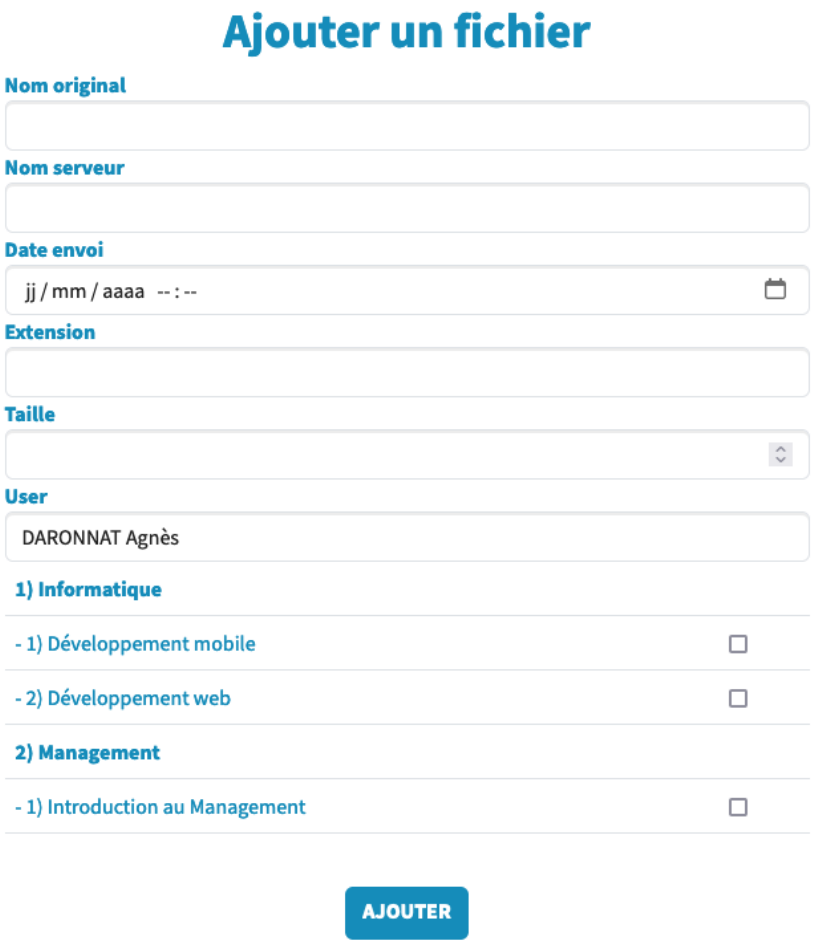
Chapitre 11 – Upload

Dans ce chapitre, nous allons modifier le formulaire d'ajout d'un fichier afin d'envoyer réellement un fichier sur le serveur.

Table des matières

A. Modification du formulaire permettant d'ajouter un fichier



Actuellement, notre formulaire permet de saisir des données concernant un fichier. Nous allons réduire ce formulaire afin de sélectionner un fichier, le propriétaire du fichier et la ou les sous-catégories du fichier. L'extension, la date d'envoi, la taille du fichier, le nom du fichier original et le nom du fichier sur le serveur seront renseignés automatiquement par des instructions à placer dans le contrôleur.

Dans «src/Form/FichierType » :

```
<?php

namespace App\Form;

use App\Entity\Fichier;
use App\Entity\User;
use App\Entity\Scategorie;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Doctrine\ORM\EntityRepository;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\FileType;
use Symfony\Component\Validator\Constraints\File;

class FichierType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('fichier', FileType::class, array('label' => 'Fichier', 'mapped'=>false,'attr' => ['class'=>
'form-control'], 'label_attr' => ['class'=> 'fw-bold'],'constraints' => [
                new File([
                    'maxSize' => '200k',
                    'mimeTypes' => [
                        'application/pdf',
                        'application/x-pdf',
                        'image/jpeg',
                        'image/png',
                    ],
                    'mimeTypesMessage' => 'Le site accepte uniquement les fichiers PDF, PNG et JPG',
                ])
            ],))
            ->add('user', EntityType::class, [
                'class' => User::class,
                'attr' => ['class'=> 'form-control'], 'label_attr' => ['class'=> 'fw-bold'],
                'choice_label' => function($user) {
                    return $user->getNom() . ' ' . $user->getPrenom();
                },
                'query_builder' => function (EntityRepository $er) {
```

```
                return $er->createQueryBuilder('u')
                    ->orderBy('u.nom', 'ASC')
                    ->addOrderBy('u.prenom', 'ASC');
            },
        ])
        ->add('scategories', EntityType::class, [
            'class' => Scategorie::class,
            'choices' => $options['scategories'],
            'choice_label' => 'libelle',
            'expanded' => true,
            'multiple' => true,
            'label' => false, 'mapped' => false])
           ->add('ajouter', SubmitType::class, ['attr' => ['class'=> 'btn bg-primary text-white m-4' ], 'row_attr' => ['class' => 'text-center'],])

        ;

    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Fichier::class,
            'scategories' => []
        ]);
    }
}
```

