

BTS SIO

Bilan atelier de professionnalisation n°1

Mediatekformation

Table des matières

Mission 1 : nettoyer et optimiser le code existant.....	4
Tâche 1: Nettoyer le code.....	4
Tâche 2 : ajouter une fonctionnalité.....	9
Mission 2 : coder la partie back-office.....	11
Tâche 1 : Gérer les formations.....	11
Tâche 2 : gérer les playlists.....	15
Tâche 3: Gérer les catégories.....	18
Tâche 4: Ajouter l'accès avec authentification.....	20
Mission 3 : tester et documenter.....	22
Tâche 1 : Gérer les tests.....	22
Plan de tests.....	23
Tests unitaires.....	23
Tests d'intégration.....	23
Tests fonctionnels.....	23
Tâche 2 : créer la documentation technique.....	24
Tâche 3 : créer la documentation utilisateur.....	26
Mission 4 : déployer le site et gérer le déploiement continu.....	27
Tâche 1:déployer le site.....	27
Tâche 2: Gérer la sauvegarde et la restauration de la base de données.....	28
Tâche 3 : mettre en place le déploiement continu.....	29

Introduction

Contexte :

« Vous travaillez en tant que technicien développeur junior pour l'ESN InfoTech Services 86 qui vient de remporter le marché pour différentes interventions au sein du réseau MediaTek86, dont certaines dans le domaine du développement d'application. Un contrat de développement a ainsi été rédigé entre MediaTek86 et InfoTech afin de réaliser 3 applications. »

Existant :

La partie front office a déjà été codées (pages Accueil, Formations et Playlists).

Langage et technologies utilisées :

Le site est codé en local sous Netbeans avec Wamp server. Le framework utilisé est Symfony 5.4 avec composer

Mission 1 : nettoyer et optimiser le code existant

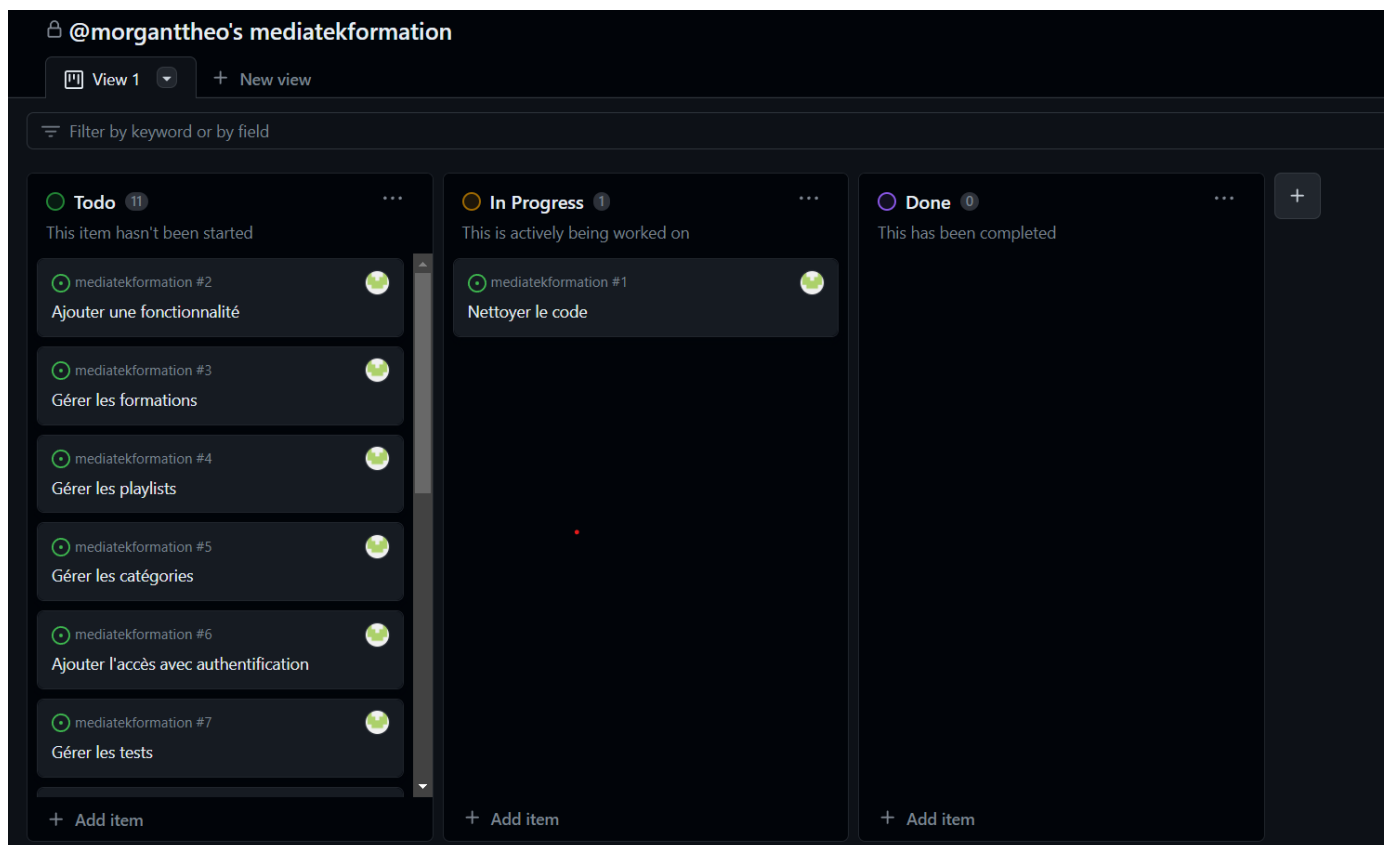
Tâche 1: Nettoyer le code

Tâche : Nettoyer le code en suivant les indications de SonarLint (excepté le code généré automatiquement par Symfony).

Temps estimé : 2h

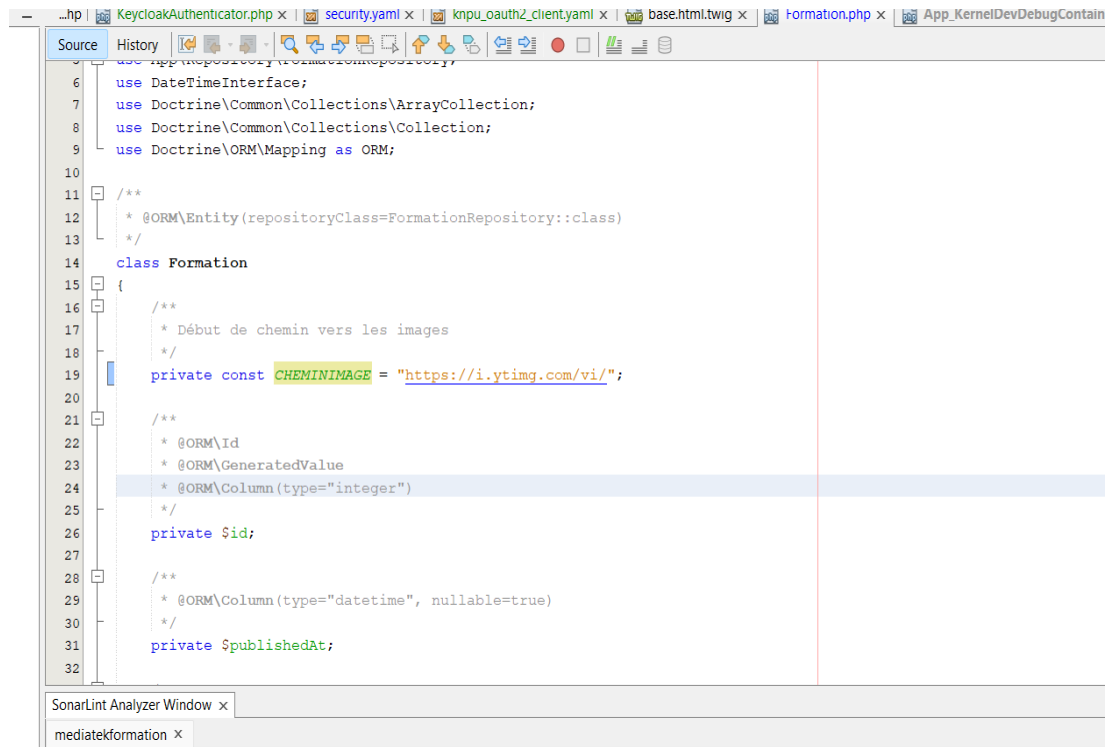
Temps réel : 2h30

Kanban



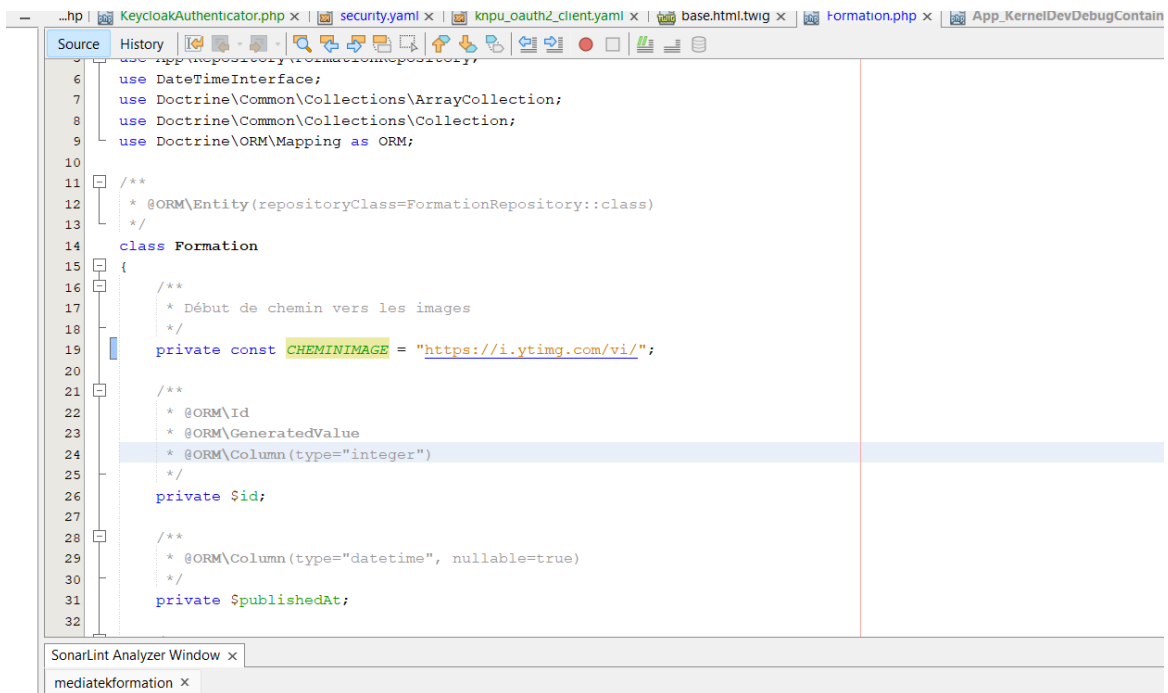
Problème de constante :

