

MEDIATEK FORMATION

Sommaire

CONTEXTE :	3
I. MISSION 1 : NETTOYER ET OPTIMISER LE CODE EXISTANT	4
TACHE 1 : NETTOYER LE CODE EN SUIVANT LES INDICATIONS DE SONARLINT (NE NETTOYER QUE LES FICHIERS CREEES PAR LE DEVELOPPEUR, DONC TRIER LES "ACTION ITEMS" DE SONARLINT PAR "LOCATION" ET S'ARRETER AU PREMIER FICHIER DANS « VENDOR».	4
TACHE 2 : DANS LE RESPECT DES BONNES PRATIQUES DE CODAGE, EN PARTICULIER SOLID (ICI, LE S : "SINGLE RESPONSABILITY"), MODIFIER LES METHODES DE FORMATIONREPOSITORY ET PLAYLISTREPOSITORY QUI CONTIENNENT DES TESTS SUR \$TABLE : A CHAQUE FOIS, CREER 2 METHODES PLUTOT QU'UNE, POUR EVITER CE TEST. LE RESTE DU CODE DE L'APPLICATION DOIT ETRE ADAPTE POUR EXPLOITER CES NOUVELLES METHODES.	6
TACHE 3 : DANS LA PAGE DES PLAYLISTS, AJOUTER UNE COLONNE POUR AFFICHER LE NOMBRE DE FORMATIONS PAR PLAYLIST ET PERMETTRE LE TRI CROISSANT ET DECREISSANT SUR CETTE COLONNE. CETTE INFORMATION DOIT AUSSI S'AFFICHER DANS LA PAGE D'UNE PLAYLIST.	11
II. MISSION 2 : CODER LA PARTIE BACK-OFFICE	18
TACHE 4 : MODIFICATION D'UNE FORMATION.	18
TACHE 5 : AJOUT D'UNE FORMATION	24
TACHE 6 : SUPPRIMER UNE FORMATION	27
TACHE 7 : AJOUT D'UNE PLAYLIST	30
TACHE 8 : MODIFICATION D'UNE PLAYLIST	37
TACHE 9 : SUPPRIMER UNE PLAYLIST	40
TACHE 10 : AJOUTER UNE CATEGORIE	42
TACHE 11 : SUPPRIMER UNE CATEGORIE	47
TACHE 12 : AJOUT DE L'ACCES AVEC AUTHENTIFICATION	49
III. MISSION 3 : TESTER ET DOCUMENTER	58
TACHE 13 : TEST UNITAIRE SUR LA METHODE QUI RETOURNE LA DATE AU FORMAT STRING.	58
TACHE 14 : TEST INTEGRATION SUR LES REGLES DE VALIDATION.	59
TACHE 15 : TEST INTEGRATION SUR LES REPOSITORY	59
TACHE 16 : TESTS FONCTIONNELS.	60
TACHE 17 : TESTS DE COMPATIBILITES NAVIGATEURS	61
TACHE 18 : CREER LA DOCUMENTATION TECHNIQUE	61
TACHE 19 : CREATION DOC UTILISATEUR.	62
IV. MISSION 4 : DEPLOYER LE SITE GERER LE DEPLOIEMENT CONTINU	63
TACHE 20 : DEPLOYER LE SITE	63
TACHE 21 : GERER LA SAUVEGARDE ET LA RESTAURATION DE LA BDD	65
TACHE 22 : METTRE EN PLACE LE DEPLOIEMENT CONTINU	67
V. BILAN FINAL	69

Contexte :

ITS 86 est une Entreprise de Services Numériques (ESN) spécialisée dans le développement informatique, l'hébergement de site web, l'infogérance, la gestion de parc informatique et l'ingénierie système et réseau.

Elle répond régulièrement à des appels d'offres en tant que société d'infogérance et prestataire de services informatiques.

Notre projet avait pour objectif de créer un site permettant d'accéder à des vidéos d'autoformations. Pour cela, nous avons utilisé le framework Symfony ainsi que le moteur de templates Twig, tous deux écrits en PHP.

Nous avons également intégré une base de données PHPMyAdmin pour stocker les informations relatives aux vidéos et aux utilisateurs.

L'existant :

Le développement de la partie front-office a été confiée à un autre développeur. Le front permet d'accéder à l'accueil du site, de consulter les formations, les playlists. On peut également accéder à une formation ou playlist particulière. Les CGU sont également visionnables.

Langages et technologies + outils utilisés :

- Langages de programmation : PHP, HTML, CSS, JavaScript
- Framework : Symfony
- Template Engine : TWIG
- SGBDR : MySQL
- Serveur Web : MAMP
- Versionning : Git
- Gestionnaires de dépendances : composer
- IDE / Éditeur : VSCODE

I. Mission 1 : Nettoyer et optimiser le code existant

Tâche 1 : Nettoyer le code en suivant les indications de Sonarlint (ne nettoyer que les fichiers créés par le développeur, donc trier les "Action items" de Sonarlint par "Location" et s'arrêter au premier fichier dans « vendor »).

Temps estimé : 2h → Temps réel : 1h30



a)

Intitulé Sonarlint :



Problème : les chemins sont écrits en dur dans le code et à plusieurs répétitions, il est préférable donc de déclarer des constantes qui contiennent les chemins et d'appeler directement chaque constante ou il faut.

Code Avant :

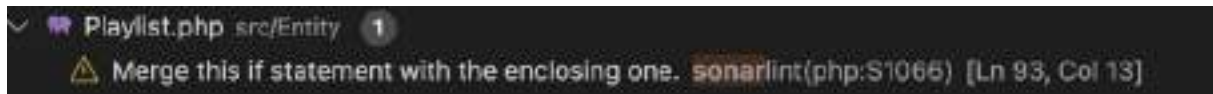
```
->select('p.id id')
->addSelect('p.name name')
->addSelect('c.name categorienome')
```

Après :

```
const P_ID = 'p.id id';
3 references
const P_NAME = 'p.name name';
3 references
const C_NAME = 'c.name categorienome';
```

b)

Intitulé Sonarlint :



Problème : Deux if sont imbriqués inutilement, il faut supprimer le deuxième if et le rajouter dans la condition d'entrée dans la boucle.

Code Avant :

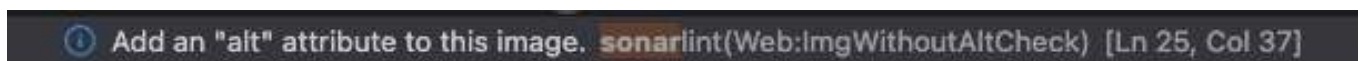
```
public function removeFormation(Formation $formation): self
{
    if ($this->formations->removeElement($formation)) {
        // set the owning side to null (unless already changed)
        if ($formation->getPlaylist() === $this) {
            $formation->setPlaylist(null);
        }
    }
}
```

Après :

```
public function removeFormation(Formation $formation): self
{
    if ($this->formations->removeElement($formation) && $formation->getPlaylist() === $this) {
        // set the owning side to null (unless already changed)
        {
            $formation->setPlaylist(null);
        }
    }
}
```

c)

Intitulé SonarLint :



Problème : Il manque un attribut alt= « » . Dans le cas où l'image ne s'afficherait pas, un texte apparaît alors à la place.

Code Avant :

```

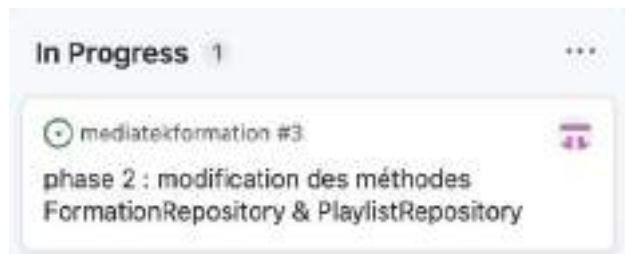
```

Après :

```
on.picture }}"alt ="Formation disponible sur Youtube"
-top" style="width:100%;height:auto;">
```

Tâche 2 : Dans le respect des bonnes pratiques de codage, en particulier SOLID (ici, le S : "Single responsibility"), modifier les méthodes de FormationRepository et PlaylistRepository qui contiennent des tests sur \$table : à chaque fois, créer 2 méthodes plutôt qu'une, pour éviter ce test. Le reste du code de l'application doit être adapté pour exploiter ces nouvelles méthodes

Temps estimé : 2h – temps réel : 3h



FormationRepository : Deux méthodes sont concernées.

```
public function findAllOrderby($champ, $ordre, $table=""): array{
    if($table==""){
        return $this->createQueryBuilder('f')
            ->orderBy('f.'.$champ, $ordre)
            ->getQuery()
            ->getResult();
    }else{
        return $this->createQueryBuilder('f')
            ->join('f.'.$table, 't')
            ->orderBy('t.'.$champ, $ordre)
            ->getQuery()
            ->getResult();
    }
}

public function findByIdContaining($champ, $valeur, $table=""): array{
    if($valeur==""){
        return $this->findAll();
    }
    if($table==""){
        return $this->createQueryBuilder('f')
            ->where('f.'.$champ, 'LIKE :valeur')
            ->orderBy('f.publishedAt', 'DESC')
            ->setParameter('valeur', '%'.$valeur.'%')
            ->getQuery()
            ->getResult();
    }else{
        return $this->createQueryBuilder('f')
            ->join('f.'.$table, 't')
            ->where('t.'.$champ, 'LIKE :valeur')
            ->orderBy('t.publishedAt', 'DESC')
            ->setParameter('valeur', '%'.$valeur.'%')
            ->getQuery()
            ->getResult();
    }
}
```

On remplace donc findAllOrderBy() par une méthode qui prend en paramètre une table et une autre qui ne prend pas de table en paramètre :

```
1 reference | 0 overrides
public function findAllOrderWithTable($champ, $ordre, $table):array{
    return $this->createQueryBuilder('f')
        ->join('f.'.$table, 't')
        ->orderBy('f.'.$champ, $ordre)
        ->getQuery()
        ->getResult();
}

1 reference | 0 overrides
public function findAllOrderWithoutTable($champ, $ordre):array{
    return $this->createQueryBuilder('f')
        ->orderBy('f.'.$champ, $ordre)
        ->getQuery()
        ->getResult();
}
```

Puis exactement la même chose avec findByContainValue :

```
1 reference | 0 overrides
public function findByContainValueWithTable($champ, $valeur, $table=""): array{
    if($valeur==""){
        return $this->findAll();
    }

    return $this->createQueryBuilder('f')
        ->join('f.'.$table, 't')
        ->where('t.'.$champ.' LIKE :valeur')
        ->orderBy('f.publishedAt', 'DESC')
        ->setParameter('valeur', '%'.$valeur.'%')
        ->getQuery()
        ->getResult();
}

1 reference | 0 overrides
public function findByContainValueWithoutTable($champ, $valeur): array{
    if($valeur==""){
        return $this->findAll();
    }
    return $this->createQueryBuilder('f')
        ->where('f.'.$champ.' LIKE :valeur')
        ->orderBy('f.publishedAt', 'DESC')
        ->setParameter('valeur', '%'.$valeur.'%')
        ->getQuery()
        ->getResult();
}
```


Il reste maintenant à modifier les deux fonctions présentes dans FormationsController, qui sont impactées par le changement du Repository :

Code Avant :

```
public function findAllContain($champ, Request $request, $table=""): Response
{
    $valeur = $request->get("recherche");
    $formations = $this->formationRepository->findByContainValue($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::MESSAGE, [
        'formations' => $formations,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}

public function sort($champ, $ordre, $table=""): Response
{
    $formations = $this->formationRepository->findAllOrderBy($champ, $ordre, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::MESSAGE, [
        'formations' => $formations,
        'categories' => $categories
    ]);
}
```

Après :

```
public function findAllContain($champ, Request $request, $table=""): Response
{
    $valeur = $request->get("recherche");
    if ($table == "") {
        $formations = $this->formationRepository->findByContainValueWithoutTable($champ, $valeur);
    }
    else
    {
        $formations = $this->formationRepository->findByContainValueWithTable($champ, $valeur, $table);
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::MESSAGE, [
        'formations' => $formations,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}

public function sort($champ, $ordre, $table=""): Response
{
    if($table == ""){
        $formations = $this->formationRepository->findAllOrderWithoutTable($champ, $ordre);
    }
    else
    {
        $formations = $this->formationRepository->findAllOrderWithTable($champ, $ordre, $table);
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::MESSAGE, [
        'formations' => $formations,
        'categories' => $categories
    ]);
}
```


On suit la même logique pour PlaylistRepository qui contient une méthode qui effectue un test sur \$table :

```
public function findByContainValue($champ, $valeur, $table=""): array{
    if($valeur==""){
        return $this->findAllOrderBy('name', 'ASC');
    }
    if($table==""){
        return $this->createQueryBuilder('p')
            ->select(self::P_ID)
            ->addSelect(self::P_NAME)
            ->addSelect(self::C_NAME)
            ->leftjoin('p. formations', 'f')
            ->leftjoin('f. categories', 'c')
            ->where('p.' . $champ . ' LIKE :valeur')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->groupBy('p.id')
            ->addGroupBy('c.name')
            ->orderBy('p.name', 'ASC')
            ->addOrderBy('c.name')
            ->getQuery()
            ->getResult();
    }else{
        return $this->createQueryBuilder('p')
            ->select(self::P_ID)
            ->addSelect(self::P_NAME)
            ->addSelect(self::C_NAME)
            ->leftjoin('p. formations', 'f')
            ->leftjoin('f. categories', 'c')
            ->where('c.' . $champ . ' LIKE :valeur')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->groupBy('p.id')
            ->addGroupBy('c.name')
            ->orderBy('p.name', 'ASC')
            ->addOrderBy('c.name')
            ->getQuery()
            ->getResult();
    }
}
```

On écrit une méthode qui effectue une recherche dans la table « playlist », et une autre méthode qui fait la même chose dans une table « catégories » :

```
public function findByPlaylistValue($champ, $valeur): array{
    if($valeur==""){
        return $this->findAllOrderBy('name', 'ASC');
    }
    return $this->createQueryBuilder('p')
        ->select(self::P_ID)
        ->addSelect(self::P_NAME)
        ->addSelect(self::C_NAME)
        ->leftjoin('p.formations', 'f')
        ->leftjoin('f.categories', 'c')
        ->where('p.'.$champ.' LIKE :valeur')
        ->setParameter('valeur', '%'.$valeur.'%')
        ->groupBy('p.id')
        ->addGroupBy('c.name')
        ->orderBy('p.name', 'ASC')
        ->addOrderBy('c.name')
        ->getQuery()
        ->getResult();
}
```

0 references | 0 overrides

```
public function findByCategorieValue($champ, $valeur): array{
    return $this->createQueryBuilder('p')
        ->select([self::P_ID])
        ->addSelect(self::P_NAME)
        ->addSelect(self::C_NAME)
        ->leftjoin('p.formations', 'f')
        ->leftjoin('f.categories', 'c')
        ->where('c.'.$champ.' LIKE :valeur')
        ->setParameter('valeur', '%'.$valeur.'%')
        ->groupBy('p.id')
        ->addGroupBy('c.name')
        ->orderBy('p.name', 'ASC')
        ->addOrderBy('c.name')
        ->getQuery()
        ->getResult();
}
```

