

22/01/2023

ATELIER PROFESSIONNEL

MEDIATEKFORMATION

Compte-Rendu

Réalisé par : Alexandre Schoukroun
BTS SIO-SLAM 2^{ÈME} ANNÉE

Table des matières

MISSION	2
LANGAGE ET OUTILS	3
ETAPE 1. NETTOYER ET OPTIMISER LE CODE EXISTANT.....	4
A. NETTOYER LE CODE	4
B. RESPECTER LES BONNES PRATIQUES DE CODAGE.....	6
C. AJOUTER UNE FONCTIONNALITE.....	11
ETAPE 2. CODER LA PARTIE BACK-OFFICE	16
A. GERER LES FORMATIONS	17
B. GERER LES PLAYLISTS	24
C. GERER LES CATEGORIES	30
D. AJOUTER L'ACCES AVEC AUTHENTIFICATION.....	34
ETAPE 3.TESTER ET DOCUMENTER	44
A. GERER LES TESTS	45
B. CREER LA DOCUMENTATION TECHNIQUE	48
C. CREER LA DOCUMENTATION UTILISATEUR.....	50
ETAPE 4. DEPLOYER LE SITE ET GERER LE DEPLOIEMENT CONTINU...	52
A. DEPLOYER LE SITE.....	52
B. GERER LA SAUVEGARDE ET LA RESTAURATION DE LA BDD.....	57
C. METTRE EN PLACE LE DEPLOIEMENT CONTINU	60
BILAN.....	62




CONTEXTE

MediaTek86, un réseau qui gère les médiathèques de la Vienne, et qui a pour rôle de fédérer les prêts de livres, DVD et CD et de développer la médiathèque numérique pour l'ensemble des médiathèques du département .

MISSION

Il m'a été confié le développement du site de formation Mediatekformation qui permet à ses utilisateurs de regarder des formations sur l'informatique en ligne.



LANGAGE ET OUTILS

LANGAGE DE
PROGRAMMATION

HTML
SQL

IDE

NETBEANS

SERVEUR

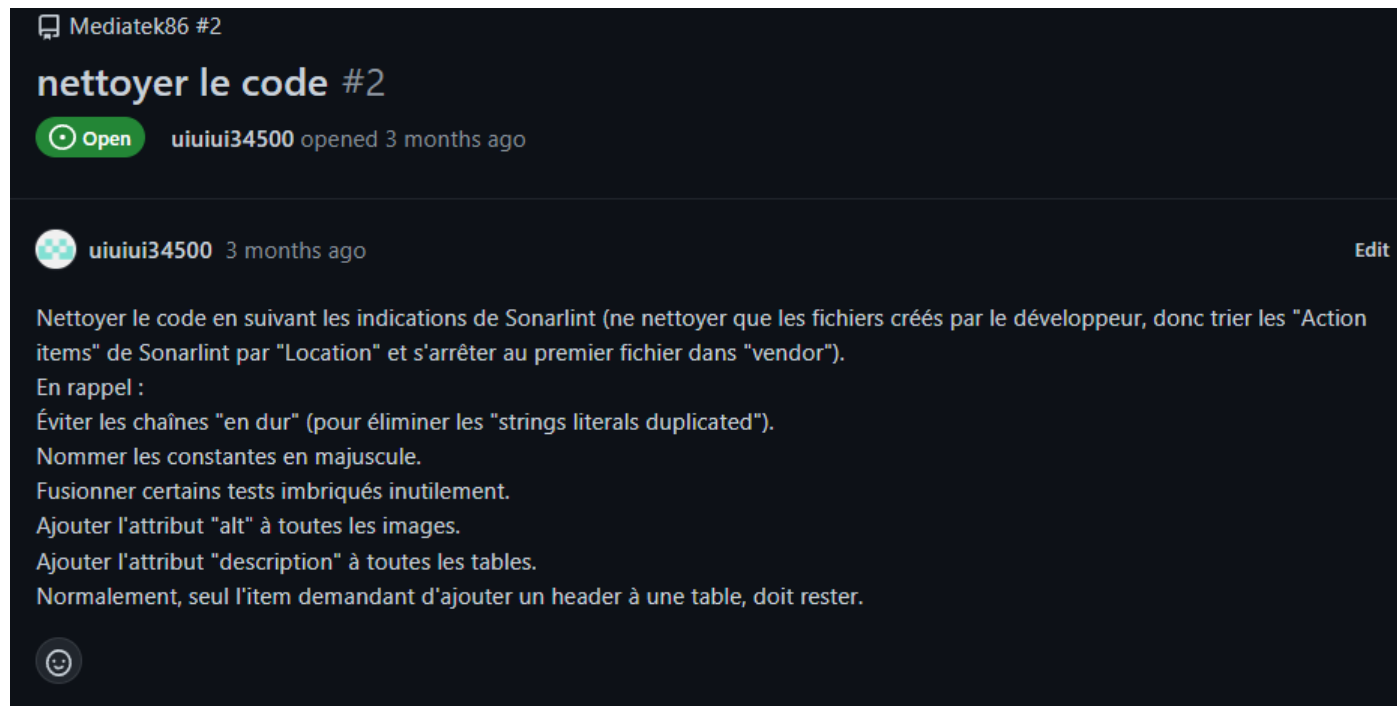
WAMP SERVER
MYSQL
APACHE
PHP

VERSIONNING

GITHUB

ETAPE 1. NETTOYER ET OPTIMISER LE CODE EXISTANT

A. NETTOYER LE CODE



Dans cette tache il m'as été confié de nettoyer le code en utilisant les indication de Sonarlint , c'est-à-dire d'éviter les chaines en dur , nommer les constantes en majuscule , Fusionner certains test imbriqué inutilement et l'ajout de l'attribut alt a toutes les images.

Extrait du code :

Ajout de constantes pour accéder aux pages du site :

↑	@@ -15,6 +15,9 @@
15	15 */
16	16 + class FormationsController extends AbstractController {
17	17
18	18 +
19	19 + const PAGE_FORMATIONS = "pages/formations.html.twig";
20	20 + const PAGE_FORMATION = "pages/formation.html.twig";
18	21 /**
19	22 *
20	23 * @var FormationRepository
↕	@@ -39,7 +42,7 @@ function __construct(FormationRepository \$formationRepository
39	42 public function index(): Response{
40	43 \$formations = \$this->formationRepository->findAll();
41	44 \$categories = \$this->categorieRepository->findAll();
42	- return \$this->render("pages/formations.html.twig", [
45	+ return \$this->render(self::PAGE_FORMATIONS, [
43	46 'formations' => \$formations,
44	47 'categories' => \$categories
45	48]);

Fusion de test imbriqué inutilement :

88	
89	public function removeFormation(Formation \$formation): self
90	{
-	if (\$this->formations->removeElement(\$formation)) {
91 +	if (\$this->formations->removeElement(\$formation) && \$formation->getPlaylist() === \$this) {
92	// set the owning side to null (unless already changed)
-	if (\$formation->getPlaylist() === \$this) {
-	\$formation->setPlaylist(null);
-	}
93 +	\$formation->setPlaylist(null);
94 +	
95	}

Ajout de l'attribut alt à toutes les images :

```

20         <div class="col-md-auto">
21             {% if formation.miniature %}
22                 <a href="{{ path('formations.showone', {id:formation.id}) }}">
23 -                 
23 +                 
24             </a>
25             {% endif %}
26         </div>

```

B. RESPECTER LES BONNES PRATIQUES DE CODAGE

Mediatek86 #1

respecter les bonnes pratiques de codage #1

Closed
 uiuiui34500 opened 3 months ago

uiuiui34500 3 months ago
 Edit

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.

Preview
 H B I ≡ <> 🔗 ≡ ≡ ≡ @ ↻ ↩

Leave a comment

Markdown is supported

Paste, drop, or click to add files

Reopen

Comment

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 .

Extrait de code avant modification pour FormationRepository :

```
* @param type $champ
* @param type $ordre
* @param type $table si $champ dans une autre table
* @return Formation[]
*/
public function findAllOrderBy($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();
    }
}

/**
 * Enregistrements dont un champ contient une valeur
 * ou tous les enregistrements si la valeur est vide
 * @param type $champ
 * @param type $valeur
 * @param type $table si $champ dans une autre table
 * @return Formation[]
 */
public function findByContainValue($champ, $valeur, $table=""): array{
    if($valeur==""){
        return $this->findAll();
    }
    if($table==""){
        return $this->createQueryBuilder('f')
            ->where('f.'.$champ.' LIKE :valeur')
            ->orderBy('f.'.$champ, 'DESC')
            ->setParameter('valeur', '%'.$valeur.'%')
            ->getQuery()
            ->getResult();
    }else{
        return $this->createQueryBuilder('f')
            ->join('f.'.$table, 't')
            ->where('t.'.$champ.' LIKE :valeur')
            ->orderBy('f.'.$champ, 'DESC')
            ->setParameter('valeur', '%'.$valeur.'%')
            ->getQuery()
            ->getResult();
    }
}

/**
```

