

Compte rendu d'activités

Réalisé lors de l'atelier professionnel n°1 au CNED

Présentation du projet :

MediaTek86 constitue un réseau responsable de la gestion des médiathèques dans le département de la Vienne. Sa principale mission est de coordonner les emprunts de livres, de DVD et de CD entre les différentes médiathèques tout en promouvant le développement de ressources médiathèques numériques pour l'ensemble du département.

Afin d'augmenter l'attrait des médiathèques, MediaTek86 aspire à élargir sa gamme de services. Cela inclut la proposition de formations sur les outils numériques pour les membres, ainsi que l'offre d'opportunités d'autoformation en ligne.

Objectif :

Une autre personne a déjà commencé à développer l'application destinée à présenter le site "Mediatek Formation". Je dois prendre la suite, corriger le code et mettre en place la partie back-office. Je dois aussi gérer le déploiement et la mise en place d'un système d'authentification externe avec Keycloak. Je dispose d'un cahier des charges pour la mission à réaliser.

Environnement de travail :

Dans le cadre de ce projet j'ai travaillé sous un OS Windows avec le IDE Apache NetBeans et le framework Symfony.

Mission 0 : préparer l'environnement de travail :

Pour commencer j'installe Wamp afin d'avoir un serveur web en local pour faire mes tests



J'installe également composer et Symfony en ligne de commande. Puis une fois le projet téléchargé je lance "composer install"

```
C:\wamp64\www\mediatekformation>composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 133 installs, 0 updates, 0 removals
```

et crée une nouvelles bdd avec PhpMyAdmin en y important celle du projet.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> categorie	★ Parcourir Structure Rechercher Insérer Vider Supprimer	9	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> doctrine_migration_versions	★ Parcourir Structure Rechercher Insérer Vider Supprimer	6	InnoDB	utf8mb3_unicode_ci	16,0 kio	-
<input type="checkbox"/> formation	★ Parcourir Structure Rechercher Insérer Vider Supprimer	237	InnoDB	utf8mb4_unicode_ci	160,0 kio	-
<input type="checkbox"/> formation_categorie	★ Parcourir Structure Rechercher Insérer Vider Supprimer	277	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> messenger_messages	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	64,0 kio	-
<input type="checkbox"/> playlist	★ Parcourir Structure Rechercher Insérer Vider Supprimer	27	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
6 tables	Somme	556	MyISAM	utf8mb3_unicode_ci	320,0 kio	0 o

↑ ☐ Tout cocher Avec la sélection : ▼

Je configure le fichier .env du projet Symfony pour l'accès à la base de données.

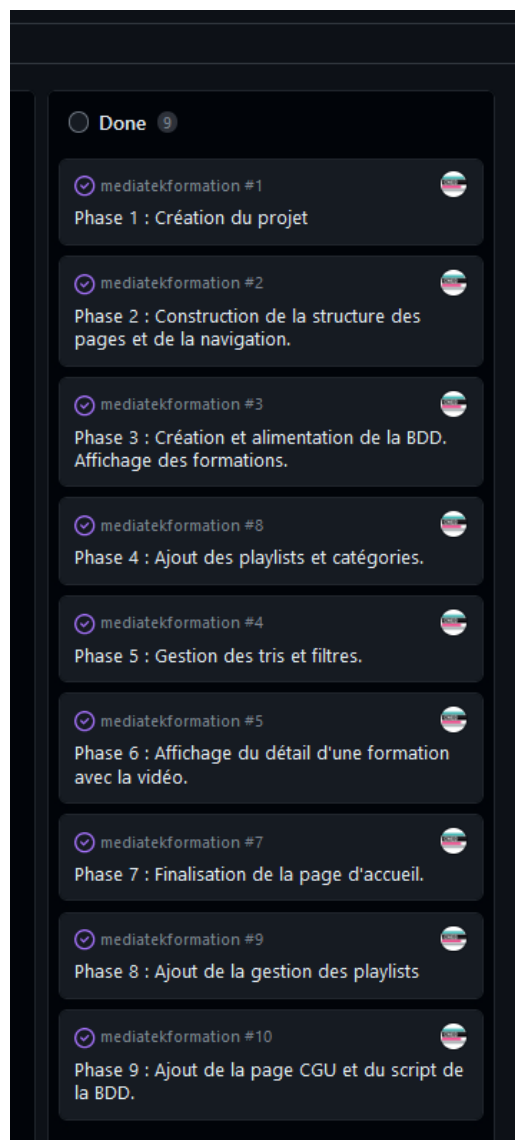
En utilisant Symfony CLI, je lance la commande « symfony server:start » dans le terminal

à la racine de mon projet et je vérifie que le site fonctionne.

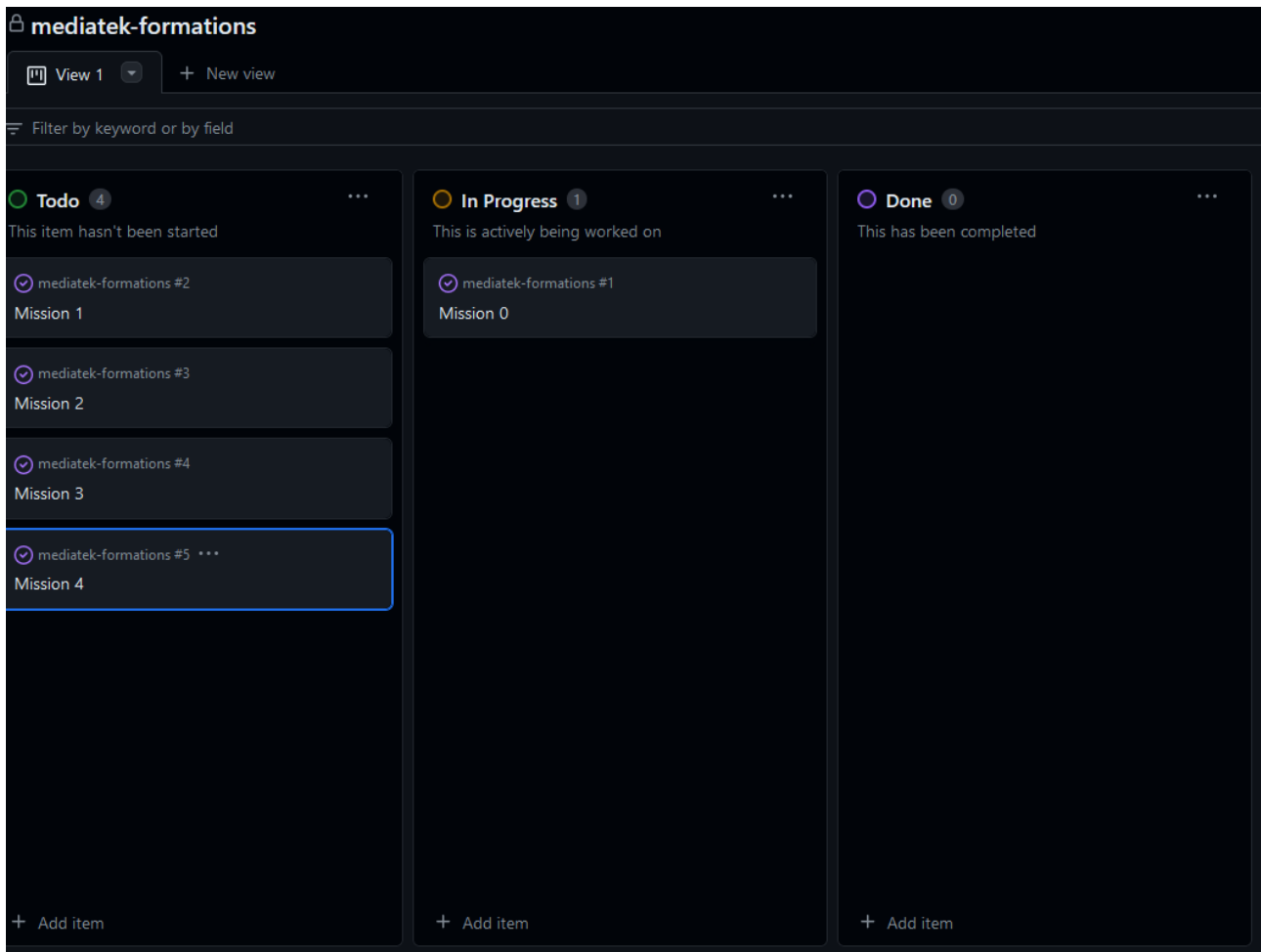
Je prends connaissance du travail déjà

réalisé avec visionnage du site, le kanban, le

code, et le readme du dépôt.