Dans PlaylistsController, la méthode précédente était appelée :

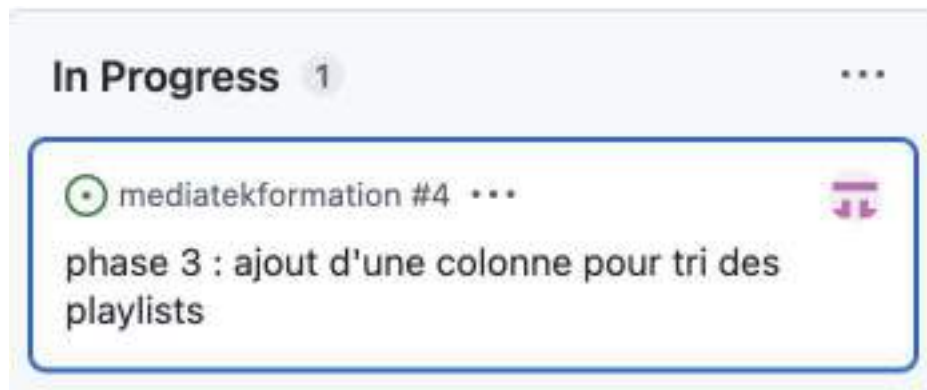
```
2 references | 0 overrides
public function findAllContain($champ, Request $request, $table=""): Response{
    $valeur = $request->get("recherche");
    $playlists = $this->playlistRepository->findByContainValue($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::PLAYLIST, [
        'playlists' => $playlists,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}
```

On incorpore donc nos deux nouvelles méthodes qui viennent d’être créées en fonction de la valeur de \$table :

```
public function findAllContain($champ, Request $request, $table=""): Response{
    $valeur = $request->get("recherche");
    if ($table == "") {
        $playlists = $this->playlistRepository->findByPlaylistValue($champ, $valeur);
    } else {
        $playlists = $this->playlistRepository->findByCategorieValue($champ, $valeur);
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::PLAYLIST, [
        'playlists' => $playlists,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}
```

Tâche 3 : Dans la page des playlists, ajouter une colonne pour afficher le nombre de formations par playlist et permettre le tri croissant et décroissant sur cette colonne. Cette information doit aussi s'afficher dans la page d'une playlist.

Temps estimé : 2h – temps réel : 5h



On commence par ajouter une fonction dans l'entity Playlist afin de récupérer les catégories et pouvoir alléger les requêtes du repository et l'affichage du twig. Cette fonction nous permet de récupérer une collection contenant les noms des catégories :

```
/*  
 * Summary of getCategoriesPlaylist  
 * @return Collection  
 */  
1 reference | 1 override  
public function getCategoriesPlaylist(): Collection  
{  
    $categories = new ArrayCollection();  
    foreach($this->formations as $formation){  
        $categoriesFormation = $formation->getCategories();  
        foreach ($categoriesFormation as $categorieFormation){  
            if(!$categories->contains($categorieFormation->getName())){  
                $categories[] = $categorieFormation->getName();  
            }  
        }  
    }  
    return $categories;  
}
```


On modifie ensuite findAllOrderBy dans PlaylistRepository (pour plus de lisibilité) en deux fonctions :

```
/**
 * Retourne toutes les playlists triées sur un champ
 * @param type $champ
 * @param type $ordre
 * @return Playlist[]
 */
public function findAllOrderBy($champ, $ordre): array{
    return $this->createQueryBuilder('p')
        ->select(self::P_ID)
        ->addSelect(self::P_NAME)
        ->addSelect(self::C_NAME)
        ->leftjoin('p.formations', 'f')
        ->leftjoin('f.categories', 'c')
        ->groupBy('p.id')
        ->addGroupBy('c.name')
        ->orderBy('p.'.$champ, $ordre)
        ->addOrderBy('c.name')
        ->getQuery()
        ->getResult();
}
```

```
/**
 * Summary of findAllOrderByName
 * @param mixed $ordre
 * @return array
 */
2 references | 0 overrides
public function findAllOrderByName($ordre) : array
{
    return $this->createQueryBuilder('p')
        ->leftjoin('p.formations', 'f')
        ->groupBy('p.id')
        ->orderBy('p.name', $ordre)
        ->getQuery()
        ->getResult();
}
```

```
/**
 * Summary of findAllOrderByNbFormations
 * @param mixed $ordre
 * @return array
 */
0 references | 0 overrides
public function findAllOrderByNbFormations($ordre) : array{
    return $this->createQueryBuilder('p')
        ->leftjoin('p.formations', 'f')
        ->groupBy('p.id')
        ->orderBy('count(f.title)', $ordre)
        ->getQuery()
        ->getResult();
}
```

On a donc une nouvelle fonction qui retourne les playlists triées en fonction de leurs noms et une autre en fonction du nombre de formations présentes dans la playlist. On enlève le select pour pouvoir accéder à tous les champs de l'entity (et non se restreindre sur certains).

Il faut maintenant modifier en conséquence la méthode sort du contrôleur pour gérer les deux nouveaux cas de tri : soit via le nom, soit via le nombre de formations :

```
2 references | 0 overrides
public function index(): Response{
    $playlists = $this->playlistRepository->findAllOrderByName('ASC');
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::PLAYLIST, [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}
```

```
/**
 * @Route("/playlists/tri/{champ}/{ordre}", name="playlists:sort")
 * @param Type $champ
 * @param Type $ordre
 * @return Response
 */
2 references | 0 overrides
public function sort($champ, $ordre): Response{
    switch($champ){
        case "name":
            $playlists = $this->playlistRepository->findAllOrderByName($ordre);
            break;
        case "nbformations":
            $playlists = $this->playlistRepository->findAllOrderByNbFormations($ordre);
            break;
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/playlists.html.twig", [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}
```

On a ensuite pour la page des playlists, un affichage beaucoup plus propre dans le twig :

```
{% for k in 0..playlists|length-1 %}
<tr class="align-middle">
<td>
<div class="text-info">
{{ playlists[k].name }}
</div>
</td>
<td class="text-left">
{% set categories = playlists[k].categoriesplaylist %}
{% if categories|length > 0 %}
{% for c in 0..categories|length-1 %}
<div class="text-left">
{{ categories[c] }}
</div>
{% endfor %}
{% endif %}
</td>
<td>
{{ playlists[k].formations|length }}
</td>
<td class="text-center">
<a href="{{ path('playlists.show', {'id': playlists[k].id}) }}" class="btn btn-secondary">Voir détail</a>
</td>
</tr>
{% endfor %}
```

Il y a par exemple 74 formations/vidéos dans la playlist « Bases de la programmation (C#) et ainsi de suite :

Bases de la programmation (C#)	C# POO	74	Voir détail
Compléments Android (programmation mobile)	Android	15	Voir détail
Cours Composant logiciel	Cours	2	Voir détail

Pour le tri, via un champ, il faut ajouter la fonction `findAllOrderByName` dans les deux méthodes précédemment créées dans la tâche 2 afin de retourner tous les enregistrements dans le cas où \$valeur est vide.


```

* Enregistrements dont un champ contient une valeur
* ou tous les enregistrements si la valeur est vide
* @param type $champ
* @param type $valeur
* @return Playlist[]
*/
1 reference | 0 overrides
public function findByPlaylistValue($champ, $valeur): array{
    if($valeur==""){
        return $this->findAllOrderByName('ASC');
    }
    return $this->createQueryBuilder('p')
        ->leftjoin('p.formations', 'f')
        ->where('p.'.$champ.' LIKE :valeur')
        ->setParameter('valeur', '%'.$valeur.'%')
        ->groupBy('p.id')
        ->orderBy('p.name', 'ASC')
        ->getQuery()
        ->getResult();
}

```

```

1 reference | 0 overrides
public function findByCategorieValue($champ, $valeur): array{
    if($valeur==""){
        return $this->findAllOrderByName('ASC');
    }
    return $this->createQueryBuilder('p')
        ->leftjoin('p.formations', 'f')
        ->leftjoin('f.categories', 'c')
        ->where('c.'.$champ.' LIKE :valeur')
        ->setParameter('valeur', '%'.$valeur.'%')
        ->groupBy('p.id')
        ->orderBy('p.name', 'ASC')
        ->getQuery()
        ->getResult();
}

```

Et même chose pour la méthode index du controlleur :

```
2 references | 0 overrides  
public function index(): Response{  
    $playlists = $this->playlistRepository->findAllOrderByNane('ASC');  
    $categories = $this->categorieRepository->findAll();  
    return $this->render(self::PLAYLIST, [  
        'playlists' => $playlists,  
        'categories' => $categories  
    ]);  
}
```

On ajoute les deux boutons de tri sur le nombre de formations pour finaliser notre nouvelle colonne, toujours dans playlists.html.twig :

```
<div class="text-left" scope="col">  
    <button type="button" class="btn btn-info btn-sm active" value="button" aria-pressed="true"></button>  
    <button type="button" class="btn btn-info btn-sm" value="button" aria-pressed="false"></button>  
</div>
```

On ajoute bien le champ « nbformations » et non pas le « name » :

	<	>
74		
19		
18		
18		
13		
11		
10		
8		
8		
8		
6		
6		

Dernière petite chose, on doit également ajouter, dans la page d'UNE playlist, l'information sur le nombre de formations présentent dans cette playlist, on ajoute dans playlist.html.twig :

```
<strong>nombre de formations : </strong>
{{ playlist.formations|length }}
<br /><br />
```

Et donc :



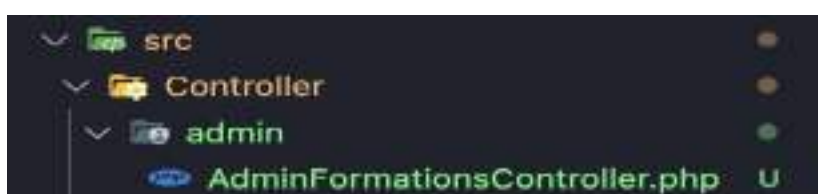
II. Mission 2 : Coder la partie Back-Office

Tâche 4 : Modification d'une formation

Temps estimé : 2h → Temps réel : 4h



On commence donc la partie back-office (admin du site) en créant un nouveau dossier admin dans src/Controller puis à l'intérieur de celui-ci : AdminFormationsController :



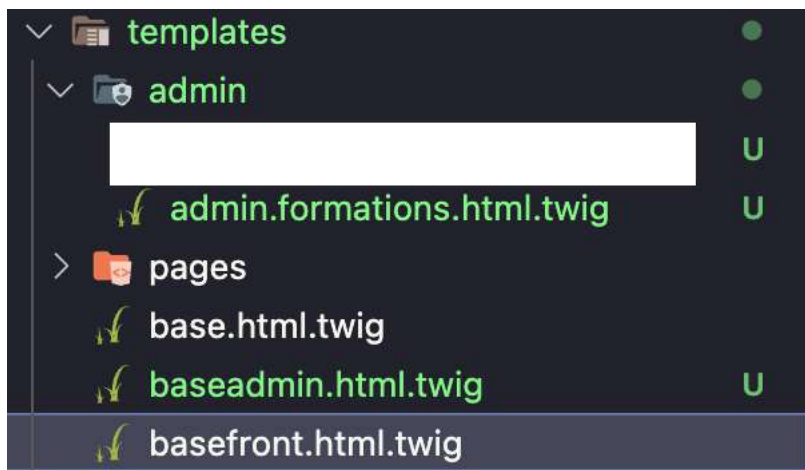
Le contenu est similaire à celui de FormationsController, excepté les routes des méthodes qui doivent accéder à la racine du dossier « admin » ainsi qu'à la future page que nous allons créer par la suite admin.formations.html.twig :

```
const ROUTE_ADM = "admin/admin.formations.html.twig";
```

Exemple avec la méthode index :

```
2 references | 0 overrides
public function index(): Response
{
    $formations = $this->formationRepository->findAll();
    $categories = $this->categorieRepository->findAll();
    return $this->render(self::ROUTE_ADM, [
        'formations' => $formations,
        'categories' => $categories
    ]);
}
```

Dans la même logique que AdminFormationsController, on crée donc admin.formations.html.twig dans un dossier « admin » dans « templates » et également un fichier baseadmin.html.twig à la racine de « template » de la même manière que le « basefront.html.twig » mais nous ne voulons pas du menu et du titre du côté admin. :



On récupère le contenu de basefront.html.twig dans baseadmin.html.twig en ajoutant juste un titre pour la gestion des formations.

Ensuite dans admin.voyages.html on intègre le baseadmin.html.twig. Et comme nous voulons la même présentation (boutons de tris et de filtres) que dans la partie front (voyages.html.twig), on copie le contenu de ce dernier dans admin.voyages.html.twig en ajoutant deux boutons (éditer et supprimer) :

```
baseadmin.html.twig U X
templates > baseadmin.html.twig
1 {% extends "base.html.twig" %}
2
3 {% block title %}{% endblock %}
4 {% block stylesheets %}{% endblock %}
5 {% block top %}
6     <div class="container">
7
8         <div class="text-center">
9             <h1>Gestion des formations</h1>
10        </div>
11    </div>
12 {% endblock %}
13 {% block body %}
14
15 {% endblock %}
16 {% block footer %}
17
18 {% endblock %}
19 {% block javascripts %}{% endblock %}
```

Dans admin.formations.sort, on change également les routes pour chaque tri, par exemple :

```
<a href="{% path('admin.formations.sort',
<a href="{% path('admin.formations.sort',
```

(auparavant : formations.sort)

formation	playlist	management	status	actions
Eclipse n°8 : Déploiement	Eclipse et Java	Java	04/01/2021	
Eclipse n°7 : Tests unitaires	Eclipse et Java	Java	04/01/2021	
Eclipse n°6 : Documentation testbeaps	Eclipse et Java	Java	04/01/2021	

On veut maintenant que le bouton « edit » nous emmène sur une page, où est présent un formulaire avec les informations d’une formation, que l’on pourra modifier.

On ajoute dans le dossier template/admin le fichier ‘admin.formation.edit.html.twig’.

Puis dans AdminFormationController, on ajoute notre fonction edit en précisant la route du fichier que l’on vient de créer. On récupère en paramètre un objet de type Formation puis on envoie \$formation à la vue par rapport à ‘formation’ reçu par la route.


```

/**
 * @Route("/admin/edit/{id}", name="admin.formation.edit")
 * @param Formation $formation
 * @return Response
 *
 */

public function edit(Formation $formation): Response{

    return $this->render("admin/admin.formation.edit.html.twig", [
        'formation' => $formation,
    ]);
}

```

Dans admin.formations.html.twig on complète le bouton « Editer » précédemment ajouté :

```

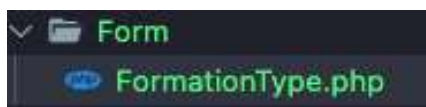
<a href="{{ path('admin.formation.edit', {id:formation.id}) }}" class="btn btn-secondary">Editer</a>

```

Maintenant, on veut créer un formulaire qui affiche les informations d'une formation et pouvoir les modifier lorsqu'on appuie sur le bouton éditer.