Avant correction



```
5 use App\Repository\FormationRepository;
6 use DateTimeInterface;
7 use Doctrine\Common\Collections\ArrayCollection;
8 use Doctrine\Common\Collections\Collection;
9 use Doctrine\ORM\Mapping as ORM;
10
11 /**
12  * @ORM\Entity(repositoryClass=FormationRepository::class)
13  */
14 class Formation
15 {
16     /**
17      * Début de chemin vers les images
18      */
19     private const CHEMINIMAGE = "https://i.ytimg.com/vi/";
20
21     /**
22      * @ORM\Id
23      * @ORM\GeneratedValue
24      * @ORM\Column(type="integer")
25      */
26     private $id;
27
28     /**
29      * @ORM\Column(type="datetime", nullable=true)
30      */
31     private $publishedAt;
```

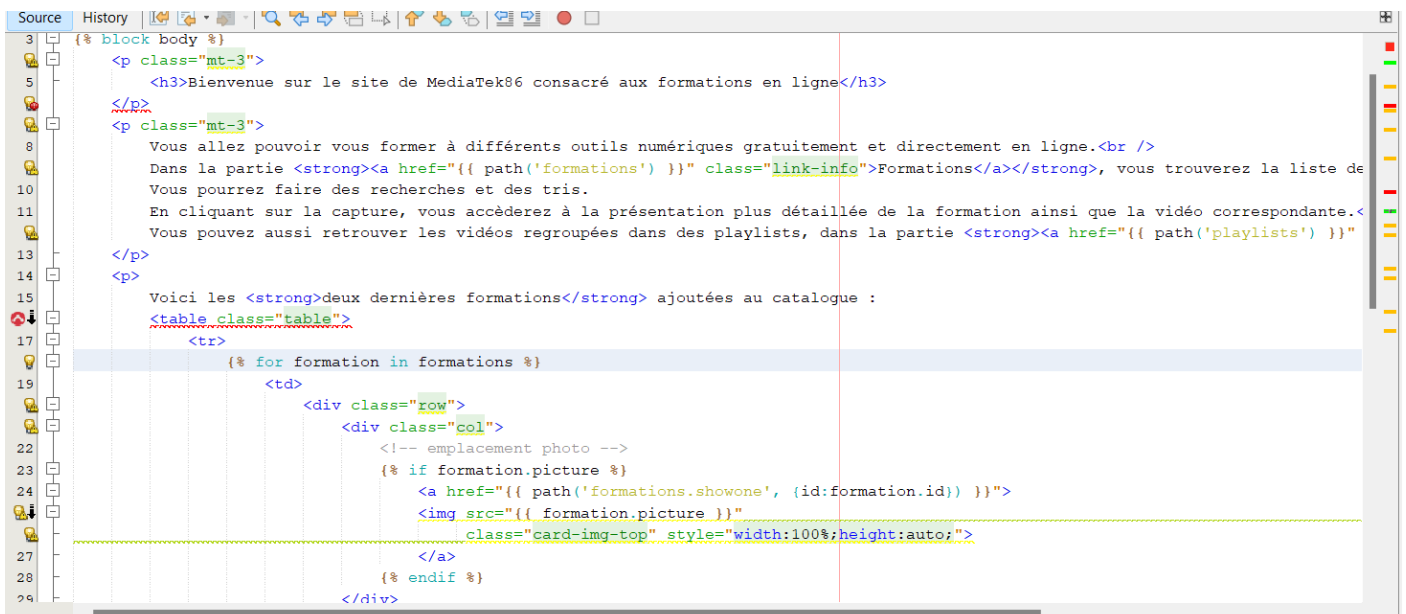
Après correction



```
5 use App\Repository\FormationRepository;
6 use DateTimeInterface;
7 use Doctrine\Common\Collections\ArrayCollection;
8 use Doctrine\Common\Collections\Collection;
9 use Doctrine\ORM\Mapping as ORM;
10
11 /**
12  * @ORM\Entity(repositoryClass=FormationRepository::class)
13  */
14 class Formation
15 {
16     /**
17      * Début de chemin vers les images
18      */
19     private const CHEMINIMAGE = "https://i.ytimg.com/vi/";
20
21     /**
22      * @ORM\Id
23      * @ORM\GeneratedValue
24      * @ORM\Column(type="integer")
25      */
26     private $id;
27
28     /**
29      * @ORM\Column(type="datetime", nullable=true)
30      */
31     private $publishedAt;
32 }
```

Problème de balise :

Avant correction



```
3 {% block body %}
4 <p class="mt-3">
5     <h3>Bienvenue sur le site de MediaTek86 consacré aux formations en ligne</h3>
6 </p>
7 <p class="mt-3">
8     Vous allez pouvoir vous former à différents outils numériques gratuitement et directement en ligne.<br />
9     Dans la partie <strong><a href="{{ path('formations') }}" class="link-info">Formations</a></strong>, vous trouverez la liste de
10     Vous pourrez faire des recherches et des tris.
11     En cliquant sur la capture, vous accéderez à la présentation plus détaillée de la formation ainsi que la vidéo correspondante.<
12     Vous pouvez aussi retrouver les vidéos regroupées dans des playlists, dans la partie <strong><a href="{{ path('playlists') }}"
13 </p>
14 <p>
15     Voici les <strong>deux dernières formations</strong> ajoutées au catalogue :
16     <table class="table">
17         <tr>
18             {% for formation in formations %}
19                 <td>
20                     <div class="row">
21                         <div class="col">
22                             <!-- emplacement photo -->
23                             {% if formation.picture %}
24                                 <a href="{{ path('formations.showone', {id:formation.id}) }}">
25                                     
27                                 </a>
28                             {% endif %}
29                         </div>
30                     </div>
31                 </td>
32             {% endfor %}
33         </tr>
34     </table>
35 </p>
36 </block body %}
```

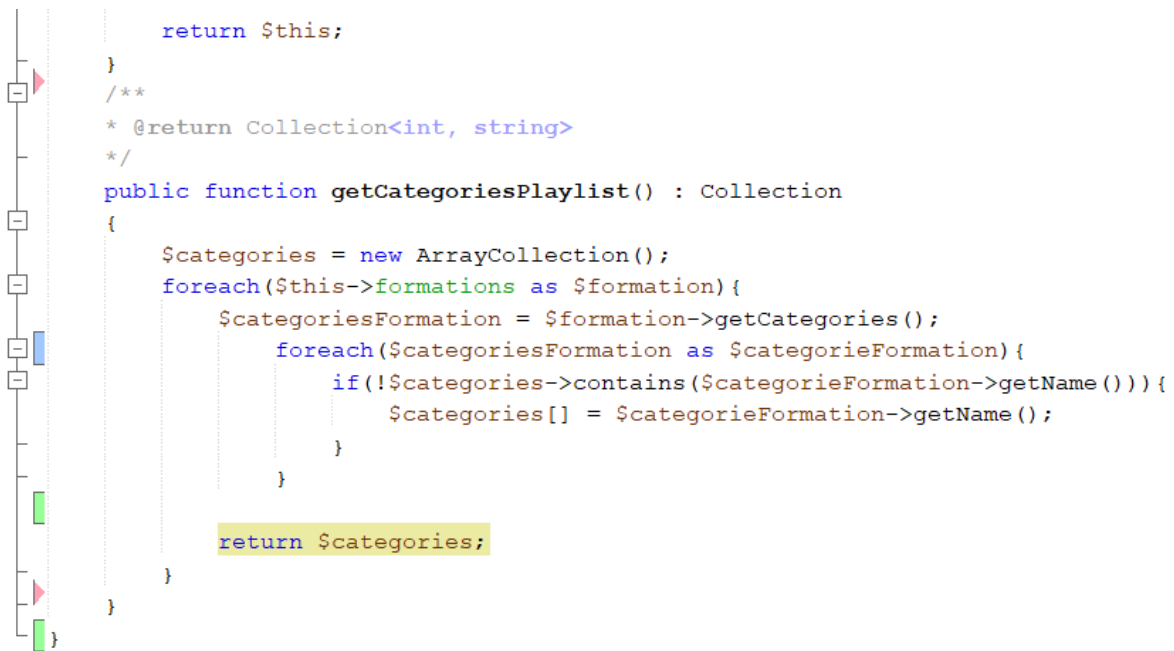
Après correction

```

{% block body %}
<p class="mt-3">
  <h3>Bienvenue sur le site de MediaTek86 consacré aux formations en ligne</h3>
</p>
<p class="mt-3">
  Vous allez pouvoir vous former à différents outils numériques gratuitement et directement en ligne.<br />
  Dans la partie <strong><a href="{{ path('formations') }}" class="link-info">Formations</a></strong>, vous trouverez la liste de
  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 <strong><a href="{{ path('playlists') }}"
</p>
<p>
  Voici les <strong>deux dernières formations</strong> ajoutées au catalogue :
</p>
<table class="table">
  <tr>
    {% for formation in formations %}
      <td>
        <div class="row">
          <div class="col">
            <!-- emplacement photo -->
            {% if formation.picture %}
              <a href="{{ path('formations.showone', {id:formation.id}) }}">
                
 */
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;
}