De même, je prends connaissance du travail demandé pour réaliser le kanban du projet.



Je configure Git pour créer un nouveau dépôt et push le premier commit.

Mission 1 : nettoyer et optimiser le code existant :

Tâche 1 : nettoyer le code :

Je vérifie la configuration SonarQube et lance le serveur et crée un nouveau projet.

Create a project

Project display name *



Up to 255 characters. Some scanners might override the value you provide.

Project key *



The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

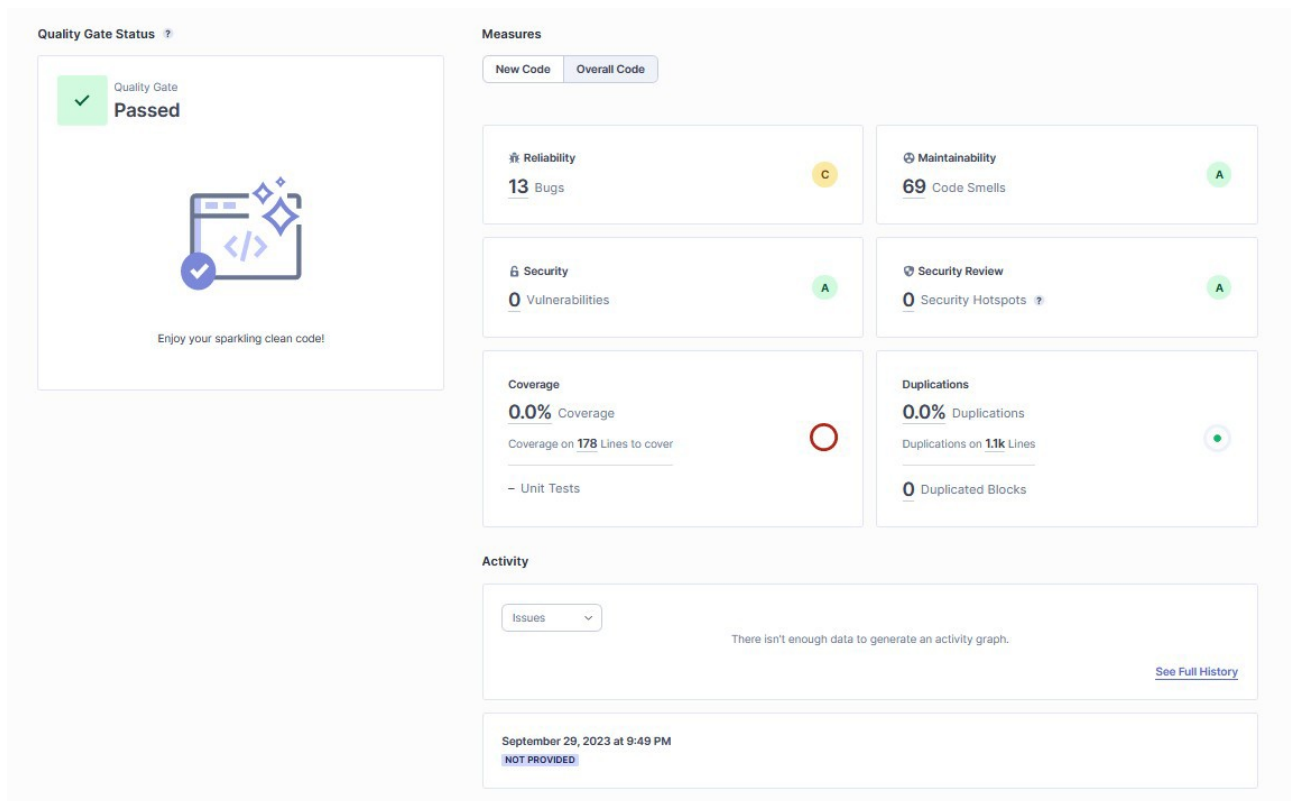
Main branch name *

The name of your project's default branch [Learn More](#)

Next

Je lance la commande pour effectuer le scan du projet a ça racine

```
INFO: Analysis Report uploaded in 33ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=mediatekformation
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AYrifu_rcce1Pm1wbliZ
INFO: Analysis total time: 1:27.469 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 1:28.732s
INFO: Final Memory: 21M/80M
INFO: -----
PS C:\wamp64\www\mediatekformation>
```



13 « bugs » et 69 « code smells » sont détecté. Je consulte les indications fournies par le logiciel pour corriger le code et le rendre en conformité. Une grande partie des corrections à effectuer concernent des espaces en trop, des URL réutilisées plusieurs fois dans le code et laissé en dur.

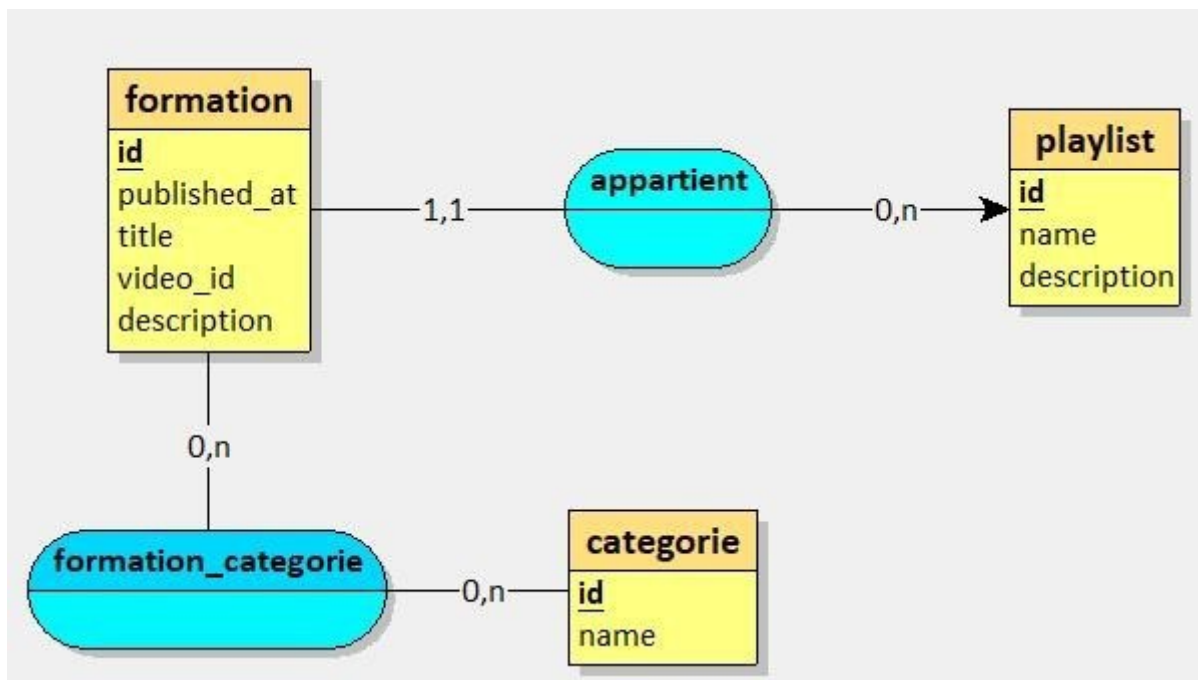
Il y a aussi une imbrication « inutile » que le logiciel demande de corriger.