Dans la console, dans le dossier du projet on tape la commande :

- php bin/console make:form
- on lui précise le nom de la classform : FormationType
- puis le nom de l'entity : Formation (car on souhaite le formulaire avec les propriétés de Formation.php)
- un nouveau fichier FormationType est créé avec une méthode buildForm qui construit le formulaire avec les champs de l'entity, et une méthode configureOptions qui gère les options.



J'ai également ajouté des paramètres à certains champs :

- on doit afficher une liste de playlist et pouvoir en sélectionner UNE seule
- même chose pour catégorie, cependant on peut sélectionner plusieurs catégories pour une formation
- la date doit être sélectionnée, et non pas saisie, et ne peut pas être postérieure à la date du jour
- tous les champs sont obligatoires excepté 'description' et 'catégorie'.

Voici donc le contenu de FormationType avec les modifications :

```
FormationType.php U X
app > Form > FormationType.php > PHP > AppFormFormationType > buildForm
15 use Symfony\Component\Validator\Constraints\LessThanOrEqual;
16
17 // namespace / implementations
18 class FormationType extends AbstractType
19 {
20     // interface / constraints / prototype
21     public function buildForm(FormBuilderInterface $builder, array $options): void
22     {
23         $builder
24             =>add('title', null, [
25                 'label' => 'Titre de la formation',
26                 'required' => true,
27             ])
28             =>add('playlist', EntityType::class, [
29                 'class' => Playlist::class,
30                 'choice_label' => 'name',
31                 'multiple' => false,
32                 'required' => true,
33             ])
34             =>add('categories', EntityType::class, [
35                 'class' => Categories::class,
36                 'choice_label' => 'name',
37                 'multiple' => true,
38                 'required' => false,
39             ])
40             =>add('publishedAt', DateType::class, [
41                 'label' => 'Date',
42                 'constraints' => [
43                     new LessThanOrEqual('today'),
44                 ],
45                 'required' => true,
46             ])
47             =>add('description', null, [
48                 'required' => false,
49             ])
50             =>add('videoId', null, [
51                 'label' => 'Lien de la formation',
52                 'required' => true,
53             ])
54             =>add('submit', SubmitType::class, [
55                 'label' => 'Enregistrer'
56             ])
57         }
58     }
59
60     // interface / constraints / prototype
61     public function configureOptions(OptionsResolver $resolver): void
62     {
63         $resolver->setDefaults([
64             'data_class' => Formation::class,
65         ]);
66     }
67 }
```


De retour dans la méthode « edit » , il faut construire le formulaire dans le contrôleur et l'envoyer à la vue :

```
public function edit(Formation $formation): Response{
    $formFormation = $this->createForm(FormationType::class, $formation);

    return $this->render("admin/admin.formation.edit.html.twig", [
        'formation' => $formation,
        'formformation' => $formFormation->createView()
    ]);
}
```

La 1^{ère} ligne ajoutée crée un objet qui contient les informations du formulaire FormationType.

La 2^{ème} ligne envoie l'objet \$formFormation à la vue.

Dans admin.formation.edit.html.twig on permet l'affichage du formulaire :

```
1 {% extends "baseadmin.html.twig" %}
2
3 {% block body %}
4     <h2>Détail Formation : </h2>
5     {{ form_start(formformation) }}
6
7     {{ form_end(formformation) }}
8 {% endblock %}
```

Lorsqu'un changement est fait, il doit être enregistré dans la BDD, on termine donc la méthode edit du contrôleur. Un second paramètre apparaît si on appuie sur le bouton Enregistrer du formulaire.

Si un formulaire a donc été soumis puis validé, il appelle la méthode add() du FormationRepository et les changements sont enregistrés dans le formulaire :

```
/**
 * Route("/admin/edit/{id}", name="admin.formation.edit")
 * @param Formation $formation
 * @param Request $request
 * @return Response
 */
public function edit(Formation $formation, Request $request): Response{
    $formFormation = $this->createForm(FormationType::class, $formation);

    $formFormation->handleRequest($request);
    if($formFormation->isSubmitted() && $formFormation->isValid()){
        $this->formationRepository->add($formation, true);
        return $this->redirectToRoute('admin.formation');
    }

    return $this->render("admin/admin.formation.edit.html.twig", [
        'formation' => $formation,
        'formformation' => $formFormation->createView()
    ]);
}
```

Avant de voir à quoi ressemble le formulaire, Symfony a des modèles bootstrap permettant d'améliorer la présentation du formulaire. On ajoute dans le fichier config/packages/twig.yaml un paramètre avec le modèle SymfonyForm :

```
1 twig:
2   default_path: '%kernel.project_dir%/templates'
3   form_themes: ['bootstrap_5_layout.html.twig']
4
5 when@test:
6   twig:
7     strict_variables: true
8
```

Nous avons maintenant un formulaire avec une présentation convenable et les champs préremplis pour une modification d'une formation :

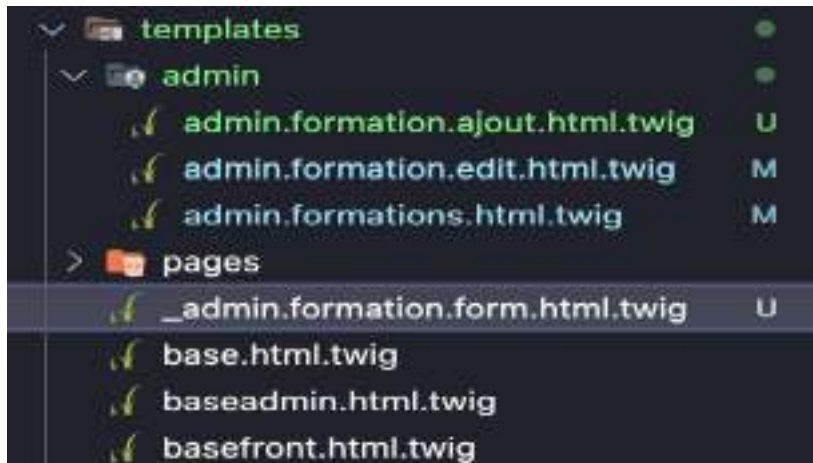


Tâche 5 : Ajout d'une formation



Temps estimé : 2h – temps réel : 1h

Nous créons un nouveau fichier « twigadmin.formation.ajout.html.twig » dans le dossier templates/admin. Comme le contenu sera similaire à celui de la modification d'une formation (excepté que certains champs seront vides). On ajoute en racine du template un fichier qui contiendra la partie formulaire : « _admin.formation.form.html.twig ». Ce dernier sera envoyé ensuite à admin.formation.ajout.html et admin.formation.edit.html :



Dans celui-ci on envoie donc le contenu du formulaire qui était présent dans admin.formation.edit.html.twig :



admin.formation.edit.html.twig contiendra désormais :



Et pareil pour admin.formation.ajout.html.twig.

Maintenant nous créons la nouvelle méthode « ajout » dans AdminFormationController.php. Elle est quasiment identique à la méthode edit mais elle ne prend pas en paramètre \$formation, car l'objectif est justement d'en créer une nouvelle. Il faut également changer sa route :

```

/**
 * @Route("/admin/ajout", name="admin.formation.ajout")
 * @param Request $request
 * @return Response
 */
2 references (0 overrides)
public function ajout(Request $request): Response{
    $formation = new Formation();
    $formFormation = $this->createForm(FormationType::class, $formation);

    $formFormation->handleRequest($request);
    if($formFormation->isSubmitted() && $formFormation->isValid()){
        $this->formationRepository->add($formation, true);
        return $this->redirectToRoute('admin.formations');
    }

    return $this->render("admin/admin.formation.ajout.html.twig", [
        'formation' => $formation,
        'formformation' => $formFormation->createView()
    ]);
}

```

Dans admin.formations.html.twig on ajoute maintenant un bouton pour l'ajout d'une nouvelle visite dans lequel on envoie la vue admin.formation.ajout :

```

{# admin.formations.html.twig M X
templates > admin > {# admin.formations.html.twig
1  {% extends "baseadmin.html.twig" %}
2  {% block body %}
3      <p class="text-end">
4          <a href="{{ path('admin.formation.ajout') }}" class="btn btn-primary">
5              Ajouter une nouvelle formation
6          </a>
7      </p>

```

Gestion des formations					
Ajouter une nouvelle formation					
formation	playlist	catégories	date	actions	
Eclipse n°6 : Déploiement	Eclipse et Java	java	04/01/2021		Editer Supprimer
Eclipse n°7 : Tests unitaires	Eclipse et Java	java	03/01/2021		Editer Supprimer
Eclipse n°8 : Documentation technique	Eclipse et Java	java	11/12/2020		Editer Supprimer

Petit changement dans FormationType (au niveau des champs du formulaire), on initialise la date, à la date du jour (car on part du principe que la date du jour est la même que la date d'ajout d'une nouvelle formation) :

```
->add('publishedAt', DataType::class, [
    'label' => 'Date',
    'constraints' => [
        new LessThanOrEqual(['today'],
    ],
    'format' => 'ddMMyyyy',
    'data' => isset($options['data']) ? $options['data']->getPublishedAt() : new DateTime('now'),
    'required' => true
])
```

Voilà à quoi ressemble la page d'ajout d'une formation lorsqu'on appuie sur le bouton : « Ajouter une nouvelle formation » :

The screenshot shows a web form titled "Gestion des formations". Below the title is a section "Nouvelle Formation :". The form contains several input fields: "Titre de la formation" (text input), "Playlist" (text input with "Eclipse et Java" selected), "Categories" (list box with "Java", "UML", "CS", "Python" visible), "Date" (date picker set to 22/02/2023), "Description" (text area), and "Lien de la formation" (text input). A blue "Créer" button is at the bottom left of the form area.

Tâche 6 : Supprimer une formation

The screenshot shows a task card in a Kanban board. The board has a header "In Progress" with a count of "1". The task card is titled "mediatekformation #7" and has a subtitle "phase 6 : suppression d'une formation". There is a green circular icon on the left and a purple icon on the right of the card.

Temps estimé : 2h → Temps réel : 4h

Pour supprimer une formation c'est légèrement plus simple. Dans FormationRepository nous avons déjà une méthode remove, qui permet de supprimer une ligne de la table Formation :

```
0 references | 0 overrides
public function remove(Formation $entity, bool $flush = false): void
{
    $this->getEntityManager()->remove($entity);

    if ($flush) {
        $this->getEntityManager()->flush();
    }
}
```

On ajoute maintenant une nouvelle méthode dans AdminFormationsController.php :

```
/**
 * @Route ("/admin/suppr/{id}", name="admin.formation.suppr")
 * @param Formation $formation
 * @return Response
 */
4 references | 0 overrides
public function suppr(Formation $formation) : Response{
    $this->formationRepository->remove($formation, true);
    return $this->redirectToRoute('admin. formations');
}
```

Il faut appeler la méthode remove de formationRepository avec en paramètre l'objet de type Formation, et « true » en second pour la suppression en BDD : RedirectToRoute permet juste de rediriger sur la même page.

Dans « admin. formations.html.twig », on appelle cette nouvelle méthode au niveau du bouton supprimer :

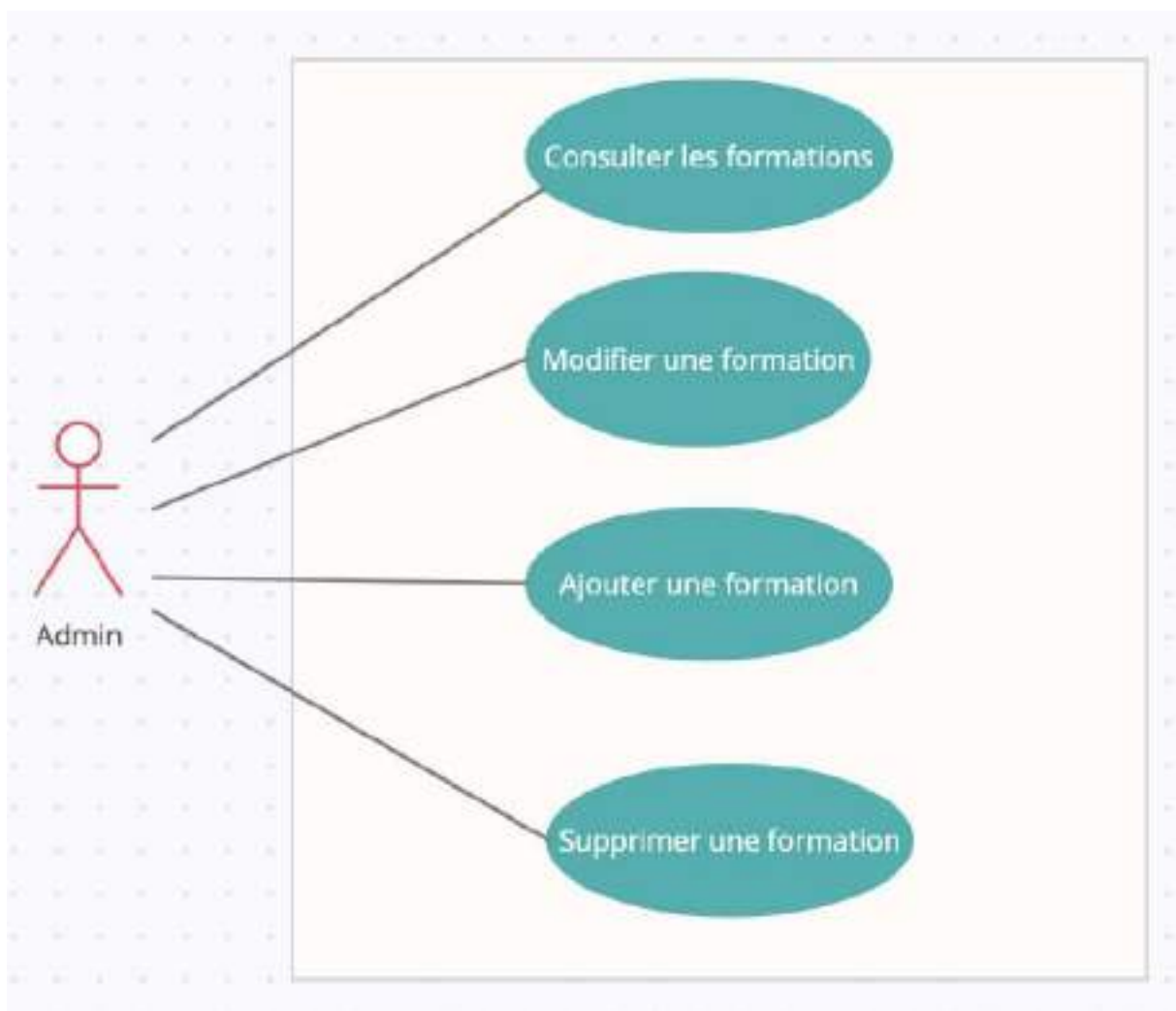
```
<a href="{% path('admin.formation.suppr', {id:formation.id}) %}" class="btn btn-danger" >Supprimer</a>
```