```

Après correction



```

        return $this;
    }
    /**
     * @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;
    }
}

```

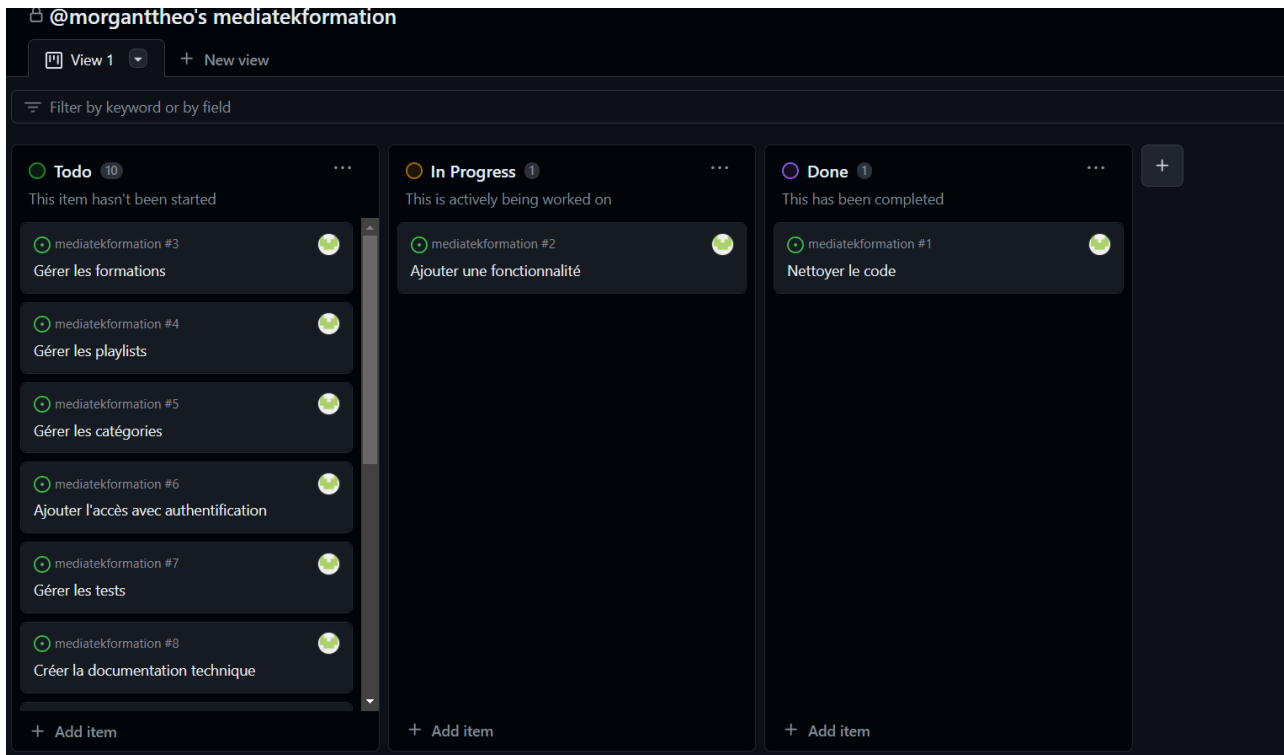
Tâche 2 : ajouter une fonctionnalité

Tâche : 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'ajouter sur la page d'une playlist

Temps estimé : 2h

Temps réel : 3 h

Kanban



Ajout d'une colonne

playlist	Nombre de formations	catégories	
Bases de la programmation (C#)	74	C# POO	Voir détail
Programmation sous Python	19	Python	Voir détail
MCD : exercices progressifs	18	MCD	Voir détail
TP Android (programmation mobile)	18	Android	Voir détail
Compléments Android (programmation mobile)	13	Android	Voir détail
Visual Studio 2019 et C#	11	C#	Voir détail
Cours UML	10	UML	Voir détail
Eclipse et Java	8	Java	Voir détail

Modification du code : Dans la table Playlist j'ai ajouté un champ pour compter le nombre de formations par playlist et je l'ai ensuite intégré dans la page « Playlist ». J'ai ensuite créé une fonction dans le repository pour récupérer et trier par ordre (croissant ou décroissant selon le paramètre), cette fonction est appelée dans le controller que la vue renvoie avec les boutons lien « < » et « > ».

Controller :

```

}
/**
 * @Route("/playlists/trier/{champ}/{ordre}", name="playlists.count")
 * @param type $champ
 * @param type $ordre
 * @return Response
 */
public function count($champ, $ordre): Response{
    switch($champ){
        case "nb_formation":
            $playlists = $this->playlistRepository->findAllOrderByNumber($ordre);
            break;
        default:
            echo 'le champ ne contient pas de vidéo';
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/playlists.html.twig", [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}

```

Repository :

```

1
2 /**
3  * Retourne toutes les playlists triées sur le nombre de formation
4  * @param type $champ
5  * @param type $ordre
6  * @return Playlist[]
7  */
8 public function findAllOrderByNumber($ordre): array{
9     return $this->createQueryBuilder('p')
10         ->leftJoin('p.formations', 'f')
11         ->groupBy('p.id')
12         ->orderBy('p.nb_formation', $ordre)
13         ->getQuery()
14         ->getResult();
15 }
16
17 /**
18  * Enregistrements dont un champ contient une valeur
19

```

Twig(vue) :

```

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

```

Mission 2 : coder la partie back-office

Tâche 1 : Gérer les formations

Temps estimé : 5 h

Temps réel : 6 h

A réaliser :

- 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.
- Les mêmes tris et filtres présents dans le front office doivent être présents dans le back office.
- Si une formation est supprimée, il faut aussi l'enlever de la playlist où elle se trouvait.
- 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.

Kanban

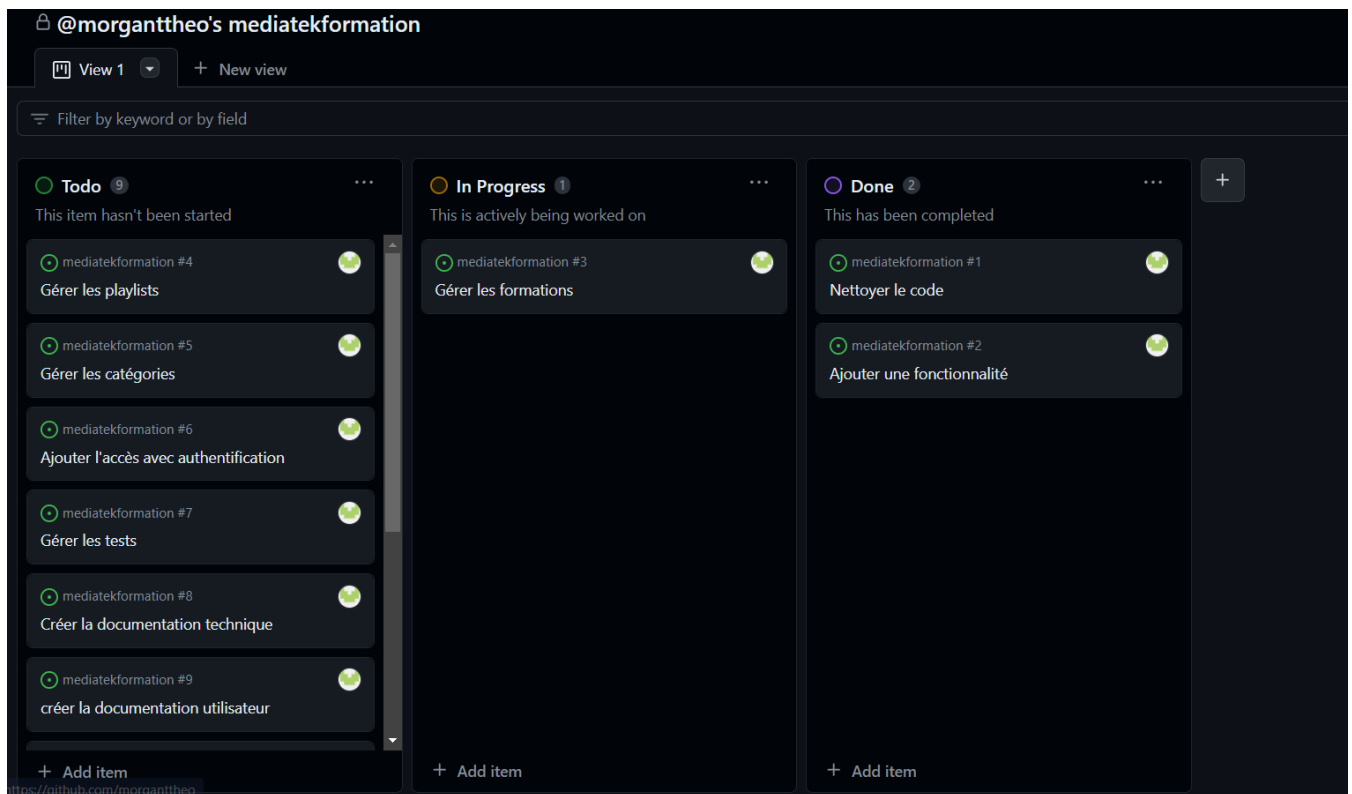
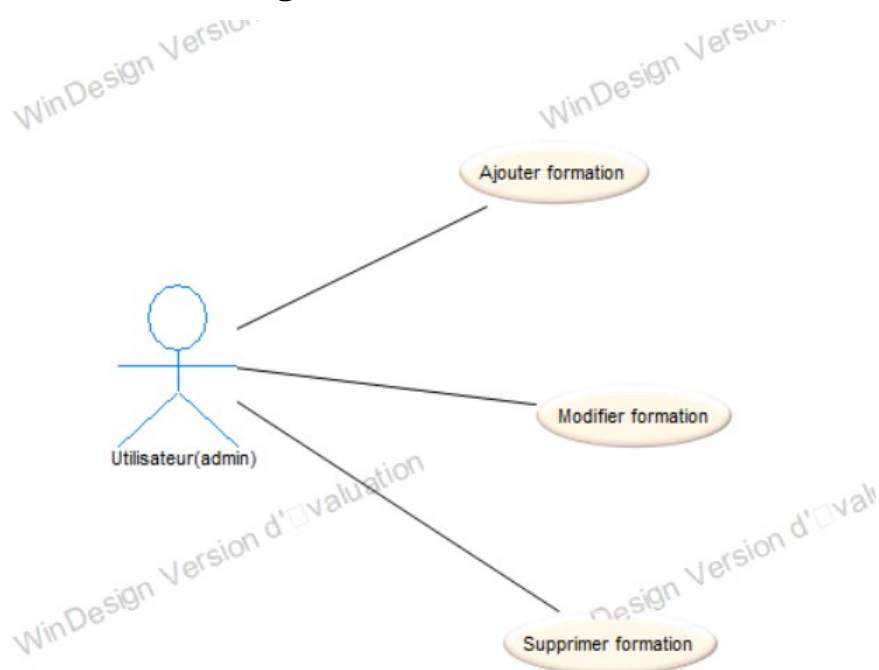