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

Dans les fichiers twig il faut compléter des alt= (le texte alternatif aux images), mettre des légendes au tableau, etc. Une partie des « bugs » et « code smells » sont des fichiers système et ou code généré automatiquement. Je n'y touche pas.

Tâche 2 : ajouter une fonctionnalité :

Dans la page playlists, je dois ajouter comme fonctionnalité le trie des playlists par nombre de formations. Je commence par regarder comment est organiser la base de données et faire quelques tests.



```
/**
 * @Route("/playlists", name="playlists")
 * @return Response
 */
public function index(): Response {
    $playlists = $this->playlistRepository->findAllOrderByName('ASC');
    $categories = $this->categorieRepository->findAll();
    dd($playlists);
    return $this->render(self::PAGE_PLAYLISTS, [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}
```

Pour pouvoir trier les playlists par nombre de formation, je vais créer une méthode dans PlaylistRepository, sur la base de celle existante dans FormationRepository

```

if($champ == "nombre"){
    $playlists = $this->playlistRepository->findAllOrderByAmount($ordre);
}

```


Je rajoute une condition à sort de PlaylistControlleur, de façon à accéder la méthode findAllorderByAmount avec le champ « nombre » et je crée les boutons associé dans le fichier playlists.html.twig

```

<th class="text-center align-top" scope="col">
    formations par playlist<br />
    <a href="{{ path('playlists.sort', {champ:'nombre', ordre:'ASC'}) }}" class="btn btn-info
    btn-sm active" role="button" aria-pressed="true"><</a>
    <a href="{{ path('playlists.sort', {champ:'nombre', ordre:'DESC'}) }}" class="btn btn-info
    btn-sm active" role="button" aria-pressed="true">></a>
</th>

```

Le résultat sur navigateur :



MediaTek86

Des formations pour tous sur des outils numériques

[Accueil](#)
[Formations](#)
[Playlists](#)

playlist

<

>

filtrer

catégories

▼

formations par playlist

<

>

Cours Informatique embarquée	Cours	1 formation	Voir détail
Cours Merise/2	MCD Cours	1 formation	Voir détail
Cours Modèle relationnel et MCD	MCD Cours	1 formation	Voir détail
Cours de programmation objet	POO Cours	1 formation	Voir détail
Cours Composant logiciel	Cours	2 formations	Voir détail
Cours MCD MLD MPD	MCD Cours	2 formations	Voir détail
Cours MCD vs Diagramme de classes	MCD Cours	2 formations	Voir détail
Cours Cours...	...	2 formations	Voir détail

Mission 2 : coder la partie back-office :

Tâche 1 : gérer les formations :

Je crée un fichier pour la vue baseAdmin.html.twig sur le même modèle que basefront.

```
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="row">
9              <div class="col-6 text-left">
10                 
11             </div>
12             <div class="col-6 text-center">
13                 <h5>Back office de Mediatek Formation</h5>
14             </div>
15         </div>
16         <!-- menu -->
17         <nav class="navbar navbar-expand-lg navbar-light bg-light">
18             <div class="collapse navbar-collapse" id="navbarSupportedContent">
19                 <ul class="navbar-nav mr-auto">
20                     <li class="nav-item">
21                         <a class="nav-link" href="#">Formations</a>
22                     </li>
23                     <li class="nav-item">
24                         <a class="nav-link" href="#">Playlists</a>
25                     </li>
26                     <li class="nav-item">
27                         <a class="nav-link" href="#">Catégories</a>
28                     </li>
29                 </ul>
30             </div>
31         </nav>
32     </div>
33 {% endblock %}
34 {% block body %}{% endblock %}
35 {% block footer %}
36     <div class="container text-center">
37         <footer>
38             <hr>
39             <p><small><i>
40                 Consultez nos <a class="link-secondary" href="{{ path('cgu') }}">Conditions Générales d'Utilisation</a>
41             </i></small></p>
42         </footer>
43     </div>
44 {% endblock %}
45 {% block javascripts %}{% endblock %}
```

Je choisis de changer les couleurs pour que l'on puisse bien discerner la partie back-office du reste de l'application. Je commence par l'image incluse dans basefront.



MediaTek86

Des formations pour tous sur des outils numériques

Lors de cette tâche, je dois gérer les formations sur le modèle du front, mais avec des options en plus, pour cela je crée le contrôleur AdminFormations.

```
<?php
namespace App\Controller\Admin;

use App\Repository\CategorieRepository;
use App\Repository\FormationRepository;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class AdminFormationController extends AbstractController
{
    const PAGE_FORMATIONS = "pages/admin/formations.html.twig";

    const PAGE_FORMATION = "pages/admin/formation.html.twig";

    /**
     *
     * @var FormationRepository
     */
    private $formationRepository;

    /**
     *
     * @var CategorieRepository
     */
    private $categorieRepository;


    public function __construct(FormationRepository $formationRepository)
    {
        $this->formationRepository = $formationRepository;
        $this->categorieRepository = $categorieRepository;
    }

    /**
     * @Route("/admin", name="admin.formations")
     * @return Response
     */
    public function index(): Response{
        $formations = $this->formationRepository->findAll();
        $categories = $this->categorieRepository->findAll();
        return $this->render(self::PAGE_FORMATIONS, [
            'formations' => $formations,
            'categories' => $categories
        ]);
    }
}
```

[illegible]

Je construis ce contrôleur sur le modèle de celui utilisé dans la partie front. Je fais de même pour le fichier twig qui lui est associé.

Test de l'affichage et des fonctionnalités avec navigateur :

**MediaTek86**
Des formations pour tous
sur des outils numériques

Back office

Formations Playlists Catégories

formation
< >





filtrer

playlist
< >

filtrer

catégories
Android ▾

date
< >

Android Studio (complément n°13) : Permissions	Compléments Android (programmation mobile)	Android	29/09/2019	
Android Studio (complément n°12) : Positionner texte sur photo	Compléments Android (programmation mobile)	Android	17/09/2019	
Sujet E5 SLAM 2019 : cas RESTILOC mission3 (SQL et Android)	Sujet E5 SLAM 2019 métropole : cas RESTILOC	Android SQL	22/05/2019	
Android Studio (complément n°11) : Transformer une image en texte	Compléments Android (programmation mobile)	Android	18/12/2018	

J'ajoute les boutons demandés dans le fichier twig. Et je crée la méthode `suppr` de mon contrôleur `AdminFormationController` qui va me permettre de pouvoir gérer les suppressions.

```
/**
 * @Route("/admin/formations/suppr/{id}", name="admin.formation.suppr")
 * @param Formation $formation
 * @return Response
 */
public function suppr(Formation $formation): Response{
    $this->formationRepository->remove($formation, true);
    $this->addFlash(
        'alert',
        'Suppression de la formation ' . $formation->getTitle() . ' prise en compte';
    );
    return $this->redirectToRoute('admin.formations');
}
```

Après suppression d'une formation, la dite formation n'apparaît plus dans la base de données.