Pour finir, on ajoute aussi un onclick sur le bouton supprimé pour confirmer la suppression (et éviter des suppressions involontaires) :

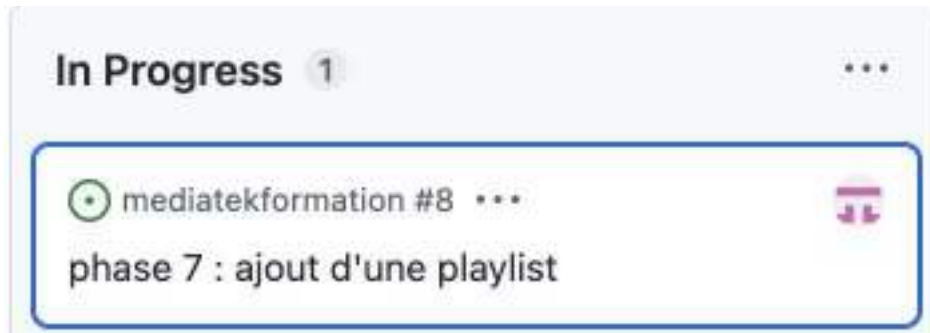
```
class="btn btn-danger" onclick="return confirm('Etes-vous sur de vouloir supprimer {{ formation.title }}?')">Supprimer</a>
```



Nous avons donc créé un back-office avec une partie admin qui peut gérer les formations. On peut illustrer la gestion de ces formations avec un diagramme de cas d'utilisation :



Tâche 7 : Ajout d'une playlist



Temps estimé : 2h – Temps réel : 2h

Nous voulons une deuxième page qui affiche la liste des playlists, puis par la suite une page qui affiche les catégories. Donc dans le baseadmin.html.twig, je vais copier le contenu du basefront.html.twig en remplaçant dans la navbar, « Accueil », « Formations », « Playlists » par « Formations », « Playlists », « Catégories ». Catégories ne contient aucune route, pour le moment, on s'en occupera plus tard. J'ai également ajouté la bannière du front que j'avais oubliée :

```
baseadmin.html.twig M x
templates > / baseadmin.html.twig
1  {% extends "base.html.twig" %}
2
3  {% block title %}{% endblock %}
4  {% block stylesheets %}{% endblock %}
5  {% block top %}
6      <div class="container">
7          <!-- titre -->
8          <div class="text-left">
9              
10          </div>
11          <!-- menu -->
12          <nav class="navbar navbar-expand-lg navbar-light bg-light">
13              <div class="collapse navbar-collapse" id="navbarSupportedContent">
14                  <ul class="navbar-nav mr-auto">
15                      <li class="nav-item">
16                          <a class="nav-link" href="{{ path('admin. formations') }}">Formations</a>
17                      </li>
18                      <li class="nav-item">
19                          <a class="nav-link" href="{{ path('admin-playlists') }}">Playlists</a>
20                      </li>
21                      <li class="nav-item">
22                          <a class="nav-link" href="">Catégories</a>
23                      </li>
24                  </ul>
25              </div>
26          </nav>
27      </div>
28  {% endblock %}
29  {% block body %}
30
31  {% endblock %}
32  {% block footer %}
33
34  {% endblock %}
35  {% block javascripts %}{% endblock %}
```



Maintenant on veut créer notre nouvelle page qui affiche la liste des playlists à la manière du front. On va d'abord créer un nouveau controller dans le dossier src/controller/admin que l'on va nommer AdminPlaylistsController.

Comme pour AdminFormationsController, on copie l'intégralité de PlaylistsController en changeant les routes des méthodes pour les « orientées » vers la nouvelle page « admin.playlists.html.twig » :



Quelques exemples de changement de routes dans AdminPlaylistsController :

```

AdminPlaylistsController.php 5 0 X
src > Controller > admin > AdminPlaylistsController.php > PHP > AppController
1  <?php
2  namespace App\Controller\Admin;
3
4  use App\Entity\Playlist;
5  use App\Form\PlaylistType;
6  use App\Repository\CategoryRepository;
7  use App\Repository\FormationRepository;
8  use App\Repository\PlaylistRepository;
9  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
10 use Symfony\Component\HttpFoundation\Request;
11 use Symfony\Component\HttpFoundation\Response;
12 use Symfony\Component\Routing\Annotation\Route;
13
14 /**
15  * Description of PlaylistsController
16  *
17  * @author ande
18  */
19
20 // ...
21
22 class AdminPlaylistsController extends AbstractController {
23
24     const PLAYLIST = "admin/admin.playlists.html.twig";

```

```

    = @route("/admin/playlists", name="admin.playlists")
    = @return Response
  end

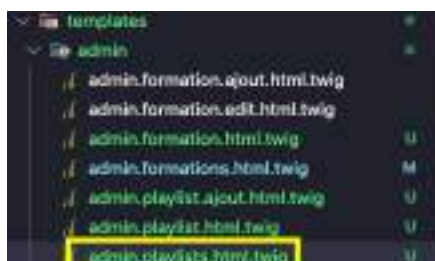
  def index
    public function index(): Response{
      $playlists = $this->playlistRepository->findAllOrderByName('ASC');
      $categories = $this->categoryRepository->findAll();
      return $this->render('admin/admin.playlists.html.twig', [
        'playlists' => $playlists,
        'categories' => $categories
      ]);
    }
  }

  = @route("/admin/playlists/{id}/{champ}/{ordre}", name="admin.playlists.sort")
  = @param type $champ
  = @param type $ordre
  = @return Response
  end

  def sort($champ, $ordre): Response{
    switch($champ){
      case "name":
        $playlists = $this->playlistRepository->findAllOrderByName($ordre);
        break;
      case "nbformations":
        $playlists = $this->playlistRepository->findAllOrderByNbformations($ordre);
        break;
    }
    $categories = $this->categoryRepository->findAll();
    return $this->render('admin/admin.playlists.html.twig', [
      'playlists' => $playlists,
      'categories' => $categories
    ]);
  }
}

```

Maintenant on ajoute « admin.playlists.html.twig » pour l’affichage des playlists dans templates/admin :



Dans ce même fichier, on copie l’intégralité de la partie front « playlists.html.twig » en faisant « extends » baseadmin.html.twig et non pas le basefront.html.twig. On change également les routes des tris et filtres pour que cela corresponde aux méthodes du « AdminPlaylistsController ». Voici quelques changements de routes (c’est le même principe partout).

De plus, on ajoute un nouveau bouton qui permettra par la suite d'ajouter une nouvelle playlist à la suite grâce au futur formulaire `admin.playlist.ajout.html.twig` :

```

1  <div class="container">
2      <div class="text-center">
3          <h2>Gestion des playlists</h2>
4      </div>
5      <div class="text-center">
6          <a href="{{ path('admin.playlist.ajout') }}" class="btn btn-primary">
7              Ajouter une nouvelle playlist
8          </a>
9      </div>
10     <table class="table">
11         <thead>
12             <tr>
13                 <th>Nom</th>
14                 <th>Catégorie</th>
15                 <th>Nombre de formations</th>
16                 <th>Actions</th>
17             </tr>
18         </thead>
19         <tbody>
20             <tr>
21                 <td>Python 3</td>
22                 <td>Python</td>
23                 <td>10</td>
24                 <td>
25                     <a href="{{ path('admin.playlist.edit', {'id': 1}) }}" class="btn btn-info">Modifier</a>
26                     <a href="{{ path('admin.playlist.delete', {'id': 1}) }}" class="btn btn-danger">Supprimer</a>
27                 </td>
28             </tr>
29             <tr>
30                 <td>Java 8</td>
31                 <td>Java</td>
32                 <td>5</td>
33                 <td>
34                     <a href="{{ path('admin.playlist.edit', {'id': 2}) }}" class="btn btn-info">Modifier</a>
35                     <a href="{{ path('admin.playlist.delete', {'id': 2}) }}" class="btn btn-danger">Supprimer</a>
36                 </td>
37             </tr>
38             <tr>
39                 <td>PHP 7</td>
40                 <td>PHP</td>
41                 <td>3</td>
42                 <td>
43                     <a href="{{ path('admin.playlist.edit', {'id': 3}) }}" class="btn btn-info">Modifier</a>
44                     <a href="{{ path('admin.playlist.delete', {'id': 3}) }}" class="btn btn-danger">Supprimer</a>
45                 </td>
46             </tr>
47             <tr>
48                 <td>JavaScript</td>
49                 <td>JavaScript</td>
50                 <td>8</td>
51                 <td>
52                     <a href="{{ path('admin.playlist.edit', {'id': 4}) }}" class="btn btn-info">Modifier</a>
53                     <a href="{{ path('admin.playlist.delete', {'id': 4}) }}" class="btn btn-danger">Supprimer</a>
54                 </td>
55             </tr>
56             <tr>
57                 <td>React</td>
58                 <td>React</td>
59                 <td>2</td>
60                 <td>
61                     <a href="{{ path('admin.playlist.edit', {'id': 5}) }}" class="btn btn-info">Modifier</a>
62                     <a href="{{ path('admin.playlist.delete', {'id': 5}) }}" class="btn btn-danger">Supprimer</a>
63                 </td>
64             </tr>
65             <tr>
66                 <td>Angular</td>
67                 <td>Angular</td>
68                 <td>1</td>
69                 <td>
70                     <a href="{{ path('admin.playlist.edit', {'id': 6}) }}" class="btn btn-info">Modifier</a>
71                     <a href="{{ path('admin.playlist.delete', {'id': 6}) }}" class="btn btn-danger">Supprimer</a>
72                 </td>
73             </tr>
74             <tr>
75                 <td>Vue.js</td>
76                 <td>Vue.js</td>
77                 <td>1</td>
78                 <td>
79                     <a href="{{ path('admin.playlist.edit', {'id': 7}) }}" class="btn btn-info">Modifier</a>
80                     <a href="{{ path('admin.playlist.delete', {'id': 7}) }}" class="btn btn-danger">Supprimer</a>
81                 </td>
82             </tr>
83             <tr>
84                 <td>Node.js</td>
85                 <td>Node.js</td>
86                 <td>1</td>
87                 <td>
88                     <a href="{{ path('admin.playlist.edit', {'id': 8}) }}" class="btn btn-info">Modifier</a>
89                     <a href="{{ path('admin.playlist.delete', {'id': 8}) }}" class="btn btn-danger">Supprimer</a>
90                 </td>
91             </tr>
92             <tr>
93                 <td>Docker</td>
94                 <td>Docker</td>
95                 <td>1</td>
96                 <td>
97                     <a href="{{ path('admin.playlist.edit', {'id': 9}) }}" class="btn btn-info">Modifier</a>
98                     <a href="{{ path('admin.playlist.delete', {'id': 9}) }}" class="btn btn-danger">Supprimer</a>
99                 </td>
100            </tr>
101            <tr>
102                <td>Kubernetes</td>
103                <td>Kubernetes</td>
104                <td>1</td>
105                <td>
106                    <a href="{{ path('admin.playlist.edit', {'id': 10}) }}" class="btn btn-info">Modifier</a>
107                    <a href="{{ path('admin.playlist.delete', {'id': 10}) }}" class="btn btn-danger">Supprimer</a>
108                </td>
109            </tr>
110        </tbody>
111    </table>
112 </div>

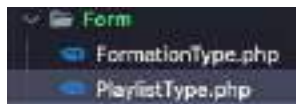
```

La page des playlists s'affiche maintenant correctement dans la partie admin, les tris et les filtres fonctionnent aussi, et le tout avec les bonnes routes et la « nouvelle » navbar.

Voici un exemple de l'affichage des playlists dans la partie admin avec un tri effectué sur le nombre de formations dans une playlist, du plus grand nombre au plus petit :



Maintenant on crée un nouveau formulaire pour pouvoir ajouter une nouvelle playlist. Dans la console, au niveau du dossier du projet on tape : `php bin/console make:form`, on lui donne le nom « PlaylistForm » et on se base sur l'entity Playlist. Le nouveau formulaire est créé avec deux champs : name et description :



```

PlaylistType.php:14
src/Form/PlaylistType.php:14:40 > App\Form\PlaylistType > buildForm
1
2
3
4 namespace App\Form;
5
6 use App\Entity\Playlist;
7 use Symfony\Component\Form\AbstractType;
8 use Symfony\Component\Form\Extension\Core\Type\SubmitType;
9 use Symfony\Component\Form\FormBuilderInterface;
10 use Symfony\Component\OptionsResolver\OptionsResolver;
11
12 class PlaylistType extends AbstractType
13 {
14     // references : 0 (overrides / prototype)
15     public function buildForm(FormBuilderInterface $builder, array $options): void
16     {
17         $builder
18             ->add('name', null, [
19                 'required' => true,
20                 'label' => 'Nom'
21             ])
22             ->add('description')
23             ->add('submit', SubmitType::class);
24     }
25
26     // references : 0 (overrides / prototype)
27     public function configureOptions(OptionsResolver $resolver): void
28     {
29         $resolver->setDefaults([
30             'data_class' => Playlist::class,
31         ]);
32     }
33 }

```

J'ai simplement rajouté l'obligation de renseigner un « name » et également un bouton submit pour valider l'ajout.

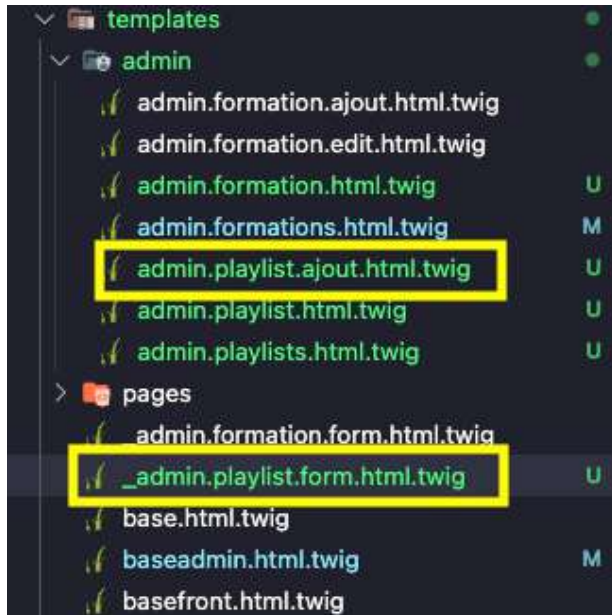
Dans AdminPlaylistsController, on ajoute maintenant la méthode ajout qui correspond quasiment à la même chose que la méthode ajout de AdminFormationsController :

```

src/Controller/AdminPlaylistsController.php:14:40 > AdminPlaylistsController > ajout
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2
```


On fait tout simplement correspondre avec l'entity `Playlist()` et en ajoutant le `use` correspondant et en mettant la bonne route à chaque fois. 'formformation' est remplacé logiquement par 'formplaylist' qui nous servira à afficher le formulaire dans la vue.

J'ai évidemment créé deux nouveaux fichiers : « `admin.playlist.ajout.html.twig` » et un fichier « `_admin.playlist.form.html.twig` » :



« `_admin.playlist.form.html.twig` » contiendra le formulaire d'ajout, mais également celui pour modifier une playlist, c'est pour cela que l'on met le formulaire dans ce fichier-là.