Diagramme de cas d'utilisation



Page « formations » coté admin avec l'ajout d'un bouton supprimer demandant une confirmation avant la suppression d'une formation. Ajout d'un bouton modifier envoyant un formulaire avec les données de la formation que l'on peut modifier et envoyer la base de données. Enfin un bouton ajouter contenant le même formulaire mais cette fois-ci vide qu'il faut compléter avec certaines contraintes.

← → ↺

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

MediaTek86

Des formations pour tous sur des outils numériques

Formations Playlists

Ajouter une formation

formation

< >

filtrer

playlist

< >

filtrer




catégories

▼

date

< >

Actions

Eclipse n°700 : Tests unitaires	Eclipse et Java Eclipse et Java	UML	02/01/2021		<div>Modifier</div> <div>Supprimer</div>
Eclipse n°6 : Documentation technique	Eclipse et Java Eclipse et Java	Java	30/12/2020		<div>Modifier</div> <div>Supprimer</div>
Eclipse n°5 : Refactoring	Eclipse et Java Eclipse et Java	Java	29/12/2020		<div>Modifier</div> <div>Supprimer</div>

Page « modifier »

Formations Playlists

Modifier une formation

Date

Dec ▼ 12 ▼ 202 ▼

07 ▼ : 49 ▼

Titre

Eclipse n°700 : Tests unitaires

Description

Intégration de sonit dans l'application et création de tests unitaires.
00:07 : rappel sur le principe du test unitaire
01:01 : intégrer JUnit au projet (une seule fois)

Video id

-nw42Xq6cYE

Playlist

Eclipse et Java ▼

Catégories

Java
UML
C#
Python

Submit

Page « Ajout »

Formations Playlists

Ajouter une formation

Date

Dec ▾

12 ▾

202 ▾

07 ▾

:

50 ▾

Titre

Description

Video id

Playlist

Eclipse et Java ▾

Catégories

Java

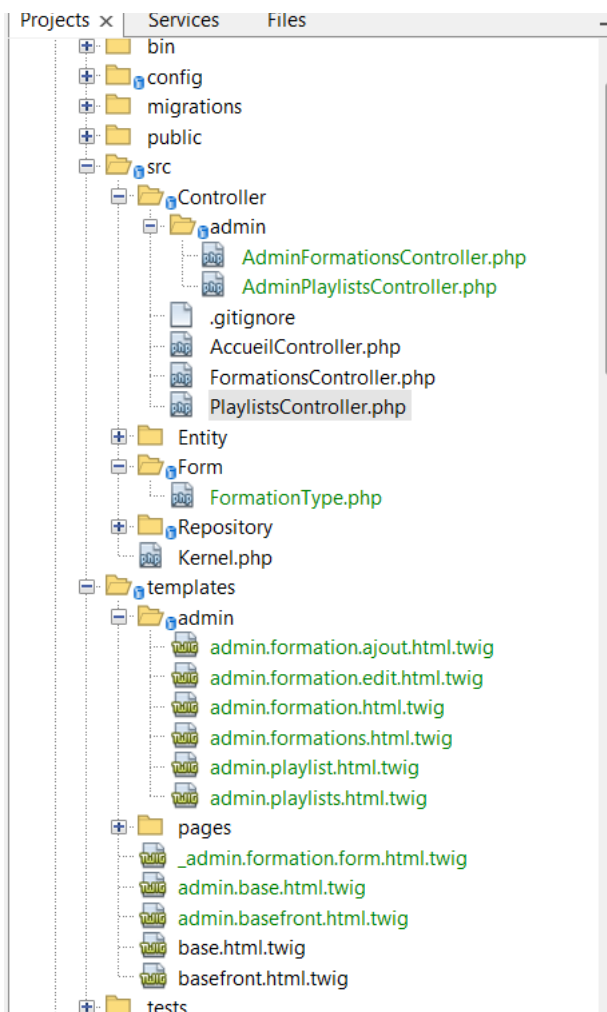
UML

C#

Python

Submit

Arborescence des fichiers



Ajout des dossier « AdminFormationsController » et « AdminPlaylistsController »

Ajout des pages de la vue coté admin renvoyées par le controller AdminFormations

Ajout du fichier « FormationType » permettant de créer le formulaire avec Buildform.

Tache 2 : gérer les playlists

Temps estimé : 5 h

Temps réel : 4h

A réaliser :

- La suppression d'une playlist n'est possible que si aucune formation n'est rattachée à elle.
- Une page doit permettre de lister les playlists et, pour chaque playlist, afficher un bouton permettant de la supprimer (après confirmation) et un bouton permettant de la modifier.
- 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 playlist. Les saisies doivent être contrôlées. L'ajout d'une playlist consiste juste à saisir son nom et sa description. Seul le champ name est obligatoire.
- Le clic sur le bouton permettant de modifier une playlist doit amener sur le même formulaire, mais cette fois prérempli. Cette fois, la liste des formations de la playlist doit apparaître, mais il ne doit pas être possible d'ajouter ou de supprimer une formation : ce n'est que dans le formulaire de la formation qu'il est possible de préciser sa playlist de rattachement.

Kanban

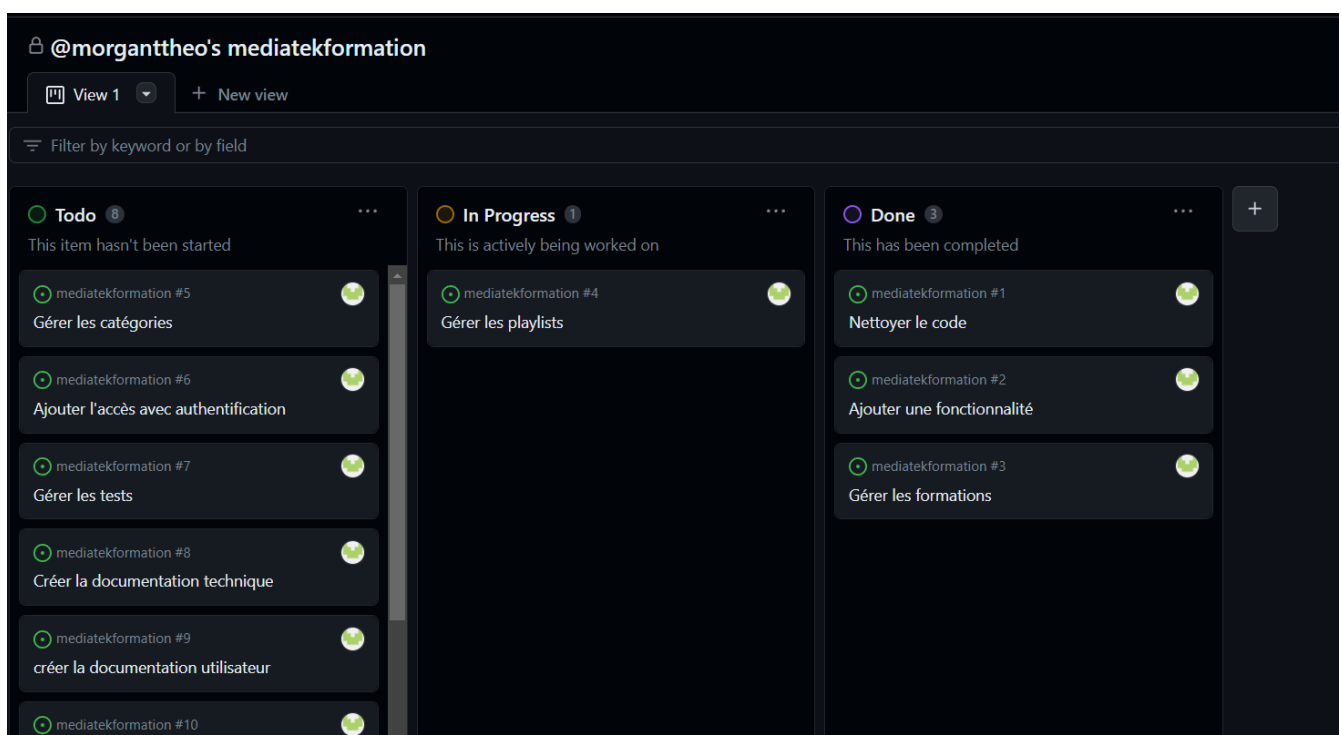
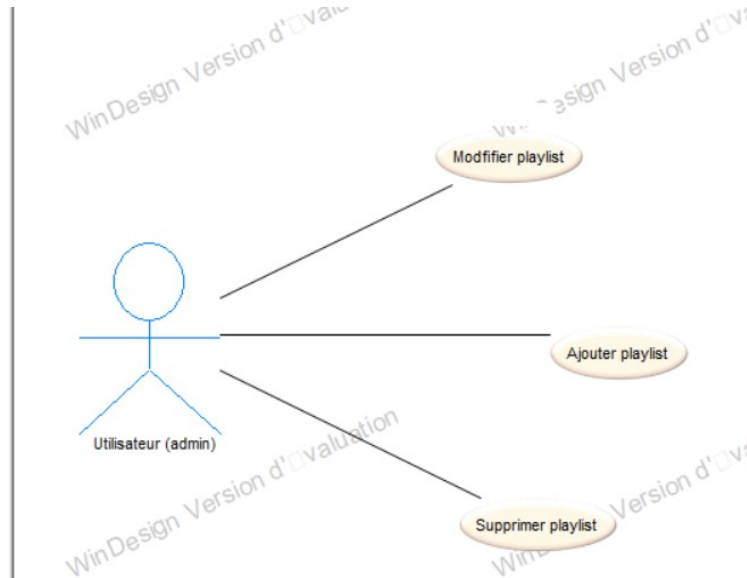


Diagramme de cas d'utilisation



Page «playlists» coté admin avec l'ajout d'un bouton supprimer demandant une confirmation avant la suppression d'une playlist. Ajout d'un bouton modifier envoyant un formulaire avec les données de la playlist que l'on peut modifier et envoyer la base de données. Enfin un bouton ajouter contenant le même formulaire mais cette fois-ci vide qu'il faut compléter.