Afficher la zone SQL

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0002 seconde(s).)

`SELECT * FROM formation_categorie WHERE formation_categorie.formation_id = 1;`

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Opérations sur les résultats de la requête

 Créer une vue

Je dois maintenant créer la méthode associée au bouton modifier. Pour cela, je commence par créer un formulaire qui permettra d'interagir avec l'utilisateur en envoyant des informations à la vue depuis le contrôleur. Puis je le configure pour qu'il puisse gérer les différentes variables de classe de l'entité formation.

```
namespace App\Form;

use DateTime;
use App\Entity\Playlist;
use App\Entity\Categorie;
use App\Entity\Formation;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\DateTimeType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;

class FormationType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('publishedAt', DateTimeType::class, [
                'widget' => 'single_text',
                'label' => 'Date'
            ])
            ->add('title')
            ->add('description')
            ->add('videoId')
            ->add('playlist', EntityType::class, [
                'class' => Playlist::class,
                'choice_label' => 'name',
                'required' => false
            ])
            ->add('categories', EntityType::class, [
                'class' => Categorie::class,
                'choice_label' => 'name',
                'multiple' => true,
                'required' => false
            ])
            ->add('submit', SubmitType::class)
        ;
    }

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

Je crée les fichiers pour la vue
formation.html.twig dans pages/admin/ et
_admin.formation.form.html.twig à la
racine des templates.

```
{% extends "baseadmin.html.twig" %}
{% block body %}
    {{ include ('_admin.formation.form.html.twig') }}
{% endblock %}
```

```
templates > {{ _admin.formation.form.html.twig
1      {{ form_start(formFormation) }}
2          <div class="row">
3              <div class="col">
4                  <div class="row">
5                      <div class="col-auto">
6                          {{ form_row(formFormation.publishedAt) }}
7                      </div>
8                      <div class="col">
9                          {{ form_row(formFormation.title) }}
10                     </div>
11                     <div class="col">
12                         {{ form_row(formFormation.videoId) }}
13                     </div>
14                 </div>
15                 <br>
16                 {{ form_row(formFormation.description) }}
17                 <br>
18                 {{ form_row(formFormation.playlist) }}
19                 <br>
20                 {{ form_row(formFormation.categories) }}
21             </div>
22         </div>
23     {{ form_end(formFormation) }}
```

Puis la méthode edit de la classe AdminFormationController

```
/**
 * @Route("/admin/formation/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);
        $this->addFlash(
            'success',
            'Modification de la formation ' . $formation->getTitle() . ' prise en compte';
        );
        return $this->redirectToRoute('admin.formations');
    }

    return $this->render(self::PAGE_FORMATION, [
        'formation' => $formation,
        'formFormation' => $formFormation->createView()
    ]);
}
```



Formations Playlists Catégories

Date Title Video id

Description

Playlist

Categories

Le formulaire et la page de formations fonctionnent, je rajoute un bouton ajouter depuis admininformations.html.twig et la méthode ajouter depuis le contrôleur AdminFormationController

```
/**
 * @Route("/admin/formation/ajout", name="admin.formation.ajout")
 * @param Request $request
 * @return Response
 */
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);
        $this->addFlash(
            'success',
            'Ajout de la formation ' . $formation->getTitle() . ' prise en compte');
        return $this->redirectToRoute('admin.formations');
    }

    return $this->render(self::PAGE_FORMATION, [
        'formation' => $formation,
        'formFormation' => $formFormation->createView()
    ]);
}
```

```
{% endfor %}
<p class="text-end">
    <a href="{{ path('admin.formation.ajout') }}" class="btn btn-secondary">
        Ajouter une formation
    </a>
</p>
<table class="table table-striped">
```

Tâche 2 : gérer les playlists :

Création du contrôleur AdminPlaylistController et de la vue sur le modèle de playlists.

```
templates > pages > admin > playlists.html.twig
```

```
1 {% extends "baseadmin.html.twig" %}
2 {% block body %}
3     <table class="table table-striped">
4         <caption>tableau des playlists</caption>
5         <thead>
6             <tr>
7                 <th class="text-left align-top" scope="col">
8                     playlist<br />
9                     <a href="{{ path('playlists.sort', {'champ':'name', 'ordre':'ASC'}) }}" class="btn btn-info btn-sm active" role="button">
10                         info
11                     </a>
12                 </th>
13             </tr>
14         </thead>
15     </table>
16 %}
```

```

/**
 * @Route("/admin/playlist/suppr/{id}", name="admin.playlist.suppr")
 * @param Playlist $playlist
 * @return Response
 */
public function suppr(Playlist $playlist): Response
{
    if(count($playlist->getFormations()) > 0){
        $this->addFlash(
            'alert',
            'Vous ne pouvez pas supprimer la playlist ' . $playlist->getName() . ' car elle contient des formations'
        );
        return $this->redirectToRoute('admin.playlists');
    }

    $this->playlistRepository->remove($playlist, true);
    $this->addFlash(
        'alert',
        'Suppression de la playlist ' . $playlist->getName() . " prise en compte"
    );
    return $this->redirectToRoute('admin.playlists');
}

```

Copie des méthodes du contrôleur PlaylistController dans AdminPlaylistController, modification du chemin des fichiers twig.

Maintenant, je dois créer mes boutons côté vu, leur contrôleur et les méthodes associées. Je pars du modèle de ma vue

Vous ne pouvez pas supprimer la playlist Bases de la programmation (C#) car elle contient des formations

Ajouter une playlist

playlist

<

>

filtrer

catégories

▼

formations par playlist

<

>

Bases de la programmation (C#)	C# POO	71 formations	Modifier	Supprimer
--------------------------------	--------	---------------	----------	-----------

précédente pour les boutons. Le bouton modifier n'est pas demandé, mais étant donné que l'on ne doit pouvoir supprimer une playlist que quand elle n'est affiliée à aucune formation, il va être nécessaire.

La fonctionnalité est OK, je passe à l'ajout de playlist. Je crée le formulaire.

```
PS C:\wamp64\www\mediatekformation> php bin/console make:form

The name of the form class (e.g. GrumpyPizzaType):
> PlaylistType

The name of Entity or fully qualified model class name that the new form will be bound to (empty for none):
> Playlist

created: src/Form/PlaylistType.php


Success!

Next: Add fields to your form and start using it.
Find the documentation at https://symfony.com/doc/current/forms.html
PS C:\wamp64\www\mediatekformation>
```

Une formation ne pouvant être associée qu'à une playlist, je modifie l'affichage de ma vue pour prévenir l'utilisateur. Je commence par créer une méthode `__toString` à Playlist, puis à Formation pour l'incorporer. J'ajoute aussi un message pour avertir l'utilisateur dans la vue.

```
<?php _admin.playlist.form.html.twig
{{ form_start(formPlaylist) }}
<div class="row">
    {{ form_row(formPlaylist.name) }}
    <br>
    {{ form_row(formPlaylist.description) }}
    <br>
    {{ form_row(formPlaylist.formations) }}
    <h5> ⚠ Une formation ne peut être rattachée qu'à une playlist ⚠ </h5>
    <br>
    {{ form_row(formPlaylist.submit, {'attr': {'onclick': 'return confirm(Etes-vous sûr?)'}}) }}
</div>
{{ form_end(formPlaylist) }}
```

Le résultat depuis un navigateur :



MediaTek86
Des formations pour tous
sur des outils numériques

🚧 Back office 🚧

Formations Playlists Catégories

Name

Bases de la programmation (C#)

Description

Exemples progressifs de programmes en procédural, événementiel et objet sous Visual Studio (version Entreprise 2017).
Prérequis : aucun

Formations

Eclipse n°4 : WindowBuilder Playlist : pouet2
Eclipse n°3 : GitHub et Eclipse
Eclipse n°2 : rétroconception avec ObjectAid Playlist : pouet2
Eclipse n°1 : installation de l'IDE Playlist : Eclipse et Java

⚠ Une formation ne peut être rattachée qu'à une playlist ⚠

Submit

Consultez nos [Conditions Générales d'Utilisation](#)

Tâche 3 : gérer les catégories :

Construction du contrôleur AdminCategoriesController que je déplace dans le dossier Admin, sur le modèle des contrôleurs précédent.