Après modification :


```

        $this->getEntityManager()->flush();
    }
}

/**
 * Retourne toutes les formations triées sur un champ
 * @param type $champ
 * @param type $ordre
 * @return Formation[]
 */
public function findAllOrderBy($champ, $ordre): array{
    return $this->createQueryBuilder('f')
        ->orderBy('f.'.$champ, $ordre)
        ->getQuery()
        ->getResult();
}

public function findAllOrderByTable($champ, $ordre, $table): array{
return $this->createQueryBuilder('f')
        ->join('f.'.$champ, 't')
        ->orderBy('t.'.$table, $ordre)
        ->getQuery()
        ->getResult();
}

/**
 * Enregistrements dont un champ contient une valeur
 * ou tous les enregistrements si la valeur est vide
 * @param type $champ
 * @param type $valeur
 * @param type $table si $champ dans une autre table
 * @return Formation[]
 */

```

```

    /**
    public function findByContainValue($champ, $valeur): array{
        if($valeur==""){
            return $this->findAll();
        }
        return $this->createQueryBuilder('f')
            ->where('f.' . $champ . ' LIKE :valeur')
            ->orderBy($this->published, 'DESC')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->getQuery()
            ->getResult();
    }

    /**
    * Enregistrements dont un champ dans une autre table contient une valeur
    * ou tous les enregistrements si la valeur est vide
    * @param type $champ
    * @param type $valeur
    * @param type $table
    * @return Formation[]
    */
    public function findByContainValueTable($champ, $valeur, $table): array {
        if ($valeur == "") {
            return $this->findAll();
        }
        return $this->createQueryBuilder('f')
            ->join('f.' . $table, 't')
            ->where('t.' . $champ . ' LIKE :valeur')
            ->orderBy($this->published, 'DESC')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->getQuery()
            ->getResult();
    }

    /**

```

Extrait de code avant modification pour PlaylistRepository :

```

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

```

Après modification pour PlaylistRepository :


```

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


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


```

C. AJOUTER UNE FONCTIONNALITE


 Mediatek86 #3


ajouter une fonctionnalité #3










 Closed uiuiui34500 opened 3 months ago

 **uiuiui34500** 3 months ago (edited) Edit



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.



 Preview

H B I         

Leave a comment

 Markdown is supported  Paste, drop, or click to add files Reopen Comment

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.

Pour effectuer cela il est nécessaire d'abord de rajouter une méthode 'getCategoriesPlaylist' dans l'entité playlist :

```

+  /**
+   * @return Collection<int, string>
+   */
+   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;
+   }
+
+

```

Par la suite il faut alors modifier la méthode FindAllOrderBy en la séparant en deux méthode différentes(FindAllOrderByName et FindAllOrderByNbFormations) au sein de playlistRepository :

```

/**
 * Retourne toutes les playlists triées sur le nom de la playlist
 * @param type $champ
 * @param type $ordre
 * @return Playlist[]
 */
public function findAllOrderByName($ordre): array{
    return $this->createQueryBuilder('p')
        ->leftjoin($this->formations, 'f')
        ->groupBy('p.id')
        ->orderBy('p.name', $ordre)
        ->getQuery()
        ->getResult();
}

/**
 * Retourne toutes les playlists triées sur le nombre de formations
 * @param type $ordre
 * @return Playlist[]
 */
public function findAllOrderByNameNbFormations($ordre): array{
    return $this->createQueryBuilder('p')
        ->leftjoin($this->formations, 'f')
        ->groupBy('p.id')
        ->orderBy('count(f.title)', $ordre)
        ->getQuery()
        ->getResult();
}

```

Dans le code html nous rajoutons alors deux boutons de tri sous forme de tableau pour pouvoir bien l'afficher :

```

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

```

Affichage de la page playlists :



MediaTek86

Des formations pour tous
sur des outils numériques

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

playlist



filtrer

catégories

Nombre de
formations



Bases de la programmation (C#)

C# POO

74

Voir détail

Compléments Android (programmation mobile)

Android

13

Voir détail

Cours Composant logiciel

Cours

2

Voir détail

Cours Curseurs

SQL Cours POO

2

Voir détail

Cours de programmation objet

POO Cours

1

Voir détail

Cours Informatique embarquée

Cours

1

Voir détail

Cours MCD MLD MPD

MCD Cours

2

Voir détail

Cours MCD vs Diagramme de classes

MCD Cours

2

Voir détail

Cours Merise/2

MCD Cours

1

Voir détail

Cours Modèle relationnel et MCD

MCD Cours

1

Voir détail



Des formations pour tous sur des outils numériques

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

Bases de la programmation (C#)

catégories : C# POO **Nombre de formation :** 74

description :

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

Prérequis : aucun

1ère partie : programmation procédurale en mode console (non graphique)

n°1 à 30 : procédural, notions élémentaires (variables, saisie/affichage, affectations/calculs, alternatives (if/switch), itérations (while/do-while/for))

n°31 à 42 : procédural, tableaux (1 et 2 dimensions, manipulations, tris, recherches)

n°43 à 59 : procédural, modules et paramètres (procédures et fonctions)

2ème partie : événementiel (en mode graphique)

n°60 à 67 : événementiel (programmation graphique)

3ème partie : initiation à l'objet

n°68 à 74 : notions de base en programmation objet sur des classes "métier"



Bases de la programmation n°1 - procédural : premier exemple

Bases de la programmation n°2 - procédural : exercice1 (affichage)

Bases de la programmation n°3 - procédural : exercice2 (saisie)

Bases de la programmation n°4 - procédural : exercice3 (calculs)

Bases de la programmation n°5 - procédural : exercice4 (calcul dans affichage)

Bases de la programmation n°6 - procédural : exercice5 (condition)

Bases de la programmation n°7 - procédural : exercice6 (conditions imbriquées)

ETAPE 2. CODER LA PARTIE BACK-OFFICE

Le back office doit permettre de gérer le contenu de la base de données.

Il doit contenir la même bannière que le front office et un menu contenant "Formations", "Playlists" et "Catégories".

A. GERER LES FORMATIONS

The screenshot shows a GitHub issue titled "gérer les formations #4" in a dark theme. The issue is marked as "Closed" and was opened 3 months ago by user "uiuiui34500". The issue description, edited 3 months ago, outlines requirements for a back-office page to manage training formations. It specifies that the page should allow listing formations, deleting them (with confirmation), and modifying them. It also mentions that if a formation is deleted, it should be removed from its associated playlist. The requirements state that the same sorting and filtering options present in the front office must be available in the back office. A button should allow adding new formations, with controlled inputs; the "description" field is optional, and categories should be selected from a list (one playlist per formation, multiple categories per formation). The date should be selected from a list and must not be in the future. Clicking the modify button should pre-fill the form. The interface includes a preview section with a rich text editor toolbar (bold, italic, link, etc.) and a comment section at the bottom.

Mediatek86 #4

gérer les formations #4

Closed uiuiui34500 opened 3 months ago

uiuiui34500 3 months ago (edited) Edit

Une page doit permettre de lister les formations et, pour chaque formation, afficher un bouton permettant de la supprimer (après confirmation) et un bouton permettant de la modifier.

Si une formation est supprimée, il faut aussi l'enlever de la playlist où elle se trouvait.

Les mêmes tris et filtres présents dans le front office doivent être présents dans le back office.

Un bouton doit permettre d'accéder au formulaire d'ajout d'une formation. Les saisies doivent être contrôlées. Seul le champ "description" n'est pas obligatoire ainsi que la sélection de catégories (une formation peut n'avoir aucune catégorie). La playlist et la ou les catégories doivent être sélectionnées dans une liste (une seule playlist par formation, plusieurs catégories possibles par formation). La date ne doit pas être saisie mais sélectionnée. Elle ne doit pas être postérieure à la date du jour.

Le clic sur le bouton permettant de modifier une formation doit amener sur le même formulaire, mais cette fois prérempli.

Preview

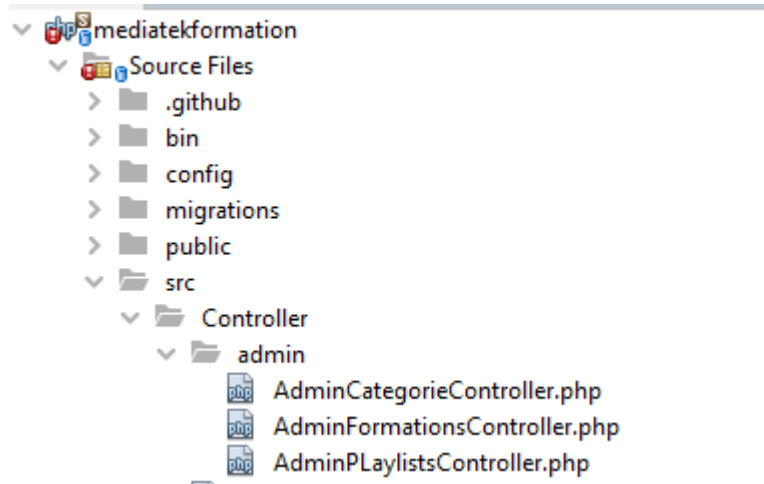
Leave a comment

Markdown is supported Paste, drop, or click to add files Reopen Comment

Cette tâche me donne pour mission de créer une page permettant de lister les formations et de pouvoir les modifier, les supprimer ou bien en ajouter. Une formation qui est supprimée sera alors supprimée de la playlist où elle est située.

