

Gestion des images Spring boot 3 et Angular

Objectifs :

1. Ajouter, coté Spring Boot, l'attribut image
2. Modifier coté Angular le composant addProduit pour upload une image
3. Modifier coté Angular le composant updateProduit pour consulter l'image
4. Ajouter, coté Angular, la colonne image à la liste des produits
5. Modifier coté Angular le composant updateProduit pour modifier l'image
6. Gérer plusieurs images par produit
7. Stocker les images dans un dossier

Ajouter, coté Spring Boot, l'attribut image

Créer l'entité Image

1. Créer l'entité Image

```
package com.nadhem.produits.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToOne;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import jakarta.persistence.Column;
import jakarta.persistence.Lob;

@Entity
@Builder
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Image {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long idImage ;
    private String name ;
    private String type ;

    @Column( name = "IMAGE" , length = 4048576 )
    @Lob
    private byte[] image;
    @OneToOne
    private Produit produit;
}
```

2. Modifier l'entité Produit

```
@OneToOne
private Image image;

public Image getImage() {
    return image;
}

public void setImage(Image image) {
    this.image = image;
}
```

Créer le Service ImageService

Créer interface ImageRepository

```
package com.nadhem.produits.repos;
import org.springframework.data.jpa.repository.JpaRepository;
import com.nadhem.produits.entities.Image;

public interface ImageRepository extends JpaRepository<Image , Long> {
}
```

Créer interface ImageService

```
package com.nadhem.produits.service;

import java.io.IOException;
import org.springframework.http.ResponseEntity;
import org.springframework.web.multipart.MultipartFile;
import com.nadhem.produits.entities.Image;

public interface ImageService {
    Image uploadImage(MultipartFile file) throws IOException;
    Image getImageDetails(Long id) throws IOException;
    ResponseEntity<byte[]> getImage(Long id) throws IOException;
    void deleteImage(Long id) ;
}
```

Créer la classe ImageServiceImpl

```
package com.nadhem.produits.service;

import java.io.IOException;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
```

```

import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import com.nadhemi.produits.entities.Image;
import com.nadhemi.produits.repos.ImageRepository;

@Service
public class ImageServiceImpl implements ImageService{

    @Autowired
    ImageRepository imageRepository;

    @Autowired
    ProduitService produitService;

    @Override
    public Image uploadImage(MultipartFile file) throws IOException {
        /*Ce code commenté est équivalent au code utilisant le design pattern Builder
        * Image image = new Image(null, file.getOriginalFilename(),
        *                               file.getContentType(), file.getBytes(), null);
        return imageRepository.save(image);*/

        return imageRepository.save(Image.builder()
            .name(file.getOriginalFilename())
            .type(file.getContentType())
            .image(file.getBytes()).build() );
    }

    @Override
    public Image getImageDetails(Long id) throws IOException{
        final Optional<Image> dbImage = imageRepository.findById(id);

        return Image.builder()
            .idImage(dbImage.get().getIdImage())
            .name(dbImage.get().getName())
            .type(dbImage.get().getType())
            .image(dbImage.get().getImage()).build() ;
    }

    @Override
    public ResponseEntity<byte[]> getImage(Long id) throws IOException{
        final Optional<Image> dbImage = imageRepository.findById(id);

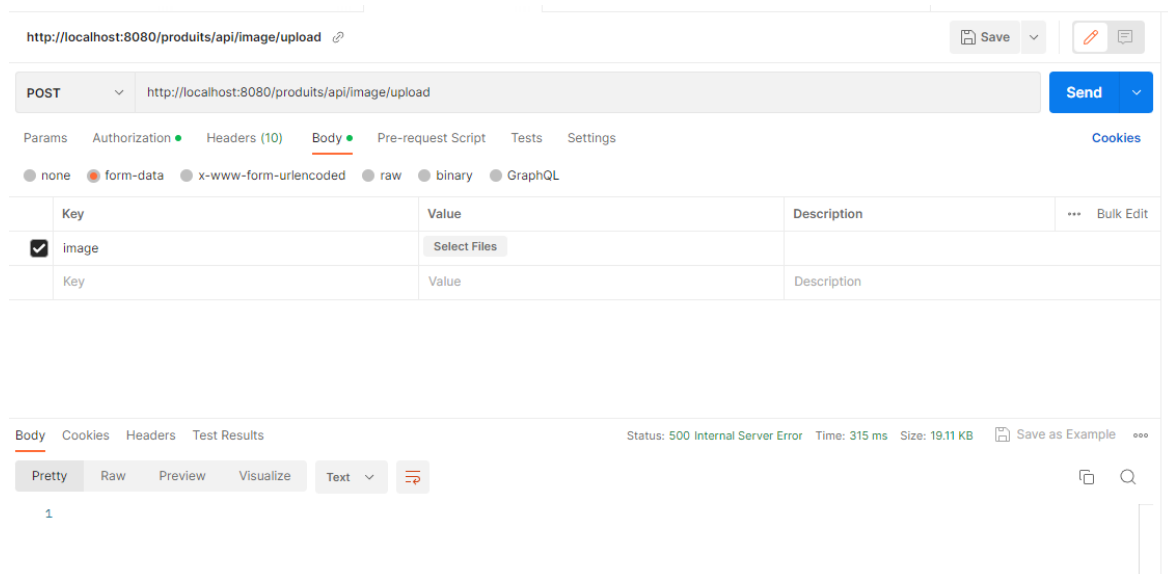
        return ResponseEntity
            .ok()
            .contentType(MediaType.valueOf(dbImage.get().getType()))
            .body(dbImage.get().getImage());
    }

    @Override
    public void deleteImage(Long id) {
        imageRepository.deleteById(id);
    }
}