```
PS C:\wamp64\www\mediatekformation> php bin/console make:controller

Choose a name for your controller class (e.g. OrangeGnomeController):
> AdminCategoriesController

created: src/Controller/AdminCategoriesController.php
created: templates/admin_categories/index.html.twig

Success!

Next: Open your new controller class and add some pages!
PS C:\wamp64\www\mediatekformation> 
```

```

namespace App\Controller\Admin;

use App\Entity\Categorie;
use App\Form\CategorieType;
use App\Repository\CategorieRepository;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class AdminCategoriesController extends AbstractController
{
    const PAGE_CATEGORIES = "pages/admin/categories.html.twig";

    const PAGE_CATEGORIE = "pages/admin/categorie.html.twig";

    /**
     *
     * @var CategorieRepository
     */
    private $categorieRepository;

    public function __construct(CategorieRepository $categorieRepository) {
        $this->categorieRepository = $categorieRepository;
    }

    /**
     * @Route("/admin/categories", name="admin.categories")
     * @return Response
     */
    public function index(): Response{
        $categories = $this->categorieRepository->findAllOrderByName('ASC');
        return $this->render(self::PAGE_CATEGORIES, [
            'categories' => $categories
        ]);
    }
}

```

Création du formulaire :

```

The name of Entity or fully qualified model class name that the new form will be bound to (empty for none):
> Categorie

```

```

created: src/Form/CategorieType.php

```

Success!

```

Next: Add fields to your form and start using it.
Find the documentation at https://symfony.com/doc/current/forms.html

```

```

class CategorieType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('name', TextType::class, [
                'required' => true
            ])
            ->add('formations', EntityType::class, [
                'class' => Formation::class,
                'multiple' => true,
                'required' => false,
            ])
            ->add('submit', SubmitType::class, [
                'label' => 'Valider'
            ])
        ;
    }

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

```

Je modifie mes formulaires avec l'ajout d'attr: { 'size' : 15 } (pour que l'élément HTML select ai 15 lignes affichées) et 'onclick' pour demander une confirmation à l'utilisateur lors de la validation (le 'submit').

```

{{ form_start(formCategorie) }}
<div class="row">
    {{ form_row(formCategorie.name) }}
    <br>
    {{ form_row(formCategorie.formations, {'attr': {'size': 15}}) }}
    <br>
    <br />
    <br />
    {{ form_row(formCategorie.submit, {'attr': {'onclick': 'return confirm(Etes-vous sûr?)"'}}) }}
</div>
    {{ form_end(formCategorie) }}
</div>
{{ form_end(formCategorie) }}

```

Je rajoute les boutons annuler et supprimer à mes formulaires.

```

> _admin.categorie.form.html.twig
{{ form_start(formCategorie) }}
    <div class="row">
        {{ form_row(formCategorie.name) }}
        <br>
        {{ form_row(formCategorie.definitions, {'attr': {'size': 15}}) }}
        <br>
        <br />
        <br />
        {{ form_row(formCategorie.submit, {'attr': {'onclick': 'return confirm(Etes-vous sûr?)'}}) }}
    </div>
    {{ form_end(formCategorie) }}
    {% if categorie.id != null %}
        <div class="d-grid gap-2 d-md-flex justify-content-md-end">
            <a href="{{ path('admin.categorie.edit', {id: categorie.id}) }}" class="btn btn-warning me-md-2">
                Annuler
            </a>
            <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>
        </div>
    {% endif %}
</div>
{{ form_end(formCategorie) }}

```

Tâche 4 : ajouter l'accès authentification :

Téléchargement de Keycloak 19.0.1 (<https://github.com/keycloak/keycloak/releases/>)

Lancement du client en local avec la commande «kc.bat start-dev» avec la console

```

C:\keycloak\bin>kc.bat start-dev
2023-10-04 15:05:35,231 INFO [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings:
FrontEnd: <request>, Strict HTTPS: false, Path: <request>, Strict BackChannel: false, Admin: <request>, Port: -1, Proxie
d: false
2023-10-04 15:05:35,973 INFO [org.keycloak.common.crypto.CryptoIntegration] (main) Detected crypto provider: org.keyclo
ak.crypto.def.DefaultCryptoProvider
2023-10-04 15:05:36,823 WARN [org.infinispan.PERSISTENCE] (keycloak-cache-init) ISPN000554: jboss-marshalling is deprec
ated and planned for removal
2023-10-04 15:05:36,884 WARN [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan int
ernal caches as no global state enabled
2023-10-04 15:05:36,891 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org
.infinispan.jboss.marshalling.core.JBossUserMarshaller'
2023-10-04 15:05:37,070 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000128: Infinispan version: Infinispa
n 'Triskaidekaphobia' 13.0.9.Final
2023-10-04 15:05:40,941 INFO [org.keycloak.connections.infinispan.DefaultInfinispanConnectionProviderFactory] (main) No
de name: node_159941, Site name: null
2023-10-04 15:05:41,468 INFO [io.quarkus] (main) Keycloak 19.0.1 on JVM (powered by Quarkus 2.7.6.Final) started in 7.2
50s. Listening on: http://0.0.0.0:8080
2023-10-04 15:05:41,468 INFO [io.quarkus] (main) Profile dev activated.
2023-10-04 15:05:41,468 INFO [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, jdbc-mariadb
, jdbc-mssql, jdbc-mysql, jdbc-oracle, jdbc-postgresql, keycloak, logging-gelf, narayana-jta, reactive-routes, resteasy,
resteasy-jackson, smallrye-context-propagation, smallrye-health, smallrye-metrics, vault, vertx]
2023-10-04 15:05:41,471 WARN [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode.
DO NOT use this configuration in production.

```



Welcome to Keycloak



Administration Console >

Centrally manage all aspects of the Keycloak server

Il faut créer un admin pour se connecter au logiciel en local. Après, nous pouvons configurer Keycloak pour notre application. Je commence par créer le « royaume » Myapplis. Puis je crée un nouveau client, mediatek86 pour l'application Symfony Mediatek Formation.

Je configure le client et récupère « l'app secret », que je renseignerai de même que le client dans le fichier .env a la racine de mon projet Symfony

```
KEYCLOAK_SECRET=nHWPw5kcwErtDNSb3DPnALzXJiV57HRw
KEYCLOAK_CLIENTID=mediatek86
KEYCLOAK_APP_URL=http://localhost:8080
```

Je crée une classe User dans le projet Symfony.

```
PS C:\wamp64\www\MediaTek86> php bin/console make:user

The name of the security user class (e.g. User) [User]:
>

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
>

Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:
>

Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed by some other system (e.g. a single sign-on server).
Does this app need to hash/check user passwords? (yes/no) [yes]:
>

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml

Success!

