

Table of Contents

Introduction :.....	3
Mission 1 :.....	4
Tâche 1.1 : nettoyer le code.....	4
Tâche 1.2 : nettoyer le code.....	11
Mission 2 :.....	15
Tâche 2.1 : gérer les formations.....	15
Tâche 2.2 : gérer les playlists.....	21
Tâche 2.3 : gérer les catégories.....	27
Tâche 2.4 : ajouter l'accès avec authentification.....	31
Mission 3 :.....	36
Tâche 3.1 : gérer les tests.....	36
Tâche 3.2 : créer la documentation technique.....	38
Tâche 3.3 : créer la documentation utilisateur.....	40
Mission 4 :.....	41
Tâche 4.1 : déployer le site.....	41
Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD.....	42
Tâche 4.3 : mettre en place le déploiement continu.....	44
Bilan final :.....	46
Problèmes rencontrés :.....	46

Introduction :

Mediatekformation est un site développé avec Symfony 6.4, et permet d'accéder aux vidéos d'auto-formation proposées par une chaîne de médiathèques et qui sont aussi accessibles sur YouTube. Actuellement, seule la partie front office a été développée. Elle contient les fonctionnalités globales suivantes :

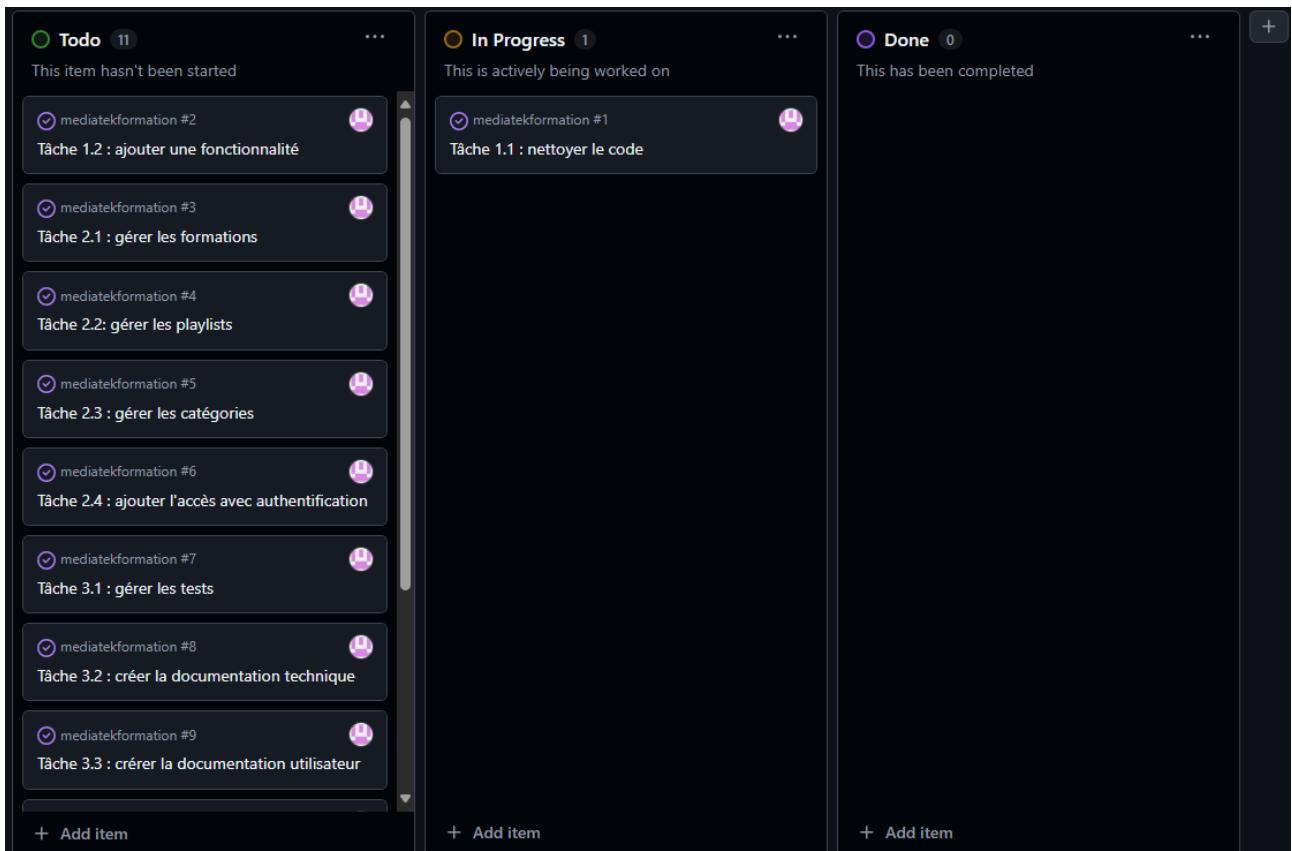
- consulter l'accueil
- consulter les formations
- consulter une formation
- consulter les playlists
- consulter une playlist
- consulter les CGU

Les langages principalement utilisés sont PHP, Twig (permet de séparer le code HTML de la logique PHP) et SQL. Le développement de cette application web prend notamment appui sur des technologies tels que Composer (gestion des dépendances PHP), Doctrine ORM (gestion des bases de données), Bootstrap (pour le CSS) ou encore PHPUnit (tests unitaires et fonctionnels).

Mission 1 :

Tâche 1.1 : nettoyer le code

- Nettoyer le code en suivant les indications de Sonarlint (excepté le code généré automatiquement par Symfony).



Temps estimé : 2h

Temps réel : 2h15min

Problème 1 :

php:S115 : Constant names should comply with a naming convention (1)

```
private const cheminImage = "https://i.ytimg.com/vi/";
```

Solution 1 :

En effet, les constantes doivent respecter une convention de nommage. De manière général, il faut les écrire en majuscule et en snake_case (underscore)

```
private const CHEMIN_IMAGE = "https://i.ytimg.com/vi/";
```

Problème 2 :

php:S1192 : String literals should not be duplicated (2)

```
#[Route('/formations', name: 'formations')]
public function index(): Response{
    $formations = $this->formationRepository->findAll();
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/formations.html.twig", [
        'formations' => $formations,
        'categories' => $categories
    ]);
}

#[Route('/formations/tri/{champ}/{ordre}/{table}', name: 'formations.sort')]
public function sort($champ, $ordre, $table=""): Response{
    $formations = $this->formationRepository->findAllOrderBy($champ, $ordre, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/formations.html.twig", [
        'formations' => $formations,
        'categories' => $categories
    ]);
}
```

Solution 2 :

Il faut créer une variable constante à laquelle est assigné le chemin de la vue formation, cela permet d'éviter les chaînes en dur dupliquées (donc si il faut changer le chemin de la vue formation, cela peut se faire à un seul endroit ce qui est plus simple et efficace)

```
private const CHEMIN_VUE_FORMATION = "pages/formations.html.twig";
```

```

#[Route('/formations', name: 'formations')]
public function index(): Response{
    $formations = $this->formationRepository->findAll();
    $categories = $this->categorieRepository->findAll();
    return $this->render(CHEMIN_VUE_FORMATION, [
        'formations' => $formations,
        'categories' => $categories
    ]);
}

#[Route('/formations/tri/{champ}/{ordre}/{table}', name: 'formations.sort')]
public function sort($champ, $ordre, $table=""): Response{
    $formations = $this->formationRepository->findAllOrderBy($champ, $ordre, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render(CHEMIN_VUE_FORMATION, [
        'formations' => $formations,
        'categories' => $categories
    ]);
}

```

Problème 3 :

php:S121 : Control structures should use curly braces (1)

```

foreach($categoriesFormation as $categorieFormation)
    if(!$categories->contains($categorieFormation->getName())){
        $categories[] = $categorieFormation->getName();
    }

```

Solution 3 :

Ici il manque les accolades de la boucle foreach

```

foreach($categoriesFormation as $categorieFormation){
    if(!$categories->contains($categorieFormation->getName())){
        $categories[] = $categorieFormation->getName();
    }
}

```

Problème 4 :

php:S131 : "switch" statements should have "default" clauses (1)

```
switch($champ) {
    case "name":
        $playlists = $this->playlistRepository->findAllOrderByNames($ordre);
        break;
}
```

Solution 4 :

Si aucune valeur du switch ne correspond à \$champ, il faut alors gérer cette situation à l'aide d'un "default". Cependant, il est préférable d'utiliser une condition if/else plutôt qu'un switch. Ici j'ai choisie d'assigner une valeur par défaut à playlists à l'aide de findAll()

```
if ($champ == "name")
{
    $playlists = $this->playlistRepository->findAllOrderByNames($ordre);
} else
{
    $playlists = $this->playlistRepository->findAll();
}
```

Problème 5 :

php:S1066 : Collapsible "if" statements should be merged (1)

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

Solution 5 :

On peut combiner les deux conditions avec l'opérateur &&, cela permet d'améliorer la lisibilité du code

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

Problème 6 :

Web:BoldAndItalicTagsCheck : "" and "" tags should be used (5)

```
<summary>A. Éditeur du site (ci-après "<i>l'Éditeur</i>")</summary>
```

Solution 6 :

Il ici il faut priviliger les tag "" et "" plutôt que "" and "<i>" respectivement

```
<summary>A. Éditeur du site (ci-après "<em>l'Éditeur</em>")</summary>
```

Problème 7 :

Web:ImgWithoutAltCheck : Image, area and button with image tags should have an "alt" attribute (4)

```

```

Solution 7 :

Si l'image ne s'affiche pas, il est vaut mieux afficher un texte alternatif à la place

```

```

Problème 8 :