			Ajouter une playlist	
playlist <input type="text"/> <input type="button" value="filtrer"/>	Nombre de formations <input type="button" value="←"/> <input type="button" value="→"/>	catégories <input type="text"/>		
			Actions	
Bases de la programmation (C#)	73	C# POO	Voir détail	Modifier
Compléments Android (programmation mobile)	13	Android	Voir détail	Modifier
Cours de programmation objet	0		Voir détail	Modifier Supprimer
Cours Composant logiciel	2	Cours	Voir détail	Modifier

Page « modifier »

Des formations pour tous sur des outils numériques

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

Modifier une playlist

Name

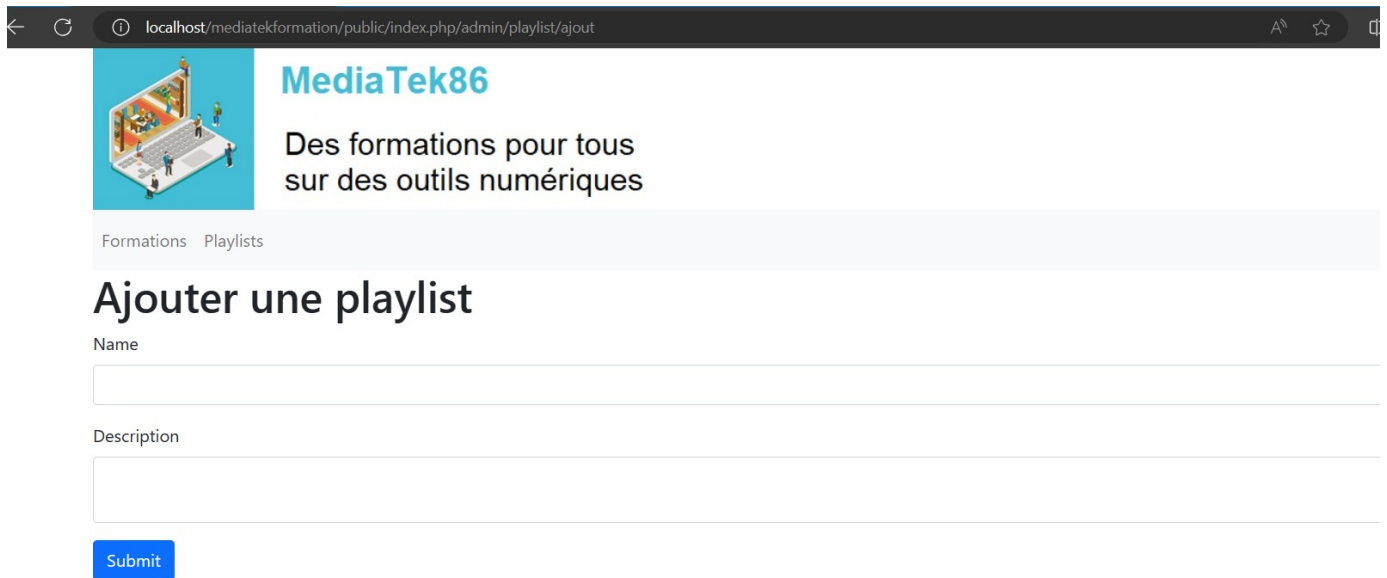
Description

Chaque vidéo est indépendante et présente une notion spécifique.
 Prérequis : avoir des notions de base en programmation sous Android et en programmation objet (si vous ne connaissez pas du tout Android, commencez par suivre la playlist)

Android Studio (complément n°1) : Navigation Drawer et Fragment

Android Studio (complément n°2) : Récupérer les contacts du mobile

Page « Ajouter »



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

MediaTek86

Des formations pour tous sur des outils numériques

Formations Playlists

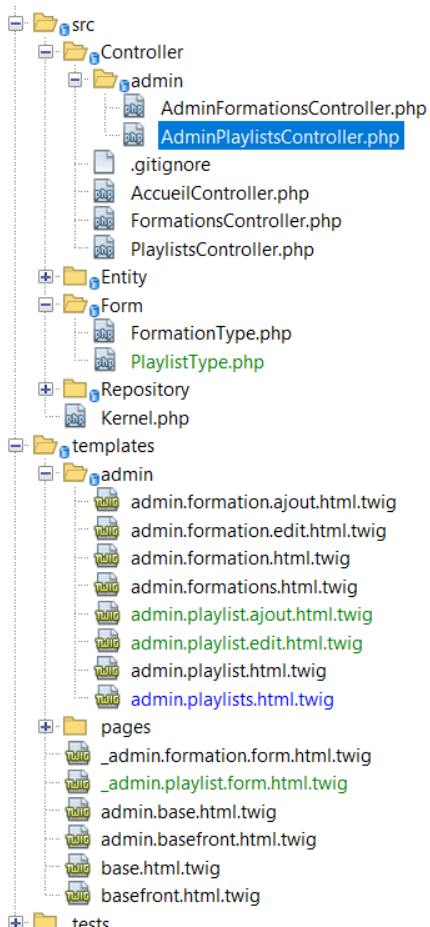
Ajouter une playlist

Name

Description

Submit

Arborescence des fichiers



Ajout des pages de la vue coté admin
(admin.playlist...) renvoyées par le controller
AdminPlaylists

Ajout du fichier « PlaylistType » permettant de créer
le formulaire avec Buildform.

Tâche 3: Gérer les catégories

Temps estimé : 3h

Temps réel : 3h30

A réaliser :

- Dans la même page, un mini formulaire doit permettre de saisir et d'ajouter directement une nouvelle catégorie, à condition que le nom de la catégorie n'existe pas déjà
- Une page doit permettre de lister les catégories et, pour chaque catégorie, afficher un bouton permettant de la supprimer. Attention, une catégorie ne peut être supprimée que si elle n'est rattachée à aucune formation.

