

### Contexte

Travaillant en tant que technicien développeur junior pour l'ESN InfoTech Services 86 et suite à l'obtention du marché pour différentes interventions au sein du réseau MediaTek86, ce travail portera sur le développement de l'application Web d'accès aux formations en ligne.

### Contenu existant

Nous utiliserons en base de travail la partie front office du site qui a été confiée en amont à un autre développeur. Ce dernier nous a transmis un dossier documentaire nous permettant de visualiser sa démarche et construction applicative. De plus, un dépôt Git est mis à disposition ainsi que la base de données déjà configurée.

### Les langages

PHP et son framework Symfony ainsi que Twig seront les langages utilisés dans le cadre de ce travail.

La base de données sera quant à elle gérée dans phpMyAdmin en format .sql.

### Les technologies utilisées

L'IDE sélectionné sera Netbeans de Apache. Les plugins SonarLint (analyse du code), Selenium (tests de compatibilité Web) et phpDocumentor (création de la documentation technique) seront installés dans l'IDE.

WampServer servira de base de données pour le développement en local.

Composer (gestionnaire de dépendance entre application et librairies) sera également configuré et utilisé en invite de commande Windows.

La solution de versioning Git sera utilisée ainsi que la plateforme Web GitHub afin de sécuriser le code. Nous utiliserons également l'outil Projets de cette plateforme afin d'y répertorier les tâches à effectuer dans un Kanban ainsi que lors du déploiement continu du projet une fois mis en ligne.

Le déploiement en ligne sera effectué grâce à la solution gratuite de l'hébergeur PlanetHoster et utiliserons la base de données MariaDB fournie par celui-ci afin d'y mettre en ligne notre BDD. Le projet enregistré en local sera transféré sur cette plateforme par l'intermédiaire de FileZilla. Une sauvegarde journalière se fera à l'aide d'un script en .sh et configurée sur la plateforme d'administration de l'hébergeur également.

Nous utiliserons également une machine virtuelle Microsoft Azure afin de :

- générer un certificat SSL via Xampp puis Certbot dans le but d'utiliser le protocole HTTPS.
- configurer Keycloak, le gestionnaire d'accès à la partie sécurisée du site Web.

# Table des matières

## Mission 1 : nettoyer et optimiser le code existant

<a href="#">Tâche 1 : nettoyer le code</a>	page 3
<a href="#">Tâche 2 : respecter les bonnes pratiques de codage</a>	page 25
<a href="#">Tâche 3 : ajouter une fonctionnalité</a>	page 33

## Mission 2 : coder la partie back-office

<a href="#">Tâche 1 : gérer les formations</a>	page 38
<a href="#">Tâche 2 : gérer les playlists</a>	page 51
<a href="#">Tâche 3 : gérer les catégories</a>	page 62
<a href="#">Tâche 4 : ajouter l'accès avec authentification</a>	page 71

## Mission 3 : tester et documenter

<a href="#">Tâche 1 : gérer les tests</a>	page 83
<a href="#">Tâche 2 : créer la documentation technique</a>	page 89
<a href="#">Tâche 3 : créer la documentation utilisateur</a>	page 90

## Mission 4 : déployer le site et gérer le déploiement continu

<a href="#">Tâche 1 : déployer le site</a>	page 91
<a href="#">Tâche 2 : gérer la sauvegarde et la restauration de la BDD</a>	page 93
<a href="#">Tâche 3 : mettre en place le déploiement continu</a>	page 96

## Mission 1 : nettoyer et optimiser le code existant.

### Tâche 1 : nettoyer le code.

Nettoyer le code en suivant les indications de SonarLint (ne nettoyer que les fichiers créés par le développeur).

Temps de travail estimé  
réel  
2 heures

Temps de travail  
4 heures

### Kanban de la tâche actuelle "In Progress"

Jlauth

Repositories 8 Projects 1

Mediatek Formation  
Updated 22 hours ago

Filter cards

Issues 0

To do 13

- M2 - T2 : gérer les playlists  
mediatekformation#5 opened by Jlauth
- M2 - T3 : gérer les catégories  
mediatekformation#6 opened by Jlauth
- M2 - T4 : ajouter l'accès avec authentification  
mediatekformation#7 opened by Jlauth
- M2 - T5 : gérer la sauvegarde et la restauration de la DBB  
mediatekformation#8 opened by Jlauth
- M3 - T1 : gérer les tests  
mediatekformation#9 opened by Jlauth
- M3 - T2 : créer la documentation technique  
mediatekformation#10 opened by Jlauth
- M3 - T3 : créer la documentation utilisateur  
mediatekformation#11 opened by Jlauth
- M4 - T1 : déployer le site  
mediatekformation#12 opened by Jlauth
- M4 - T2 : mettre en place le déploiement continu

In progress 1

- M1 - T1 : nettoyer le code  
mediatekformation#1 opened by Jlauth

Done 0

Afin de respecter l'arborescence du projet, nous commençons par l'analyse du fichier **Controller** dans **src**.