Web:TableWithoutCaptionCheck : "<table>" tags should have a description (3)

```
<table class="table">
    <tr>
        {% for formation in formations %}
            <td>
                <div class="row">
                    <div class="col">
                        <!-- emplacement photo --&gt;
                        {% if formation.picture %}
                            &lt;a href="{{ path('formations.showone', {id:formation.id}) }}"&gt;
                                &lt;img src="{{ formation.picture }}" style="width:100%;height:auto;" alt="formation thumbnail"&gt;
                            &lt;/a&gt;
                        {% endif %}
                    &lt;/div&gt;
                    &lt;div class="col"&gt;
                        &lt;p&gt;{{ formation.publishedatstring }}&lt;/p&gt;
                        &lt;h5 class="text-info mt-1"&gt;
                            {{ formation.title }}
                        &lt;/h5&gt;
                        &lt;strong&gt;playlist : &lt;/strong&gt;{{ formation.playlist.name }}&lt;br /&gt;
                        &lt;strong&gt;catégories : &lt;/strong&gt;
                        {% for categorie in formation.categories %}
                            {{ categorie.name }}&ampnbsp;
                        {% endfor %}
                    &lt;/div&gt;
                &lt;/div&gt;
            {% endfor %}
        &lt;/td&gt;
    &lt;/tr&gt;
&lt;/table&gt;</pre>
```

Solution 8 :

Il suffit de rajouter une description à la table à l'aide de la balise <caption> pour expliquer le contenu du tableau

```
<table class="table">
    <caption>Formations récente</caption>
    <tr>
        {% for formation in formations %}
            <td>
                <div class="row">
                    <div class="col">
                        <!-- emplacement photo --&gt;
                        {% if formation.picture %}
                            &lt;a href="{{ path('formations.showone', {id:formation.id}) }}"&gt;
                                &lt;img src="{{ formation.picture }}" style="width:100%;height:auto;" alt="formation thumbnail"&gt;
                            &lt;/a&gt;
                        {% endif %}
                    &lt;/div&gt;
                    &lt;div class="col"&gt;
                        &lt;p&gt;{{ formation.publishedatstring }}&lt;/p&gt;
                        &lt;h5 class="text-info mt-1"&gt;
                            {{ formation.title }}
                        &lt;/h5&gt;
                        &lt;strong&gt;playlist : &lt;/strong&gt;{{ formation.playlist.name }}&lt;br /&gt;
                        &lt;strong&gt;catégories : &lt;/strong&gt;
                        {% for categorie in formation.categories %}
                            {{ categorie.name }}&ampnbsp;
                        {% endfor %}
                    &lt;/div&gt;
                &lt;/div&gt;
            {% endfor %}
        &lt;/td&gt;
    &lt;/tr&gt;
&lt;/table&gt;</pre>
```

Tâche 1.2 : nettoyer le code

- 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.

The image shows a digital task board with three vertical columns: Todo, In Progress, and Done. Each column has a header with a status icon, a count of items, and a '...' button. Below the headers are lists of tasks, each with a circular icon, a task ID, and a task name. At the bottom of each column is a '+ Add item' button.

Column	Status	Count	Task Description
Todo	Green circle	10	This item hasn't been started
In Progress	Orange circle	1	This is actively being worked on
Done	Purple circle	1	This has been completed

Todo Column:

- mediatekformation #3 Tâche 2.1 : gérer les formations
- mediatekformation #4 Tâche 2.2: gérer les playlists
- mediatekformation #5 Tâche 2.3 : gérer les catégories
- mediatekformation #6 Tâche 2.4 : ajouter l'accès avec authentification
- mediatekformation #7 Tâche 3.1 : gérer les tests
- mediatekformation #8 Tâche 3.2 : créer la documentation technique
- mediatekformation #9 Tâche 3.3 : créer la documentation utilisateur
- mediatekformation #10 Tâche 4.1 : déployer le site

In Progress Column:

- mediatekformation #2 Tâche 1.2 : ajouter une fonctionnalité

Done Column:

- mediatekformation #1 Tâche 1.1 : nettoyer le code

Temps estimé : 2h

Temps réel : 1h50min

Voici le résultat :

The screenshot shows a website for 'MediaTek86' with a header featuring a logo of people working on laptops and the text 'Des formations pour tous sur des outils numériques'. Below the header, there's a navigation bar with links for 'Accueil', 'Formations', and 'Playlists'. The main content area displays a table of playlists. The columns are 'playlist' (with navigation arrows), 'catégories' (with dropdown and sorting arrows), and 'nombre de formations' (with sorting arrows). A red vertical line highlights the 'nombre de formations' column. The data in the table is as follows:

playlist	catégories	nombre de formations
Bases de la programmation (C#)	C# POO	74
Programmation sous Python	Python POO	19
MCD : exercices progressifs	MCD	18
TP Android (programmation mobile)	Android SQL Java	18
Compléments Android (programmation mobile)	Android	13
Visual Studio 2019 et C#	C# POO	11

La classe PlaylistRepository.php possédaient seulement une fonction permettant de récupérer toutes les playlists triés sur le nom de la playlist, il a donc fallu créer une fonction permettant de les triés selon le nombre de formation :

```
public function findAllOrderByFormation($ordre) : array
{
    return $this->createQueryBuilder('p')
        ->leftJoin('p.formations', 'f')
        ->groupBy('p.id')
        ->orderBy('COUNT(f.id)', $ordre)
        ->getQuery()
        ->getResult();
}
```

Cette requête récupère toutes les playlist, compte combien de formations elles contiennent et les trie selon ce nombre par ordre croissant ou décroissant.

Après cela, il faut modifier la méthode sort de PlaylistsController.php pour qu'elle gère le champ "formation", en utilisant la nouvelle fonction :

```

if ($champ == "name") {
    $playlists = $this->playlistRepository->findAllOrderByNome ($ordre);
} elseif ($champ == "formation") {
    $playlists = $this->playlistRepository->findAllOrderByFormation ($ordre);
} else {
    $playlists = $this->playlistRepository->findAll();
}

```

Ensuite, au niveau de la vue on créer une nouvelle colonne "nombre de formations" avec deux boutons pour le tri croissant ou décroissant, et on utilise la fonction sort de playlists en renseignant les paramètres (champ et ordre) :

```

<th class="text-left align-top" scope="col">
    nombre de formations<br />
    <a href="{{ path('playlists.sort', {champ:'formation', ordre:'ASC'}) }}"></a>
    <a href="{{ path('playlists.sort', {champ:'formation', ordre:'DESC'}) }}"></a>
</th>

```

Et Pour chacune des playlists on affiche dans la colonne correspondante le nombre de formations :

```

{% for k in 0..playlists|length-1 %}
    <tr class="align-middle">
        <td>
            <h5 class="text-info">
                {{ playlists[k].name }}
            </h5>
        </td>
        <td class="text-left">
            {% set categories = playlists[k].categoriesplaylist %}
            {% if categories|length > 0 %}
                {% for c in 0..categories|length-1 %}
                    &nbsp;{{ categories[c] }}
                {% endfor %}
            {% endif %}
        </td>
        <td class="text-left">
            {{ playlists[k].formations|length }}
        </td>
        <td class="text-center">
            <a href="{{ path('playlists.showone', {id:playlists[k].id}) }>
        </td>
    </tr>
{% endfor %}

```

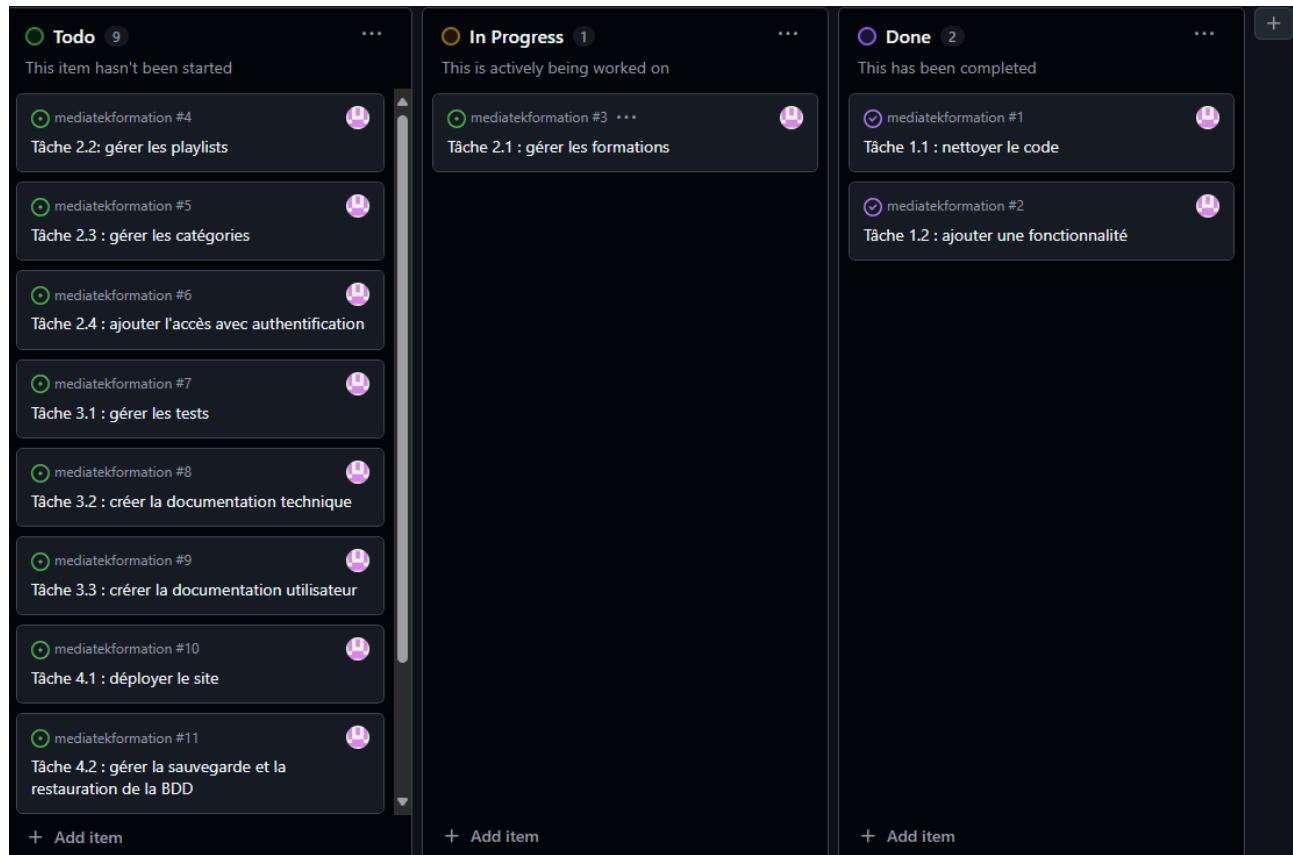
Enfin, pour la vue de la page d'une playlist, on ajoute simplement cette ligne qui affiche le nombre de formation en utilisant le filtre twig `|length` :

```
<div class="col">
    <h4 class="text-info mt-5">{{ playlist.name }}</h4>
    <strong>catégories : </strong>
    <!-- boucle pour afficher les catégories --&gt;
    {% set anccategorie = '' %}
    {% for playlist in playlistcategories %}
        {{ playlist.name }}&ampnbsp;;
    {% endfor %}
    &lt;br /&gt;&lt;br /&gt;
    &lt;strong&gt;description :&lt;/strong&gt;&lt;br /&gt;
        {{ playlist.description|nl2br }}
    &lt;br /&gt;&lt;br /&gt;
    &lt;strong&gt;nombre de formations : &lt;/strong&gt;{{ playlist.formations|length }}</pre>
```