```

Créer la classe ImageRestController

Tester avec Postman l'upload d'une image



Modifier coté Angular le composant addProduit pour upload une image

Ajouter la classe model Image et modifier la classe model Produit

```
export class Image {  
  idImage! : number ;  
  name! : string ;  
  type! : string ;  
  image! : number[] ;  
}
```

```
export class Produit {  
  idProduit! : number;  
  nomProduit! : string;  
  prixProduit! : number;  
  dateCreation! : Date ;  
  categorie! : Categorie;  
  image! : Image  
  imageStr!:string  
}
```

Ajouter les méthodes de gestion des images au service ProduitService

```
uploadImage(file: File, filename: string): Observable<Image>{  
  const imageFormData = new FormData();
```

```

        imageFormData.append('image', file, filename);
        const url = `${apiURL} /image/upload`;
        return this.http.post<Image>(url, imageFormData);
    }

    loadImage(id: number): Observable<Image> {
        const url = `${this.apiURL} /image/get/info/${id}`;
        return this.http.get<Image>(url);
    }
}

```

Ajouter un input de type file au composant AddProduitComponent :

```

<div class="col-sm-16 col-md-16 col-lg-16">
    <label class="mb-2">Image :</label>
    <input type="file" (change)="onImageUpload($event)"
name="logo" class="form-control">
</div>

```

Ajouter une image composant AddProduitComponent pour prévisualiser l'image avant son upload :

```

<div class="col-sm-16 col-md-16 col-lg-16">
    <img [src]="imagePath" class="card-img-top img-fluid rounded-start"
style="width: 500px; height: 300px" />
</div>

```

Ajouter la méthode OnImageUpload

```

uploadedImage!: File;
imagePath: any;

```

```

onImageUpload(event: any) {
    this.uploadedImage = event.target.files[0];

    var reader = new FileReader();
    reader.readAsDataURL(this.uploadedImage);
    reader.onload = (_event) => { this.imagePath = reader.result; }
}

```

Modifier la méthode addProduit :

```

addProduit(){

    this.produitService
    .uploadImage(this.uploadedImage, this.uploadedImage.name)
    .subscribe((img: Image) => {
        this.newProduit.image=img;
        this.newProduit.categorie = this.categories.find(cat => cat.idCat
== this.newIdCat)!;

        this.produitService
        .ajouterProduit(this.newProduit)
        .subscribe(() => {
            this.router.navigate(['produits']);
        });
    });
}

```

Tester l'upload d'une image en ajoutant un produit

Ajouter un Produit :

Nom Produit

Prix Produit

Date création

Categorie

Image :