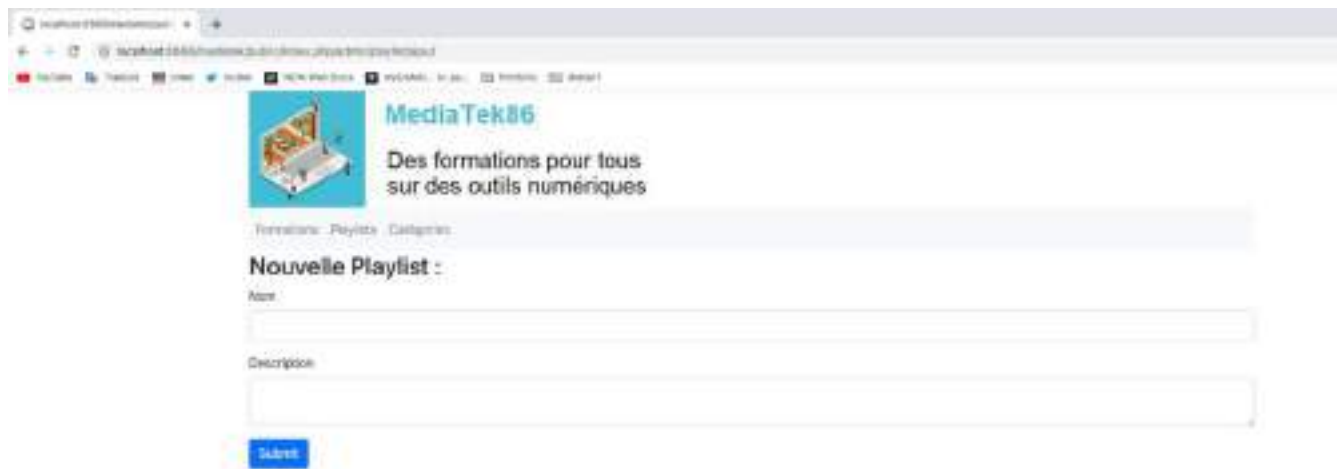
Dans « `_admin.playlist.form.html.twig` », on permet l'affichage du nouveau formulaire :



Et enfin dans « admin.playlist.ajout.html.twig » il faut inclure le form :

```
admin.playlist.ajout.html.twig U X
templates > admin > ./ admin.playlist.ajout.html.twig
1 {% extends "baseadmin.html.twig" %}
2
3 {% block body %}
4     <h2>Nouvelle Playlist :</h2>
5     {{ include ('_admin.playlist.form.html.twig') }}
6 {% endblock %}
```

Voici donc à quoi ressemble le formulaire après le clic sur « ajouter une nouvelle playlist » :



En revanche, rien à voir avec l'ajout d'une playlist, mais j'avais oublié de créer « admin.playlist.html.twig » et « admin.formation.html.twig » pour l'affichage des détails d'UNE formation et d'UNE playlist dans la liste des formations et des playlists. J'ai donc corrigé cela :

```
templates
└─ admin
    ├── admin.formation.ajout.html.twig
    ├── admin.formation.edt.html.twig
    ├── admin.formation.html.twig U
    ├── admin.formation.html.twig M
    ├── admin.playlist.ajout.html.twig U
    ├── admin.playlist.html.twig U
    └── admin.playlist.html.twig U
```


[illegible]

In Progress 1

mediatekformation #9

phase B : modification d'une playlist

```
<th class="text-left">
  actions
</th>
```

```

</td>
<div class="btn-group" role="group">
  <a href="" class="btn btn-secondary">Editor</a>
  &nbsp;
  <a href="" class="btn btn-danger">Supprimer</a>
</div>
</td>

```



Dans AdminPlaylistsController, on ajoute la nouvelle méthode « edit » :

```

/...
* @Route("/admin/playlist/edit/{id}", name="admin.playlist.edit")
* @param Playlist $playlist
* @param Request $request
* @return Response
*
*
4 references (0 overrides)
public function edit(Playlist $playlist, Request $request, $id): Response{
    $formPlaylist = $this->createForm(PlaylistType::class, $playlist);

    $playlistFormations = $this->formationRepository->findAllForOnePlaylist($id);

    $formPlaylist->handleRequest($request);
    if($formPlaylist->isSubmitted() && $formPlaylist->isValid()){
        $this->playlistRepository->add($playlist, true);
        return $this->redirectToRoute('admin.playlists');
    }

    return $this->render("admin/admin.playlist.edit.html.twig", [
        'playlist' => $playlist,
        'formplaylist' => $formPlaylist->createView(),
        'playlistformations' => $playlistFormations
    ]);
}

```

J'ai copié la méthode edit de celle créée lors de la modification d'une formation, en remplaçant simplement par le bon entity, le bon Repository mais également la bonne route que l'on va créer juste après.

A la manière de la méthode « showone » j'ai également appelé la méthode `findAllForOnePlaylist()` à partir de l'objet `formationRepository`. Celle-ci récupère toutes les formations liées à une playlist (d'où l'`$id` en paramètre) et les stockent dans « `playlistFormations` ». Cela nous permettra d'afficher la liste des formations présentes dans une playlist, au sein du formulaire « edit ».

On crée maintenant le fichier « `admin.playlist.edithtml.twig` », on y insère le form créé pour l'ajout d'une playlist, en rajoutant une colonne permettant d'afficher la liste des playlists présent dans la formation :

```
admin playlist.edit.html.twig U X
templates > admin > admin.playlist.edit.html.twig
1  {% extends "baseadmin.html.twig" %}
2
3  {% block body %}
4      <h2>Détail Playlist :</h2>
5      <div class="row">
6          <div class="col-md-6">
7              {{ include('_admin.playlist.form.html.twig') }}
8          </div>
9          <div class="col-md-6">
10              <div class="list-group">
11                  {% for formation in playlistformations %}
12                      <div class="row mt-1">
13                          <div class="col-md-auto">
14                              {% if formation.miniature %}
15                                  <a href="{{ path('admin.formation.showone', {id:formation.id}) }}">
16                                      
17                                  </a>
18                              {% endif %}
19                          </div>
20                          <div class="col d-flex align-items-center">
21                              <a href="{{ path('admin.formation.showone', {id:formation.id}) }}"
22                                  class="link-secondary text-decoration-none">
23                                  {{ formation.title }}
24                              </a>
25                          </div>
26                      </div>
27                  {% endfor %}
28              </div>
29          </div>
30      {% endblock %}
31
```

J'ai simplement utilisé le code du front pour l'affichage d'une playlist. Je peux l'utiliser car souvenez-vous, j'ai créé de la même manière `playlistformations` dans la méthode `édit`, donc je peux m'en servir comme ceci également.

Il nous reste simplement à compléter le href du bouton éditer :

```
<a href="{{ path('admin.playlist.edit', {id:playlists[k].id}) }}" class="btn btn-secondary">Editer</a>
```

Tâche 9 : Supprimer une playlist



Temps estimé : 1h – Temps réel : 1h

Pour terminer la gestion des playlists, on ajoute la méthode « delete » dans « AdminPlaylistsController » en reprenant celle des formations en changeant la route, le repository, et l'entity pour correspondre :

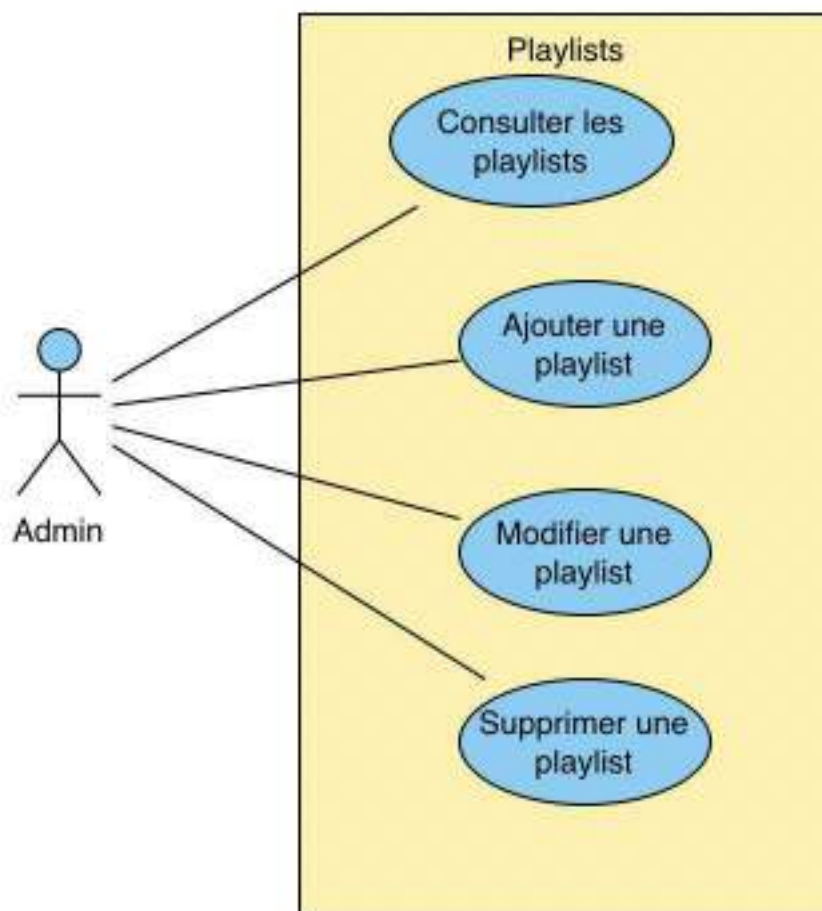
```
144
+ private ("admin/playlists/suppr/{id}", name="admin.playlist.suppr")
+ delete Playlist $playlist
+ return Response
+
+ @RequestMapping(methods = {RequestMethod.DELETE})
+ public Function<Response> supprPlaylist($playlist : Response)
+ {
+     $this->playlistRepository->remove($playlist, true);
+     return $this->redirect($this->adminPlaylists());
+ }
```

Enfin, on met la route de la méthode correspondante dans le « playlists.html.twig » au niveau du bouton supprimer. De plus on ajoute un attribut onclick, qui définit un code JavaScript qui sera déclenché au moment du clic sur le bouton. On utilise une instruction ternaire pour afficher un message d'alerte qui précise que la playlist ne peut pas être supprimée si elle contient des formations. Sinon, on demande à l'utilisateur s'il souhaite vraiment supprimer la playlist (vide) :

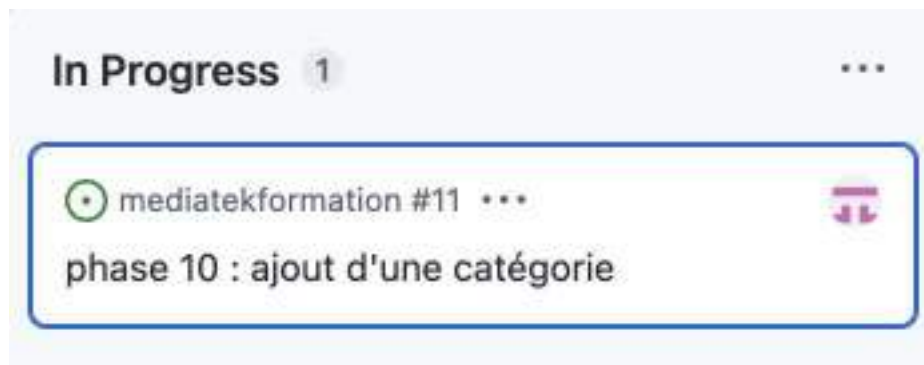
```
< a href="{% path('admin.playlist.suppr', id:playlists[k].id) %}" class="btn btn-danger" onclick="{% if playlists[k].formations|length > 0 %}
'alert('Impossible de supprimer la playlist car elle contient des formations.');" return false; {% else %}
'return confirm('Êtes-vous sûr de vouloir supprimer ' ~ playlists[k].name ~ ' ?');" %}>Supprimer</a>
```



On peut conclure cette partie par un diagramme du cas d'utilisation « Gestion des playlists » :



Tâche 10 : Ajouter une catégorie

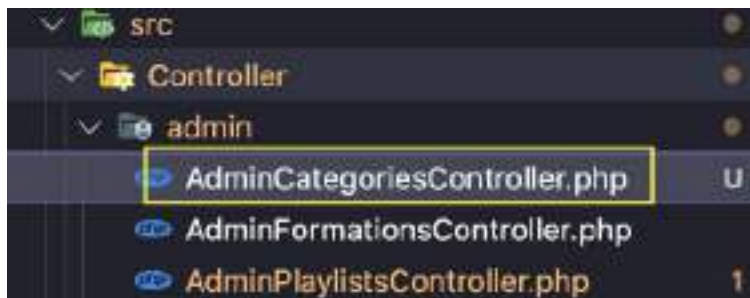


Temps estimé : 2h – Temps réel : 2h

On ajoute d'abord le path dans le baseadmin.html.twig vers la future page qui affichera la liste des catégories : admin.categories.html.twig :

```
baseadmin.html.twig M X
templates > baseadmin.html.twig
1  {% extends "base.html.twig" %}
2
3  {% block title %}{% endblock %}
4  {% block stylesheets %}{% endblock %}
5  {% block top %}
6      <div class="container">
7          <!-- titre -->
8          <div class="text-left">
9              
10             </div>
11          <!-- menu -->
12          <nav class="navbar navbar-expand-lg navbar-light bg-light">
13              <div class="collapse navbar-collapse" id="navbarSupportedContent">
14                  <ul class="navbar-nav mr-auto">
15                      <li class="nav-item">
16                          <a class="nav-link" href="{{ path('admin. formations') }}">Formations</a>
17                      </li>
18                      <li class="nav-item">
19                          <a class="nav-link" href="{{ path('admin.playlists') }}">Playlists</a>
20                      </li>
21                      <li class="nav-item">
22                          <a class="nav-link" href="{{ path('admin.categories') }}">Catégories</a>
23                      </li>
24                  </ul>
25              </div>
26          </nav>
27      </div>
28  {% endblock %}
29  {% block body %}
30
31  {% endblock %}
32  {% block footer %}
33
34  {% endblock %}
35  {% block javascripts %}{% endblock %}
```

Ensuite on ajoute un nouveau Controller « AdminCategoriesController.php » :



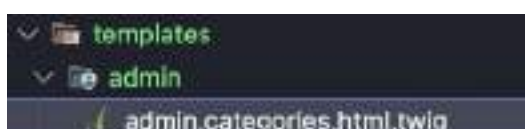
Globalement, je copie majoritairement le contenu du `FormationsController` en changeant comme d'habitude le namespace et en le faisant hériter de `AbstractController`.

Je déclare la même méthode `_construct` et la même méthode `index()` en changeant bien évidemment les routes :

```

AdminCategoriesController.php U X
src > Controller > admin > AdminCategoriesController.php
7 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
8 use Symfony\Component\HttpFoundation\Request;
9 use Symfony\Component\HttpFoundation\Response;
10 use Symfony\Component\Routing\Annotation\Route;
11 use Symfony\Component\Routing\Annotation\Route;
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
```

Maintenant nous créons le fichier « admin.categories.html.twig » :



Dans ce dernier, on va étendre le `baseadmin.html.twig` évidemment, mais aussi :

- afficher un mini formulaire qui nous permettra d'ajouter une catégorie
- afficher un tableau, avec deux colonnes « name » et « action »
- name contient le nom des catégories existantes et action contiendra un bouton permettant de supprimer.

