d'une référence à nos livres pour la collection et quelle est obtenue en transmettant à deux arguments : la variable du Firestore et le nom de la collection à utiliser (là j'ai francisé le nom comme déclaré dans Firebase).

- ***Construction du template de l'ajout***

Nous allons d'abord définir le template de cet ajout en fonction du modèle, « à la sauce » Bootstrap, ensuite, on s'occupera de la logique métier.

**T2.2** Intégrez le code complet du template dans `book.component.html` :

```
<div>
    <h4>Ajout d'un livre dans la bibliothèque</h4>
    <form #bookForm="ngForm" (ngSubmit)="onSubmit(bookForm)">
      <div class="form-group">
        <i class="bi bi-book"></i>
         <input class="form-control" name="titre" #name="ngModel"
          [(ngModel)]="book.titre" placeholder="Titre du livre" required>
      </div>

      <div class="form-group">
        <i class="bi bi-person"></i>
        <input class="form-control" name="auteur" #name="ngModel"
          [(ngModel)]="book.auteur" placeholder="Auteur du livre" required>
      </div>

      <div class="form-group">
        <i class="bi bi-person-lines-fill"></i>
        <input class="form-control" name="editeur" #name="ngModel"
          [(ngModel)]="book.editeur" placeholder="Editeur" required>
      </div>

       <div class="form-group">
        <i class="bi bi-gem"></i>
         <select class="form-control" name="genre" #name=ngModel
           [(ngModel)]="book.genre">
           <option value="Aventures">Aventures</option>
           <option value="Policier">Policier</option>
           <option value="Science Fiction">Science Fiction</option>
         </select>
        </div>

        <div class="form-group">
          <div class="form-group float-right">
             <button type="submit" class="btn btn-success"
               [disabled]="!bookForm.valid" [disabled]="bookForm.invalid">
                <i class="bi bi-plus"></i>Ajout du livre</button>
          </div>
        </div>
    </form>
  </div>
```

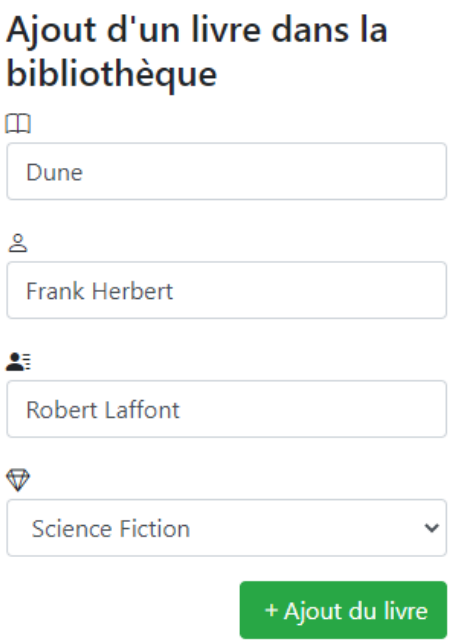
**T2.3** Et celui du contrôleur dans `book.component.ts` :

```
import { Component, ElementRef, OnInit, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { IBook } from 'src/app/models/book.model';
import { BookService } from 'src/app/services/book.service';

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

  book: IBook = { titre: '', auteur: '', editeur: '', genre: '' };

  constructor(private bookService: BookService) { }

  ngOnInit() {
  }

  onSubmit(form: NgForm) {
    this.bookService.addBook(form.value).
      then(() => form.reset());
  }
}
```

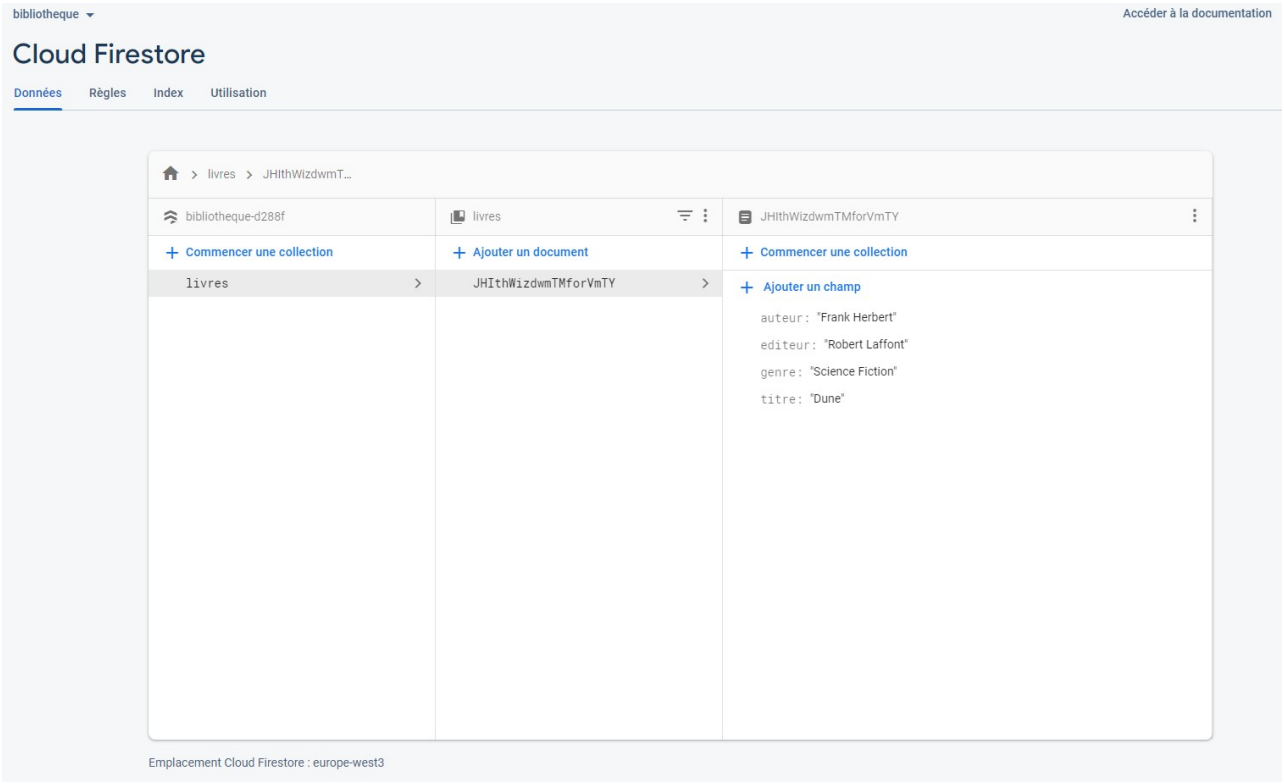
Outre l'importation des modules nécessaires, on voit l'utilisation du service (dans le constructeur) et d'une de ses méthodes (en fait pour l'instant il n'y en a qu'une).

- ***Démonstration***

L'ajout d'un livre comme celui-ci :




Doit aboutir au résultat suivant dans la console Firebase au niveau du Cloud Firestore (après un rafraîchissement) :


bibliotheque
Accéder à la documentation
Cloud Firestore
Données
Règles
Index
Utilisation
livres > JHIthWizdwmT...
bibliotheque-d288f
livres
JHIthWizdwmTMforVmTY
Commencer une collection
Ajouter un document
Commencer une collection
livres
JHIthWizdwmTMforVmTY
Ajouter un champ
auteur: "Frank Herbert"
editeur: "Robert Laffont"
genre: "Science Fiction"
titre: "Dune"
Emplacement Cloud Firestore : europe-west3

## T2.4 Testez...