Ajouter

Vérifier l'ajout dans la BD Mysql

Options

	id_image	image	name	type
1	[BLOB - 24.6 KiB]	textRotate.jpg	image/jpeg	

☐ Check all With selected: Edit Copy Delete Export

Modifier coté Angular le composant updateProduit pour consulter l'image

Modifier la méthode ngOnInit() comme suit :

```
myImage! : string;
```

```
ngOnInit(): void {
  this.produitService.listeCategories().
    subscribe(cats => {this.categories = cats._embedded.categories;
    console.log(cats);
  });

  this.produitService.consulterProduit(this.activatedRoute.snapshot.params['id']).
    subscribe( prod =>{ this.currentProduit = prod;
    this.updatedCatId = prod.categorie.idCat;



    this.produitService
      .loadImage(this.currentProduit.image.idImage)
      .subscribe((img: Image) => {
        this.myImage = 'data:' + img.type + ';base64,' + img.image;
      });
  } ) ;
}
```

Ajouter l'élément image au fichier update-produit.component.html

```
<div class="col-sm-16 col-md-16 col-lg-16">
  <img [src]="myImage" class="card-img-top img-fluid rounded-start"
  style="width: 500px; height: 300px" />
</div>
```

Tester la consultation d'un produit

Ajouter, coté Angular, la colonne image à la liste des produits

Id	Nom	Prix	Date Création	Catégorie	Image		
8	iphone 14	4000	24/03/2023	SmartPhones		Supprimer	Modifier
9	dell inspiron	5000	08/03/2023	PC		Supprimer	Modifier

Ajouter la colonne image au « table » au niveau du fichier produits.component.html :

```
<tr> <th>Id</th><th>Nom</th> <th>Prix</th> <th>Date  
Création</th><th>Catégorie</th> <th>Image</th></tr>
```

```
<td><img class="card-img-top img-responsive"  
        src = "{{produit.imageStr}}"  
        style = "height :50px;width:100px">  
</td>
```

Modifier la méthode chargerProduits() de la classe ProduitsComponent, pour qu'elle récupère l'image du produit grâce à l'api *loadImage* :

```
chargerProduits(){  
  this.produitService.listeProduit().subscribe(prods => {  
    this.produits = prods;  
  
    this.produits.forEach((prod) => {  
      this.produitService  
        .loadImage(prod.image.idImage)  
        .subscribe((img: Image) => {  
          prod.imageStr = 'data:' + img.type + ';base64,' + img.image;  
        });  
    });  
  });  
}
```


Modifier coté Angular le composant updateProduit pour modifier l'image

Ajouter un input de type file au composant UpdateProduitComponent :

```
<div class="col-sm-16 col-md-16 col-lg-16">

  <label class="mb-2">Image :</label>
  <input type="file" (change)="onImageUpload($event)" class="form-control">
</div>
```

Ajouter la méthode OnImageUpload

```
uploadedImage!: File;
isImageUpdated: Boolean=false;
```

```
onImageUpload(event: any) {
  if(event.target.files && event.target.files.length) {
    this.uploadedImage = event.target.files[0];
    this.isImageUpdated =true;
    const reader = new FileReader();
    reader.readAsDataURL(this.uploadedImage);
    reader.onload = () => { this.myImage = reader.result as string; };
  }
}
```

Modifier la méthode updateProduit() comme suit :

```
updateProduit() {
  this.currentProduit.categorie = this.categories.find(cat => cat.idCat ==
this.updatedCatId!);
  //tester si l'image du produit a été modifiée
  if (this.isImageUpdated)
  {
    this.produitService
      .uploadImage(this.uploadedImage, this.uploadedImage.name)
      .subscribe((img: Image) => {
        this.currentProduit.image = img;

        this.produitService
          .updateProduit(this.currentProduit)
          .subscribe((prod) => {
```

```

        this.router.navigate(['produits']);
    });
}
else{
    this.produitService
        .updateProduit(this.currentProduit)
        .subscribe((prod) => {
            this.router.navigate(['produits']);
        });
}
}
}