Ajout des fonctionnalités :

Création d'un dossier admin dans le dossier Controller , permettant alors de mettre tous les adminController :



Création du fichier AdminFormationsController , qui contient toutes les méthodes pour ajouter , modifier et supprimer les formations :

Méthode suppr pour supprimer une formation :

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

Méthode edit pour modifier une formation :

```
/**
 * @Route("/admin/edit/{id}", name="admin.formation.edit")
 * @param Formation $formation
 * @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.formations');
    }

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

/**
```

Méthode ajout pour ajouter une formation :

```

* @Route("/admin/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);
        return $this->redirectToRoute('admin.formations');
    }

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

```

Création du formulaire formationType.php dans le dossier Form qui permet d'avoir un formulaire avec les données des formations à rentrer pour modifier ou ajouter une formation :

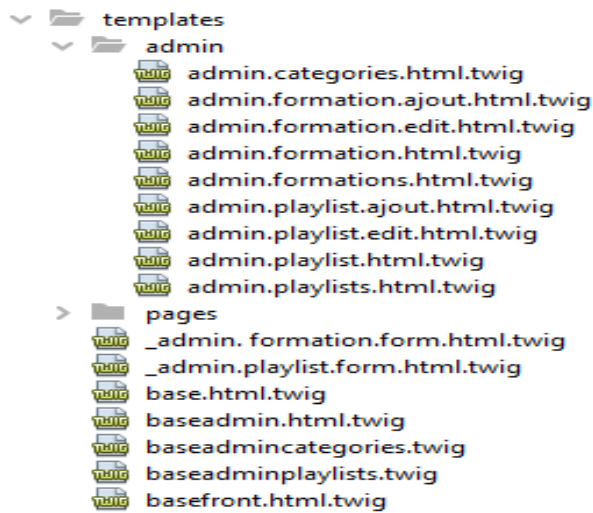
```

class FormationType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('publishedAt', null, [
                'widget' => 'single_text',
                'data' => isset($options['data']) &&
                    $options['data']->getPublishedAt() != null ? $options['data']->getPublishedAt() : new DateTime('now'),
                'label' => 'Date'
            ])
            ->add('title', TextType::class, [
                'label' => 'Titre'
            ])
            ->add('description', TextType::class, [
                'label' => 'Description',
                'required' => false
            ])
            ->add('videoId', TextType::class, [
                'label' => 'Id de la vidéo'
            ])
            ->add('playlist', EntityType::class, [
                'class' => Playlist::class,
                'choice_label' => 'name',
            ])
            ->add('categories', EntityType::class, [
                'class' => Categorie::class,
                'choice_label' => 'name',
                'required' => false,
                'multiple' => true
            ])
            ->add('submit', SubmitType::class, [
                'label' => 'Enregistrer'
            ])
        ;
    }

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


```

Après cela il m'as fallu crée un dossier admin dans le dossier template pour mettre tout l'affichage html coté admin :



Voici alors l'affichage de la page de gestion des formations :

Gestion des formations



MediaTek86

Des formations pour tous
sur des outils numériques

Formations Playlists Catégories
Ajouter une nouvelle formation

formation	playlist	catégories	date	
<div style="display: flex; align-items: center;"> < > <input style="width: 100px;" type="text"/> filtrer </div>	<div style="display: flex; align-items: center;"> < > <input style="width: 100px;" type="text"/> filtrer </div>	<div style="display: flex; align-items: center;"> v <input style="width: 100px;" type="text"/> </div>	<div style="display: flex; align-items: center;"> < > </div>	
Eclipse n°4 : WindowBuilder	Eclipse et Java	Java	09/11/2020	<div style="display: flex; justify-content: space-between;"> Editer Supprimer </div>
Eclipse n°2 : rétroconception avec ObjectAid	Eclipse et Java	Java UML	05/11/2020	<div style="display: flex; justify-content: space-between;"> Editer Supprimer </div>
Eclipse n°1 : installation de l'IDE	Eclipse et Java	Java	03/11/2020	<div style="display: flex; justify-content: space-between;"> Editer Supprimer </div>
UML : Diagramme de paquetages	Cours UML	UML	01/11/2020	<div style="display: flex; justify-content: space-between;"> Editer Supprimer </div>

Affichage d'ajout d'une formation :

[se déconnecter](#)

Gestion des formations

**MediaTek86**

Des formations pour tous
sur des outils numériques

[Formations](#) [Playlists](#) [Catégories](#)

Nouvelle formation

Titre

Date

Playlist

Description

Categories

Java
UML
C#
Python

Id de la vidéo

Affichage de modification d'une formation :

[se deconnecter](#)

Gestion des formations

**MediaTek86**

Des formations pour tous
sur des outils numériques

[Formations](#) [Playlists](#) [Catégories](#)

Détail de la formation

Titre

Date

Playlist

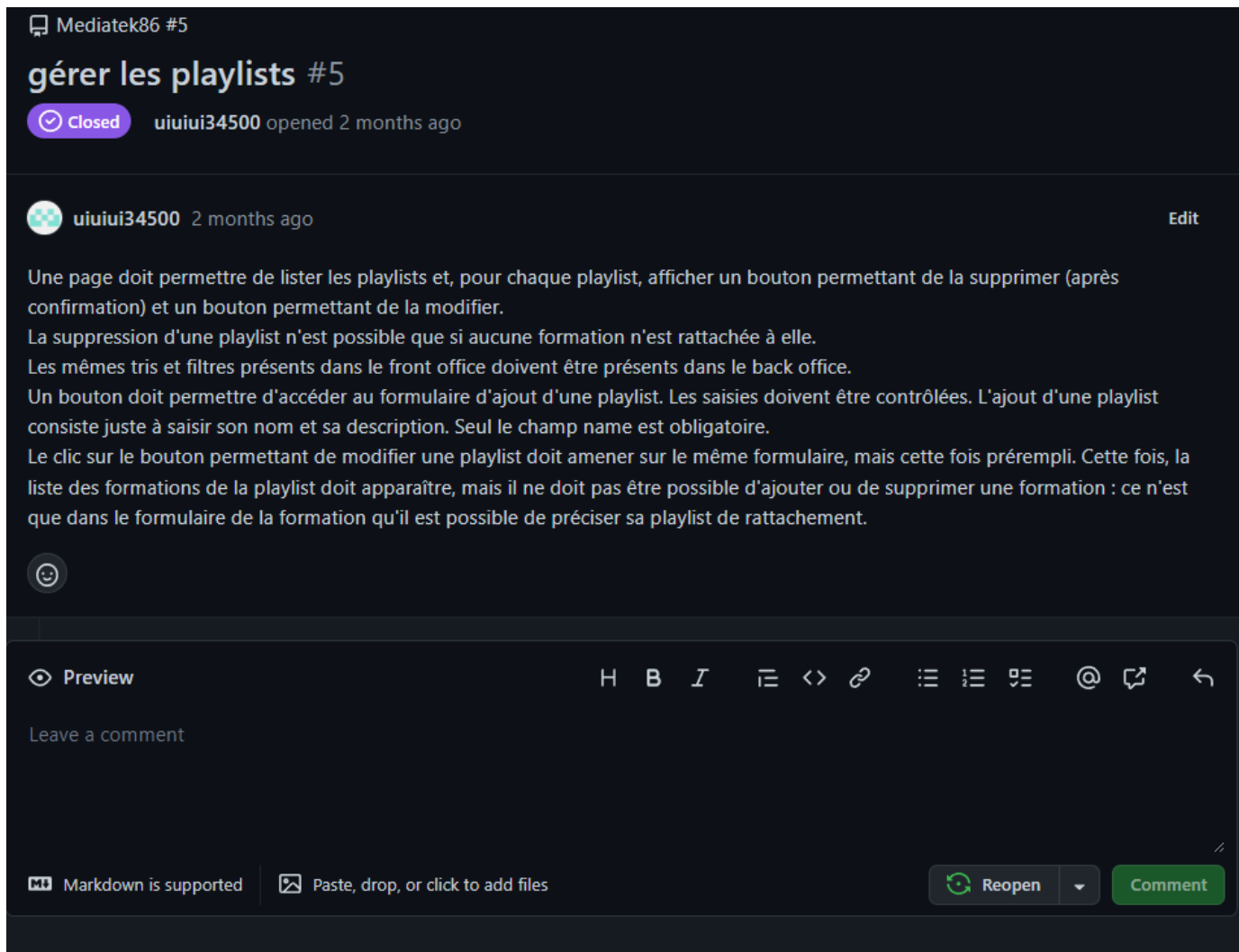
Description

Categories

Java
UML
C#
Python

Id de la vidéo

B. GERER LES PLAYLISTS



Cette tache me donne pour mission de crée une page permettant de lister les playlists et de pouvoir les modifier, les supprimer ou bien en ajouter. Une playlist qui est supprimé seras supprimé seulement si elle ne contient aucune formation.