Next Steps:
- Review your new App\Entity\User class.
- Use make:entity to add more fields to your User entity and then run make:migration.
- Create a way to authenticate! See https://symfony.com/doc/current/security.html
PS C:\wamp64\www\MediaTek86>
```

```
PS C:\wamp64\www\MediaTek86> php bin/console make:entity User

Your entity already exists! So let's add some new fields!

New property name (press <return> to stop adding fields):
> keycloakId

Field type (enter ? to see all types) [integer]:
> string

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
> yes

updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!

Next: When you're ready, create a migration with php bin/console make:migration
```

J'effectue la commande `php bin/console make:migration` (pour me permettre d'avoir un nouveau fichier migration avec des commandes SQL, ici permettant la création de la table user avec les caractéristiques de l'entité). Avec `php bin/console d:m:m` (doctrine migrations migrate) j'exécute les instructions des fichiers de migration n'ayant pas encore été utilisés.

Installation des bundles (paquets, extensions) nécessaire à la connexion et authentification avec Keycloak. Je lance les deux commandes :

- `composer require knpuniversity/oauth2-client-bundle 2.10`
- `composer require stevenmaguire/oauth2-keycloak 3.1 --with-all-dependencies`

Configuration du fichier `config/packages/knpu_oauth2_client.yaml` créé avec `oauth2-client-bundle`.

```
config > packages > ! knpu_oauth2_client.yaml
1 knpu_oauth2_client:
2   clients:
3     keycloak:
4       type: keycloak
5       auth_server_url: '%env(KEYCLOAK_APP_URL)%'
6       realm: 'myapplis'
7       client_id: '%env(KEYCLOAK_CLIENTID)%'
8       client_secret: '%env(KEYCLOAK_SECRET)%'
9       redirect_route: 'oauth_check'
```

Configuration du firewall `config/packages/security.yaml`.

```

config > packages > ! security.yaml
1  security:
2      enable_authenticator_manager: true
3      # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
4      password_hashers:
5          Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
6      # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
7      providers:
8          # used to reload user from session & other features (e.g. switch_user)
9          app_user_provider:
10             entity:
11                 class: App\Entity\User
12                 property: email
13     firewalls:
14         dev:
15             pattern: ^/(_(profiler|wdt)|css|images|js)/
16             security: false
17         main:
18             lazy: true
19             provider: app_user_provider
20             form_login:
21                 login_path: oauth_login
22
23             # activate different ways to authenticate
24             # https://symfony.com/doc/current/security.html#the-firewall
25
26             # https://symfony.com/doc/current/security/impersonating\_user.html
27             # switch_user: true
28
29             # Easy way to control access for large sections of your site
30             # Note: Only the *first* access control that matches will be used
31             access_control:
32                 - { path: ^/admin, roles: ROLE_ADMIN }
33                 # - { path: ^/profile, roles: ROLE_USER }
34
35     when@test:

```

Le chemin du formulaire d'authentification est assigné à `oauth_login`. L'accès control (contrôle d'accès) a aussi été modifié, en enlevant le commentaire '#' en début de ligne je spécifie au framework Symfony que chaque contrôleur ayant une route incluant « /admin » nécessitera un statut « ROLE_ADMIN » de l'utilisateur (donc connexion authentification). Création du contrôleur `OauthController` avec la commande :

`php bin/console make:controller OAuthController --no-template`

```

PS C:\wamp64\www\MediaTek86> php bin/console make:controller OAuthController --no-template

created: src/Controller/OAuthController.php

Success!

Next: Open your new controller class and add some pages!
PS C:\wamp64\www\MediaTek86>

```

Configuration du contrôleur.

```

namespace App\Controller;

use KnpU\OAuth2ClientBundle\Client\ClientRegistry;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class OAuthController extends AbstractController
{
    /**
     * @Route("/oauth/login", name="oauth_login")
     */
    public function index(ClientRegistry $clientRegistry): RedirectResponse{
        return $clientRegistry->getClient('keycloak')->redirect();
    }

    /**
     * @Route("/oauth/callback", name="oauth_check")
     */
    public function connectCheckAction(Request $request, ClientRegistry $clientRegistry){

        * Description of KeycloakAuthenticator
        *
        * @author emds
        */
class KeycloakAuthenticator extends OAuth2Authenticator implements AuthenticationEntryPointInterface {
    private $clientRegistry;
    private $entityManager;
    private $router;

    public function __construct(ClientRegistry $clientRegistry,
        EntityManagerInterface $entityManager, RouterInterface $router){
        $this->clientRegistry = $clientRegistry;
        $this->entityManager = $entityManager;
        $this->router = $router;
    }

    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 avec Keycloak
                $email = $keycloakUser->getEmail();
                /** @var User $userInDatabase */
                $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) 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;
            })
        );
    }
}

```

Création de
KeycloakAuthenticator
SIO SLAM Quentin Saujot

dans /src/ . Configuration du fichier en utilisant le modèle de celui utilisé en travaux pratique.

L'authentification étant géré par Keycloak, à partir du moment où la connexion est validée (utilisateur et mot de passe OK) il existe trois cas de figure.

- L'utilisateur c'est déjà connecté a Mediatek Formation et existe dans la base de données de l'application.
- L'utilisateur (son email) a été déjà créé/existe dans la data base de données, mais le keycloakId n'existe pas, ou n'est pas le même. Dans ce cas les informations sont enregistrées dans la base de données pour la prochaine connexion.
- L'utilisateur n'existe pas, création d'un nouvel « User » grant (« ROLE_ADMIN ») qui sera sauvegardé dans la base de données.

Modification du fichier security.yaml pour qu'il prenne en compte le fichier, nouvellement créé pour l'authentification Keycloak. Ajout du chemin 'logout' (dans le même fichier de configuration) pour permettre la déconnexion de l'utilisateur.

```
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
  main:
    lazy: true
    entry_point: form_login
    # provider: app_user_provider
    form_login:
      login_path: oauth_login
    custom_authenticators:
      - App\Security\KeycloakAuthenticator
    logout:
      path: logout
```

Mission 3 : tester et documenter :

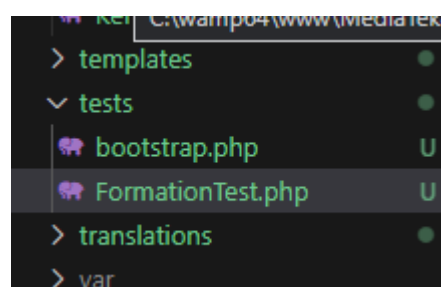
Tâche 1 : gérer les tests :

Je crée une base mediatekformation_test pour les tests.



Test unitaires :

Création de FormationTest qui me permettra de contrôler le fonctionnement de la méthode qui retourne la date de parution au format string.