Commentaires :

```
'constraints' => [ // Nous ajoutons une contrainte sur le composant « FileType ».
            new File([ // La contrainte est de type File
                'maxSize' => '200k', // Nous limitons à 200ko la taille du fichier
                'mimeTypes' => [ // Nous mettons la liste des MIMES que nous acceptons
                    'application/pdf',
                    'application/x-pdf',
                    'image/jpeg',
                    'image/png',
                ],
```

Dans « templates/fichier/ajout-fichier.hml.twig

```
{% extends 'base.html.twig' %}

{% block title %}
        {{parent()}}
        Ajout fichier
{% endblock %}

{% block body %}
```

```
        <div class="container-fluid">
                <div class="row justify-content-center">

                        <div class="col-12 col-sm-10 col-md-8 text-primary">
                                <h1 class="text-center text-primary mt-4 pt-4 fw-bold">
                                        Ajouter un fichier</h1>
                                {{ form_start(form) }}
                                {{ form_row(form.fichier) }}
                                {{ form_row(form.user) }}
                                <div class="table-responsive">
                                        <table class="table">
                                        {% set id = null %}
                                                {% for scategorie in scategories %}
                                                        <tr>
                                                                {% if id != scategorie.categorie.id %}
                                                                        {% set id =
scategorie.categorie.id %}
                                                                        <tr><td class="text-primary fw-
bold">{{scategorie.categorie.id}}) {{scategorie.categorie.libelle}}<td></td></tr>
                                                                {% endif %}
                                                                <td class="text-primary">
                                                                        -
                                                                        {{scategorie.numero}})
{{scategorie.libelle}}</td>
                                                                <td class="text-primary">

        {{ form_widget(form.scategories[scategorie.id]) }}</td>
                                                        </tr>
                                                {% endfor %}
                                        </table>
                                </div>
                                {{ form_row(form.ajouter) }}

                                {{ form_end(form) }}
                        </div>
                </div>
        </div>

{% endblock %}
```

Commentaire :
Nous modifions l'affichage du formulaire en retirant les anciens composants et en mettant le composant « file ».

Voici le résultat :

B. Création du répertoire qui accueillera les fichiers

À la racine du projet, créez un répertoire « uploads ». À l'intérieur de celui-ci, créez un répertoire « fichiers ».

```
su login ????
```

```
mkdir uploads
```

```
mkdir uploads/fichiers
```

Enfin, donnez les droits suivants au répertoire « uploads » :

```
chmod -R 777 uploads
```

Le répertoire « upload » nous permettra de stocker des fichiers envoyés par l'utilisateur. Le répertoire « fichiers » contiendra les fichiers qui seront partagés entre utilisateurs.

Dans le fichier « services.yaml », créez une clé permettant de configurer le chemin vers votre répertoire « fichiers ».

Dans « config/services.yaml » :

```
parameters:
    file_directory: '%kernel.project_dir%/uploads/fichiers'
```

Commentaire :
La clé est file_directory. Nous lui mettons le chemin où se situe l'application à l'aide de « %kernel.project_dir% » et nous mettons à la suite les répertoires jusqu'à « fichiers ».