Tout d'abord j'ai donc du crée un fichier AdminPlaylistsController avec les méthodes suivantes

Méthode suppr pour supprimer une playlist :

```
/**
 * @Route("/admin.playlists/suppr/{id}", name="admin.playlist.suppr")
 * @param Playlist $playlist
 * @return Response
 */
public function suppr(Playlist $playlist): Response {

    $this->playlistRepository->remove($playlist, true);
    return $this->redirectToRoute('admin.playlists');
}
```

Méthode edit pour modifier une playlist :

```
/**
 * @Route("/admin.playlists/edit/{id}", name="admin.playlist.edit")
 * @param Playlist $playlist
 * @return Response
 */
public function edit(Playlist $playlist, Request $request): Response {
    $formPlaylist = $this->createForm(PlaylistType::class, $playlist);
    $playlistFormations = $this->formationRepository->findAllForOnePlaylist($playlist);

    $formPlaylist->handleRequest($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
    ]);
}
```

Méthode ajout pour ajouter une playlist :

```

/**
 * @Route("/admin/playlist/ajout", name="admin.playlist.ajout")
 * @param Request $request
 * @return Response
 */
public function ajout(Request $request): Response {
    $playlist = new Playlist();
    $formPlaylist = $this->createForm(PlaylistType::class, $playlist);

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

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

```

Pour faire tout cela il m'a alors fallu créer un formulaire pour les saisies des playlists en cas de modification ou d'ajout d'une playlist. J'ai donc créé un formulaire dans le dossier Form voici le code correspondant :

```

class PlaylistType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('name', TextType::class, [
                'label' => 'Nom',
                'required' => true
            ])
            ->add('description')
            ->add('enregistrer', SubmitType::class)
        ;
    }

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

```

Une fois tout cela fait il ne manque plus que l’affichage en html , dans le fichier html des playlists j’ai fais une conditions a remplir pour supprimer une playlist en vérifiant si la playlist contient bien zéro formations , car une playlist ne peux etre supprimer seulement si elle ne contient aucune formation , j’ai donc rajouter ce code suivant :

```

<td class="text-center">
    <a href="{{ path('admin.playlist.edit', {id:playlists[k].id}) }}" class="btn btn-secondary">Modifier</a>
    {% if playlists[k].formations|length == 0 %}
    <a href="{{ path('admin.playlist.suppr', {id:playlists[k].id}) }}" class="btn btn-danger" onclick="return cc
    {% else %}
        <a href="#" class="btn btn-danger" onclick="return confirm('{{ playlists[k].name }} ne doit pas contenir de
    {% endif %}
</td>

```

Affichage de la page de gestion des playlists :

Gestion des playlists



MediaTek86

Des formations pour tous
sur des outils numériques

Formations Playlists Catégories

Ajouter une nouvelle playlist

playlist



filtrer

catégories

Nombre de
formations



Bases de la programmation (C#)

C# POO

74

Voir détail

Modifier

Supprimer

Compléments Android (programmation mobile)

Android

13

Voir détail

Modifier

Supprimer

Cours Composant logiciel

Cours

2

Voir détail

Modifier

Supprimer

Cours Curseurs

SQL Cours POO

2

Voir détail

Modifier

Supprimer

Cours de programmation objet

POO Cours

1

Voir détail

Modifier

Supprimer

Cours Informatique embarquée

Cours

1

Voir détail

Modifier

Supprimer

Cours MCD MLD MPD

MCD Cours

2

Voir détail

Modifier

Supprimer

Cours MCD vs Diagramme de classes

MCD Cours

2

Voir détail

Modifier

Supprimer

Cours Merise/2

MCD Cours

1

Voir détail

Modifier

Supprimer

Affichage d'ajout d'une playlist :



MediaTek86

Des formations pour tous
sur des outils numériques

Formations Playlists Catégories

Nouvelle playlist

Nom

Description

Enregistrer

Affichage de modification d'une playlist :

Gestion des playlists



MediaTek86

Des formations pour tous
sur des outils numériques

Formations Playlists Catégories

Détail de la playlist

Nom

Bases de la programmation (C#)

Description

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

Enregistrer

C. GERER LES CATEGORIES



Pour cette tache je dois crée une page de gestion des catégories qui permet de lister les catégories et de les supprimer ou bien d'en ajouter.

Pour commencer cette tache j'ai dû crée un fichier AdminCategorieController au sein du dossier admin dans le dossier Controller. Dans ce fichier j'ai créé plusieurs méthodes :

Première méthode « findAllForOnePlaylist » qui permet de lister toutes les catégories des formations d'une playlist :

```

/**
 * Retourne la liste des catégories des formations d'une playlist
 * @param type $idPlaylist
 * @return Catégorie[]
 */
public function findAllForOnePlaylist($idPlaylist): array {
    return $this->createQueryBuilder('c')
        ->join('c.formations', 'f')
        ->join('f.playlist', 'p')
        ->where('p.id=:id')
        ->setParameter('id', $idPlaylist)
        ->orderBy('c.name', 'ASC')
        ->getQuery()
        ->getResult();
}

```

Ensuite nous avons la méthode `suppr` qui comme son nom l'indique permet de supprimer une catégorie :

```

/**
 * @Route("admin/categorie/suppr/{id}", name="admin.categorie.suppr")
 * @param Catégorie $categorie
 * @return Response
 */
public function suppr(Catégorie $categorie): Response {
    $this->categorieRepository->remove($categorie, true);
    return $this->redirectToRoute('admin.categories');
}

```

Par la suite il y a la méthode `ajout`, qui permet d'ajouter des catégories :


```

    /**
    * @Route("/admin/categorie/ajout", name="admin.categorie.ajout")
    * @param Request $request
    * @return Response
    */
    public function ajout(Request $request): Response {
        $nomCategorie = $request->get("nom");
        $nomTest = $this->categorieRepository->findAllNameEqual($nomCategorie);
        if ($nomTest == false) {
            $categorie = new Categorie();
            $categorie->setName($nomCategorie);
            $this->categorieRepository->add($categorie, true);
            return $this->redirectToRoute('admin.categories');
        }
        else{
            return $this->redirectToRoute('admin.categories');
        }
    }
}

```

Pour finir tout cela il a fallu ajouter une page html pour afficher tout cela :

```

{% extends "baseadmincategories.twig" %}

{% block body %}

<div class="row">

  <div class="col text-center">

    <h1>Gestion des categories</h1>

  </div>

  <div class="form-group col text-end">

    <form class="form-inline mt-1" method ="POST" action="{{ path('admin.categorie.ajout') }}" >

      <input type="text" class="sm" name="nom">

      <button type="submit" class="btn btn-primary mb-2 btn-sm">Ajouter</button>

    </form>

  </div>

</div>

<table class="table table-striped">

  <thead>

    <tr>

      <th class="text-left align-top" scope="col">

        Catégorie<br />

      </th>

      <th class="text-left align-top" scope="col">

        Action<br />

      </th>

    </tr>

  <tbody>

    {% for categorie in categories %}

      <tr class="align-middle">

        <td>

          <h5 class="text-info">

            {{ categorie.name }}

          </h5>

        </td>

        <td>

          {% if categorie.formations|length == 0 %}

            <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>

          {% else %}

            <a href="#" class="btn btn-danger" onclick="return confirm('{{ categorie.name }} ne doit pas contenir de formations pour pouvoir être supprimé')">Supprimer</a>

          {% endif %}

        </td>

      </tr>

    {% endfor %}

  </tbody>

</table>

{% endblock %}

```

Affichage de la page :

[se déconnecter](#)

Gestion des catégories

**MediaTek86**

Des formations pour tous
sur des outils numériques

[Formations](#) [Playlists](#) [Catégories](#)

Gestion des categories

 [Ajouter](#)

Catégorie	Action
Java	Supprimer
UML	Supprimer
C#	Supprimer
Python	Supprimer
MCD	Supprimer
Android	Supprimer
POO	Supprimer
SQL	Supprimer
Cours	Supprimer

D.AJOUTER L'ACCES AVEC AUTHENTIFICATION

Mediatek86 #7

ajouter l'accès avec authentification #7

[Closed](#) uiuiui34500 opened 2 months ago

uiuiui34500 2 months ago

[Edit](#)

Le back office ne doit être accessible qu'après authentification : un seul profil administrateur doit avoir le droit d'accès. Pour gérer l'authentification, utiliser Keycloak.

Il doit être possible de se déconnecter, sur toutes les pages (avec un lien de déconnexion).

[Preview](#)[H](#) [B](#) [I](#) [≡](#) [<>](#) [🔗](#) [☰](#) [☷](#) [☰](#) [@](#) [🔗](#) [↩](#)

Leave a comment

Markdown is supported

Paste, drop, or click to add files

[Reopen](#)[Comment](#)

Dans cette tâche j'ai donc dû ajouter un accès avec authentification pour accéder au côté admin du site, tout cela grâce à keycloak.

Tout d'abord j'ai dû paramétrer keycloak en créant un nouveau client nommé « mediatekformation » qui nous permettra par la suite de lier l'application avec l'accès d'authentification.

Client ID	Type
account	OpenID Connect
account-console	OpenID Connect
admin-cli	OpenID Connect
broker	OpenID Connect
mediatekformation	OpenID Connect
realm-management	OpenID Connect
security-admin-console	OpenID Connect

Par la suite j'ai créé un utilisateur pour la connexion à la partie admin :

Users

Users are the users in the current realm. [Learn more](#)

User list

Permissions

Q Search user

→

Add user

Delete user

1-1

<

>

<input type="checkbox"/>	Username	Email	Last name	First name	Status
<input type="checkbox"/>	adminmediatekformation	<div><div></div>adminmediatekformation@dom.com</div>	-	-	-

1-1

<

>

Dans le projet Netbeans j'ai dû modifier le fichier .env en rajouter les donnée de connexion pour se connecter à keycloak.

Ligne de code modifier :

```
@@ -42,3 +42,7 @@ DATABASE_URL="mysql://root:@127.0.0.1:3306"
42 42 ###> symfony/mailer ###
43 43 # MAILER_DSN=null://null
44 44 ###< symfony/mailer ###
45 +
46 + KEYCLOAK_SECRET=8laQHtehmq309Jg3wMiwtyPbVBUSA2LE
47 + KEYCLOAK_CLIENTID=mediatekformation
48 + KEYCLOAK_APP_URL=http://localhost:8080
```

Ensuite j'ai alors crée la classe « user » et sa table correspondante, cela permet d'enregistrer l'utilisateur dans la

bdd du site dès qu'il est connecter pour pouvoir gérer la déconnexion sans dépendre de keycloak .