```

use App\Entity\Formation;
use PHPUnit\Framework\TestCase;

class FormationTest extends TestCase
{
    public function testGetPublishedAtString()
    {
        $Formation = new Formation();
        $Formation->setTitle("Formation test");
        $Formation->setPublishedAt(new \DateTime("2022-04-14"));
        $this->assertEquals("14/04/2022", $Formation->getPublishedAtString());
    }
}

```

Tests d'intégration sur les règles de validation :

```

FormationValidationTest.php U X
tests > Validations > FormationValidationTest.php > FormationValidationsTest > testValidDateFormation
4 use App\Entity\Formation;
5 use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;
6 use Symfony\Component\Validator\Validator\ValidatorInterface;
7
8
9 class FormationValidationsTest extends KernelTestCase {
10
11     /**
12      * Création d'un objet de type Formation, avec informations minimales
13      * @return Formation
14      */
15     public function getFormation(): Formation{
16         return (new Formation())
17             ->setTitle("Formation test");
18     }
19
20     /**
21      * Utilisation du Kernel pour tester une règle de validation
22      * @param Formation $formation
23      * @param int $nbErreursAttendues
24      * @param string $message
25      */
26     public function assertErrors(Formation $formation, int $nbErreursAttendues, string $message=""){
27         self::bootKernel();
28         $validator = self::getContainer()->get(ValidatorInterface::class);
29         $error = $validator->validate($formation);
30         echo $error."\n";
31         $this->assertCount($nbErreursAttendues, $error, $message);
32     }
33
34     public function testValidDateFormation(){
35         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('yesterday')), 0, " 04/10/2023 devrait réussir");
36         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('first day of January 2008')), 0, " 01/01/2008 devrait réussir");
37         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('last sat of July 2008')), 0, " 26/07/2008 devrait réussir");
38     }
39
40     public function testNonValidDateFormation(){
41         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('06/08/2026')), 1, " devrait échouer");
42         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('11/02/2025')), 1, " devrait échouer");
43         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('06/11/2028')), 1, " devrait échouer");
44         $this->assertErrors($this->getFormation()->setPublishedAt(new \DateTime('09/07/2025')), 1, " devrait échouer");
45     }
46
47 }

```

Je rajoute un assert `LessThanOrEqual('today')` à la date `publishedAt` de `Formation`. Si une date renseignée est postérieure à la date actuelle, la donnée ne sera pas acceptée et le test ne sera pas valide.

```

/**
 * @ORM\Column(type="datetime", nullable=true)
 * @Assert\LessThanOrEqual('today')
 */
private $publishedAt;

```

Tests d'intégration sur les Repository :

Pour tester FormationRepository je crée FormationRepositoryTest qui hérite de KernelTestCase.

```
namespace App\Tests\Validations\Repository;

use App\Entity\Formation;
use Doctrine\ORM\EntityManager;
use App\Repository\FormationRepository;
use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

class FormationRepositoryTest extends KernelTestCase
{
    private \Doctrine\ORM\EntityManager $entityManager;

    protected function setUp(): void
    {
        $kernel = self::bootKernel();

        $this->entityManager = $kernel->getContainer()
            ->get('doctrine')
            ->getManager();
    }
}
```

Je commence par la méthode testNbFormations qui vérifie le nombre de formation. Je regarde combien j'ai de formations avec une commande SQL pour définir le « assertEquals ».

```
public function testNbFormations()
{
    $repository = $this->recupRepository();
    $nbFormation = $repository->count([]);
    $this->assertEquals( 237, $nbFormation);
}
```

Puis je teste en ajoutant une formation que le nombre de formation correspond. De même pour la suppression d'une formation.

```
public function newFormation() : Formation
{
    $formation = (new Formation())
        ->setTitle("Un titre")
        ->setDescription("Description blabla")
        ->setPublishedAt(new \DateTime("yesterday"));
    return $formation;
}

public function testAddFormation()
{
    $repository = $this->recupRepository();
    $formation = $this->newFormation();
    $nbFormation = $repository->count([]);

    $this->entityManager->persist($formation);
    $this->entityManager->flush();
    $this->assertEquals($nbFormation + 1, $repository->count([]), "erreur lors de l'ajout");
}

public function testSupprFormation()
{
    $repository = $this->recupRepository();

    $nbFormation = $repository->count([]);
    $formation = $repository->findOneBy(['title' => "Un titre"]);
    $formation = $this->entityManager->merge($formation);
    $this->entityManager->remove($formation);
    $this->entityManager->flush();
    $this->assertEquals($nbFormation - 1, $repository->count([]), "erreur lors de la suppression");
}
```

✓ Affichage des lignes 0 - 0 (total de 1, traitement

```
SELECT count(*) FROM `formation`;
```

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer s

☐ Tout afficher | Nombre de lignes : 25

Options supplémentaires

count(*)

237

Tests fonctionnels :

Je mets en place des tests fonctionnel pour tester mes contrôleurs. Je commence par créer un dossier « Controller » dans mon dossier « tests ». Puis je crée FormationControllerTest.

```
namespace App\tests\Controller;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class FormationsControllerTest extends WebTestCase
{
    public function testAccesPage()
    {
        $client = static::createClient();
        $client->catchExceptions(false);
        $client->request('GET', '/formations');
        $response = $client->getResponse();
        $this->assertEquals(Response::HTTP_OK, $response->getStatusCode());
    }
}
```

Mon premier test vérifie que la page /formations renvoie une réponse OK (code 200). Je teste aussi le contenu de la page. Avec assertSelectorTextContains je vérifie que le premier « th » contient « formation ». Avec assertCount(5, \$crawler->filter('th')) ; je vérifie qu'il y a bien 5 « th » (élément HTML pour constituer l'entête de tableaux) affiché dans ma page.

```
public function testContenuPage()
{
    $client = static::createClient();
    $crawler = $client->request('GET', '/formations');
    $this->assertSelectorTextContains('th', 'formation');
    $this->assertCount(5, $crawler->filter('th'));
}
```

Puis je teste un lien de ma page, voir si il m'amène au bon endroit. Les liens des formations dans la page /formations étant des images. Pour cliquer sur la première formation proposée, je compte le nombre de liens qui le précède.

```
public function testLinkFormation()
{
    $client = static::createClient();
    $crawler = $client->request('GET', '/formations');
    //click sur un lien ( l'image de la première formation)
    $link = $crawler->filter('a')->eq(9)->attr('href');
    $crawler = $client->request('GET', $link);
    // récupération du résultat du clic
    $response = $client->getResponse();
    //control si le lien existe
    $this->assertEquals(Response::HTTP_OK, $response->getStatusCode());
    //récupération de la route et controle qu'elle est correcte
    $uri = $client->getRequest()->server->get("REQUEST_URI");
    $this->assertEquals('/formations/formation/1', $uri);
}
```

Tests de compatibilité :

Je crée un scénario de test sur Sélénium avec Firefox.

The screenshot shows the Selenium IDE interface with a project named 'Mediatek86'. The test scenario 'test1' consists of 25 steps. The first step is 'open' with target 'http://192.168.1.10:8000/'. The subsequent steps involve clicking on various elements identified by CSS selectors, such as 'css=col:nth-child(1) > row.card-img-top', 'linkText=Formations', 'name=recherche', 'id=recherche', 'css=option:nth-child(3)', 'css=option:nth-child(2)', 'linkText=Playlists', 'linkText=<', 'id=recherche', 'css=option:nth-child(3)', 'css=option:nth-child(2)', and 'css=option:nth-child(3)'. The 'Value' column contains values like 'CF', 'label=CF', 'label=Android', and 'label=CF'. The 'Log' tab at the bottom shows the execution results, indicating that the test completed successfully.

Step	Command	Target	Value
1	open	http://192.168.1.10:8000/	
2	set window size	1122x691	
3	click	css=col:nth-child(1) > row.card-img-top	
4	click	linkText=Formations	
5	click	name=recherche	
6	click	name=recherche	
7	double click	name=recherche	
8	type	name=recherche	CF
9	click	css=col:nth-child(1) > form-inline .btn	
10	click	css=col:nth-child(2) > .btn:nth-child(3)	
11	click	css=col-center > .btn:nth-child(2)	
12	click	id=recherche	
13	select	id=recherche	label=CF
14	click	css=option:nth-child(3)	
15	click	id=recherche	
16	select	id=recherche	label=Android
17	click	css=option:nth-child(2)	
18	click	linkText=Playlists	
19	click	linkText=>	
20	click	linkText=<	
21	click	id=recherche	
22	select	id=recherche	label=CF
23	click	css=option:nth-child(3)	
24	click	css=col-center > .btn:nth-child(2)	
25	click	css=col-center > .btn:nth-child(3)	

Scénario que je relance avec Chrome pour effectuer le teste de compatibilité.

The screenshot shows the Selenium IDE interface with a project named 'Mediatek86'. The test scenario 'test1' consists of 25 steps, identical to the previous one. The 'Log' tab at the bottom shows the execution results, indicating that the test completed successfully.

Step	Command	Target	Value
1	open	http://192.168.1.10:8000/	
2	set window size	1122x691	
3	click	css=col:nth-child(1) > row.card-img-top	
4	click	linkText=Formations	
5	click	name=recherche	
6	click	name=recherche	
7	double click	name=recherche	
8	type	name=recherche	CF
9	click	css=col:nth-child(1) > form-inline .btn	
10	click	css=col:nth-child(2) > .btn:nth-child(3)	
11	click	css=col-center > .btn:nth-child(2)	
12	click	id=recherche	
13	select	id=recherche	label=CF
14	click	css=option:nth-child(3)	
15	click	id=recherche	
16	select	id=recherche	label=Android
17	click	css=option:nth-child(2)	
18	click	linkText=Playlists	
19	click	linkText=>	
20	click	linkText=<	
21	click	id=recherche	
22	select	id=recherche	label=CF
23	click	css=option:nth-child(3)	
24	click	css=col-center > .btn:nth-child(2)	
25	click	css=col-center > .btn:nth-child(3)	

Tâche 2 : créer la documentation technique :

Pour générerr la documentation technique j'utilise phpDocumentor

mediatekformation

Namespaces

- App
 - Controller
 - Entity
 - Form
 - Repository
 - Security

Packages

- Application

Reports

- Deprecated
- Errors
- Markers

Indices

- Files

Documentation

Table of Contents

Packages

P [Application](#)

Namespaces

N [App](#)

Tâche 2 : créer la documentation utilisateur :

Vidéo présentant les fonctionnalités du site disponible sur le portfolio

Mission 4 : déployer le site :

Tâche 1 : déployer le site :

Le site est déployé sur une VM Linux sur Azure avec un serveur Apache2