```

Optimisation

Si l'image du produit a été modifiée alors il faut supprimer l'ancienne image de la BD et ce pour ne pas surcharger la BD. Pour ce faire on modifie la méthode updateProduit() de la classe ProduitServiceImpl comme suit :

```

@Override
    public Produit updateProduit(Produit p) {
        Long oldProdImageId =
this.getProduit(p.getIdProduit()).getImage().getIdImage();
        Long newProdImageId = p.getImage().getIdImage();

        Produit prodUpdated = produitRepository.save(p);

        if (oldProdImageId != newProdImageId) //si l'image a été modifiée
            imageRepository.deleteById(oldProdImageId);

        return prodUpdated;
    }

```

Gérer plusieurs images par produit

Entité Produit

```

@OneToMany (mappedBy = "produit")
private List<Image> images;

```

Entité Image

```

@ManyToOne()
@JoinColumn (name="PRODUIT_ID")
@JsonIgnore
private Produit produit;

```

Interface ImageService

```

Image uploadImageProd(MultipartFile file, Long idProd) throws IOException;

List<Image> getImagesParProd(Long prodId);

```

Classe ImageServiceImpl

@Service

... @Override

```
public Image uplaodImageProd(MultipartFile file, Long idProd)

    throws IOException {
    Produit p = new Produit();
    p.setIdProduit(idProd);
    return imageRepository.save(Image.builder()
        .name(file.getOriginalFilename())
        .type(file.getContentType())
        .image(file.getBytes())
        .produit(p).build() );
}

@Override
public List<Image> getImagesParProd(Long prodId) {
    Produit p = produitRepository.findById(prodId).get();
    return p.getImages();
}
```

Classe ImageRestController

```
@PostMapping(value = "/uplaodImageProd/{idProd}" )
public Image uploadMultiImages(@RequestParam("image")MultipartFile file,
    @PathVariable("idProd") Long idProd)
    throws IOException {
    return imageService.uplaodImageProd(file, idProd);
}

@RequestMapping(value = "/getImagesProd/{idProd}" ,
    method = RequestMethod.GET)
public List<Image> getImagesProd(@PathVariable("idProd") Long idProd)
throws IOException {
    return imageService.getImagesParProd(idProd);
}
```

Tester avec Postman l'api "/ uplaodImageProd/{idProd}"

Tester avec Postman l'api "/getImagesProd/{idProd}"

Modifications coté Angular

Model Produit, ajouter l'attribut suivant :

```
images!: Image[];
```

updateProduit.html

```

<div class="mt-2">
    <button type="submit" (click)="onAddImageProduit()" class="btn
btn-success">Ajouter Image</button>
</div>

```

```

<div class="card shadow mb-4">
    <div class="card-body">
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>Images</th>
                </tr>
            </thead>
            @for(img of currentProduit.images; track currentProduit.idProduit)
            {
                <tbody >
                    <tr>
                        <td></td>
                    </tr>
                </tbody>
            }
        </table>
    </div>
</div>

```

update-produit.component.ts

```

ngOnInit(): void {
    this.produitService.listeCategories().
    subscribe(cats => {this.categories = cats._embedded.categories;
    });

    this.produitService.consulterProduit(this.activatedRoute.snapshot.params['id'])
    .subscribe( prod =>{ this.currentProduit = prod;
        this.updatedCatId = prod.categorie.idCat;
    } ) ;
}

```

```

onAddImageProduit() {
    this.produitService
    .uploadImageProd(this.uploadedImage,
this.uploadedImage.name,this.currentProduit.idProduit)
    .subscribe( (img : Image) => {

```

```

        this.currentProduit.images.push(img);
    });
}

```

```

supprimerImage(img: Image){
    let conf = confirm("Etes-vous sûr ?");
    if (conf)
        this.produitService.supprimerImage(img.idImage).subscribe(() => {
            //supprimer image du tableau currentProduit.images
            const index = this.currentProduit.images.indexOf(img, 0);
            if (index > -1) {
                this.currentProduit.images.splice(index, 1);
            }
        });
    }
    updateProduit() {
        this.currentProduit.categorie = this.categories.find(cat => cat.idCat ==
this.updatedCatId!);
        this.produitService
            .updateProduit(this.currentProduit)
            .subscribe((prod) => {
                this.router.navigate(['produits']);
            });
    }
}