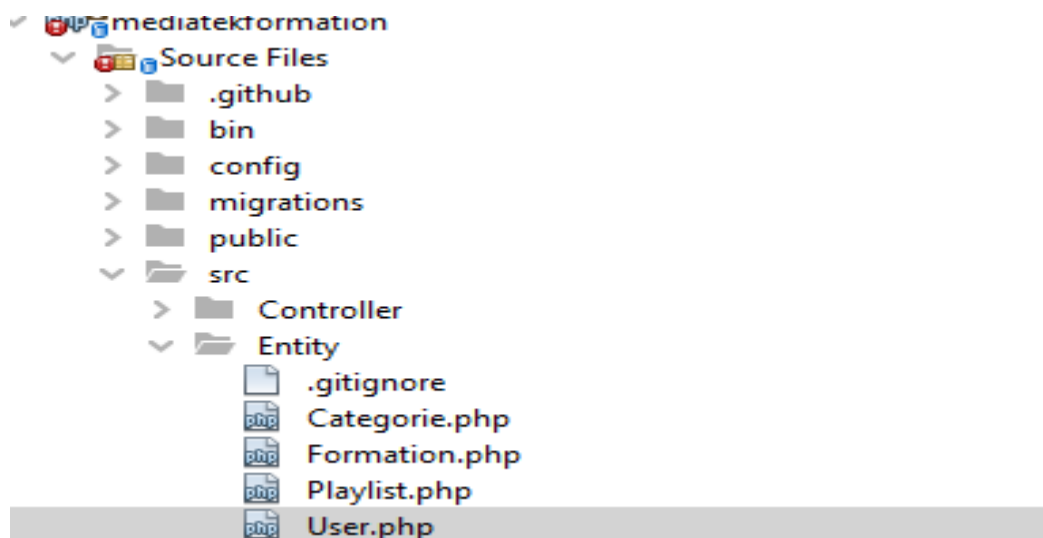
Dans la fenêtre de commande au dossier du projet j'ai donc taper la commande suivante :

php bin/console make:user

Il faut ajouter un champ à cette classe pour faire le lien avec Keycloak avec la commande.

php bin/console make:entity User

L'entité User est alors créée au sein du dossier Entity :



Il faut maintenant créer le fichier de migration qui va contenir la requête pour créer la table dans la BDD. Dans la fenêtre de commandes il faut donc faire :

php bin/console make:migration

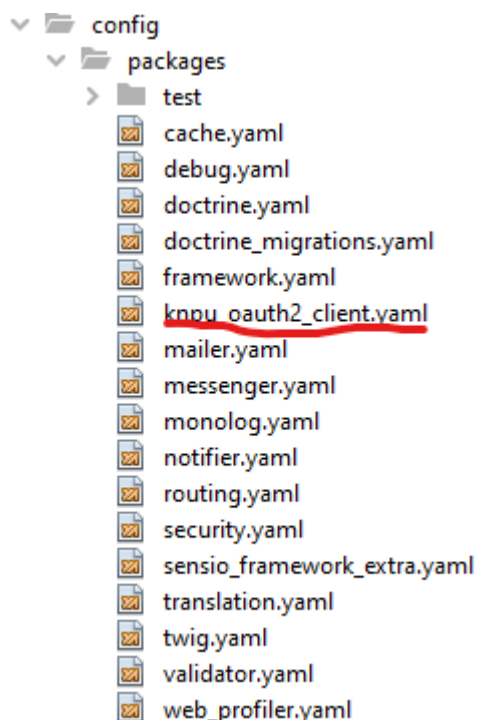
Puis il faut lancer la migration avec :

php bin/console doctrine:migrations:migrate

Par la suite dans la base de donnée la table user a été créée :

Table ▲	Action
<input type="checkbox"/> categorie	★ Parcourir Structure Rechercher Insérer Vider Supp
<input type="checkbox"/> doctrine_migration_versions	★ Parcourir Structure Rechercher Insérer Vider Supp
<input type="checkbox"/> formation	★ Parcourir Structure Rechercher Insérer Vider Supp
<input type="checkbox"/> formation_categorie	★ Parcourir Structure Rechercher Insérer Vider Supp
<input type="checkbox"/> messenger_messages	★ Parcourir Structure Rechercher Insérer Vider Supp
<input type="checkbox"/> playlist	★ Parcourir Structure Rechercher Insérer Vider Supp
<input type="checkbox"/> <u>user</u>	★ Parcourir Structure Rechercher Insérer Vider Supp

Par la suite il faut installer les bundles pour Oauth et keycloak, dans la fenêtre de commande j'ai alors tapé la première commande **composer require knpuniversity/oauth2-client-bundle 2.10** avec cette commande on a créé le fichier « knpu_oauth2_client.yam » dans config > packages :



Après cela il faut alors installer un second bundle avec la commande **composer require stevenmaguire/oauth2-keycloak 3.1 --with-all-dependencies** . Une fois ce bundle installer il faut compléter dans Netbeans le fichier `Knpu_oauth2_client.yaml` :

```
knpu_oauth2_client:
  clients:
    keycloak:
      type: keycloak
      auth_server_url: '%env(KEYCLOAK_APP_URL)%'
      realm: 'myapplis'
      client_id: '%env(KEYCLOAK_CLIENTID)%'
      client_secret: '%env(KEYCLOAK_SECRET)%'
      redirect_route: 'oauth_check'
```

Il faut ensuite configurer le firewall, dans le fichier `security.yaml` qui se trouve dans « config>packages » :


```

@@ -5,14 +5,25 @@ security:

    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider

    providers:
-     users_in_memory: { memory: null }
+     # 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
-     provider: users_in_memory
+     entry_point: form_login
+     form_login:
+         login_path: oauth_login
+     custom_authenticators:
+         - App\Security\KeycloakAuthenticator
+     logout:
+         path: logout
+

    # activate different ways to authenticate
    # https://symfony.com/doc/current/security.html#the-firewall

```

```

@@ -23,6 +34,7 @@ security:

    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used

    access_control:
+     - { path: ^/admin, roles: ROLE_ADMIN }
+
    # - { path: ^/admin, roles: ROLE_ADMIN }
    # - { path: ^/profile, roles: ROLE_USER }

```

Après avoir fais cela il faut crée le contrôleur qui va gérer l'authentification , dans la fenêtre de commande j'ai écrit **php bin/console make:controller OAuthController --no-template** ce qui crée le fichier OAuthController.php dans le dossier Controller :

```
<?php
```

```
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){
```

```
    }
```

```
    /**
```

```
     * @Route("/logout", name="logout")
```

```
     */
```

```
    public function logout() {
```

```
    }
```

```
}
```

Il faut ensuite , crée la classe qui va gérer l'authentification il faut alors crée un nouveau dossier « Security » dans le dossier src avec une classe « KeycloakAuthenticator.php » qui contiens plusieurs méthodes :

```

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;
        })
    );
}

```

```

public function onAuthenticationFailure(Request $request, AuthenticationException $exception): ?Response {

    $message = strtr($exception->getMessageKey(), $exception->getMessageData());
    return new Response($message, Response::HTTP_FORBIDDEN);
}

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

public function start(Request $request, AuthenticationException $authException = null): Response {

    return new RedirectResponse(
        'oauth/login',
        Response2::HTTP_TEMPORARY_REDIRECT
    );
}

public function supports(Request $request): ?bool {

    return $request->attributes->get('_route') === 'oauth_check';
}

```

Une fois que tout est configuré il faut ensuite mettre en place la déconnexion . Dans les dossiers templates de haut de pages cotés admin il faut ajouter un lien « se déconnecter » :

```

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

```

Affichage de connexion :