```
root@localhost:/var/www/html/MediaTek86# nano .env
root@localhost:/var/www/html/MediaTek86# composer require symfony/apache-pack
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? y
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been updated
Running composer update symfony/apache-pack
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking symfony/apache-pack (v1.0.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Installing symfony/apache-pack (v1.0.1): Extracting archive
Generating optimized autoload files
```

Je vérifie l'installation.

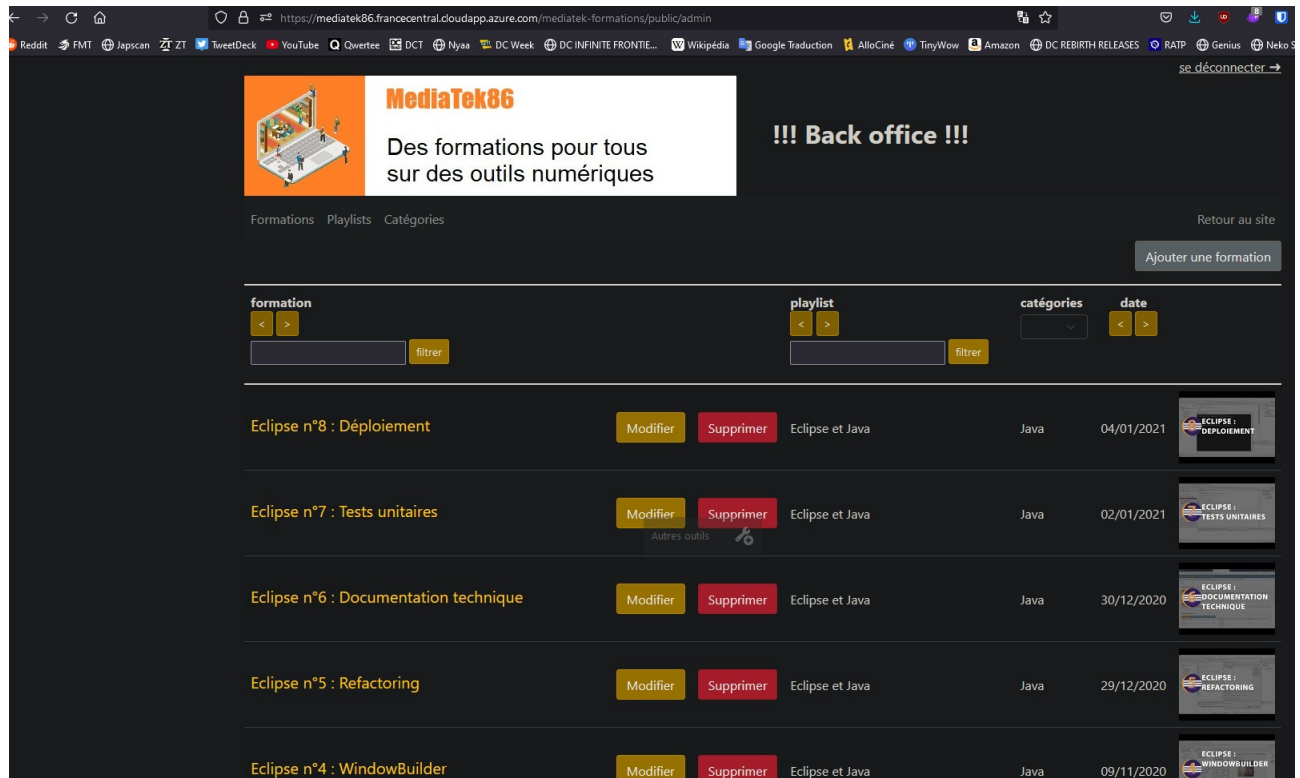
```
root@localhost:/var/www/html/MediaTek86# composer check-platform-reqs
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]?
Checking platform requirements for packages in the vendor dir
composer-plugin-api 2.6.0 success
composer-runtime-api 2.2.2 success
ext-ctype 7.4.33 success
ext-filter 7.4.33 success
ext-iconv 7.4.33 success
ext-json 7.4.33 success
ext-tokenizer 7.4.33 success
ext-xml 7.4.33 success
php 7.4.33 success
```

Installation de Keycloak :

Keycloak est installé dans une autre VM Linux en suivant ce tuto :

<https://github.com/CNED-SLAM/mediatekformation/wiki/Keycloak-en-ligne-et-en-HTTPS#solution-1--keycloak-dans-une-vm-linux>

Le site est en ligne et accessible



Tâche 2 : gérer la sauvegarde et la restauration de la BDD :

La sauvegarde de la BDD est assurée par le script backup.sh lancé via un cronjob tous les jours à 00:00

```
#!/bin/bash
DATE=$(date +%Y-%m-%d)
find /var/www/html/rest_mediatekdocuments/savebdd/bdd* -mtime -6 -exec rm {} \;
mysqldump -u Alpha-Echo -p'$Dig$Gilt$' --databases mediatekformation --single-transaction | gzip > /var/www/html/mediatek-formations/savebdd/bddbackup_${DATE}.sql.gz
```

Ce script définit une variable avec la date du jour au format yyyy-mm-dd ; puis recherche dans le dossier savebdd des sauvegardes de plus de 6 jours pour les supprimer puis il fait une sauvegarde complète de la BDD.

Tâche 3 : mettre en place le déploiement continu :

Le site est accessible à l'adresse suivante :

<https://mediatek86.francecentral.cloudapp.azure.com/mediatek-formations/public/>