Mission 2 :

Tâche 2.1 : gérer les formations

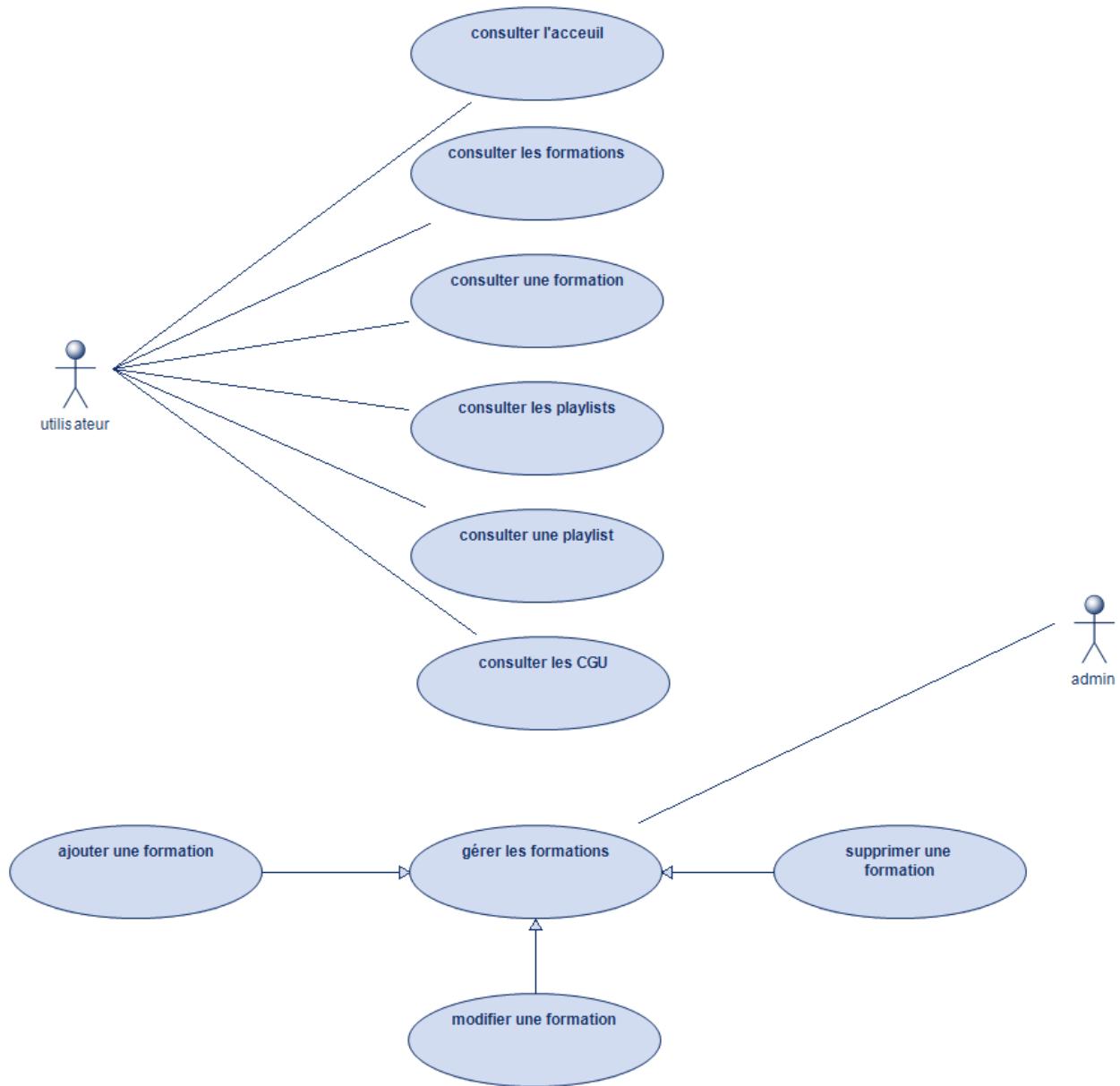
- Permettre l'ajout, la modification et la suppression d'une formation.



Temps estimé : 5h

Temps réel : 5h30min

Diagramme de cas d'utilisation :



Pour cette tâche, il faudra une nouvelle page permettant de lister toutes les formations et pour chacune d'elle un bouton permettant de modifier la formation et un bouton permettant de supprimer la formation. Il faudra notamment un bouton permettant d'ajouter une nouvelle formation sur cette page. Cela correspond à de la gestion propre au back-office, ce sera donc gérer dans la partie admin du site, voici un aperçu de cette page ainsi que de son URL:

localhost/mediatekformation/public/index.php/admin

Gestion des formations

Ajouter une nouvelle formation

formation	playlist	catégories	date	
< >	< >	< >	< >	
<input type="text"/> filtrer	<input type="text"/> filtrer	<input type="text"/>		
Eclipse n°8 : Déploiement	Eclipse et Java	Java	04/01/2021	<button>Editer</button> <button>Supprimer</button>
Eclipse n°7 : Tests unitaires	Eclipse et Java	Java	02/01/2021	<button>Editer</button> <button>Supprimer</button>
Eclipse n°6 : Documentation technique	Eclipse et Java	Java	30/12/2020	<button>Editer</button> <button>Supprimer</button>
Eclipse n°5 : Refactoring	Eclipse et Java	Java	29/12/2020	<button>Editer</button> <button>Supprimer</button>
Eclipse n°4 : WindowBuilder	Eclipse et Java	Java	09/11/2020	<button>Editer</button> <button>Supprimer</button>
Eclipse n°3 : GitHub et Eclipse	Eclipse et Java	Java	07/11/2020	<button>Editer</button> <button>Supprimer</button>
				<button>Editer</button>

Ensute, il faudra une page vers laquelle sera diriger l'admin lorsqu'il cliquera sur le bouton éditer, d'une formation. Cette page doit contenir un formulaire prérempli avec les informations existantes de la formation et d'un bouton permettant d'enregistrer les modifications, voici un aperçu :

localhost/mediatekformation/public/index.php/admin/edit/1

Gestion des formations

Détail formation :

date

01/04/2021



titre

Eclipse n°8 : Déploiement

Description

Exécution de l'application en dehors de l'IDE, en invite de commande.
Création d'un fichier jar pour le déploiement de l'application.



Video id

Z4yTTXka958

Playlist

Eclipse et Java



Categories

Java
UML
C#
Python



Enregistrer

Enfin, une dernière page sera requise pour ajouter une nouvelle formation, similaire à la page de modification, mais cette fois-ci les champs ne seront pas pré-rempli (mise à par la date et une playlist assigné par défaut) :

localhost/mediatekformation/public/index.php/admin/ajout

Gestion des formations

Nouvelle formation :

date

03/05/2025



titre

Description

Video id

Playlist

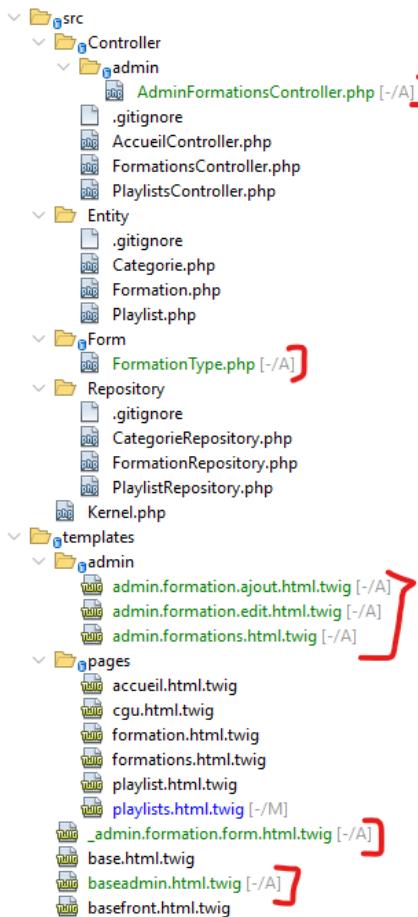


Catégories

- Java
- UML
- C#
- Python

Enregistrer

Voici l'arborescence des fichiers de l'application avec les nouveaux fichiers ajoutés :



On observe clairement la présence de trois nouvelles pages dans le dossier *templates/admin*, ainsi que l'ajout d'un nouveau contrôleur, "AdminFormationController.php" car comme mentionné précédemment, ces pages sont spécifiquement dédiées à la gestion de la partie administrateur du site. Cette classe contient les fonctions d'ajout/modification/suppression d'une formation, qui sont appelé lors de l'interaction avec les boutons correspondant.

Un formulaire nommé "FormationType.php" (généré à l'aide de Symfony) est également présent. Ce formulaire est liée à l'entité Formation de la BDD et génère donc automatiquement tous les champs de cette entité. Ces donc ce formulaire qui sera utilisé pour l'ajout et la modification d'une formation.

Les pages d'ajout/modification/suppression sont toutes des extensions de la page *baseadmin.html.twig* car ces pages partage le même titre("Gestion des formations"). La page d'ajout et la page de modification incluent toutes deux le fichier *_admin.formation.form.html.twig*, qui contient le formulaire.

Tâche 2.2 : gérer les playlists

- Permettre l'ajout, la modification et la suppression d'une playlist.

The image shows a digital task board with three vertical columns: Todo, In Progress, and Done. Each column has a header with a color-coded circle (green for Todo, yellow for In Progress, purple for Done), the status name, a count of items, and a '...' button. Below the headers are lists of tasks, each with a small profile picture icon and a plus sign icon to its right.