MYAPPLIS

Sign in to your account

Username or email

adminmediatekformation

Password

Sign In

se déconnecter

Gestion des formations



MediaTek86

Des formations pour tous
sur des outils numériques

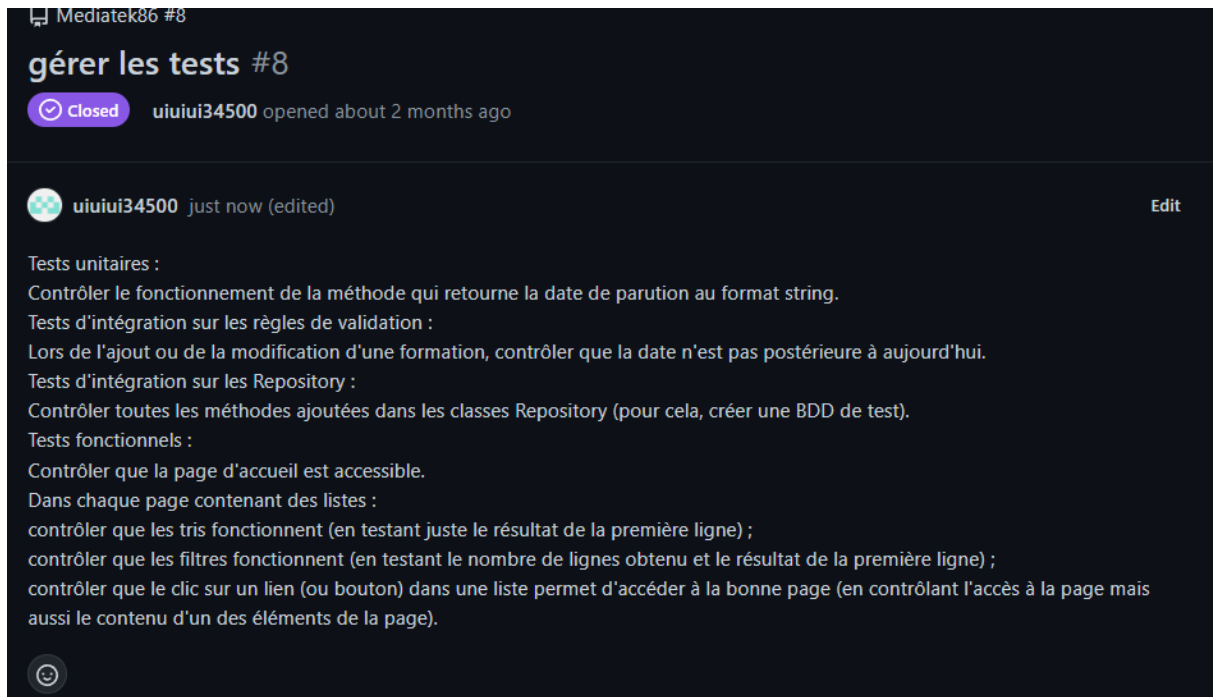
Formations Playlists Catégories

Ajouter une nouvelle formation

formation	playlist	catégories	date	
< >	< >		< >	
<input type="text"/>	<input type="text"/>			
<input type="button" value="filtrer"/>	<input type="button" value="filtrer"/>			
Eclipse n°4 : WindowBuilder	Eclipse et Java	Java	09/11/2020	<input type="button" value="Editer"/> <input type="button" value="Supprimer"/>
Eclipse n°2 : rétroconception avec ObjectAid	Eclipse et Java	Java UML	05/11/2020	<input type="button" value="Editer"/> <input type="button" value="Supprimer"/>
Eclipse n°1 : installation de l'IDE	Eclipse et Java	Java	03/11/2020	<input type="button" value="Editer"/> <input type="button" value="Supprimer"/>
UML : Diagramme de paquetages	Cours UML	UML	01/11/2020	<input type="button" value="Editer"/>

ETAPE 3.TESTER ET DOCUMENTER

A. GERER LES TESTS



Test unitaires :

Contrôler le fonctionnement de la méthode qui retourne la date de parution au format string.

Tests d'intégration sur les règles de validation :

Lors de l'ajout ou de la modification d'une formation, contrôler que la date n'est pas postérieure à aujourd'hui

Tests fonctionnels :

Contrôler que la page d'accueil est accessible.

Dans chaque page contenant des listes :

contrôler que les tris fonctionnent (en testant juste le résultat de la première ligne) ;

contrôler que les filtres fonctionnent (en testant le nombre de lignes obtenu et le résultat de la première ligne) ;

contrôler que le clic sur un lien (ou bouton) dans une liste permet d'accéder à la bonne page (en contrôlant l'accès à la page mais aussi le contenu d'un des éléments de la page).

Test unitaires :

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la méthode <code>getPublishedAtString()</code> de la classe Formation pour voir si elle retourne la bonne date au bon format.	Test unitaire lancé avec la date : 2021-01-04 17:00:12	04/01/2021	OK


Tests d'intégration sur les règles de validation :

But du test	Action de contrôle	Résultat attendu	Bilan
Lors de l'ajout ou de la modification d'une formation, contrôler que la date n'est pas postérieure à aujourd'hui. récupérer le nombre d'enregistrements contenus dans la table Formation.	Test d'intégration sur les règles de validations lancé avec la date: 2026/01/18, et avec un nombre d'erreur attendue de 1.	2026/01/18 → Erreur attendue 1	OK
Contrôler la méthode Add() du «FormationRepository», pour voir si elle ajoute bien une formation à la table Formation.	Test d'intégration lancé avec un count()	SnbFormations +1 → 239 formations	OK
Contrôler la méthode Remove() du «FormationRepository», pour voir si elle supprime bien une formation de la table Formation.	Test d'intégration lancé avec un count()	SnbFormations -1 → 238 formations	OK
Contrôler la méthode FindAllOrderBy() du «FormationRepository» pour voir si elle fait bien le tri d'un champ dans l'ordre défini.	Test d'intégration lancé avec le tri sur le champ «title» des formations, et dans l'ordre «ASC».	«Android Studio (complément n°1) : Navigation Drawer et Fragment»	OK
Contrôler la méthode FindAllOrderByTable() du «FormationRepository» pour voir si elle fait bien le tri d'un champ d'une table qui est définie.	Test d'intégration lancé avec le tri sur le champ «name» de la table «playlist», et dans l'ordre «ASC».	"Bases de la programmation n°74 - POO : collections"	OK
Contrôler la méthode FindByContainValue() du «FormationRepository» pour voir si elle filtre les formations dont un champ contient une valeur (chaîne) spécifiée.	Test d'intégration lancé avec le filtre sur le champ «title» avec pour valeur «C#».	"C# : ListBox en couleur" → 11 formations	OK
Contrôler la méthode FindByContainValueTable() du «FormationRepository» pour voir si elle filtre les formations dont un champ dans une autre table contient une valeur (chaîne) spécifiée.	Test d'intégration lancé avec le filtre sur le champ «name» de la table «playlist», avec pour valeur «Compléments Android (programmation mobile)».	"Android Studio (complément n°13) : Permissions" → 12 formations	OK
Contrôler la méthode FindAllLasted() du «FormationRepository» pour voir si elle tri les formations selon la date la plus récente de publication.	Test d'intégration lancé avec la date «2023-01-16 13:33:39»	"2023-01-16 13:33:39"	OK
Contrôler la méthode FindAllForOnePlaylist() du «FormationRepository» pour voir si elle récupère bien les formations d'une playlist selon son id, et réalise le tri ascendant.	Test d'intégration lancé avec la playlist ayant pour Sid «3».	"Python n°0 : installation de Python" → 19 formations	OK
Contrôler la méthode Add() du «PlaylistRepository», pour voir si elle ajoute bien une playlist à la table Playlist.	Test d'intégration lancé avec un count()	SnbPlaylists + 1 .	OK
Contrôler la méthode Remove() du «PlaylistRepository», pour voir si elle supprime bien une playlist à la table Playlist.	Test d'intégration lancé avec un count()	SnbPlaylists - 1	OK
Contrôler la méthode FindAllOrderByName() du «PlaylistRepository» pour voir si elle fait bien le tri selon le nom de la playlist et dans l'ordre défini.	Test d'intégration lancé avec un ordre «ascendant».	«Android – Test playlist»	OK
Contrôler la méthode FindAllOrderByNbFormations() du «PlaylistRepository» pour voir si elle fait bien le tri selon le nombre de formations et dans l'ordre défini.	Test d'intégration lancé avec un ordre «ascendant», sur la première case du tableau des playlists.	"Cours Informatique embarquée"	OK
Contrôler la méthode FindByContainValue() du «PlaylistRepository» pour voir si elle filtre les playlists dont un champ contient une valeur (chaîne) spécifiée.	Test d'intégration lancé avec le filtre sur le champ «name» avec pour valeur «Sujet».	«Exercices objet (sujets EDC BTS SIO)» → 8 playlists doivent être récupérées	OK
Contrôler la méthode FindByContainValueTable() du «PlaylistRepository» pour voir si elle filtre les playlists dont un champ d'une autre table, contient une valeur (chaîne) spécifiée.	Test d'intégration lancé avec le filtre sur le champ «name» de la table «categories» avec pour valeur «MCD».	"Cours MCD MLD MPD" → 5 playlists doivent être récupérées	OK
Contrôler la méthode Add() du «CategorieRepository», pour voir si elle ajoute bien une catégorie à la table Categorie.	Test d'intégration lancé avec un count()	SnbCategories + 1 → 11	OK
Contrôler la méthode Remove() du «CategorieRepository», pour voir si elle supprime bien une catégorie à la table Categorie.	Test d'intégration lancé avec un count()	SnbCategories - 1 → 10	OK
Contrôler la méthode FindAllForOnePlaylist() du «CategorieRepository» pour voir si elle récupère les catégories des formations d'une nlavlist. dans l'ordre ascendant.	Test d'intégration lancé avec la playlist qui possède l'id «3»	"POO" → 2 catégories doivent être récupérées	OK
Contrôler la méthode FindAllOrderBy() du «CategorieRepository» pour voir si elle fait bien le tri d'un champ dans l'ordre défini.	Test d'intégration lancé sur le champ «name», dans l'ordre «ascendant».	«Android»	OK


Tests fonctionnels :