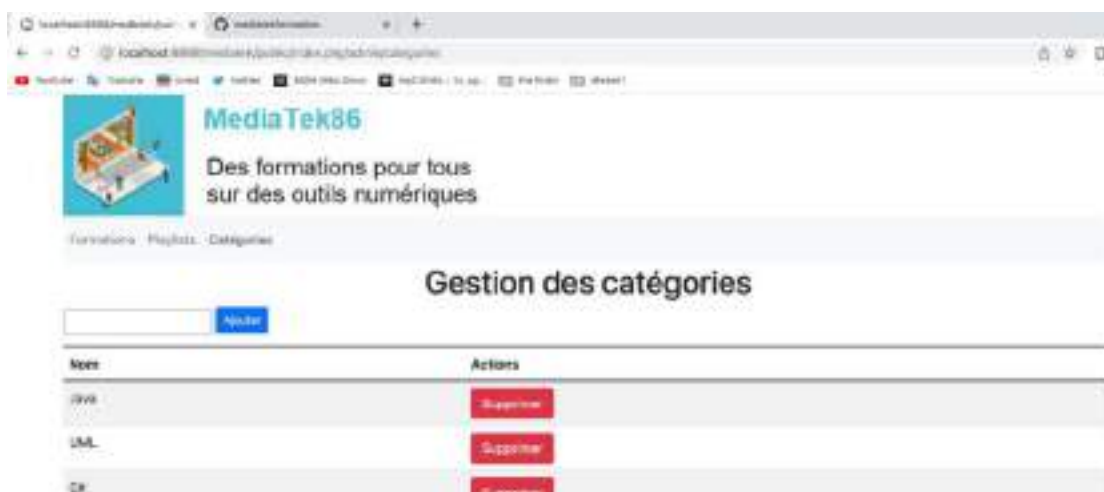
Formulaire :

```
{% extends "baseadmin.html.twig" %}

{% block body %}
<div class="container">
  <div class="text-center">
    <h1>Gestion des catégories</h1>
  </div>
</div>
<form class="form-inline mt-1" method="POST" action="{{ path('admin.categorie.ajout') }}">
  <div class="form-group mr-1 mb-2">
    <input type="text" class="sm" name="nom">
    <button type="submit" class="btn btn-primary mb-2 btn-sm">Ajouter</button>
  </div>
</form>
```

Tableau :

```
<table class="table table-striped">
  <thead>
    <tr>
      <th>Nom</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    {% for categorie in categories %}
      <tr>
        <td>{{ categorie.name }}</td>
        <td>
          <a href="{{ path('admin.categorie.suppr', {'id': categorie.id}) }}"
            class="btn btn-danger"
            onclick="return confirm('Etes-vous sûr de vouloir supprimer {{ categorie.name }} ?') ">
            Supprimer
          </a>
        </td>
      </tr>
    {% endfor %}
  </tbody>
</table>
```



Désormais on veut ajouter une catégorie. Dans le AdminCategoriesController.php on crée une nouvelle méthode ajout qui reprend le principe des précédentes.

Dans cette méthode on va bien évidemment lui donner la bonne route, dans \$nomCategorie on stocke le nom d'une catégorie depuis le bouton ajouter du « formulaire » de la page qui contient l'input « nom ».

Dans \$categories, on stocke grâce à la méthode findAll() du repository une liste de catégories.

Maintenant dans la boucle foreach, on cherche si dans la liste des catégories récupérées, une catégorie correspond déjà à celle que l'on vient d'ajouter. Si c'est le cas alors on est simplement redirigé sur la page des catégories et un message d'erreur est affiché.

Sinon, un nouvel objet de type Categorie est créé, avec un nom et ajouté dans la liste et en base de données. On est aussi redirigé sur la page avec cette fois avec un message de confirmation :

```
<?php
* @Route("/admin/categorie/ajout", name="admin.categorie.ajout")
* @param Request $request
* @return Response
*/
2 references | 0 overrides
public function ajout(Request $request): Response
{
    $nomCategorie = $request->get("nom");

    $categories = $this->categorieRepository->findAll();

    foreach ($categories as $categorieExistante) {
        if ($categorieExistante->getName() == $nomCategorie) {
            $this->addFlash('failed', 'La catégorie existe déjà');
            return $this->redirectToRoute('admin.categories');
        }
    }

    $categorie = new Categorie();
    $categorie->setName($nomCategorie);
    $this->addFlash('success', 'catégorie correctement ajoutée');
    $this->categorieRepository->add($categorie, true);
    return $this->redirectToRoute('admin.categories');
}
```

Pour les addFlash (messages) « success » et « failed » , on les ajoute comme suit dans la vue :

```
{% for message in app.flashes('success') %}
    <div class="alert alert-success">
        {{ message }}
    </div>
{% endfor %}

{% for message in app.flashes('failed') %}
    <div class="alert alert-danger">
        {{ message }}
    </div>
{% endfor %}
```

Vérifions ce qu'il se passe si j'essaie d'ajouter la catégorie Java qui existe déjà :

Gestion des catégories

La catégorie existe déjà.

Nom	Actions
Java	<input type="button" value="Supprimer"/>
UML	<input type="button" value="Supprimer"/>
CA	<input type="button" value="Supprimer"/>
Python	<input type="button" value="Supprimer"/>
MCD	<input type="button" value="Supprimer"/>
Android	<input type="button" value="Supprimer"/>
POO	<input type="button" value="Supprimer"/>
SQL	<input type="button" value="Supprimer"/>
Cours	<input type="button" value="Supprimer"/>

Catégories

Avec une catégorie Symfony maintenant :

catégorie correctement ajoutée	
Nom	Actions
Java	Supprimer
UML	Supprimer
CA	Supprimer
Python	Supprimer
MCD	Supprimer
Android	Supprimer
POO	Supprimer
SQL	Supprimer
Cours	Supprimer
Symfony	Supprimer

Tâche 11 : Supprimer une catégorie



Temps estimé : 1h – Temps réel : 1h

Maintenant, on ajoute la méthode suppr dans le même controller :

```
//...
+ @Route("/admin/categorie/suppr/{id}", name="admin_categorie_suppr")
+ @param Catégorie $categorie
+ @return Response
+}

/**
 * Supprime une catégorie
 */
public function suppr(Catégorie $categorie): Response
{
    $formations = $categorie->getFormations();

    if ($formations->isEmpty()) {
        $this->addFlash('success', 'Suppression effectuée');
        $this->categorieRepository->remove($categorie, true);
        return $this->redirectToRoute('admin.categorie');
    }

    $this->addFlash('failed', 'Echec de la suppression. La catégorie est reliée à une ou plusieurs formations');
    return $this->redirectToRoute('admin.categorie');
}
```

On veut pouvoir supprimer une catégorie uniquement si elle n'est pas reliée à une ou plusieurs formations.

Dans le if, on cherche si \$formation (contient l'ensemble des formations d'une catégorie) est vide. Si c'est le cas alors, on peut supprimer la catégorie et nous sommes redirigés sur la page.

En revanche, dans le cas inverse, on est redirigé sur la page mais sans aucun changement avec évidemment un message d'erreur nous prévenant pourquoi la catégorie n'est pas supprimée.

Exemple avec Java qui est reliée à plusieurs formations :

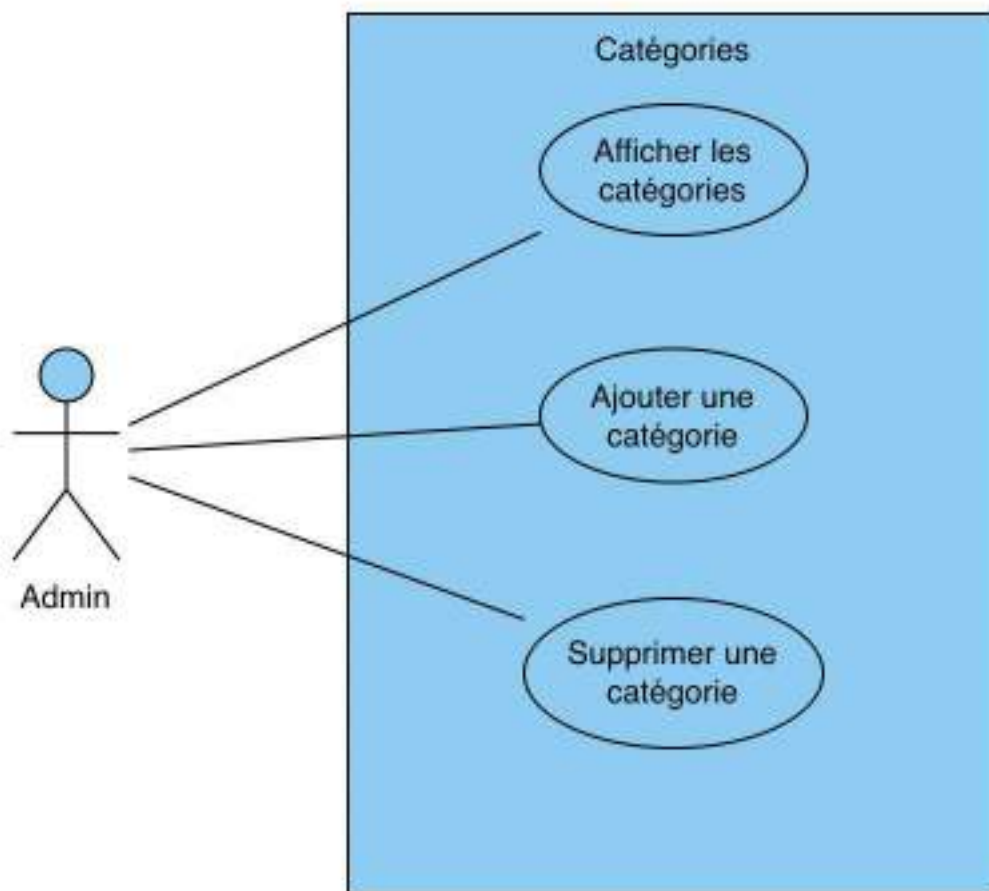
Echec de la suppression. La catégorie est relié à une ou plusieurs formations	
Nom	Actions
Java	Supprimer

Maintenant avec Symfony (qui ne contient rien car nous venons de la créer) :

Suppression effectuée	
Nom	Actions
Java	Supprimer
LML	Supprimer
CF	Supprimer
Python	Supprimer
MCD	Supprimer
Android	Supprimer
POO	Supprimer
SQL	Supprimer
Cours	Supprimer

Catégories

On peut imaginer la création de la page catégorie du côté admin avec le diagramme de cas d'utilisation suivant :

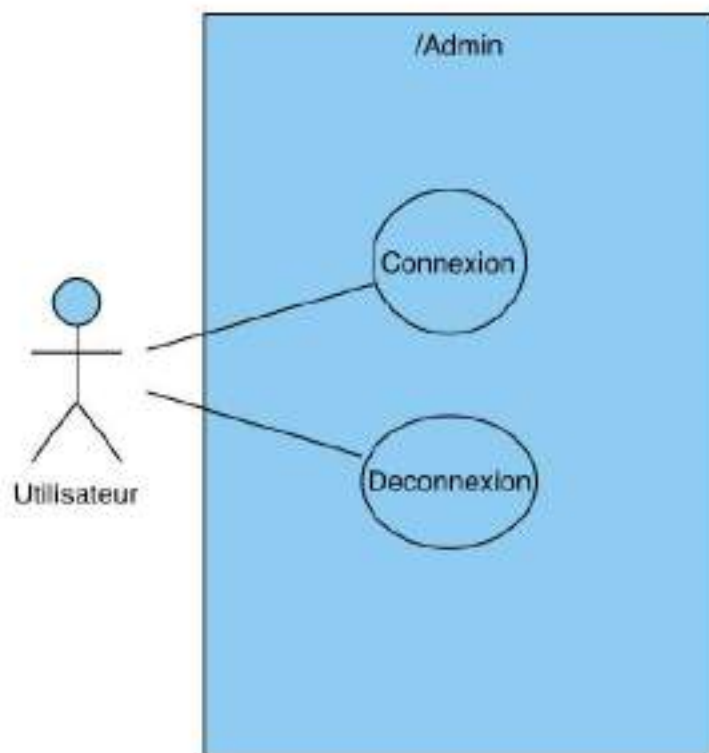


Tâche 12 : Ajout de l'accès avec authentification

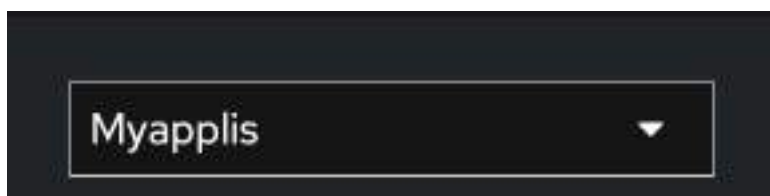


Temps estimé : 4h – Temps réel : 4h

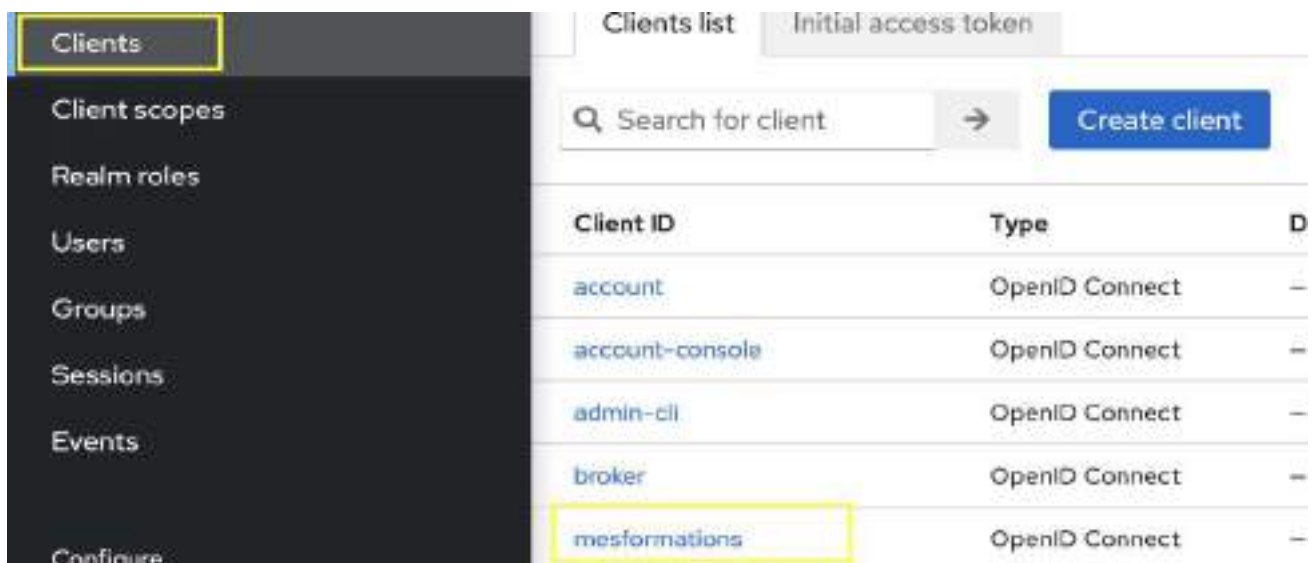
Voici ce qui doit être possible de faire sur le site à travers ce diagramme de cas d'utilisation :



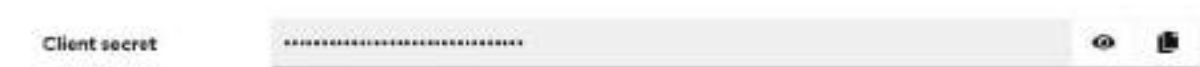
On commence par paramétrer Keycloak. On crée un royaume, ici « Myapplis » :



Dans celui-ci, on ajoute un nouveau client que l'on nomme « mesformations » :



Dans l'onglet « Credentials » on copie le code du « client secret » et on l'enregistre précieusement :

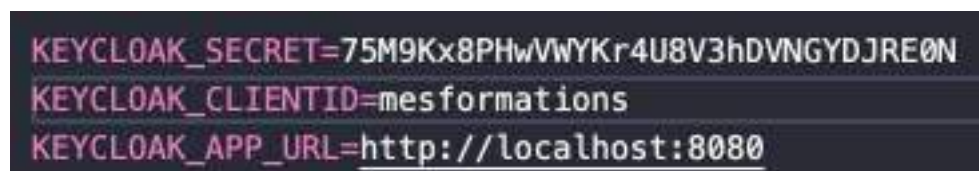


Dans « Users » on ajoute un nouvel utilisateur « adminmesformations » avec un email :



Dans Credentials on définit un mot de passe également.