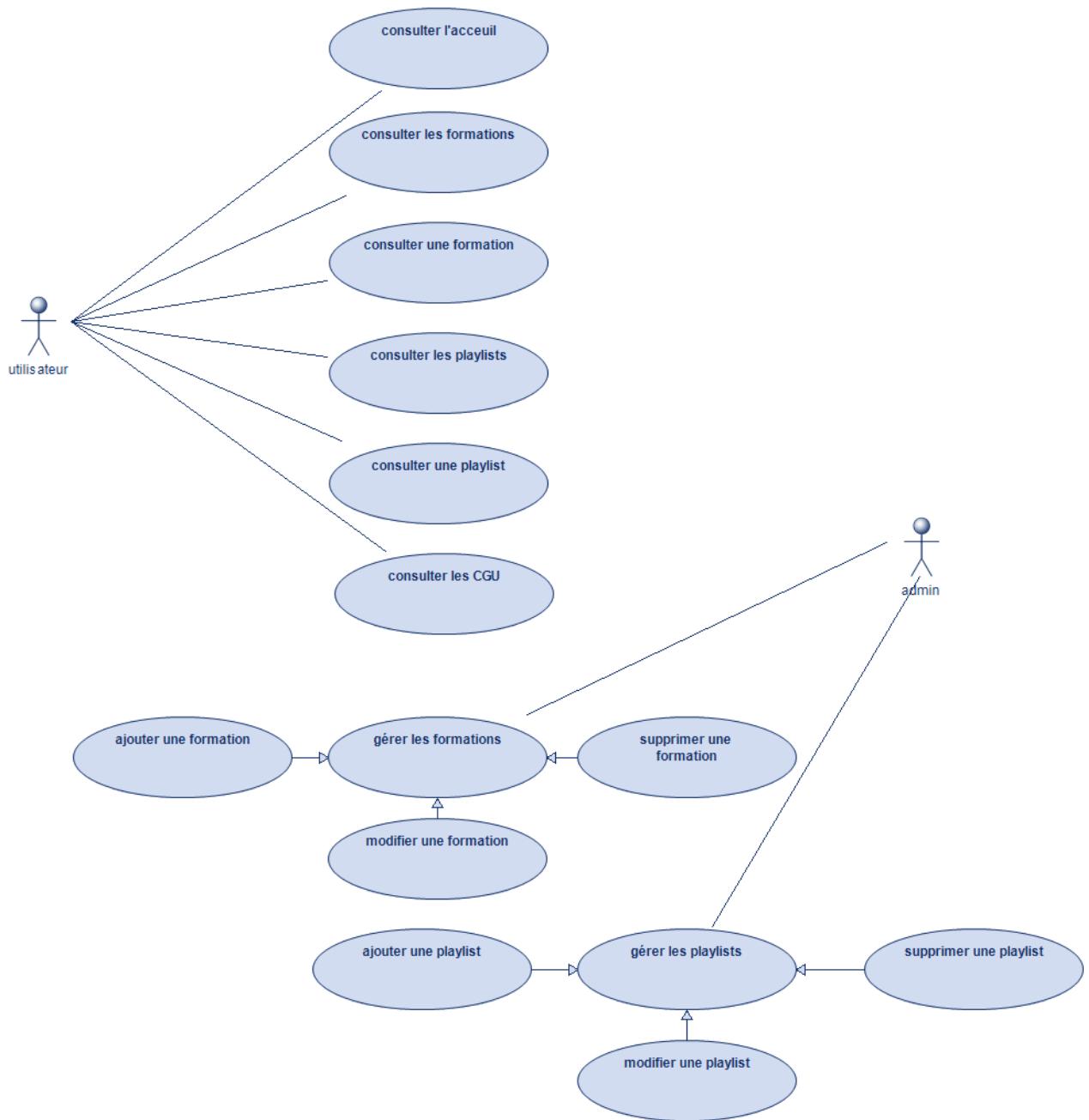
- Todo:** 8 items. This item hasn't been started.
 - mediatekformation #5: Tâche 2.3 : gérer les catégories
 - mediatekformation #6: Tâche 2.4 : ajouter l'accès avec authentification
 - mediatekformation #7: Tâche 3.1 : gérer les tests
 - mediatekformation #8: Tâche 3.2 : créer la documentation technique
 - mediatekformation #9: Tâche 3.3 : créer la documentation utilisateur
 - mediatekformation #10: Tâche 4.1 : déployer le site
 - mediatekformation #11: Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD
 - mediatekformation #12: Tâche 4.3 : mettre en place le déploiement continu
- In Progress:** 1 item. This is actively being worked on.
 - mediatekformation #4: Tâche 2.2: gérer les playlists
- Done:** 3 items. This has been completed.
 - mediatekformation #1: Tâche 1.1 : nettoyer le code
 - mediatekformation #2: Tâche 1.2 : ajouter une fonctionnalité
 - mediatekformation #3: Tâche 2.1 : gérer les formations

At the bottom of each column, there is a '+ Add item' button.

Temps estimé : 5h

Temps réel : 5h

Diagramme de cas d'utilisation :



Ici le principe et le même que pour la gestion des formations. Il y aura donc une nouvelle page permettant de lister toutes les playlists et pour chacune d'elle un bouton permettant de modifier la playlist et un bouton permettant de la supprimer, ainsi qu'un bouton permettant d'ajouter une nouvelle playlist sur cette page :

localhost/mediatekformation/public/index.php/admin/playlists

Gestion des formations

Formations Playlists

Ajouter une nouvelle playlist

playlist	catégories	nombre de formations	Editer	Supprimer
Bases de la programmation (C#)	C# POO	74	Editer	Supprimer
Compléments Android (programmation mobile)	Android	13	Editer	Supprimer
Cours Composant logiciel	Cours	2	Editer	Supprimer
Cours Curseurs	SQL Cours POO	2	Editer	Supprimer
Cours de programmation objet	POO Cours	1	Editer	Supprimer
Cours Informatique embarquée	Cours	1	Editer	Supprimer
Cours MCD MLD MPD	MCD Cours	2	Editer	Supprimer
Cours MCD vs Diagramme de classes	MCD Cours	2	Editer	Supprimer
Cours Merise/2	MCD Cours	1	Editer	Supprimer
Cours Modèle relationnel et MCD	MCD Cours	1	Editer	Supprimer

Une page pour la modification d'une playlist après l'interaction avec le bouton éditer :

localhost/mediatekformation/public/index.php/admin/playlist/edit/13

Gestion des formations

Formations Playlists

Détail 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

Submit



Bases de la programmation n°74 - POO : collections

Bases de la programmation n°73 - POO : polymorphisme et abstraction

Bases de la programmation n°72 - POO : héritage

Bases de la programmation n°71 - POO : petit qcm

Bases de la programmation n°70 - POO : encapsulation

Et une page pour l'ajout d'une nouvelle playlist :

localhost/mediatekformation/public/index.php/admin/playlist/ajout

Gestion des formations

Formations Playlists

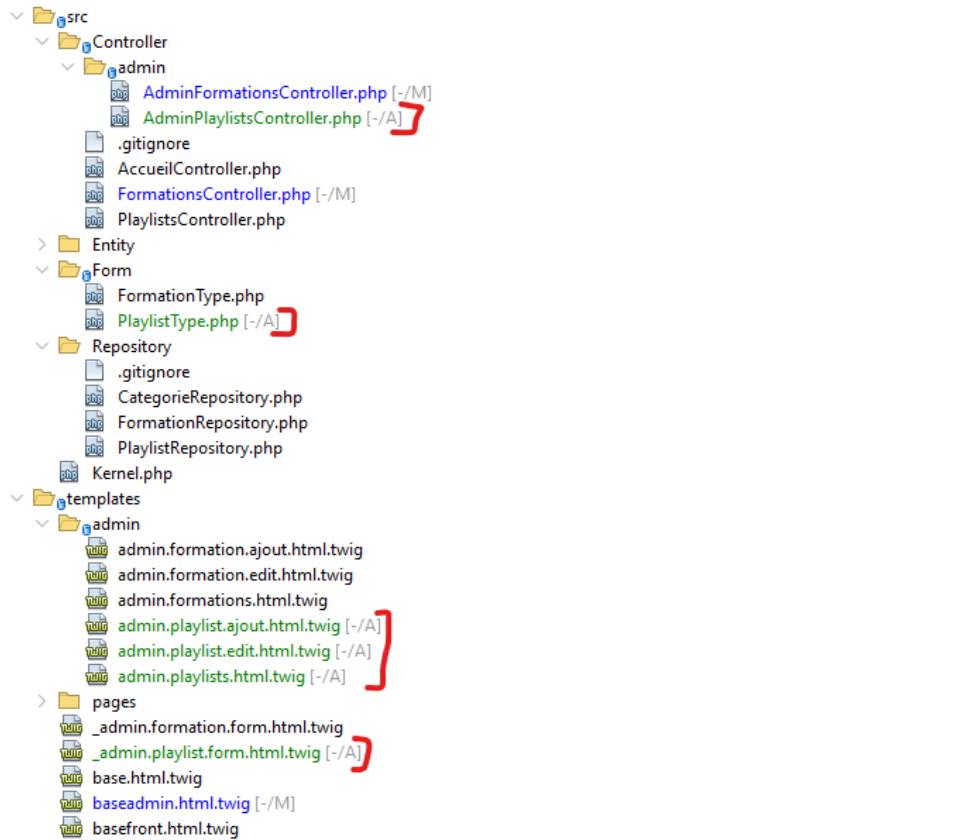
Nouvelle playlist :

nom

Description

Submit

Voici l'arborescence des fichiers de l'application avec les nouveaux fichiers ajoutés :



Il y a bien la présence de trois nouvelles pages dans le dossier *templates/admin*, ainsi que l'ajout d'un nouveau contrôleur, "AdminPlaylistController.php". Cette classe contient les fonctions d'ajout/modification/suppression d'une formation, qui sont appelé lors de l'interaction avec les boutons correspondant.

Comme pour les formations, une class nommée "PlaylistType.php" est également présente pour générer un formulaire qui servira pour la page page d'ajout et de modification d'une playlist.

La page d'ajout et la page de modification incluent toutes deux le fichier *_admin.playlist.form.html.twig*, qui contient le formulaire.