C. Mettre en place le « listener » :

Dans le répertoire « EventListener » créez le fichier « FichierListener.php » puis mettez le code suivant :

```
<?php
namespace App\EventListener;
use App\Entity\Fichier;
use Doctrine\Persistence\Event\LifecycleEventArgs;

class FichierListener
{
    public function prePersist(Fichier $fichier, LifecycleEventArgs $event): void
    {
        $fichier->setDateEnvoi(new \Datetime());

    }
}
```

Commentaire :
Le "listener » a pour objectif d'affecter la date d'envoi avec la date et l'heure actuelle.

D. Modification du contrôleur :

Installez le package permettant de vérifier le type de fichier que vous enverrez.

```
composer require symfony/mime
```

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Annotation\Route;
use App\Form\FichierType;
use App\Entity\Fichier;
use App\Repository\UserRepository;
use App\Repository\ScategorieRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\String\Slugger\SluggerInterface;


class FichierController extends AbstractController
{
   #[Route('/ajout-fichier', name: 'app_ajout_fichier')]
      public function ajoutFichier(Request $request, ScategorieRepository $scategorieRepository, EntityManagerInterface $em, SluggerInterface $slugger): Response
   {
      $fichier = new Fichier();
      $scategories = $scategorieRepository->findBy([], ['categorie'=>'asc', 'numero'=>'asc']);
      $form = $this->createForm(FichierType::class, $fichier, ['scategories'=>$scategories]);
      $form->handleRequest($request);

      if ($form->isSubmitted() && $form->isValid()) {
         $selectedScategories = $form->get('scategories')->getData();
         foreach ($selectedScategories as $scategorie) {
            $fichier->addScategory($scategorie);
         }

         $file = $form->get('fichier')->getData();

         if($file){
                                    $nomFichierServeur = pathinfo($file->getClientOriginalName(), PATHINFO_FILENAME);
            $nomFichierServeur = $slugger->slug($nomFichierServeur);
            $nomFichierServeur = $nomFichierServeur.'-'.uniqid().'.'.$file->guessExtension();
            try{
               $fichier->setNomServeur($nomFichierServeur);
               $fichier->setNomOriginal($file->getClientOriginalName());
               $fichier->setDateEnvoi(new \Datetime());
               $fichier->setExtension($file->guessExtension());
               $fichier->setTaille($file->getSize());
               $em->persist($fichier);
               $em->flush();
               $file->move($this->getParameter('file_directory'), $nomFichierServeur);
               $this->addFlash('notice', 'Fichier envoyé');
               return $this->redirectToRoute('app_ajout_fichier');
            }
            catch(FileException $e){
               $this->addFlash('notice', 'Erreur d\'envoi');
            }
         }
```

```
        }
        return $this->render('fichier/ajout-fichier.html.twig', [
            'form'=>$form,
            'scategories'=> $scategories
        ]);
    }
```

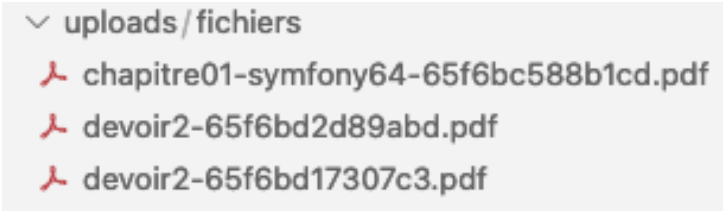
Commentaires :
SluggerInterface $slugger // nous récupérons une variable de type « SluggerInterface » qui nous permettra de formater une chaine de caractères en y retirant les espaces et les caractères spéciaux.
$nomFichier = pathinfo($fichier->getClientOriginalName(), PATHINFO_FILENAME); // nous récupérons le nom du fichier sans son extension. Le nom original du fichier se trouve dans getClientOriginalName.
pathinfo est une fonction qui permet de récupérer l'élément que nous souhaitons dans
$nomFichier = $slugger->slug($nomFichier); // Nous appliquons le slug afin de formater le nom du fichier.
$nomFichier = $nomFichier.'-'.uniqid().'.'.$fichier->guessExtension(); // Nous prenons le nom du fichier formaté auquel nous ajoutons un numéro unique et son extension. Son extension est récupérée à l'aide du MIME du fichier et non pas en lisant celle du fichier qui peut être modifiée facilement.
$fichier->move($this->getParameter('file_directory'), $nomFichier);
Nous déplaçons le fichier de la zone temporaire du serveur au répertoire « fichiers » que nous avons défini dans le fichier « services.yaml ». C'est la méthode « getParameter » qui permet de retrouver la clé « file_directory ».

Méthode	Description
getSize()	Récupère la taille du fichier en octets
guessExtension()	Récupère le type du fichier en lisant son MIME
getClientOriginalName()	Récupère le nom du fichier

Vous pouvez consulter la liste des différents Mimes : Mimes

Testez maintenant le formulaire :

Le fichier a dû apparaitre dans le répertoire de votre serveur :



Commentaire :
Vous pouvez constater que le fichier est en partie renommé ce qui permet aux utilisateurs d'envoyer des fichiers avec le même nom et de ne pas savoir quel nom il a sur le serveur.

E. Contrôleur de téléchargement du fichier :

Dans « src/Controller/FichierController.php » :