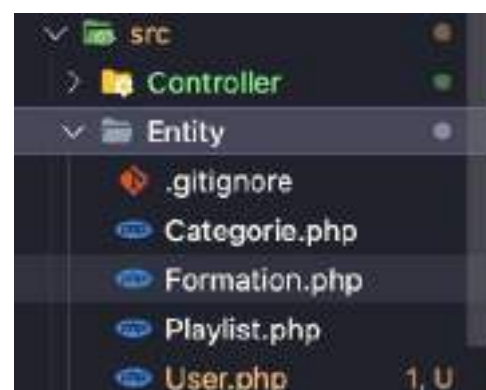
Dans Visual Studio Code maintenant, on va ajouter en bas du fichier .env, les informations de connexion Keycloak :



KEYCLOAK_SECRET correspond au client secret que nous avons copié plus haut

On veut maintenant ajouter une ligne dans la base de données pour enregistrer l'utilisateur qui se connecte ou se déconnecte.

Dans l'invite de commande, au niveau du projet : php bin/console make:user avec les options par défaut. Puis on crée les champs en créant l'entity User : php bin/console make:entity User (name = keycloakId, type=string,length par défaut, nullable dans la bdd) :

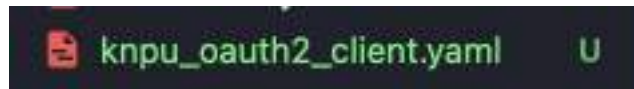


Ensuite on crée le fichier de migration permettant de créer la table dans la base de données : php bin/console make:migration puis php bin/console doctrine:migrations:migrate :



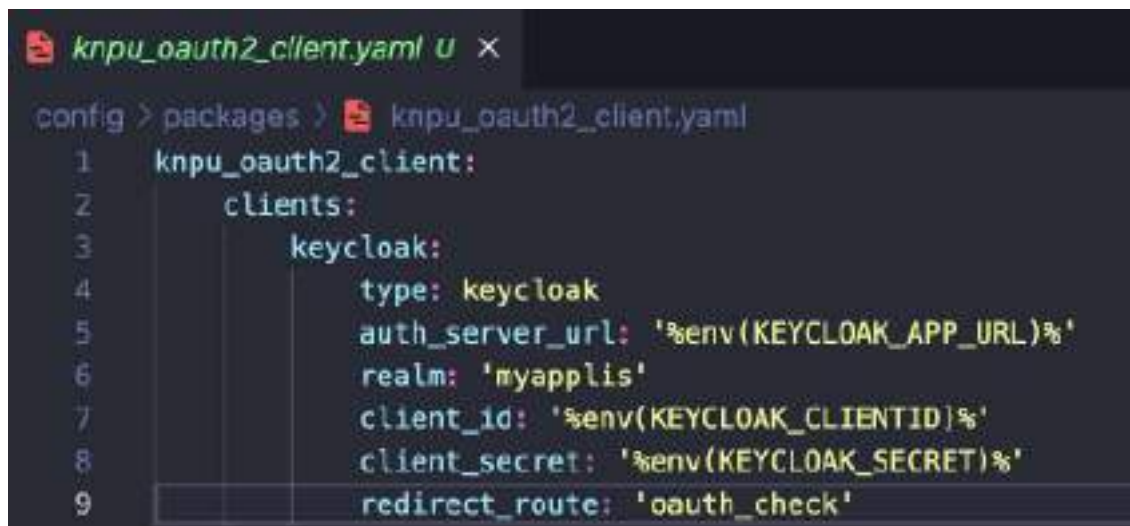
Nous créons deux bundles pour faire le lien entre symfony et keycloak, toujours dans l'invite de commandes : `Composer require knpuniversity/oauth2-client-bundle 2.10.`

Dans `config/packages` le fichier suivant est créé :



Puis on ajoute le deuxième bundle avec cette commande : `composer require stevenmaguire/oauth2-keycloak 3.1 --with-all-dependencies.`

Dans « `knpu_oauth2_client.yaml` » on ajoute les paramètres suivants, en respectant les noms précédemment mis dans le fichier `.env` :



Ensuite dans `config > packages > security.yaml` :



`Oauth_login` correspond à la route de redirection en cas d'authentification et dans `access_control` c'est le chemin qui nécessite une authentification + le rôle nécessaire pour pouvoir y accéder.

Dans src > Controller on vient créer OAuthController qui va gérer l'authentification :

```
OAuthController.php 1,0 X
src > Controller > OAuthController.php ? ...
1 <?php
2
3 namespace App\Controller;
4
5 use Keycloak\OAuth2ClientBundle\Client\ClientRegistry;
6 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
7 use Symfony\Component\HttpFoundation\RedirectResponse;
8 use Symfony\Component\HttpFoundation\Request;
9 use Symfony\Component\Routing\Annotation\Route;
10
11 /**
12  * @Route("/oauth/login", name="oauth_login")
13  */
14 class OAuthController extends AbstractController
15 {
16     /**
17      * @Route("/oauth/login", name="oauth_login")
18      */
19     public function index(ClientRegistry $clientRegistry): RedirectResponse
20     {
21         return $clientRegistry->getClient('keycloak')->redirect();
22     }
23
24     /**
25      * @Route("/oauth/callback", name="oauth_check")
26      */
27     public function connectCheckAction(Request $request, ClientRegistry $clientRegistry)
28     {
29     }
30 }
```

Index est la méthode reçue lors de la connexion, elle récupère le client 'keycloak' de l'objet reçu en paramètre et appelle la méthode redirect pour la rediriger vers l'authentification géré par keycloak.

La méthode connectCheckAction prend en charge la route de redirection après connexion. Le contenu est vide car géré par Symfony.

À la suite de l'appel redirect(), l'authentification est sollicitée par keycloak, et renvoie un code. L'application récupère ce code pour l'échanger avec un token d'accès. Pour la récupération du code, nous ajoutons une nouvelle classe « KeycloakAuthenticator.php » dans src > Security :



Pour construire cette classe on suit l'exemple de MyFacebookAuthenticator de la documentation « github.com/knpuniversity/oauth2-client-bundle » .

On ajoute les propriétés suivantes en début de classe :

```
7 references
private $clientRegistry;
7 references
private $entityManager;
2 references
private $router;

2 references | 0 overrides
public function __construct(
    ClientRegistry $clientRegistry,
    EntityManagerInterface $entityManager, RouterInterface $router)
{
    $this->clientRegistry = $clientRegistry;
    $this->entityManager = $entityManager;
    $this->router = $router;
}
```

- ClientRegistry est le gestionnaire de clients OAuth
- EntityManagerInterface permet de gérer la base de données
- RouterInterface pour lire la route

5 méthodes dans KeycloakAuthenticator.php (basé sur la documentation et MyFacebookAuthenticator) :

1) Méthode start

Envoie vers une route temporaire définie dans le controlleurOAuthController :

```
2 references | 0 overrides
public function start(Request $request, AuthenticationException $authException = null): Response
{
    return new RedirectResponse(
        '/oauth/login',
        Response::HTTP_TEMPORARY_REDIRECT
    );
}
```

2) Méthode support

Vérifie si l'url reçu correspond à « oauth_check », si c'est le cas, renvoie true :

```
0 references | 0 overrides
public function supports(Request $request): bool
{
    return $request->attributes->get('_route') === 'oauth_check';
}
```

3) Méthode authenticate

Récupère le client dans Keycloak et le token. A partir de là, elle récupère l'utilisateur pour l'enregistrer dans la base de données.

```
0 references | 0 events | 0 logs
public function authenticate(Request $request): Passport
{
    $client = $this->clientRegistry->getClient('keycloak');
    $accessToken = $this->fetchAccessToken($client);
    return new SelfValidatingPassport(
        new UserBadge($accessToken->getToken(), function () use ($accessToken, $client) {
            /** @var KeycloakUser $keycloakUser */
            $keycloakUser = $client->fetchUserFromToken($accessToken);
            // 1) recherche du user dans la BDD à partir de son id Keycloak
            $existingUser = $this->entityManager
                ->getRepository(User::class)
                ->findOneBy(['keycloakId' => $keycloakUser->getId()]);
            if ($existingUser) {
                return $existingUser;
            }
            // 2) le user existe mais n'est pas encore connecté à keycloak
            $email = $keycloakUser->getEmail();
            $userInDatabase = $this->entityManager
                ->getRepository(User::class)
                ->findOneBy(['email' => $email]);
            if ($userInDatabase) {
                $userInDatabase->setKeycloakId($keycloakUser->getId());
                $this->entityManager->persist($userInDatabase);
                $this->entityManager->flush();
                return $userInDatabase;
            }
            // 3) si le user n'existe pas encore dans la bdd
            $user = new User();
            $user->setKeycloakId($keycloakUser->getId());
            $user->setEmail($keycloakUser->getEmail());
            $user->setPassword('');
            $user->setRoles(['ROLE_ADMIN']);
            $this->entityManager->persist($user);
            $this->entityManager->flush();
            return $user;
        })
    );
}
```

4) Méthode « onAuthenticationFailure »

Exécution en cas de problème dans une autre méthode :

```
0 references | 0 events | 0 logs
public function onAuthenticationFailure(Request $request, AuthenticationException $exception): ?Response
{
    $message = strtr($exception->getMessageKey(), $exception->getMessageData());
    return new Response($message, Response::HTTP_FORBIDDEN);
}
```


5) Méthode « onAuthenticationSuccess »

Execution quand tout s'est bien passé, on redirige donc vers la partie /admin du site :

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response
{
    $targetUrl = $this->router->generate('admin_formation');
    return new RedirectResponse($targetUrl);
}
```

Maintenant on ajoute dans le firewall le chemin de la classe qui contient ces 5 méthodes :

```
security:
    enable_authenticator_manager: true
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
    providers:
        # used to reload user from session & other features (e.g. switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            entry_point: form_login
            # providers: app_user_provider
            form_login:
                login_path: oauth_login
            custom_authenticators:
                - App\Security\KeycloakAuthenticator
            logout:
                path: logout
```

Il faut gérer maintenant la déconnexion ; dans le « baseadmin.html.twig » on ajoute un lien « se déconnecter » avec une route « logout » et dans le contrôleur « OAuthController » on ajoute une méthode logout() vide (car c'est le firewall qui s'en occupe) et on y met la route /logout :

```
<div class="container">
    <div class="text-end">
        <a href="{ path('logout') }">se déconnecter</a>
    </div>
</div>
```



```

/**
 * @Route ("/logout", name ="logout")
 * @return void
 */
↑reference | 0 overrides
public function logout()
{

}

```

Maintenant, essayons de nous rendre dans la partie /admin :



Sign in to your account

Username or email

ruben-gallienpro@outlook.fr

Password

.....

Sign In

Je rentre mon mail et le mot de passe précédemment créés et je peux accéder à la partie admin de mon site sur lequel apparaît (sur chaque page) un lien pour se déconnecter :



MediaTek86

Des formations pour tous sur des outils numériques

se déconnecter

Formations Playlist Catégories

Gestion des formations

Ajouter une nouvelle formation

formation	playlist	categories	date	actions
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

III. Mission 3 : Tester et documenter

Tâche 13 : Test unitaire sur la méthode qui retourne la date au format string



Temps estimé : 30min – Temps réel : 30min

Voici la capture de l’arborescence test qui présente FormationTest ainsi que la méthode nécessaire testGetPublishedAtString() pour le test de la date au bon format :



```
namespace App\Tests;

use App\Entity\Formation;
use DateTimeImmutable;
use PHPUnit\Framework\TestCase;
use Symfony\Component\Validator\Constraints\DateTime;

/* Tests */
class FormationTest extends TestCase
{
    /**
     * @test
     */
    public function testGetPublishedAtString()
    {
        $formation = new Formation();

        $date = new DateTimeImmutable('2023-03-03 14:30:00');
        $formation->setPublishedAt($date);