Tâche 2.3 : gérer les catégories

- Permettre l'ajout et la suppression d'une catégorie .

The image shows a digital task board with three vertical columns: Todo, In Progress, and Done. Each column has a header with a color-coded circle (green for Todo, orange for In Progress, purple for Done), the status name, the count of items, and a '...' button.

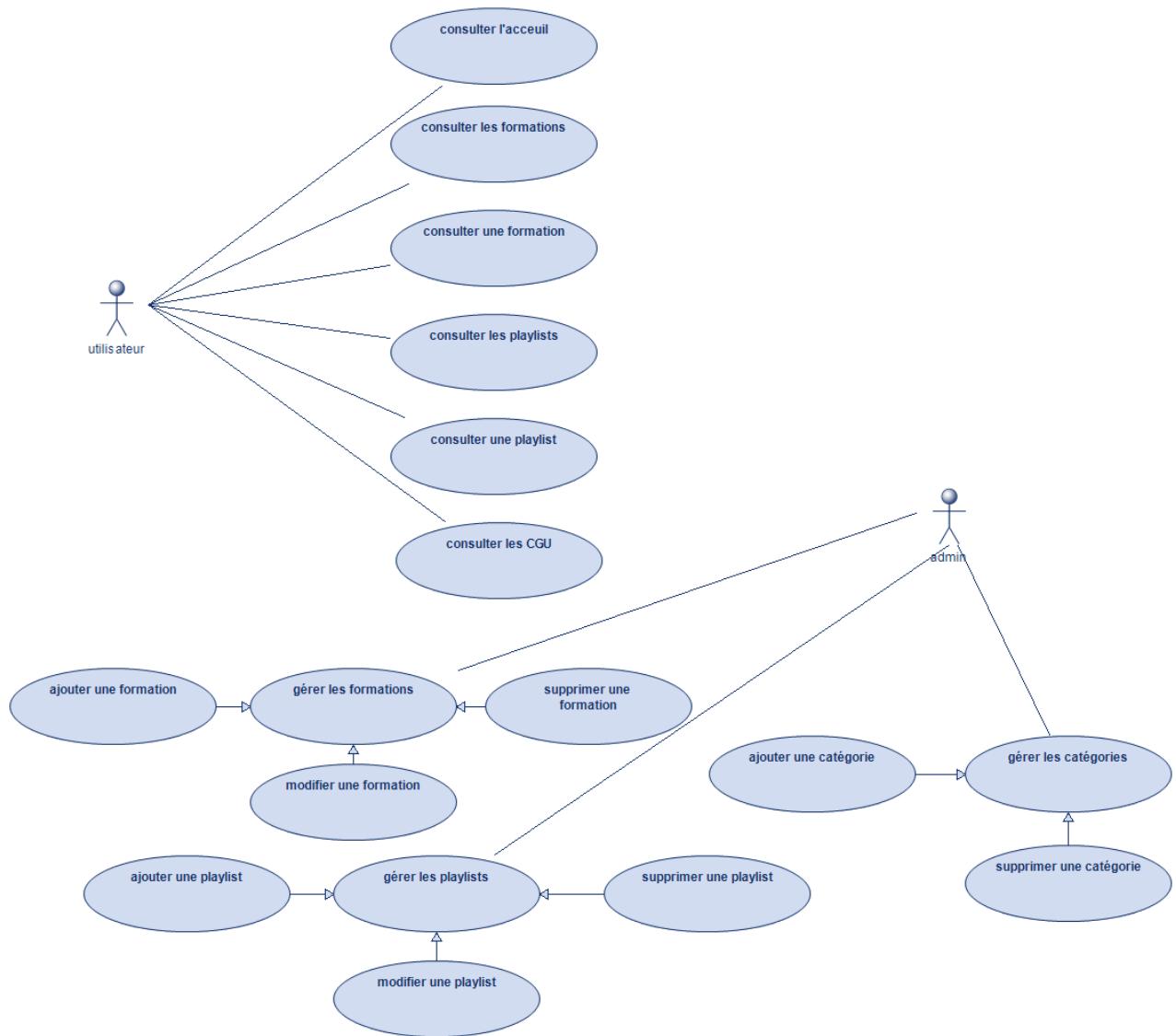
- Todo:** 7 items. This item hasn't been started.
 - mediatekformation #6: Tâche 2.4 : ajouter l'accès avec authentification
 - mediatekformation #7: Tâche 3.1 : gérer les tests
 - mediatekformation #8: Tâche 3.2 : créer la documentation technique
 - mediatekformation #9: Tâche 3.3 : créer la documentation utilisateur
 - mediatekformation #10: Tâche 4.1 : déployer le site
 - mediatekformation #11: Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD
 - mediatekformation #12: Tâche 4.3 : mettre en place le déploiement continu
- In Progress:** 1 item. This is actively being worked on.
 - mediatekformation #5: Tâche 2.3 : gérer les catégories
- Done:** 4 items. This has been completed.
 - mediatekformation #1: Tâche 1.1 : nettoyer le code
 - mediatekformation #2: Tâche 1.2 : ajouter une fonctionnalité
 - mediatekformation #3: Tâche 2.1 : gérer les formations
 - mediatekformation #4: Tâche 2.2: gérer les playlists

At the bottom of each column, there is a '+ Add item' button.

Temps estimé : 3h

Temps réel : 2h40min

Diagramme de cas d'utilisation :

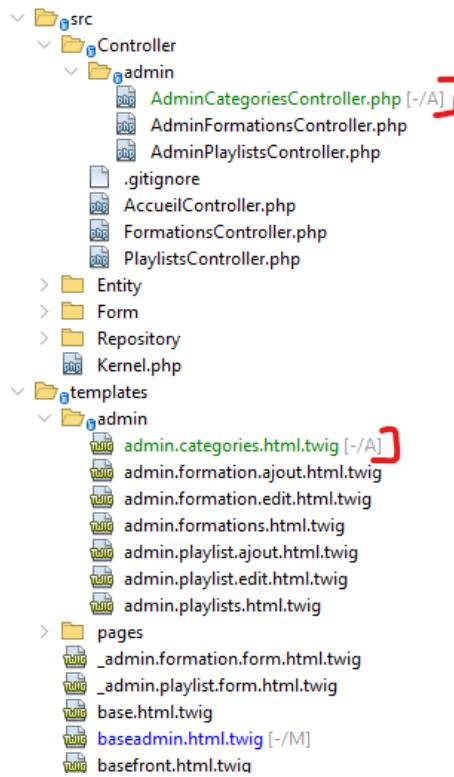


Pour cette tâche, il faudra une nouvelle page permettant de lister toutes les catégories et pour chacune d'elle un bouton permettant de la supprimer, ainsi qu'un simple formulaire permettant d'ajouter une nouvelle catégorie :

localhost/mediatekformation/public/index.php/admin/categories

Gestion des formations	
Nom	Actions
Java	<button>Supprimer</button>
UML	<button>Supprimer</button>
C#	<button>Supprimer</button>
Python	<button>Supprimer</button>
MCD	<button>Supprimer</button>
Android	<button>Supprimer</button>

Voici l'arborescence des fichiers de l'application avec les nouveaux fichiers ajoutés :



On observe la présence d'une nouvelle page "admin.categories.html.twig" dans le dossier *templates/admin*, ainsi que l'ajout d'un nouveau contrôleur, "AdminCategoriesController.php". Cette classe contient les fonctions d'ajout/suppression d'une formation, qui sont appelées lors de l'interaction avec les boutons correspondants.

Étant donné la simplicité du formulaire d'une catégorie (seulement un nom), ce dernier a été directement implémenté dans la page "admin.categories.html.twig" :

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

Tâche 2.4 : ajouter l'accès avec authentification

- Ajouter l'accès avec authentification à la partie back office.

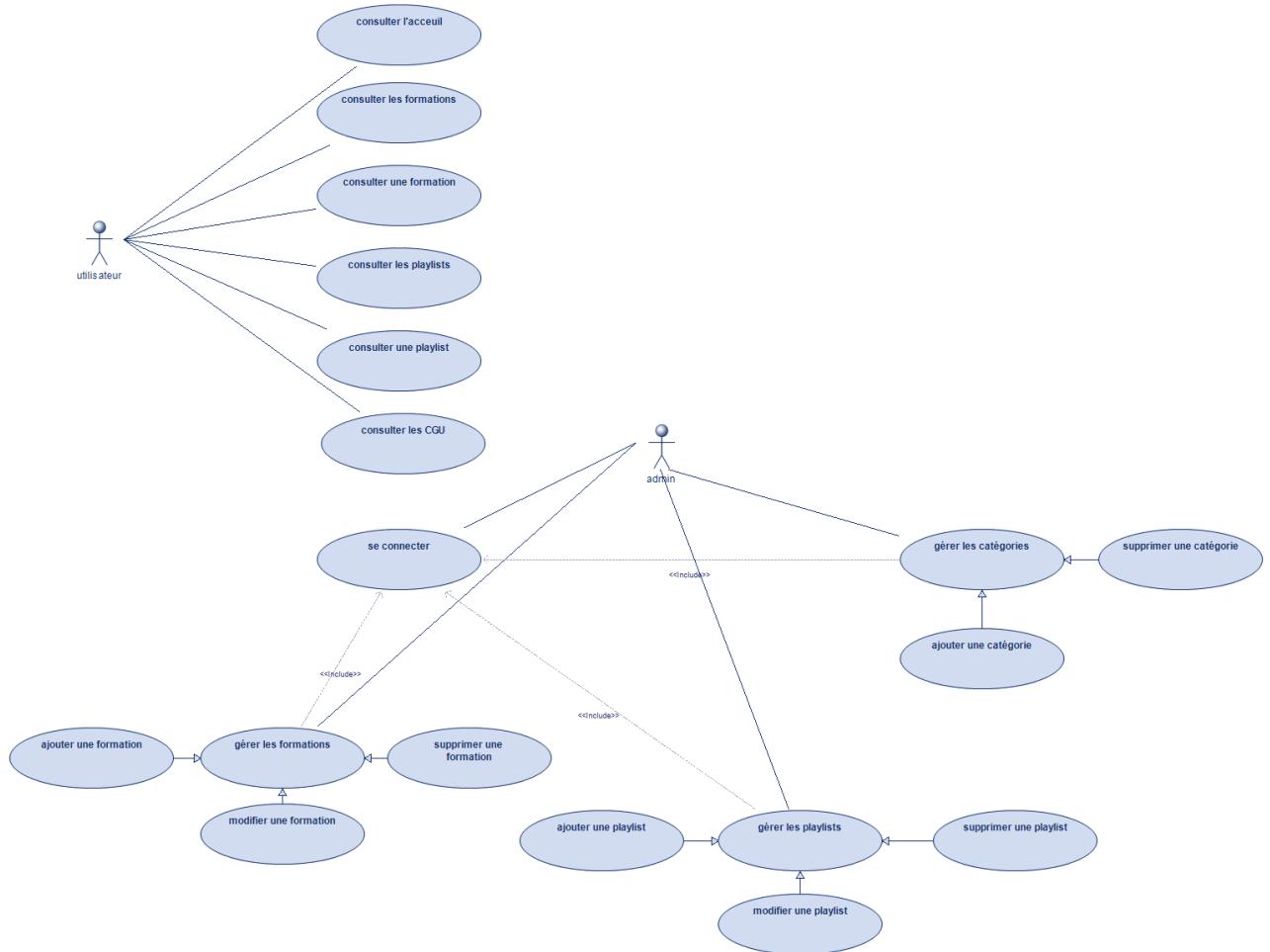
The image shows a digital task board with three vertical columns: Todo, In Progress, and Done. Each column has a header with a status indicator (green for Todo, yellow for In Progress, purple for Done), the count of items, and a 'More options' button. Below the headers, each column lists items with a user icon, a task ID, and a task name. At the bottom of each column is a '+ Add item' button.