Kanban

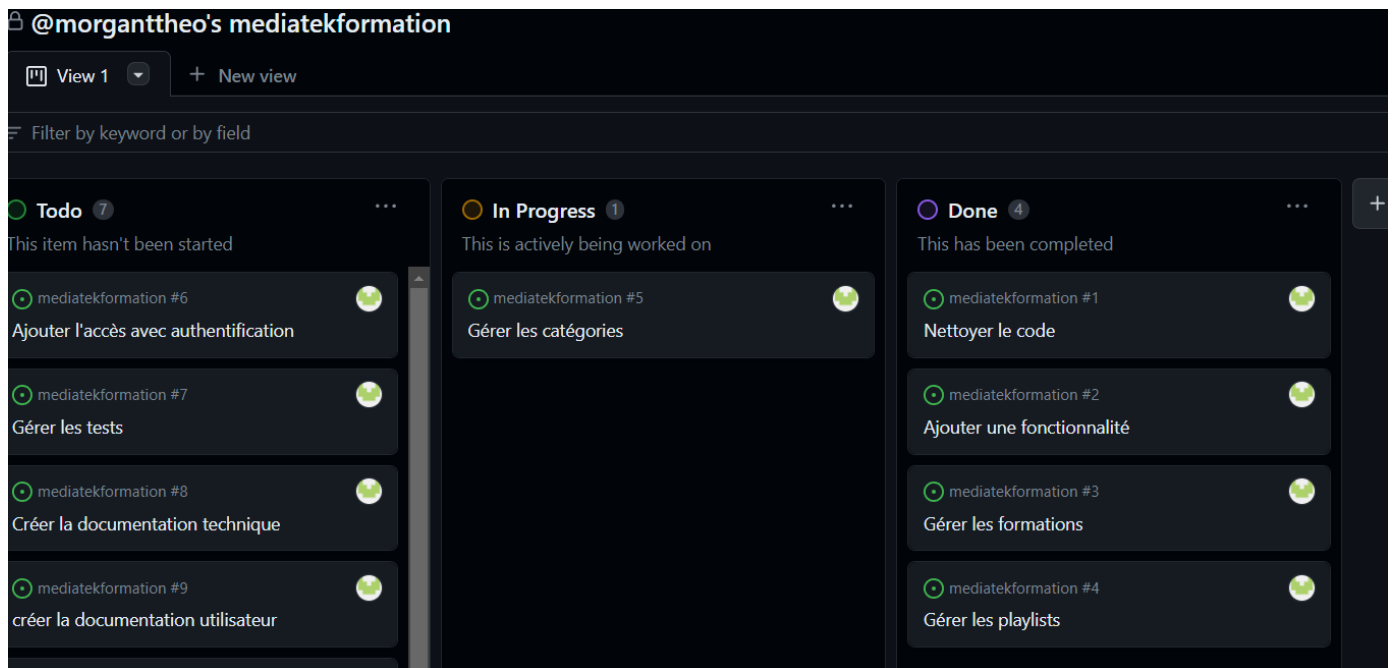
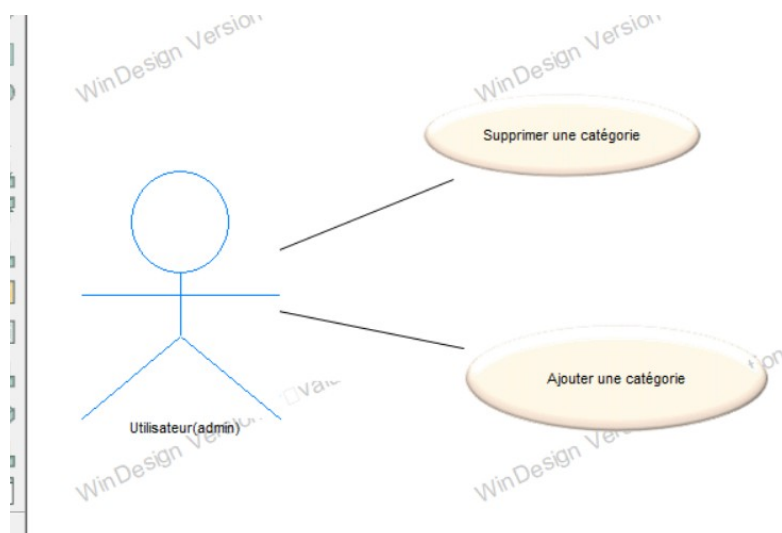
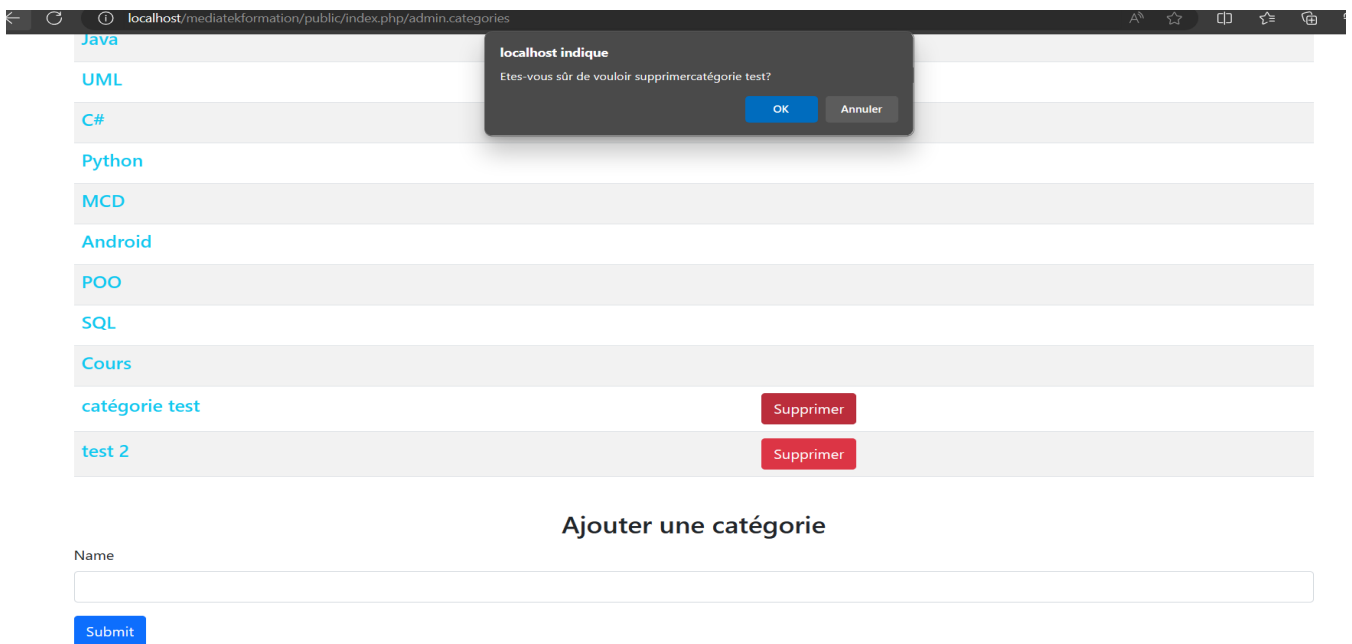


Diagramme de cas d'utilisation

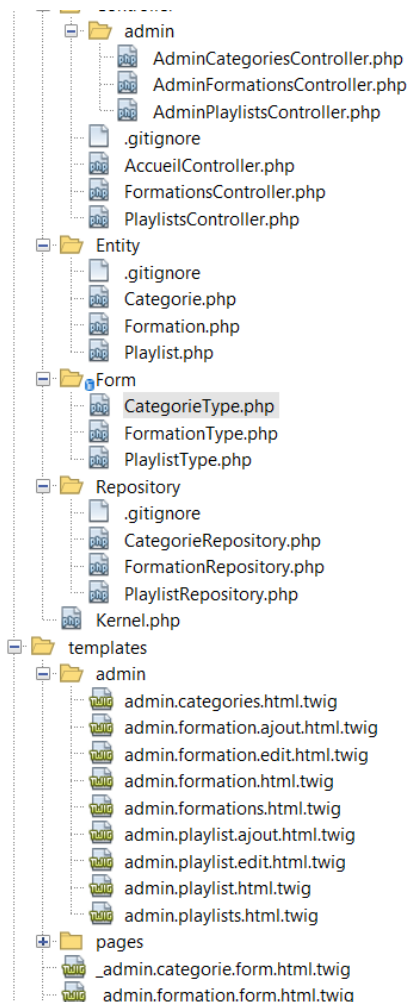


Page catégorie



Ajout d'une page avec la liste des catégories non modifiables. Possibilité de supprimer une catégorie n'ayant aucun formation reliée. Formulaire d'ajout de formation.

Arborescence



Ajout du fichier AdminCategoriesController avec la fonction index et une fonction « supprimer »

Ajout d'un formulaire (CategorieType) avec comme champ un nom et un bouton valider.

Ajout d'une page admin.categories.html pour afficher la table des catégories et le formulaire

Ajout de la page _admin.categorie.form avec le formulaire qui est appelé dans la page admin.categories.html

Tâche 4: Ajouter l'accès avec authentification

Temps estimé : 4h

Temps réel : 4h

A réaliser :

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

Kanban

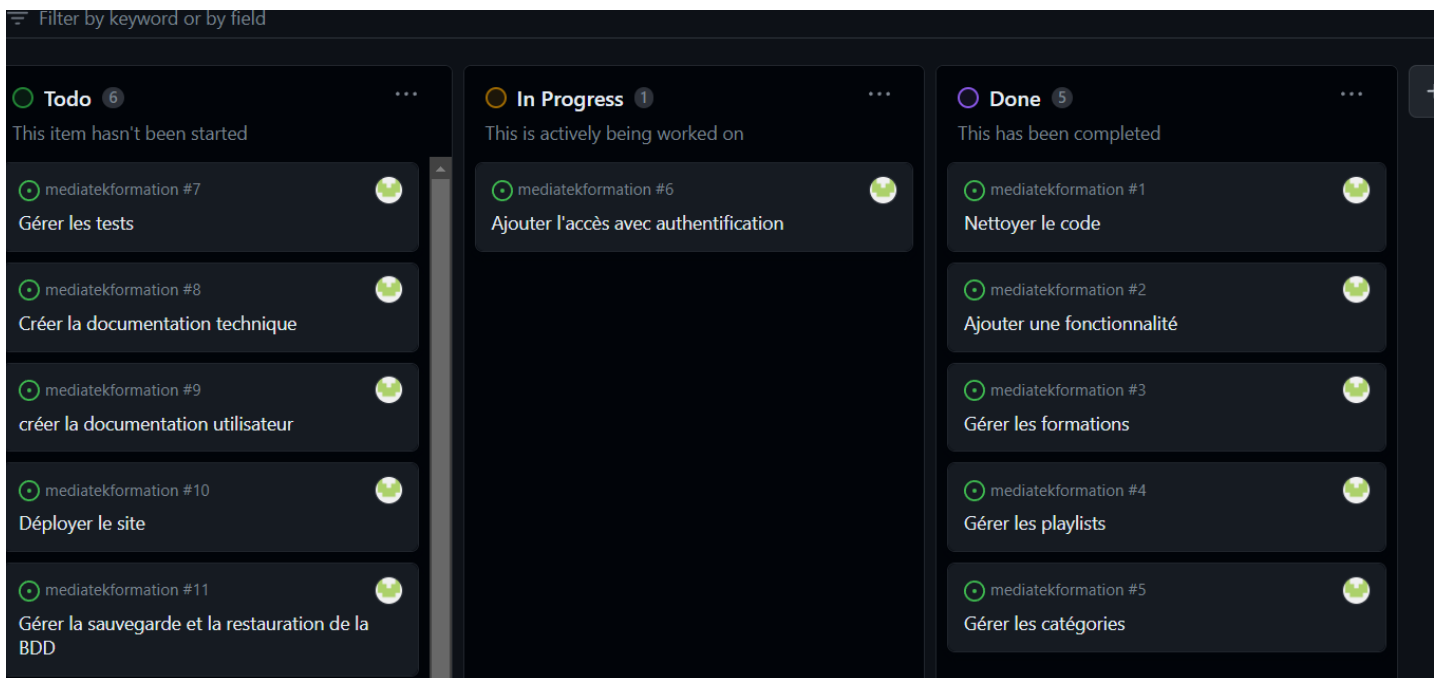
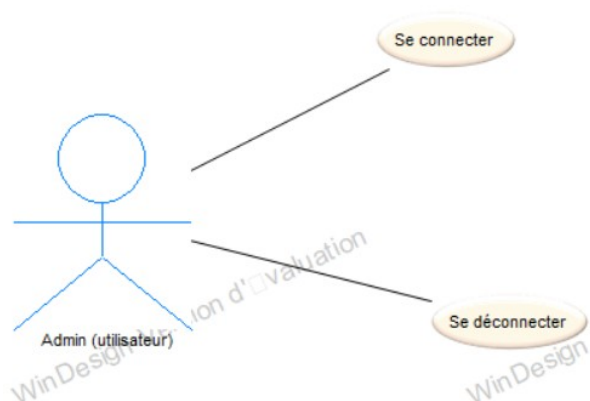


Diagramme de cas d'utilisation



Page de connection

localhost:8080/realms/myapplis/protocol/openid-connect/auth?state=4447aaf13ba542df1b90ce05037f0dc4&scope=profile%20email&response_type=code&approv...