        $this->assertEquals('03/03/2023', $formation->getPublishedAtString());
    }
}
```

Tests unitaires

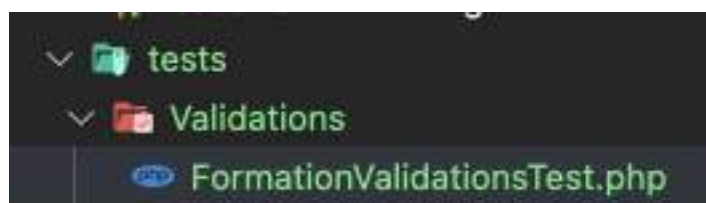
Objet du test	Action de contrôle	Résultat attendu	Réel
Contrôler la méthode <code>getPublishedAtString()</code> de la classe <code>Formation</code> pour voir si elle retourne la bonne date au bon format.	Test unitaire lancé avec la date : 2023-03-03 14:30:00	03/03/2023	OK

Tâche 14 : Test intégration sur les règles de validation

In Progress 1

mediatekformation #16
 phase 14 : test intégration sur règles de validation

Temps estimé : 1h – Temps réel : 1h



But du test	Action de contrôle	Résultat attendu	Bilan
Création méthode <u>getFormation()</u> qui retourne un objet de type Formation avec les initialisations obligatoires. Puis contrôle de cette méthode avec une <u>TestValideDateFormation</u> (non postérieure à la date du jour)	Test d'intégration lancé avec la date : 2023-03-03	Ras d'erreurs	OK
Contrôle avec méthode <u>TestNonValideDateFormation</u> (postérieure à la date du jour)	Test d'intégration lancé avec la date : 2024-03-03	Une erreur car on oblige la méthode à faire une erreur (pour l'instant on n'a pas <u>ajouter</u> la restriction dans le code)	OK
Ajout <u>ajouts</u> à la <u>contraintes du formulaire</u> dans l' <u>objet</u> formation qui <u>contrôle</u> bien que la date ne soit pas postérieure à <u>'today'</u> .	Test d'intégration sur les deux méthodes tests	Plus aucune erreur, la restriction est <u>plus</u> en compte. On ne peut plus saisir de date postérieure	OK

Tâche 15 : Test intégration sur les Repository

In Progress 1

mediatekformation #17 ...
 phase 15 : test intégration sur les Repository

Temps estimé : 1h – Temps réel : 2h



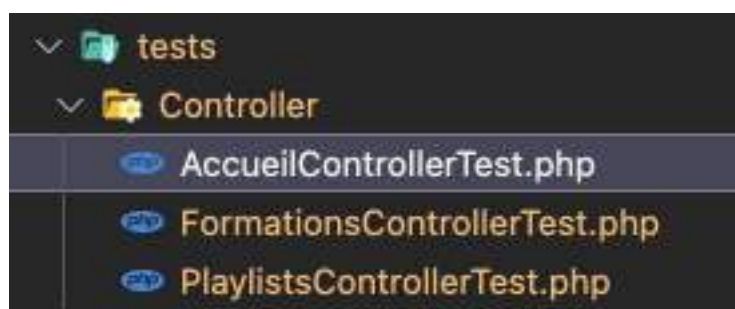
Tests d'intégration

but du test	Action de contrôle	Résultats attendus	Bilan
Test des méthodes <code>add()</code> et <code>remove()</code> de <code>FormationRepository</code> avec une BDD <code>mediatekformation_test</code>	Test d'intégrations avec la création d'une nouvelle formation avec un nom et une date.	La formation est ajoutée puis supprimée en base de données	OK
Test des méthodes <code>add()</code> et <code>remove()</code> de <code>PlaylistRepository</code> avec une BDD <code>mediatekformation_test</code>	Test d'intégrations avec la création d'une nouvelle playlist avec un nom	La playlist est ajoutée puis supprimée en base de données	OK
Test des méthodes <code>add()</code> et <code>remove()</code> de <code>CategorieRepository</code> avec une BDD <code>mediatekformation_test</code>	Test d'intégrations avec la création d'une nouvelle catégorie avec un nom	La catégorie est ajoutée puis supprimée en base de données	OK

Tâche 16 : Tests fonctionnels



Temps estimé : 1h – Temps réel : 2h



Tests fonctionnels

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler que la page d'accueil est accessible (la méthode <code>index()</code> de <code>AccountController</code> est testée).	Test avec la route « / »	code 200	OK
Contrôler que les tris fonctionnent dans la page des formations.	On envoie la route <code>/formations/tri/{type}/ASC</code> et la première formation en haut de la liste	Android Studio (complément n°1) ; Navigation <code>Drawer</code> et Fragment	OK
contrôler que le clic sur une image miniature permet d'accéder à la page de la formation en question (on contrôlant l'accès à la page mais aussi le contenu d'un des éléments de la page)	On simule le clic sur « formation miniature » de la 1 ^{re} formation qui doit sembler sur <code>/formations/formation/1</code> puis vérifier que le titre de la formation est présent sur cette page	Eclipses n°8 : Déploiement	OK
contrôler que les filtres fonctionnent (ex. filtrant le nombre de lignes obtenus et le résultat de la première ligne) ;	On simule un formulaire de recherche avec le <code>test</code> en SQL puis on cherche le nombre de formations affichées et le résultat de la 1 ^{re} ligne	9 formations sont affichées et la 1 ^{re} ligne en Eclipse n°8 : Déploiement	OK
On contrôle les mêmes choses pour la page des playlists	Seules les nom des routes changent (<code>/playlists</code>) et pour le clic, on cherche le clic de « voir détail »	Tout fonctionne	OK

Tâche 17 : Tests de compatibilités navigateurs

In Progress 1

mediatekformation #19 ...

phase 17 : test compatibilité de navigateurs

Temps estimé : 1h – Temps réel : 1h

Tests de compatibilité

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la compatibilité de l'application sur Google Chrome	Scénario d'un <code>TestFormations</code> avec Selenium sur la partie front-office	Tests Suites : 1 <code>passed</code> Tests : 1 <code>passed</code>	PASS
Contrôler la compatibilité de l'application sur Mozilla Firefox	Même scénario que pour Google Chrome	Tests Suites : 1 <code>passed</code> Tests : 1 <code>passed</code>	PASS



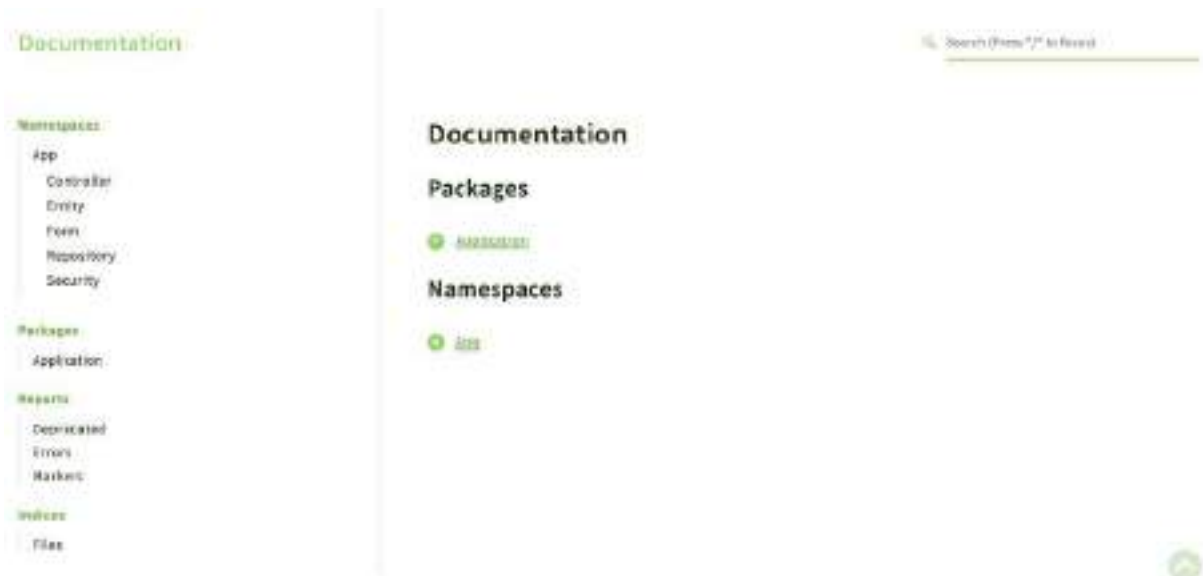
Tâche 18 : Créer la documentation technique

In Progress 2

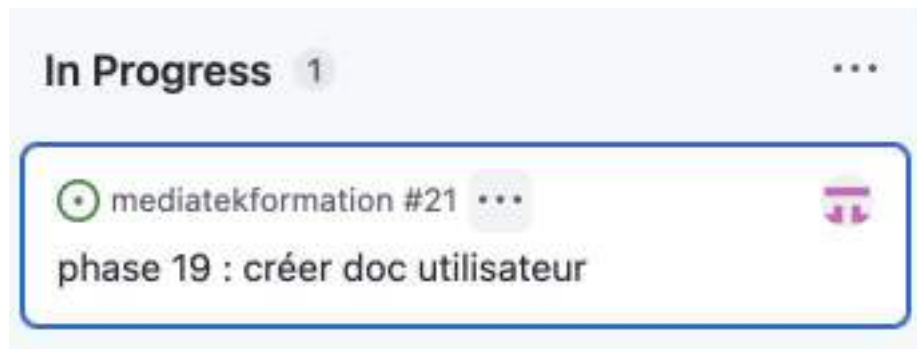
mediatekformation #20 ...

phase 18 : créer la doc technique

Temps estimé : 1h – Temps réel : 1h



Tâche 19 : Création doc utilisateur



Temps estimé : 2h – Temps réel : 2h



IV. Mission 4 : Déployer le site gérer le déploiement continu

Tâche 20 : Déployer le site



Temps estimé : 2h – Temps réel : 8h

Serveur d'authentification dans une VM en ligne :

On commence par installer et configurer le serveur d'authentification Keycloak dans une VM azure.

Pour cela on crée une machine virtuelle sur Azure, puis on la lance grâce à l'adresse ip fourni par azure pour la VM en question.

Sur cette VM, on installe JDK puis keycloak. On se place dans une invite de commande de la VM au niveau du fichier bin de keycloak puis on lance keycloak.

Dans le navigateur, sur la VM, au localhost :8080. Dans keycloak, on crée un royaume avec le même nom que dans le fichier "knpu_oauth2_client.yaml" qui est dans "config > packages" («myapplis») puis un client « mesformations » (CLIENT ID).

De retour dans la fenêtre de commande on finalise la configuration de keycloak en exécutant kc.bat build.

Ensuite on veut que keycloak soit accessible en HTTPS. On installe XAMPP sur la machine virtuelle, et sur ce dernier on démarre Apache. XAMPP ne sert qu'à avoir Apache pour obtenir le certificat « certbot » via un serveur. On télécharge donc certbot. Puis une fois dans l'invite de commande :

- on exécute : « certbotcertonly –webroot »
- on renseigne un mail et le nom de domaine Dns.

Dans le dossier keycloak/bin on peut maintenant lancer le serveur keycloak en https. Keycloak est désormais accessible en dehors de la VM.

Déploiement/Hébergement du site :

Pour ce qui est de l'hébergement du site, j'ai choisi PlanetHoster, qui est gratuit et plutôt simple d'utilisation.

On exporte la base de données en local puis on l'importe dans le phpMyAdmin de PlanetHoster.

Dans le .env de notre projet on renseigne la nouvelle chaîne de connexion de la base de données via les informations de PlanetHoster.

Pour transférer nos fichiers du projet à planetHoster, on fait un zip de notre application local, et via FileZilla, on transfère notre application zip dans le dossier www/ de PlanetHoster. On dézippe ensuite le fichier .zip dans à l'intérieur du gestionnaire de fichiers de l'hébergeur.

Dans le .env on passe en « prod » et non plus en dev. Dans le templates/pages on change la page des cgus en renseignant l'URL de notre application hébergée.

Notre site est maintenant fonctionnel en ligne et la partie /admin est aussi accessible via keycloak :

Partie front :



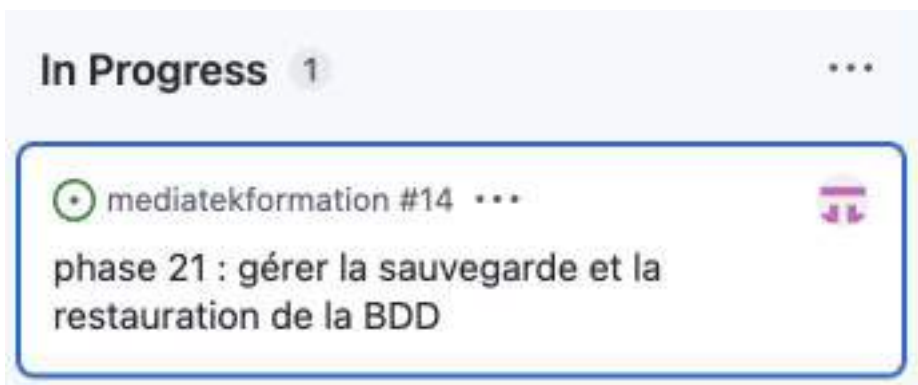
Partie Admin :



On ajoute aussi la documentation technique dans le www/ du gestionnaire de projet de PlanetHoster. On peut désormais y accéder en rajoutant /doc à la suite de l'URL du site « mediatekformations.go.yj.fr/doc » :



Tâche 21 : Gérer la sauvegarde et la restauration de la BDD



Temps estimé : 1h – Temps réel : 1h

On crée un fichier de script en local que l'on nomme backup.sh et que l'on remplit ainsi en fonction des informations transmises sur notre hébergeur [planetHoster] :

```
#!/bin/sh
DATE=`date -l`
find /home/[logincompte]/savebdd/bdd* -mtime -1 -exec rm {} \;
mysqldump -u [loginuserbdd] -p[pwduserbdd] --databases [namebdd] --single-transaction |
gzip> /home/[logincompte]/savebdd/bddbbackup_`${DATE}`.sql.gz
```

Il faut ensuite convertir le fichier au format linux, avec le logiciel dos2linux.

Puis, on copie le fichier backup.sh avec FileZilla sur le site de l'hébergeur. On crée un fichier « savebdd » à la racine du gestionnaire de fichier PlanetHoster. C'est à l'intérieur de ce dossier que l'on ajoute backup.sh.

Dans l'hébergeur on crée une « tâche crons » qui va s'effectuer avec une fréquence de 1 fois par jour. Et on tape la commande :

/home/[logincompte]/savebdd/backup.sh



Le fichier est bien ajouté dans le dossier savebdd en racine, dans le gestionnaire. Le script est automatisé sous le nom « bddbbackup_....sql.gz »



Tâche 22 : Mettre en place le déploiement continu



Temps estimé : 1h – Temps réel : 1h

A chaque push en local vers le dépôt distant, le site hébergé sur PlanetHoster doit être mis à jour.

Pour cela dans Github, dans notre projet > Actions > « set up a workflow yourself » > « Edit new file ».

On y insère ce code (le ftp_password sera rempli plus tard via Github) :

```
on:
  push:
  #
name: Deploy website on push
#
jobs:
  web-deploy:
    name: Deploy
    runs-on: ubuntu-latest
    steps:
      - name: Get latest code
        uses: actions/checkout@v2
      - name: Sync files
        uses: SamKirkland/FTP-Deploy-Action@4.3.0
        with:
          server: node3-eu.n0c.com
          server-dir: /public_html/mediatekformations/
          username: rubenformations@mediatekformations.go.yj.fr
          password: ${ secrets.ftp_password }
```

Puis on enregistre : “Start commit” > “Commit new File”. Le fichier yml est maintenant en racine du dépôt github. On pull en local sur VScode pour récupérer le fichier.

De retour sur Github> Settings > Secrets > Actions > New repository secret et on remplit les champs pour ajouter le mot de passe ftp_password avec la valeur de notre password ftp de PlanetHoster.

Désormais à chaque push sur github depuis vscode, le site va se mettre à jour également.

V. BILAN FINAL

- Développement du site complet : front + back-office (partie admin)
- Déploiement : Accès à la partie admin sécurisé via keycloak, site hébergé/déployer sur PlanetHoster + Base de données créer avec sauvegarde journalière.
- Déploiement continu via le dépôt distant sur Github
- Documentation technique accessible en ligne
- Documentation d'utilisation en vidéo