Todo	In Progress	Done
This item hasn't been started	This is actively being worked on	This has been completed
mediatekformation #7 Tâche 3.1 : gérer les tests	mediatekformation #6 Tâche 2.4 : ajouter l'accès avec authentification	mediatekformation #1 Tâche 1.1 : nettoyer le code
mediatekformation #8 Tâche 3.2 : créer la documentation technique		mediatekformation #2 Tâche 1.2 : ajouter une fonctionnalité
mediatekformation #9 Tâche 3.3 : créer la documentation utilisateur		mediatekformation #3 Tâche 2.1 : gérer les formations
mediatekformation #10 Tâche 4.1 : déployer le site		mediatekformation #4 Tâche 2.2: gérer les playlists
mediatekformation #11 Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD		mediatekformation #5 Tâche 2.3 : gérer les catégories
mediatekformation #12 Tâche 4.3 : mettre en place le déploiement continu		

Temps estimé : 4h

Temps réel : 4h10min

Diagramme de cas d'utilisation :



Pour cette tâche, il faudra une page d'authentification quand l'utilisateur tente d'accéder au back-office (admin), voici un aperçu :

localhost/mediatekformation/public/index.php/login

Authentifiez-vous

Login:

Password:

se connecter

Ainsi que d'un moyen de se déconnecter sous forme de lien accessible sur toutes les pages après authentification :

localhost/mediatekformation/public/index.php/



MediaTek86

Des formations pour tous
sur des outils numériques

Accueil Formations Playlists

se déconnecter

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

Pour ce faire, nous aurons besoin de créer une classe User.php pour mémoriser le users admin et l'enregistrer dans la BDD. Il faut aussi créer un formulaire, ce qui implique une nouvelle page "login/index.html.twig" et un nouveau controller "LoginController.php" :

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

{% block title %}Hello LoginController!{% endblock %}

{% block body %}
    {% if error %}
        <div>{{ error.messageKey|trans(error.messageData, 'security') }}</div>
    {% endif %}

    <form action="{{ path('app_login') }}" method="post">
        <h3>Authentifiez-vous</h3>
        <label for="username">Login:</label>
        <input type="text" id="username" name="_username" value="{{ last_username }}"
               class="form-control" required autofocus/>

        <label for="password">Password:</label>
        <input type="password" id="password" name="_password"
               class="form-control" required autofocus/>

        <input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}>

        <br/>
        <button class="btn btn-lg btn-primary" type="submit">se connecter</button>
    </form>
{% endblock %}
```

```
class LoginController extends AbstractController
{
    #[Route('/login', name: 'app_login')]
    public function index(AuthenticationUtils $authenticationUtils): Response
    {

        // récupération éventuelle de l'erreur
        $error = $authenticationUtils->getLastAuthenticationError();
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('login/index.html.twig', [
            'last_username' => $lastUsername,
            'error' => $error
        ]);
    }

    #[Route('/logout', name: 'logout')]
    public function logout()
    {
    }
}
```

La classe controller récupère les éventuelles erreurs si l'utilisateur ne saisit pas les bonnes informations, et ré-affiche son login. Il y a aussi une fonction qui permet de fixer la route pour la déconnexion.

Notez que le formulaire dans la vue est protégé grâce à l'utilisation de jeton (token CSRF)

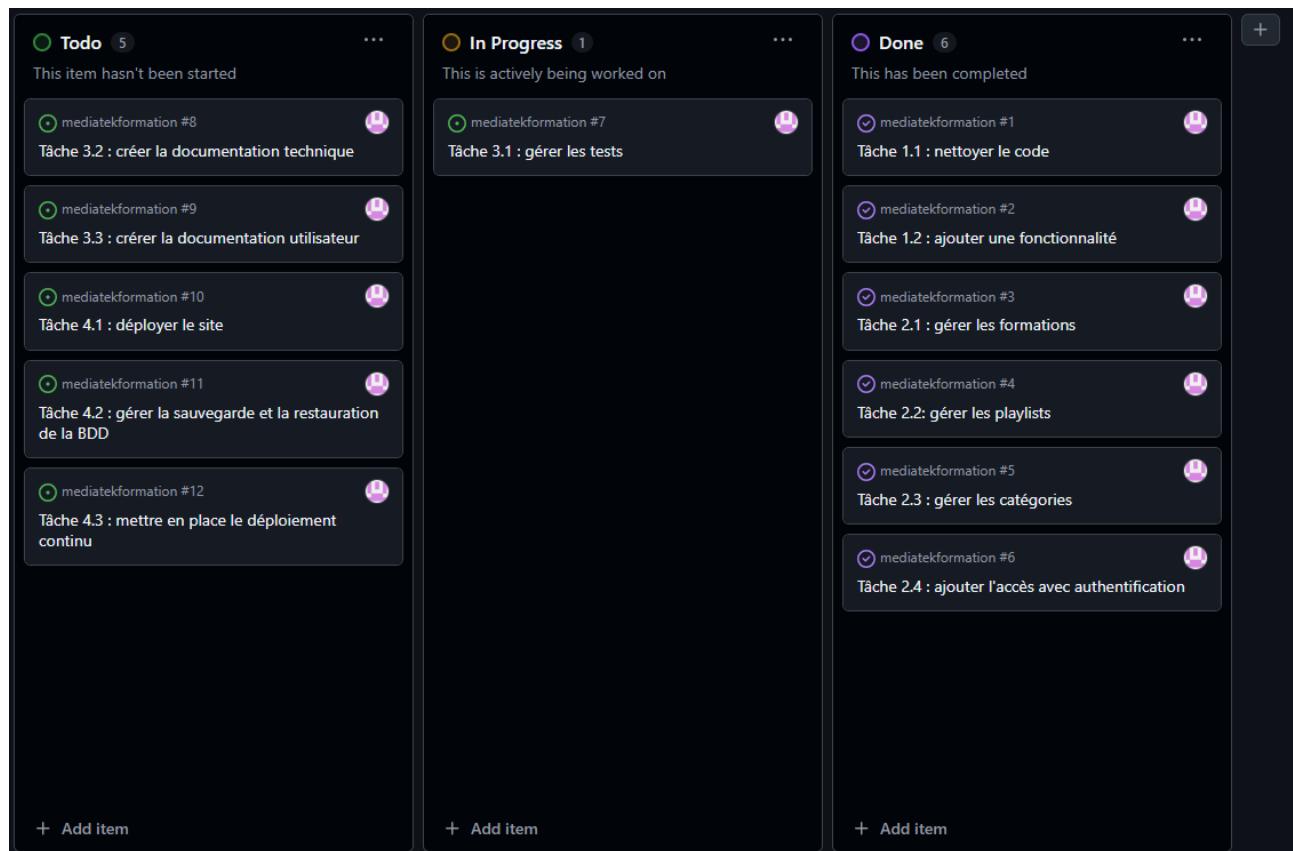
Enfin dans la vue "base.html.twig", on ajoute le lien de déconnexion seulement si l'utilisateur est connecté :