Contrôler que la page des formations est accessible	Test fonctionnel lancé par: <code>assertResponseStatusCodeSame(Response::HTTP_OK);</code>	HTTP_OK	OK
Contrôler le tri ascendant selon le nom des playlists, <code>PlaylistsTriAsc()</code> du « <code>FormationsController</code> » pour voir si le tri se fait correctement sur le nom des playlists dans l'ordre ascendant.	Test fonctionnel lancé avec un ordre «ascendant», sur le champ «name» de la table «playlist».	'Bases de la programmation n°74 - POO : collections'	OK
Contrôler le tri ascendant selon le nom des formations, <code>FormationsTriAsc()</code> du « <code>FormationsController</code> » pour voir si le tri se fait correctement sur le nom des formations dans l'ordre ascendant.	Test fonctionnel lancé sur le champ «title» de la table «formation», avec un ordre «ascendant».	'Android Studio (complément n°1) : Navigation Drawer et Fragment'	OK
Contrôler le tri des dates, selon la plus récente ou la plus ancienne publication d'une formation. « <code>TriDate()</code> », du « <code>FormationsController</code> ».	Test fonctionnel lancé sur le champ «publishedAt» dans l'ordre «ascendant»	'Cours UML (1 à 7 / 33) : introduction'	OK
Contrôler le filtre des formations, pour récupérer la liste des formations contenant la valeur recherchée « <code>FiltreFormations()</code> », du « <code>FormationsController</code> ».	Test fonctionnel lancé sur un formulaire lors du clic sur le bouton «filtrer», dans le champ de recherche, avec «UML» comme valeur.	10 formations doivent être récupérées, contenant la valeur 'UML'	OK
Contrôler le filtre des playlists, pour récupérer la liste des playlists contenant la valeur recherchée « <code>FiltrePlaylists()</code> » du « <code>FormationsController</code> ».	Test fonctionnel lancé sur un formulaire lors du clic sur le bouton «filtrer», dans le champ de recherche, avec «Eclipse» comme valeur.	9 formations doivent être récupérées, contenant la valeur «Eclipse»	OK
Contrôler le filtre des catégories, pour récupérer la liste des catégories contenant la valeur recherchée « <code>FiltreCategories()</code> » du « <code>FormationsController</code> ».	Test fonctionnel lancé sur un formulaire lors du clic sur le bouton «filtrer», dans le champ de recherche, avec «Android» comme valeur.	8 formations doivent être récupérées, la première à s'afficher doit être «Java REACT»	OK
Contrôler que le lien de l'image redirige l'utilisateur vers la page détaillant la formation. « <code>LinkFormations()</code> », du « <code>FormationsController</code> ».	Test fonctionnel lancé lors du clic sur l'image de la formation. Redirection vers la page du détail de la formation.	Redirection vers '/formations/formation/1'	OK

B. CREER LA DOCUMENTATION TECHNIQUE

 Mediatek86 #9


créer la documentation technique #9


 Closed uiuiui34500 opened about 1 month ago










 uiuiui34500 1 month ago Edit

Contrôler que tous les commentaires normalisés nécessaires à la génération de la documentation technique ont été correctement insérés.




Générer la documentation technique du site complet : front et back office excluant le code automatiquement généré par Symfony (voir l'article "Génération de la documentation technique sous NetBeans" dans le wiki du dépôt).



 Preview

H B I         

Leave a comment

 Markdown is supported  Paste, drop, or click to add files  Reopen Comment

Pour créer la documentation technique je m'assure avant d'avoir bien insérés tout les commentaire normalisé . Je télécharge par la suite le fichier phpDocumentor.phar. Dans netbeans onglet Frameworks & Tools je sélectionne phpDocumentor et sur browse pour sélectionner le fichier phar que j'ai installé ultérieurement. Ensuite je fais un clique droit sur le projet et je clique sur generate documentation elle dure 8min .

Affichage de la documentation :

Namespaces

[DoctrineMigrations](#)[App](#)[Controller](#)[Entity](#)[Form](#)[Repository](#)[Security](#)[Tests](#)[ContainerPUMWbrn](#)[ContainerRivoDeU](#)[Proxies](#)[__CG__](#)[Symfony](#)[Config](#)[Component](#)[Contracts](#)[Bundle](#)[Bridge](#)[Flex](#)[Polyfill](#)[Runtime](#)[ContainerKQNCxwg](#)[ContainerSKAcQsC](#)[Composer](#)[Autoload](#)[DAMA](#)[DoctrineTestBundle](#)

Documentation


Packages

[P Application](#)[P Authentication](#)


Namespaces


[N DoctrineMigrations](#)[N App](#)[N ContainerPUMWbrn](#)[N ContainerRivoDeU](#)[N Proxies](#)[N Symfony](#)[N ContainerKQNCxwg](#)[N ContainerSKAcQsC](#)[N Composer](#)[N DAMA](#)[N Doctrine](#)[N Egulias](#)[N Firebase](#)[N ProxyManager](#)[N GuzzleHttp](#)

C. CREER LA DOCUMENTATION UTILISATEUR


 Mediatek86 #10


créer la documentation utilisateur #10








 Closed uiuiui34500 opened about 1 month ago

 uiuiui34500 1 month ago Edit



Créer en vidéo qui permet de montrer toutes les fonctionnalités du site (front et back office).
Cette vidéo ne doit pas dépasser les 5mn et doit présenter clairement toutes les fonctionnalités, en montrant les manipulations qui doivent être accompagnées d'explications orales.





 Preview

H B I  <>     @  

Leave a comment

 Markdown is supported  Paste, drop, or click to add files

 Reopen 

Comment

La documentation utilisateur est une vidéo de présentation des fonctionnalités du site web.

Pour réaliser l'enregistrement vidéo j'ai opter pour la fonctionnalité de Windows pour enregistrer des vidéos mp4.

Affichage de l'enregistrement vidéo :



MediaTek86

Des formations pour tous
sur des outils numériques

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

Bienvenue sur le site de MediaTek86 consacré aux formations en ligne

Vous allez pouvoir vous former à différents outils numériques gratuitement et directement en ligne.

Dans la partie [Formations](#), vous trouverez la liste des formations proposées. Vous pourrez faire des recherches et des tris. En cliquant sur la capture, vous accéderez à la présentation plus détaillée de la formation ainsi que la vidéo correspondante.

Vous pouvez aussi retrouver les vidéos regroupées dans des playlists, dans la partie [Playlists](#).

Voici les **deux dernières formations** ajoutées au catalogue :



09/11/2020

Eclipse n°4 : WindowBuilder

playlist : Eclipse et Java
catégories : Java



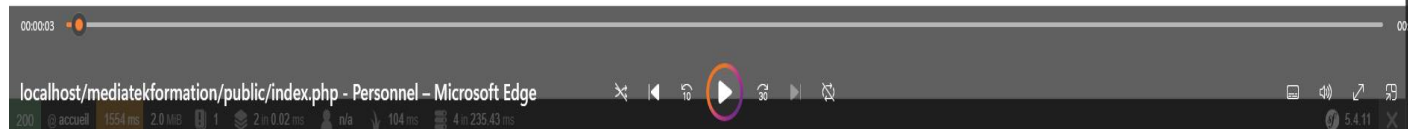
05/11/2020

**Eclipse n°2 : rétroconception avec
ObjectAid**

playlist : Eclipse et Java
catégories : Java UML


Deux dernières formations

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





ETAPE 4. DEPLOYER LE SITE ET GERER LE DEPLOIEMENT CONTINU

A. DEPLOYER LE SITE

 Mediatek86 #11

déployer le site , la bdd et la documentation utilisateur #11

 **Closed** uiuiui34500 opened about 1 month ago


 **uiuiui34500** 1 month ago (edited)

Edit

Installer et configurer le serveur d'authentification Keycloak dans une VM en ligne (voir l'article "Keycloak en ligne et en HTTPS" dans le wiki du dépôt).

Déployer le site, la BDD et la documentation technique chez un hébergeur.

Mettre à jour la page de CGU avec la bonne adresse du site.



Afin de procéder à la mise en ligne de mon serveur d'authentification keycloak, j'ai dû procéder à différentes étapes.

Tout d'abord, il a fallu que je crée une machine virtuelle windows, dans mon compte Azure, dans laquelle il a ensuite été nécessaire que j'installe le serveur keycloak.

J'ai défini un nom DNS à ma machine virtuelle "keycloakmediatekformation", qui va m'être utile durant la suite des étapes.

Grâce à la connexion bureau à distance de windows, je peux accéder à ma VM à partir de mon ordinateur, et procéder à l'installation du serveur keycloak.

Dans ma machine virtuelle, je dois installer "openJDK" et créer la variable d'environnement "JAVA_HOME", puis mettre le chemin vers mon installation jdk.

Je vais ensuite sur le site keycloak, pour le télécharger en version 19.0.1, je lance ensuite keycloak en tapant la commande "kc.bat start-dev».

Je me rends ensuite à l'adresse "localhost:8080" dans le navigateur de la VM, puis je crée un compte admin, j'accède enfin à la console d'administration après m'être authentifié avec ce compte.

Je crée ensuite mon royaume "Myapplies", et lui affecte les mêmes paramètres que lors de la configuration de mon projet symfony en localhost. Je crée également un nouvel utilisateur et lui affecte tout les droits nécessaires.

Je finalise la configuration en tapant dans la fenêtre de commandes "kc.bat build" .

Il faut ensuite que j'installe le certificat nécessaire à l'accessibilité de keycloak en HTTPS, pour cela, je commence par installer "xampp", une fois l'installation terminée, je clique sur le "start" de "apache" dans mon serveur xampp. Je dois ensuite installer "Certbot" pour obtenir un certificat, je choisis "Apache" on "Windows", une fois téléchargé, je lance l'installation, puis j'ouvre ma fenêtre de commandes en mode admin et je tape "certbot certonly --webroot".

Je rentre un email valide, je valide les termes, je rentre mon nom de domaine "DNS" -->

keycloakmediatekformation.francecentral.cloudapp.azure.com, et je rentre le web root "C:\xampp\htdocs" .