MYAPPLIS

Sign in to your account

Username or email

Password

[Forgot Password?](#)

Sign In

Lien de déconnection

← ↻ ⓘ localhost/mediatekformation/public/index.php/admin



MediaTek86

Des formations pour tous
sur des outils numériques

[se déconnecter](#)

Mission 3 : tester et documenter

Tâche 1 : Gérer les tests

Temps estimé : 7h

Temps réel : 7h

A réaliser :

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

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

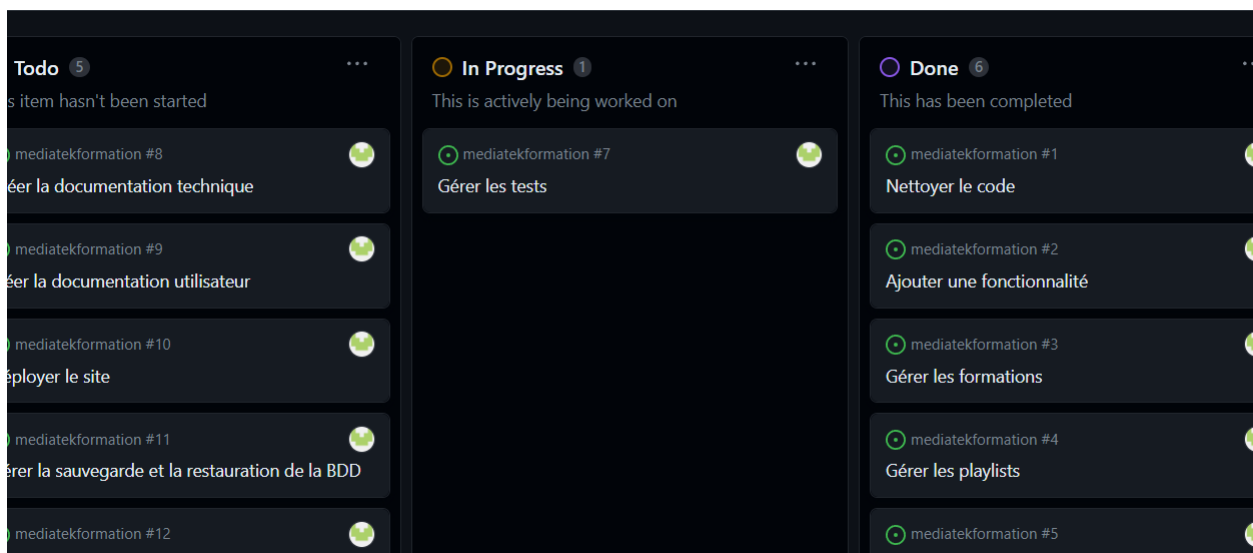
Contrôler toutes les méthodes ajoutées dans les classes Repository (pour cela, créer une BDD de test).

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).

Kanban



Plan de tests

Tests unitaires

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la méthode <code>getPublishedAtString()</code> de la classe <code>Formation</code> pour voir si elle retourne la bonne date au bon format.	Test unitaire lancé avec la date : 2022-01-04 17:00:12	04/01/2022	OK

Tests d'intégration

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler qu'une date postérieure à aujourd'hui ne puisse pas être ajoutée	Premier teste lancé avec date =« 2023-05-10 17:00:15 »	Date inférieur à aujourd'hui = 0 erreur	OK
	Deuxième teste lancé avec date =« 2025-05-10 17:00:15 »	Date supérieur à aujourd'hui = 1 erreur	
Contrôler que les méthodes d'ajout et de suppression de playlists, formations et catégories fonctionnent	Test sur le nombre d'entité avant et après la méthode	Nb ancien = nb nouveau + 1 ou nb nouveau -1	OK

Tests fonctionnels

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler que les tris fonctionnent (en testant juste le résultat de la première ligne)	Test lancé sur la première ligne du tableau, demande de renvoi du nom de la formation	'Eclipse n°6 : Documentation technique'	OK
contrôler que les filtres fonctionnent (en testant le nombre de lignes obtenu et le résultat de la première ligne) ;	Test lancé avec comme paramètre de recherche : 'Eclipse n°5 : Refactoring'	Eclipse n°5 : Refactoring, 1 ligne	OK
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 lancé avec le lien d'une playlist Test sur le titre de cette playlist	/playlists/playlist/13	OK

Tâche 2 : créer la documentation technique

Temps estimé : 1h

Temps réel : 2h

A réaliser

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

Kanban

The Kanban board is divided into three columns: **Todo** (4 items), **In Progress** (1 item), and **Done** (7 items). Each item is represented by a card with a title and a status icon.

Column	Item	Status
Todo (4)	mediatekformation #9 créer la documentation utilisateur	Not Started
	mediatekformation #10 Déployer le site	Not Started
	mediatekformation #11 Gérer la sauvegarde et la restauration de la BDD	Not Started
	mediatekformation #12 Mettre en place le déploiement continu	Not Started
In Progress (1)	mediatekformation #8 Créer la documentation technique	In Progress
Done (7)	mediatekformation #2 Ajouter une fonctionnalité	Completed
	mediatekformation #3 Gérer les formations	Completed
	mediatekformation #4 Gérer les playlists	Completed
	mediatekformation #5 Gérer les catégories	Completed

Première page de la documentation

mediatekformation

Namespaces

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

Packages

- Application

Reports

- Deprecated
- Errors
- Markers

Indices

- Files

Application

Table of Contents

Classes

-  [AccueilController](#)
Controleur de l'accueil
-  [AdminCategoriesController](#)
Description of AdminCategoriesController
-  [AdminFormationsController](#)
Controleur des formations
-  [AdminPlaylistsController](#)
Description of AdminPlaylistsController
-  [FormationsController](#)
Controleur des formations
-  [OAuthController](#)
-  [PlaylistsController](#)
Description of PlaylistsController