```
<body>
    <!-- entête -->
    <div class="container text-end">
        {% if app.user %}
            <a href="{{ path('logout') }}>se déconnecter</a>
        {% endif %}
    </div>
    {% block top %}{% endblock %}
    <!-- corps -->
    <div class="container">
        {% block body %}{% endblock %}
    </div>
    <!-- footer -->
    {% block footer %}{% endblock %}
    <!-- javascript -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4ZUI0qizoJcfXp20ke��" crossorigin="anonymous"></script>
</body>
```

Mission 3 :

Tâche 3.1 : gérer les tests

- test unitaire sur la méthode qui retourne la date au format string;
- tests d'intégration sur les règles de validation;
- tests d'intégration sur les Repository;
- tests fonctionnels(accès à l'accueil, tris, filtres...);
- tests de compatibilité de navigateurs.



Temps estimé : 7h

Temps réel : 7h 30min

Contexte : MediaTek86
 Situation professionnelle : Symfony
 Application : mediatekformation (site de mise à disposition des auto-formations).

Plan de tests

Tests unitaires

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la méthode getPublishedAtString() de la classe Formation pour voir si elle retourne la bonne date au bon format.	Test unitaire lancé avec la date : 20XX-01-04 17:00:12	04/01/20XX	OK

Tests d'intégration

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler toutes les méthodes ajoutées dans les classes FormationRepository, PlaylistRepository et CategorieRepository	Test d'intégration sur chacune des méthodes des ces 3 classes	Chacun des tests sur les méthodes vérifie leur bon fonctionnement et renvoie le résultat OK à la console	OK
Lors de l'ajout ou de la modification d'une formation, contrôler que la date n'est pas postérieure à aujourd'hui	Deux test permettant la validité et la non validité d'une date	Le test de validité est correct si la date est égal ou antérieur à aujourd'hui. Le test de non validité est correct si la date est postérieur à aujourd'hui.	OK

Tests fonctionnels

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler que la page d'accueil est accessible	Test fonctionnel avec une requête "GET" sur la route de la page d'accueil	La route est trouvée avec un code 200 à la requête	OK
Contrôler que les tris fonctionnent	Deux Tests Fonctionnel sur les routes selon les tris (ASC et DESC)	La première ligne trouvé correspond aux attentes selon le tri croissant ou décroissant.	OK
Contrôler que les filtres fonctionnent	Test fonctionnel qui simule la soumission du formulaire avec un filtre de recherche spécifique	La ligne trouvé doit être identique à la recherche	OK
Contrôler que le clic sur un lien dans une liste permet d'accéder à la bonne page.	Test fonctionnel simulant un clic sur un lien	Le code de la réponse est 200	OK

1

Tests de compatibilité

But du test	Action de contrôle	Résultat attendu	Bilan
Vérifier l'affichage correct de la page d'accueil	Ouvrir l'application dans différents navigateurs (Chrome, Firefox, Edge) et observer l'affichage	La mise en page, les couleurs et les éléments s'affichent correctement et de manière identique sur tous les navigateurs	OK
Tester la navigation entre les pages	Cliquer sur les liens du menu et observer si les pages se chargent correctement	Les liens fonctionnent et redirigent vers les bonnes pages sans erreur	OK
Vérifier le bon fonctionnement des formulaires	Remplir et soumettre un formulaire dans chaque navigateur	Les données sont envoyées et traitées correctement, sans erreur d'affichage ou de validation	OK

Tâche 3.2 : créer la documentation technique

- Générer la documentation technique pour l'ensemble de l'application (excepté le code généré automatiquement)

The image shows a digital task board with three columns: Todo, In Progress, and Done.

- Todo:** 4 items. This item hasn't been started.
 - mediatekformation #9: Tâche 3.3 : créer la documentation utilisateur
 - mediatekformation #10: Tâche 4.1 : déployer le site
 - mediatekformation #11: Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD
 - mediatekformation #12: Tâche 4.3 : mettre en place le déploiement continu
- In Progress:** 1 item. This is actively being worked on.
 - mediatekformation #8: Tâche 3.2 : créer la documentation technique
- Done:** 7 items. This has been completed.
 - mediatekformation #1: Tâche 1.1 : nettoyer le code
 - mediatekformation #2: Tâche 1.2 : ajouter une fonctionnalité
 - mediatekformation #3: Tâche 2.1 : gérer les formations
 - mediatekformation #4: Tâche 2.2: gérer les playlists
 - mediatekformation #5: Tâche 2.3 : gérer les catégories
 - mediatekformation #6: Tâche 2.4 : ajouter l'accès avec authentification
 - mediatekformation #7: Tâche 3.1 : gérer les tests

Each card includes a small profile picture icon next to the task name. There are also "Add item" buttons at the bottom of each column.

Temps estimé : 1h

Temps réel : 40min

mediatekformation

Search (Press "/" to focus)

Namespaces

DoctrineMigrations

App

Controller

DataFixtures

Entity

Form

Repository

Tests

Container0v7yyK

ContainerSe4CxSN

Proxies

CG

Symfony

Config

Component

Contracts

Bundle

Bridge

Flex

Polyfill

UX

ContainerOp44cFK

ContainerWmqHkw0

Composer

Autoload

Semver

Documentation

Table of Contents ⚡

Packages

Application

Namespaces

DoctrineMigrations

App

Container0v7yyK

ContainerSe4CxSN

Proxies

Symfony

ContainerOp44cFK

ContainerWmqHkw0

Composer

DAMA

Doctrine

Command

Fixtures

Egulias

Masterminds

Monolog

DeepCopy

PhpParser

Tâche 3.3 : créer la documentation utilisateur

- Créer une vidéo de moins de 5mn qui présente l'ensemble des fonctionnalités.

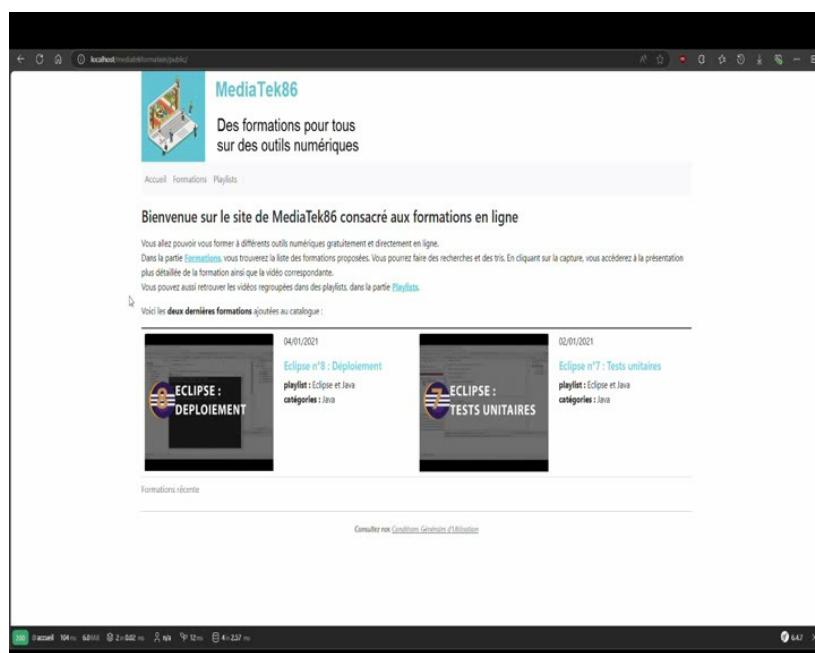
The image shows a digital task board with three columns: Todo, In Progress, and Done.

- Todo:** Contains three items:
 - mediatekformation #10: Tâche 4.1 : déployer le site
 - mediatekformation #11: Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD
 - mediatekformation #12: Tâche 4.3 : mettre en place le déploiement continu
- In Progress:** Contains one item:
 - mediatekformation #9: Tâche 3.3 : créer la documentation utilisateur
- Done:** Contains eight items, all with checkmarks:
 - mediatekformation #1: Tâche 1.1 : nettoyer le code
 - mediatekformation #2: Tâche 1.2 : ajouter une fonctionnalité
 - mediatekformation #3: Tâche 2.1 : gérer les formations
 - mediatekformation #4: Tâche 2.2: gérer les playlists
 - mediatekformation #5: Tâche 2.3 : gérer les catégories
 - mediatekformation #6: Tâche 2.4 : ajouter l'accès avec authentification
 - mediatekformation #7: Tâche 3.1 : gérer les tests
 - mediatekformation #8: Tâche 3.2 : créer la documentation technique

Each card includes a small profile picture and a plus sign (+) icon at the bottom right.

Temps estimé : 1h

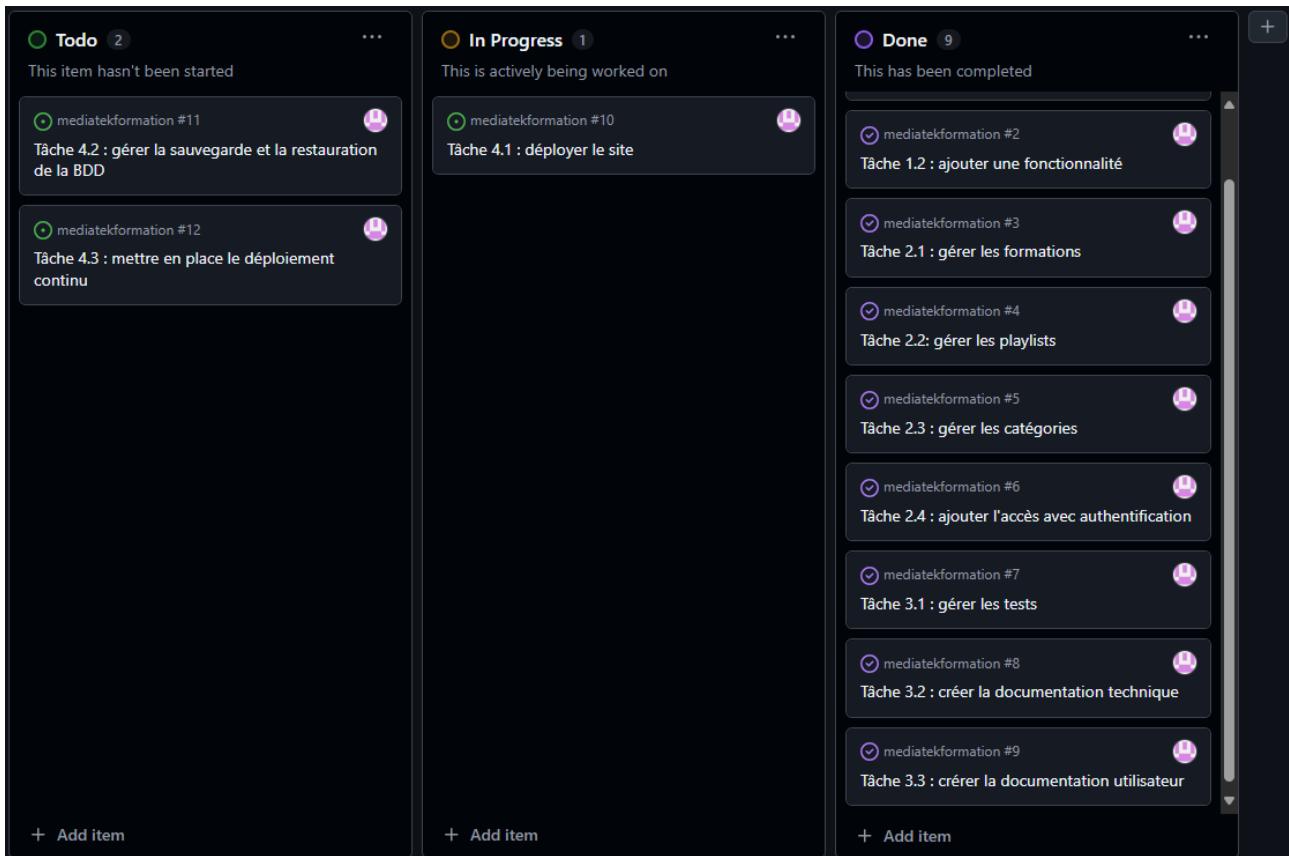
Temps réel : 30min



Mission 4 :

Tâche 4.1 : déployer le site

- Déployer le site chez un hébergeur ainsi que la documentation technique.



Temps estimé : 2h

Temps réel : 2h30min

Pour la première phase du déploiement du site, il a été nécessaire de sélectionner un hébergeur. J'ai opté pour InfinityFree, qui offre toutes les fonctionnalités requises pour assurer la compatibilité avec le site, notamment une version PHP adaptée et la prise en charge de MySQL.

Dans un second temps, j'ai configuré la base de données via phpMyAdmin (proposé par l'hébergeur) en exécutant un fichier SQL pré-rempli contenant un ensemble de données.

Ensuite, le transfert des fichiers du site a été réalisé à l'aide du logiciel FileZilla, en utilisant les informations FTP fournies par l'hébergeur. Pour accélérer le processus, les fichiers ont été compressés en archive ZIP avant leur envoi.

Enfin, il a suffi de modifier le fichier .htaccess afin d'assurer la redirection vers le dossier "public".

Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD

- Automatiser une tâche de sauvegarde quotidienne de la BDD.

The image shows a digital task board with three columns: Todo, In Progress, and Done. Each column has a title and a brief description below it. A plus sign icon is located at the bottom right of each column.

Todo	In Progress	Done
This item hasn't been started	This is actively being worked on	This has been completed
<ul style="list-style-type: none">meditekformation #12 Tâche 4.3 : mettre en place le déploiement continu	<ul style="list-style-type: none">meditekformation #11 Tâche 4.2 : gérer la sauvegarde et la restauration de la BDD	<ul style="list-style-type: none">meditekformation #3 Tâche 2.1 : gérer les formationsmeditekformation #4 Tâche 2.2 : gérer les playlistsmeditekformation #5 Tâche 2.3 : gérer les catégoriesmeditekformation #6 Tâche 2.4 : ajouter l'accès avec authentificationmeditekformation #7 Tâche 3.1 : gérer les testsmeditekformation #8 Tâche 3.2 : créer la documentation techniquemeditekformation #9 Tâche 3.3 : créer la documentation utilisateurmeditekformation #10 Tâche 4.1 : déployer le site
+ Add item	+ Add item	+ Add item

Temps estimé : 1h

Temps réel : 50min

Pour gérer la sauvegarde journalière, il faut commencer par créer un script qui permettra d'enregistrer le script de la bdd dans un fichier (qui dans son nom contient la date et l'extension “.sql.gz”) :