Je stoppe Apache, je ferme et désinstalle Xampp qui n'est plus utile. Dans une fenêtre de commandes lancée en mode admin, dossier C:\keycloak\bin, je lance la commande suivante, dans ma machine virtuelle : kc.bat start --hostname=keycloakmediatekformation.francecentral.cloudap

```
p.azure.com -- https-  
certificatefile=C:\Certbot\live\keycloakmediatekformation.francecentral.cloudapp.azure.com\fullchain.pem --https-  
certificate-  
keyfile=C:\Certbot\live\keycloakmediatekformation.francecentral.cloudapp.azure.com\privkey.pem --https-port=443
```

Pour accéder au serveur keycloak en dehors de la VM, je me rends à l'adresse

"keycloakmediatekformation.francecentral.cloudapp.azure.com". Dans l'application "mediatek-formation", il faut donner les bonnes valeurs aux 3 variables

KEYCLOAK du fichier.env

KEYCLOAK_APP_URL :

<https://keycloakmediatekformation.francecentral.cloudapp.azure.com>

KEYCLOAK_SECRET

:MTbgKp42DyoaAMu42alcUV9MxZnWO0y6

KEYCLOAK_CLIENTID :

mediatekformation

J'ai choisi d'utiliser l'hébergeur "planethoster", pour toute la partie de déploiement de l'application. Afin de déployer la bdd dans mon hébergeur, je fais une exportation de la bdd de l'application, et je copie le contenu de mon fichier. Je vais ensuite dans l'onglet "bases de données > phpmyadmin" de

mon hébergeur, je crée ma base de données, et un utilisateur associé. Ma base de données aura en paramètres :

Nom base de données : wtvuygnw_mediaformation

Nom utilisateur : wtvuygnw_alexandre

Pwd : aZeRtY121413

J'ouvre ensuite la base de données, et je colle le contenu de mon fichier dans l'onglet "sql" de phpmyadmin. Je clique ensuite sur "exécuter", et ma base de données apparaît avec les tables correctement implantées.

J'attribue également un site web à ma base de données "mediatek.go.yj.fr ", il s'agit du site dans lequel l'application sera implantée.

Je vais ensuite dans Netbeans, puis je modifie les informations d'accès à la base données dans le fichier ".env" :

```
31 - DATABASE_URL="mysql://root:@127.0.0.1:3308/mediatekformation"
31 + DATABASE_URL="mysql://wtvuygnw:hrkK3XZHBbJuz6@localhost:3308/wtvuygnw_mediatekformation"
```

Par la suite il faut avec Fille zilla dans le gestionnaire de fichier de mon hébergeur je transfère en format zip le dossier du site dans la partie public_html que j'extrait par la suite . le site est donc en ligne il ne manque plus que la documentation technique , je procède de la même manière sauf que j'insère le fichier zip de la doc dans le fichier www et je l'extrait .

Le site est accessible en ligne à l'adresse :

<https://mediatek.go.yj.fr/mediatekformation/public/>

La documentation technique est accessible en ligne à l'adresse :

<https://mediatek.go.yj.fr/DocumentationTechnique/index.html>

B. GERER LA SAUVEGARDE ET LA RESTAURATION DE LA BDD



The screenshot shows a GitHub issue interface. At the top, it says 'Mediatek86 #13' and the issue title 'gérer la sauvegarde et la restauration de la BDD (#13'. Below the title, there's a green 'Open' button and text indicating it was opened 'less than a minute ago' by user 'uiuiui34500'. The issue content, edited 'just now', describes the need for an automated journal backup for the database, referencing a wiki article 'Automatiser la sauvegarde d'une BDD'. It also states that restoration will be manual, using a backup script. The interface includes a 'Preview' section with a rich text editor toolbar (bold, italic, link, etc.) and a 'Leave a comment' field. At the bottom, there are buttons for 'Close issue' and 'Comment', along with a note that 'Markdown is supported'.

Pour gérer la sauvegarde d'une bdd il faut crée un fichier de script qui permet d'enregistrer le script de la bdd dans un fichier nommé « bddbbackup_ » suivi de la date et l'extension « .sql.gz », dans le dossier « savebdd » en racine du compte. Sous PlanetHoster, le chemin pour arriver à un dossier est “\home\” suivi du login du compte (ftp), suivi de “\” et du

nom du dossier concerné. J'ai créé alors un fichier en local nommé backup.sh avec le code suivant :

```
#!/bin/sh
```

```
DATE=`date -l`
```

```
find /home/ wtvuygnw savebdd/bdd* -mtime -1 -exec rm {} \;
```

```
mysqldump -u wtvuygnw_alexandre -paZeRtY121413—databases
```

```
wtvuygnw_mediaformation —single-transaction | gzip >
```

```
/home/wtvuygnw/savebdd/bddbbackup_${DATE}.sql.gz
```

Par la suite il faut mettre le fichier au format linux.

Il faut installer le logiciel dos2linux , le dézipper , aller dans le dossier bin et copier le fichier dos2linux.exe et le coller dans le dossier ou le fichier backup.sh a été placer . Ensuite j'ai donc ouvert la fenêtre de commande en mode admin en me localisant dans le dossier qui contient les deux fichiers pour exécuter la commande suivante :

```
dos2unix.exe backup.sh
```



Le fichier backup.sh est alors convertit au format linux .

Il faut ensuite transférer le fichier chez l'hébergeur ; avec FilleZilla il faut se connecter au compte ,Créer en racine, le dossier "savebdd", faire un clic droit sur le dossier > "Droits d'accès au fichier", cocher toutes les cases de permissions puis "OK". - Transférer le fichier backup.sh dans le dossier savebdd, faire un clic droit sur le fichier transféré > "Droits d'accès au fichier", cocher toutes les cases de permissions puis "OK".

Dans planetHoster il faut aller dans le panneau et dans le menu de gauche sélectionner « Crons » puis dans « Taches crons » il faut en ajouter une . Dans « commande » il faut mettre :

```
/home/wtvuygnw/savebdd/backup.sh
```

De retour dans la page des tâches, en bas, partie “Cron courriel”, mettre une adresse mail valide qui permet de recevoir les éventuelles erreurs. Normalement, si tout fonctionne, un fichier portant un nom similaire à celui-ci (avec une date différente) : bddbbackup_2023-01-05.sql.gz est enregistré dans le même dossier que backup.sh.

🏠 > savebdd	
Nom ↑	Taille
 backup.sh	254 B
 bddbbackup_2023-01-05.sql.gz	361 B

C. METTRE EN PLACE LE DEPLOIEMENT CONTINU



Afin de réaliser le déploiement de l'application sur le dépôt Github, je dois procéder à plusieurs étapes.

Le but est de faire en sorte qu'à chaque push vers GitHub, le site en ligne soit aussi mis à jour. Cela va se faire en paramétrant GitHub pour lui dire vers quel ftp envoyer les fichiers.

Je dois créer un fichier "yaml" d'automatisation, dans le dépôt github, je sélectionne "actions", puis je clique sur le lien "set up a workflow yourself". Je supprime le contenu de "Edit new

file", et je le remplace par le contenu suivant.

```
1  on: push
2  name: Deploy website on push
3  jobs:
4    web-deploy:
5      name: Deploy
6      runs-on: ubuntu-latest
7      steps:
8        - name: Get latest code
9          uses: actions/checkout@v2
10
11        - name: Sync files
12          uses: SamKirkland/FTP-Deploy-Action@4.3.0
13          with:
14            server: node127-eu.n0c.com
15            server-dir: /public_html/mediatekformation/
16            username: alexandre@mediatek.go.yj.fr
17            password: ${ secrets.ftp_password }
```

Je clique sur "start", puis sur "commit new file", le dossier ".github/workflows" a été ajouté et contient le fichier yml de déploiement qui vient d'être créé. Je vais dans Netbeans et je fais un "Git > Remote > Pull > Next > Finish", le dossier est ainsi récupéré en local. J'enregistre maintenant le password du ftp dans le dépôt github, pour cela je sélectionne "Settings > Secrets > Actions > New repository secret", en racine du dépôt. Je remplis ensuite le champ "name" et le champ "value". Je clique ensuite sur "Add secret".

Je peux désormais utiliser le déploiement continu, pour cela je procède à l'ajout d'un titre dans ma page d'accueil, dans Netbeans, puis je fais un commit > push. Je vérifie en allant

dans l'onglet "actions" de mon dépôt GitHub, pour voir si tout à fonctionné.

BILAN

Lors de la réalisation de ce projet, j'ai atteint les objectifs concernant le nettoyage et l'optimisation du code, et l'ajout de plusieurs fonctionnalités.

Cependant, j'ai rencontré de grosses difficultés au début du projet pour comprendre la tâche 2 ainsi que la tâche 3, de la 1ère mission. En effet, le code ayant déjà été créé par un autre développeur, je me suis senti assez perdu lors de la relecture du code, et de l'application des consignes, notamment sur la compréhension de chacune des fonctions. J'ai finalement persévéré et je pense avoir réussi la mission.

J'ai également réussi à créer les pages d'administrations des formations, des playlists et des catégories, puis à créer une authentification keycloak sur ces pages. J'ai rencontré un problème au début de cette tâche car je n'avais pas la bonne version de keycloak, une fois la bonne version installée tout allait bien.

Le projet m'a permis d'apprendre à configurer une machine virtuelle keycloak, permettant l'accès au serveur d'authentification en HTTPS, et j'ai ainsi pu réaliser le

déploiement en ligne avec authentification sécurisée.
Globalement je suis satisfait du projet que j'ai réalisé.