Tâche 3 : créer la documentation utilisateur

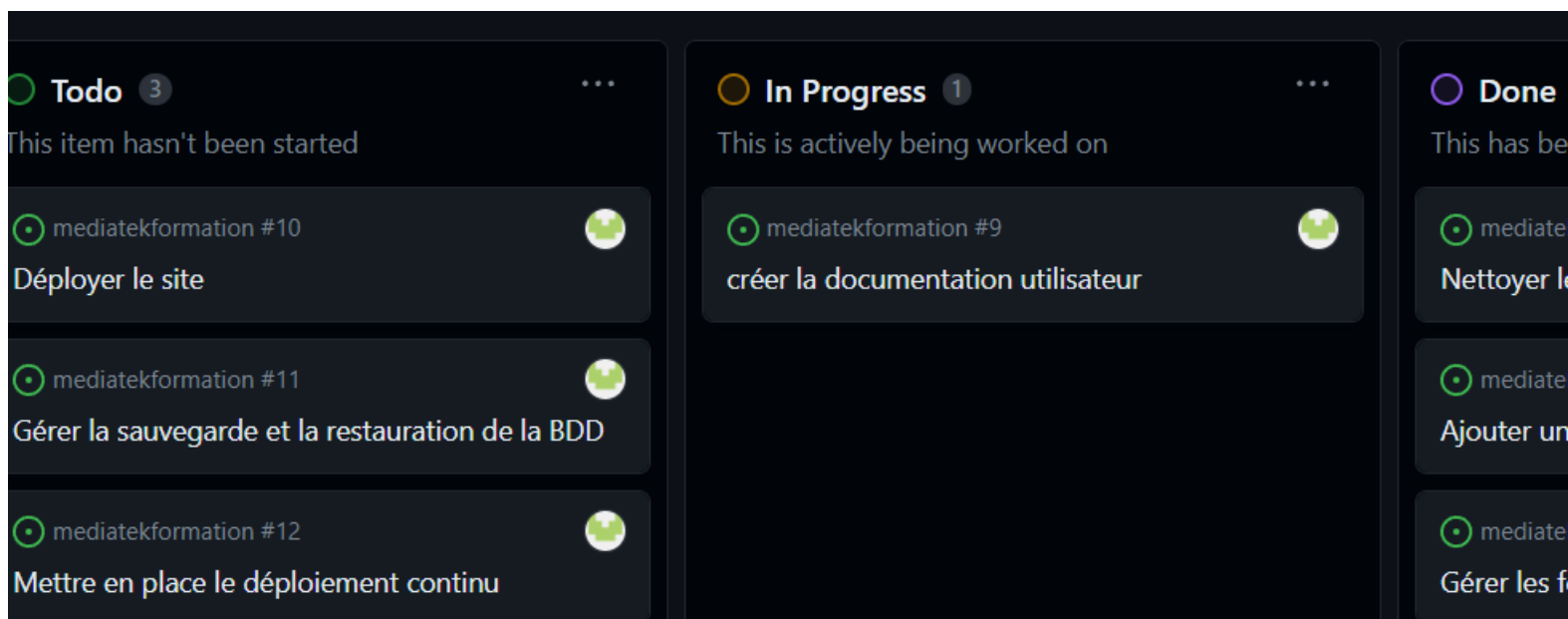
Temps estimé : 2 h

Temps réel : 1 h

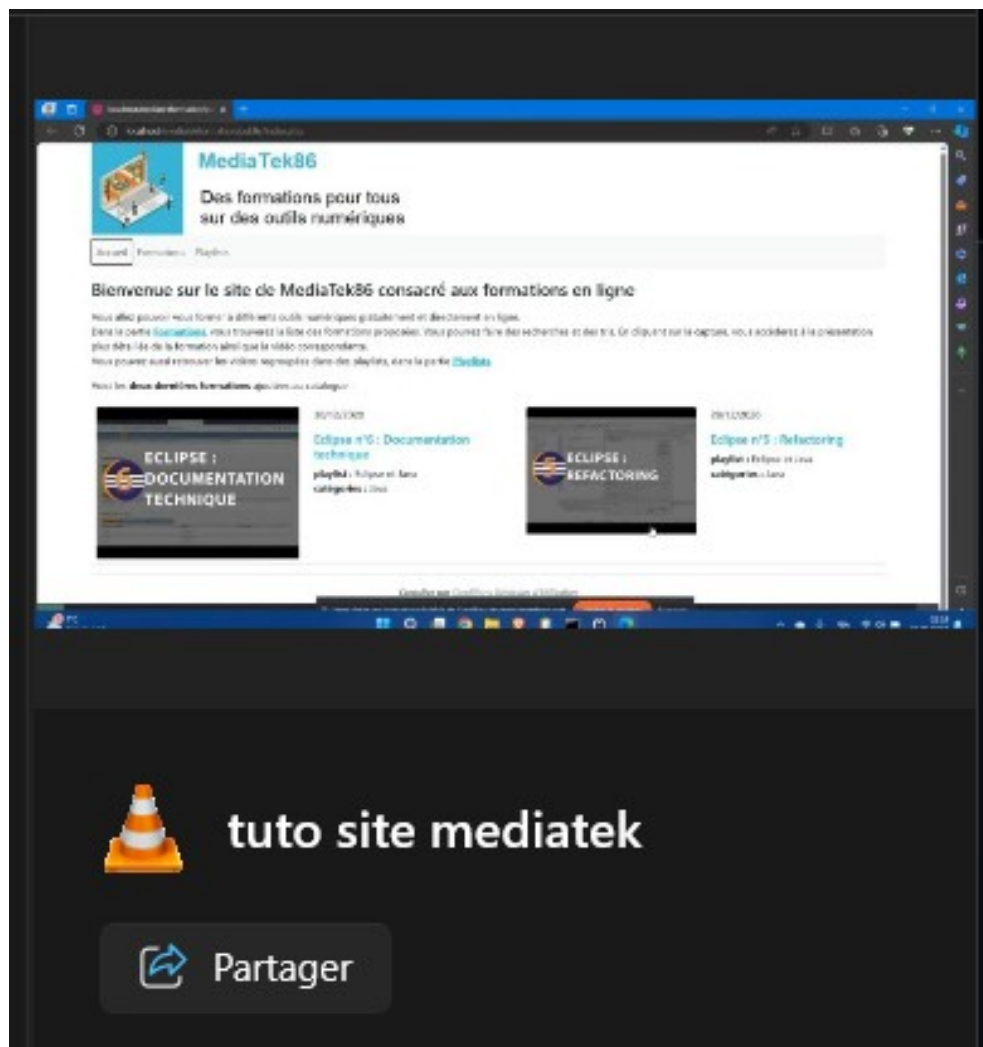
A réaliser :

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.

Kanban



Capture vidéo



Mission 4 : déployer le site et gérer le déploiement continu

Tâche 1: déployer le site

Temps estimé : 2h

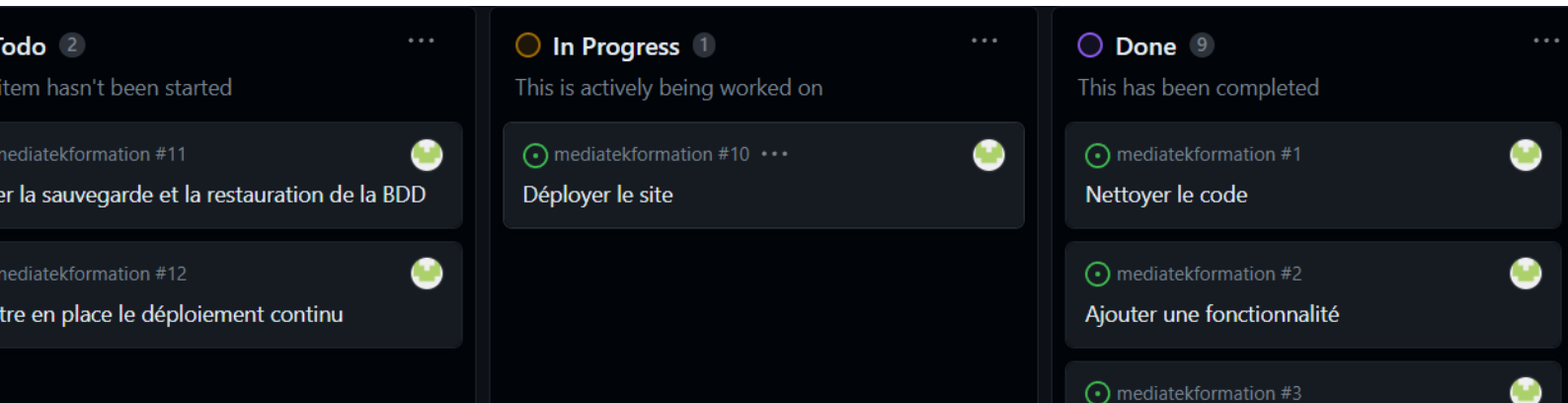
Temps réel : 3h

A faire :

- 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.
- 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).

Kanban



J'ai commencé par créer une machine virtuelle avec Azure, que j'ai ensuite configuré en paramétrant le dns et les ports. J'ai par la suite configuré le jdk en ligne de commande, puis installé keycloak, apache et cerbot (certificat) sur cette machine. J'ai aussi configuré keycloak en créant un royaume, un client ainsi qu'un utilisateur.

Pour le déploiement du site j'ai créé un nom de domaine sur l'hébergeur PlanetHoster, créé la base de donnée et copié le contenu de ma base locale. Pour finir j'ai apporté les modifications au fichier .env (adresse bdd, environnement = prod) puis j'ai transféré le dossier du site via FileZilla pour pouvoir accéder au site

Tâche 2: Gérer la sauvegarde et la restauration de la base de données

Temps estimé : 1h

Temps réel : 2h

A faire :

- La restauration pourra se faire manuellement, en exécutant le script de sauvegarde.

•Une sauvegarde journalière automatisée doit être programmée pour la BDD (voir l'article "Automatiser la sauvegarde d'une BDD" dans le wiki du dépôt).

Kanban



Démarche :

Création en local d'un fichier « Backup.sh » permettant l'enregistrement, ce fichier est convertit en format linux avant d'être transféré sur le site de l'hébergeur (planetHoster), il faut ensuite donner les autorisations sur le dossier et ajouter une tâche Cron pour sauvegarder la BDD à la fréquence souhaitée, ici toutes les minutes. J'ai rencontré un problème d'autorisation lors de la création de ce Cron, j'ai donc ouvert un ticket support à l'équipe de l'hébergeur qui m'ont informé que l'abonnement gratuit ne permettait pas l'ajout de Cron.

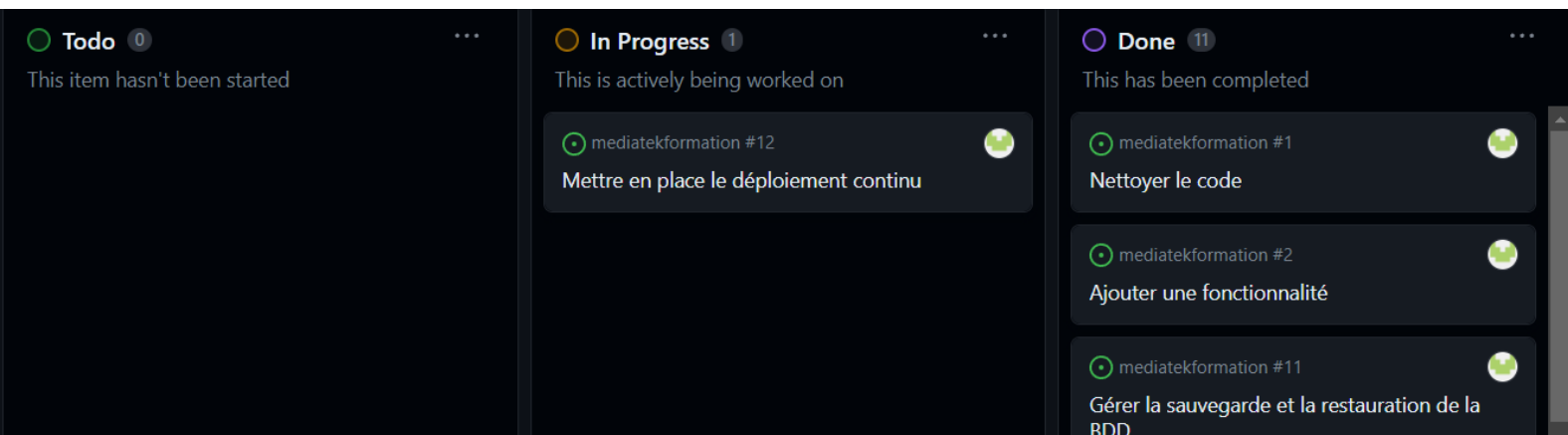
Tâche 3 : mettre en place le déploiement continu

Temps estimé : 1h

Temps réel : 1h

A faire : Configurer le dépôt Github pour que le site en ligne soit mis à jour à chaque push reçu dans le dépôt

Kanban



Etapes : Création d'un workflow sur GitHub, informations de connexions ftp écrites dans le script, puis un tirage (Pull) dans Netbeans pour récupérer le fichier. Ensuite il ne reste plus qu'à faire un Push pour envoyer les modifications au site.

Fichier main.yaml

on: push

name: Deploy website on push

jobs:

web-deploy:

name: Deploy

runs-on: ubuntu-latest

steps:

- name: Get latest code

uses: actions/checkout@v2

- name: Sync files

uses: SamKirkland/FTP-Deploy-Action@4.3.0

with:

server: node181-eu.n0c.com

server-dir: /public_html/mediatekformation/

username: oehtnagrom@morgantmediatek.go.yj.fr

password: \${ secrets.ftp_password }