```
#!/bin/sh
DATE=`date -I`
find /htdocs/if0_38521742/savebdd/bdd* -mtime -1 -exec rm {} \;
mysqldump -u if0_38521742 -p[REDACTED] --databases if0_38521742_mEDIATEkformation --single-transaction |
gzip > /home/if0_38521742/savebdd/bddbackup_${DATE}.sql.gz
```

La première ligne enregistre la date dans une variable (qui servira à nommer le fichier).

La deuxième ligne supprime l'ancien fichier du script de la bdd.

Et les deux dernières lignes permettent de récupérer le script sql de la bdd et l'enregistrer dans un fichier zippé portant le nom de “bddbackup_” suivi de la date et l'extension “.sql.gz” dans le “savebdd”.

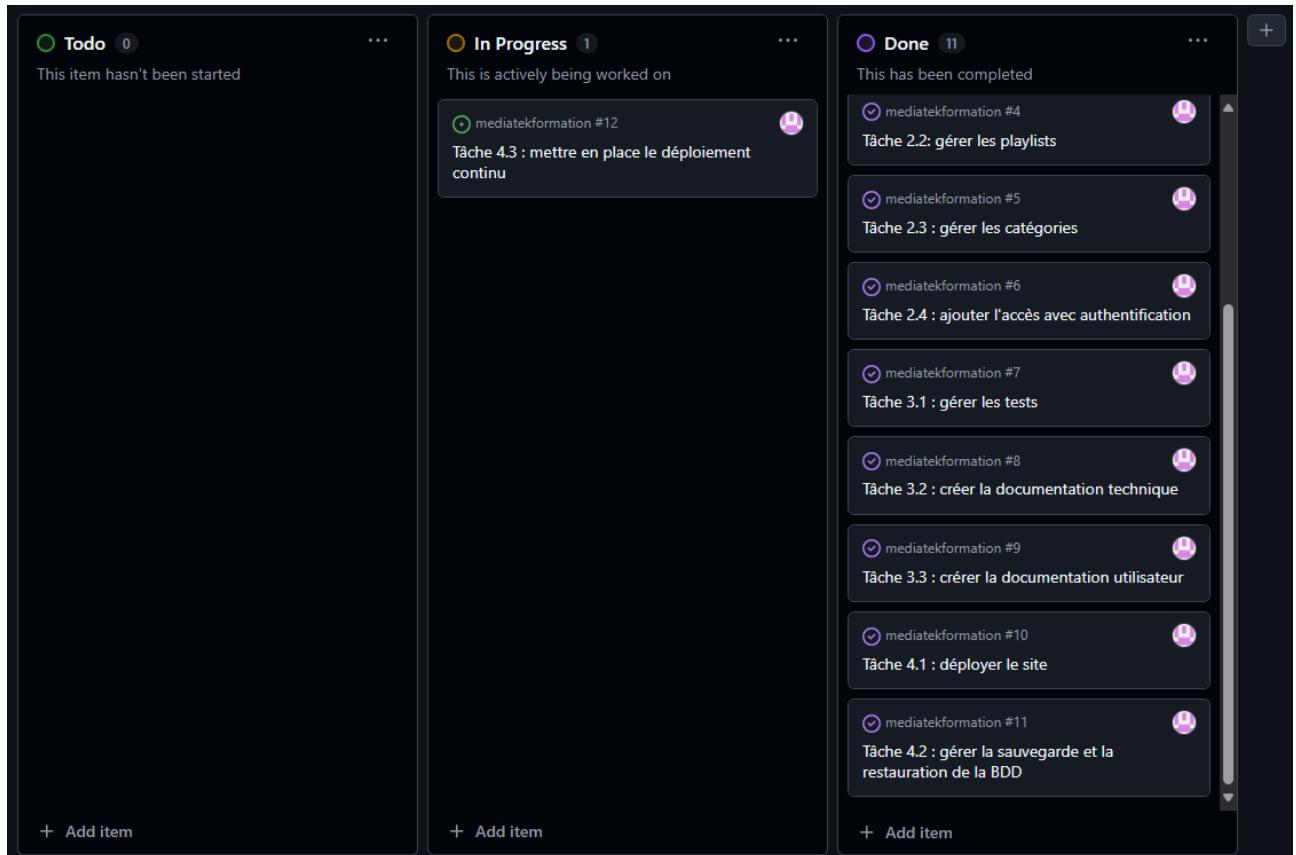
Après cela il faut mettre le fichier au format linux (grâce au logiciel dos2linux et le transférer chez l'hébergeur avec FileZilla.

Enfin il ne reste plus qu'à créer une tâche Cron automatisé pour exécuter le script en choisissant une fréquence journalière, cependant il n'est plus possible d'utiliser des tâches cron sur les comptes d'hébergement gratuit d'InfinityFree (cette fonctionnalité a été désactivée en août 2023 en raison d'une utilisation excessive qui affectait les performances des serveurs).

Pour restaurer la base de données en cas de perte ou de corruption des données, il suffit de récupérer le fichier bddbackup_XXXX-XX-XX.sql.gz dans le dossier savebdd, de le décompresser, de vider la base de données, puis d'exécuter le script SQL.

Tâche 4.3 : mettre en place le déploiement continu

- Créer un script dans GitHub pour gérer le déploiement continu.



Temps estimé : 1h

Temps réel : 30 min

Pour que le site en ligne soit mis à jour à chaque push reçu dans le dépôt il faut paramétrer GitHub pour lui dire vers quel ftp envoyer les fichiers. Pour ce cela, on créer un fichier yml d'automatisation dans GitHub :

```
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        - name: Sync files
11          uses: SamKirkland/FTP-Deploy-Action@4.3.0
12          with:
13            server: ftpupload.net
14            server-dir: /htdocs/
15            username: if0_38521742
16            password: ${{ secrets.ftp_password }}
```

Dans ce script est renseigné l'adresse ftp du site, ainsi que le dossier où il se trouve et le login de connexion ftp.

On commit les changements et on pull dans Netbeans afin de récupérer le dossier ".github/workflows" qui contient le fichier yml. Il faut aussi renseigner et enregistrer le password d'accès au ftp dans GitHub (Settings > Secrets and variables > Actions > New repository secret).

Après cela, un commit puis un push va automatiquement mettre à jour les fichiers sur le ftp, ce qui permet de mettre à jour le site en ligne.

Bilan final :

Le projet a bien progressé avec plusieurs améliorations et fonctionnalités mises en place :

- **Code amélioré** : Le code a été nettoyé afin d'assurer une meilleure lisibilité et maintenabilité.
- **Back-office opérationnel** : Un système de gestion des formations, playlists et catégories a été développé, incluant un mécanisme d'authentification pour sécuriser l'accès.
- **Tests et validation** :
 - Des **tests unitaires** ont été implémentés pour vérifier l'efficacité des méthodes de classe.
 - Des **tests d'intégration** ont été réalisés sur les repositories.
 - Des **tests fonctionnels** ont été menés sur l'accès à l'accueil, les tris et les filtres.
- **Documentation technique** : Une documentation complète a été générée pour faciliter la compréhension et la maintenance du projet.
- **Déploiement** : Le site a été déployé avec succès et un pipeline de déploiement continu via GitHub a été mis en place.

Problèmes rencontrés :

Malheureusement, la gestion des sauvegardes n'a pas pu être mise en œuvre en raison des limitations de l'hébergeur InfinityFree, qui ne permet pas l'utilisation de tâches Cron.

Globalement, le projet est fonctionnel et bien structuré, malgré cette contrainte technique qui pourrait nécessiter un changement d'hébergement pour être résolue.