```
#[Route('/private-telechargement-fichier/{id}', name: 'app_telechargement_fichier', requirements: ["id"=>"\d+"] )]
    public function telechargementFichier(Fichier $fichier) {
        if ($fichier == null){
            $this->redirectToRoute('app_liste_fichiers_par_utilisateur'); }
        else{
                    return  $this->file($this->getParameter('file_directory').'/'.$fichier->getNomServeur(), $fichier->getNomOriginal());
        }
    }
```

Commentaires :
Le fichier est récupéré directement à partir de son identifiant et stocké dans la variable « $fichier ».
Si la variable est « null » alors nous redirigeons l'utilisateur sur la liste des fichiers, sinon nous envoyons le fichier à l'utilisateur en le renommant avec son nom original.

F. Modification de l'affichage des fichiers par utilisateur :

Dans «templates/fichier/liste-fichiers-par-utilisateur.html.twig » :

```
{% extends 'base.html.twig' %}

{% block title %}
        {{parent()}}Liste des fichiers par utilisateur
{% endblock %}

{% block body %}
        <div class="container-fluid">
                <h1 class="text-center text-primary mt-4 pt-4 fw-bold">
                        Liste des fichiers par utilisateur</h1>
```

```
<div class="row justify-content-center">
    <div class="col-12 col-md-8 bg-white p-4 m-0 text-primary">
        <div class="table-responsive">
            <table class="table table-hover">
                {% for u in users %}
                <tr>
                    <td class="text-white bg-primary">{{u.nom | upper}}
                        {{u.prenom | capitalize }}</td>
                </tr>
                {% if u.fichiers | length > 0 %}
                    {% for f in u.fichiers %}
                    <tr>
                        <td class="text-center">


                            <fieldset>

    <legend><i class="bi bi-filetype-pdf"></i> <a href="{{path('app_telechargement_fichier', {'id':f.id})}}" style="text-decoration: none">{{f.nomOriginal}}</a></legend>
                                <ul style="list-style: none">

    {% for s in f.scategories %}


    <li>{{s.numero}})

        {{s.libelle | capitalize}}</li>

    {% endfor %}
                                </ul>
                            </fieldset>


                        </td>
                    </tr>
                    {% endfor %}
                {% else %}
                    <tr>
                        <td class="text-center">Aucun fichier</td>
                    </tr>
                {% endif %}
                {% endfor %}
            </table>
        </div>
    </div>
</div>
```

```
        {% endblock %}
```

Commentaire :
Nous pouvons maintenant télécharger un fichier en cliquant sur le nom du fichier. Pour cela, nous avons créé un lien qui va ouvrir un nouvel onglet. Nous appelons la route « app_telechargement_fichier » que nous avons mise en place précédemment. Nous précisons l'identifiant du fichier que nous souhaitons télécharger dans la fonction « path » => path('app_telechargement_fichier', {'id':f.id}
Le nom de la variable est ici « id » que nous mettons dans un tableau. Sa valeur est f.id qui contient l'identifiant du fichier dans la base de données.
Le fichier est récupéré dans le répertoire « uploads/fichiers » (inaccessible directement par un navigateur), puis renommé avec son nom d'origine et enfin envoyé à l'utilisateur.