```

ProduitService

```

uploadImageProd(file: File, filename: string, idProd:number): Observable<any>{
    const imageFormData = new FormData();
    imageFormData.append('image', file, filename);
    const url = `${this.apiUrl + '/image/uplaodImageProd'}/${idProd}`;
    return this.http.post(url, imageFormData);
}

```

```

supprimerImage(id : number) {
    const url = `${this.apiUrl}/image/delete/${id}`;
    return this.http.delete(url, httpOptions);
}

```

Ajouter le bouton supprimer à la liste des images

```

<td style="width:20%" ><a class="btn btn-danger"
(click)="supprimerImage(img)">Supprimer</a></td>

```

Afficher la première image de chaque produit dans la liste initiale

An niveau fichier produits.component.ts

```
chargerProduits(){
  this.produitService.listeProduit().subscribe(prods => {
    this.produits = prods;

    this.produits.forEach((prod) => {
      prod.imageStr = 'data:' + prod.images[0].type + ';base64,' +
prod.images[0].image;
    });
  });
}
```

An niveau fichier produits.component.html

```
src = "{{produit.imageStr}}"
```

Stocker les images dans un dossier

Ajouter l'attribut suivant à l'entité produit

```
private String imagePath;
```

Ajouter les méthodes suivantes à la classe ImageRestController

```
@RequestMapping(value = "/uploadFS/{id}" , method = RequestMethod.POST)
public void uploadImageFS(@RequestParam("image") MultipartFile
file,@PathVariable("id") Long id) throws IOException {
  Produit p = produitService.getProduit(id);
  p.setImagePath(id+".jpg");

Files.write(Paths.get(System.getProperty("user.home")+"/images/"+p.getImagePath())
, file.getBytes());
  produitService.saveProduit(p);
}

@RequestMapping(value = "/loadfromFS/{id}" ,
method = RequestMethod.GET,
produces = org.springframework.http.MediaType.IMAGE_JPEG_VALUE)
```

```

    public byte[] getImageFS(@PathVariable("id") Long id) throws IOException {

        Produit p = produitService.getProduit(id);
        return
Files.readAllBytes(Paths.get(System.getProperty("user.home")+"/images/"+p.getImagePath()));
    }

```

Tester avec Postman l'upload d'une image et son affichage

http://localhost:8080/produits/api/image/loadfromFS/8

POST http://localhost:8080/produits/api/image/uploadFS/9

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
<input checked="" type="checkbox"/> image	dell inspiron.jpg ×
Key	Value

http://localhost:8080/produits/api/image/loadfromFS/8

GET http://localhost:8080/produits/api/image/loadfromFS/8

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value



Modifications coté Angular

Ajouter la méthode uploadImageFS à la classe ProduitService

```

uploadImageFS(file: File, filename: string, idProd : number): Observable<any>{
    const imageFormData = new FormData();

```

```

imageFormData.append('image', file, filename);
const url = `${this.apiUrl + '/image/uploadFS'}/${idProd}`;
return this.http.post(url, imageFormData);
}

```

Modifier la méthode addProduit() du composant addProduit

```

addProduit(){
    this.newProduit.categorie = this.categories.find(cat => cat.idCat
== this.newIdCat)!;
    this.produitService
        .ajouterProduit(this.newProduit)
        .subscribe((prod) => {

            this.produitService
                .uploadImageFS(this.uploadedImage,
this.uploadedImage.name,prod.idProduit)
                .subscribe((response: any) => {}
            );

            this.router.navigate(['produits']);
        });
}

```

Modifier la composant *ProduitsComponent* pour afficher les images des produits dans la liste :

Fichier **produits.component.ts**

```

apiurl:string='http://localhost:8080/produits/api';

```

```

chargerProduits(){
    this.produitService.listeProduit().subscribe(prods => {
        this.produits = prods;
    });
}

```

produits.component.html

```

src="{{apiurl+'/image/loadfromFS/'+produit.idProduit}}"

```

Supprimer le produit et son image du dossier images

@Override

```

public void deleteProduitById(Long id) {
    Produit p = getProduit(id);
}

```



```
        //supprimer l'image avant de supprimer le produit
        try {
Files.delete(Paths.get(System.getProperty("user.home")+"/images/"+p.getImagePath()));
        } catch (IOException e) {
            e.printStackTrace();
        }
        produitRepository.deleteById(id);
    }
}
```