### Résultat SonarLint pour **FormationsController**

```
/**
 * @Route("/formations", name="formations")
 * @return Response
 */
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")
 * @param type $champ
 * @param type $ordre
 * @param type $table
 * @return Response
 */
public function sort($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
    ]);
}

/**
 * @Route("/formations/recherche/{champ}/{table}", name="formations.findallcontain")
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
public function findAllContain($champ, Request $request, $table=""): Response{
    $valeur = $request->get("recherche");
    $formations = $this->formationRepository->findByContainValue($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/formations.html.twig", [
        'formations' => $formations,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}
```

### Modifications apportées

"pages/formations.html.twig" est dupliqué dans les fonctions *index()*, *sort()* et *findAllContain()*. A noter que *showOne()* n'affiche qu'une formation et redirige vers la page d'une seule formation, elle ne sera donc pas réécrite.

On écrit et assigne à la variable *\$pagesFormations* la valeur "pages/formations.html.twig", puis on l'implémente dans nos trois fonctions.

## Extraits de code corrigés

```
/**
 *
 * @var type String
 */
private $pagesFormations = "pages/formations.html.twig";

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

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

/**
 * @Route("/formations/recherche/{champ}/{table}", name="formations.findallocontain")
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
public function findAllContain($champ, Request $request, $table=""): Response{
    $valeur = $request->get("recherche");
    $formations = $this->formationRepository->findByContainValue($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesFormations, [
        'formations' => $formations,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}
```

## Nouvelle analyse SonarLint

SonarLint Analyzer Window ×

FormationsController.php ×

Nodes

 Analyze done, 0 issue found

## Résultat SonarLint pour **PlaylistsController**.

SonarLint Analyzer Window ×

PlaylistsController.php ×

Nodes

Analyze done, 1 issue found

critical (1 issue)

php:S1192 : String literals should not be duplicated (1)

52:29: PlaylistsController.php

```
/**
 * @Route("/playlists", name="playlists")
 * @return Response
 */
public function index(): Response{
    $playlists = $this->playlistRepository->findAllOrderBy('name', 'ASC');
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/playlists.html.twig", [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}

/**
 * @Route("/playlists/tri/{champ}/{ordre}", name="playlists.sort")
 * @param type $champ
 * @param type $ordre
 * @return Response
 */
public function sort($champ, $ordre): Response{
    $playlists = $this->playlistRepository->findAllOrderBy($champ, $ordre);
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/playlists.html.twig", [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}
```

```

/**
 * @Route("/playlists/recherche/{champ}/{table}", name="playlists.findallcontain")
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
public function findAllContain($champ, Request $request, $table=""): Response{
    $valeur = $request->get("recherche");
    $playlists = $this->playlistRepository->findByContainValue($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render("pages/playlists.html.twig", [
        'playlists' => $playlists,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}

/**
 * @Route("/playlists/playlist/{id}", name="playlists.showone")
 * @param type $id
 * @return Response
 */
public function showOne($id): Response{
    $playlist = $this->playlistRepository->find($id);
    $playlistCategories = $this->categorieRepository->findAllForOnePlaylist($id);
    $playlistFormations = $this->formationRepository->findAllForOnePlaylist($id);
    return $this->render("pages/playlist.html.twig", [
        'playlist' => $playlist,
        'playlistcategories' => $playlistCategories,
        'playlistformations' => $playlistFormations
    ]);
}

```

### Modifications apportées

Ecriture et assignation à **\$pagesPlaylists** la valeur **"pages/playlists.html.twig"** puis affectation de cette variable dans les trois fonctions **index()**, **sort()** et **findAllContain()**. La dernière fonction redirigeant vers une seule playlist, elle reste en "dur".

### Extraits de code corrigés

---

```

/**
 *
 * @var type String
 */
private $pagesPlaylists = "pages/playlists.html.twig";

```

```

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

```

```

/**
 * @Route("/playlists/tri/{champ}/{ordre}", name="playlists.sort")
 * @param type $champ
 * @param type $ordre
 * @return Response
 */
public function sort($champ, $ordre): Response {
    $playlists = $this->playlistRepository->findAllOrderBy($champ, $ordre);
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesPlaylists, [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}

```

```

/**
 * @Route("/playlists/recherche/{champ}", name="playlists.findallcontain")
 * @param type $champ
 * @param Request $request
 * @return Response
 */
public function findAllContain($champ, Request $request): Response {
    $valeur = $request->get("recherche");
    $playlists = $this->playlistRepository->findByContainValue($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesPlaylists, [
        'playlists' => $playlists,
        'categories' => $categories,
        'valeur' => $valeur
    ]);
}

```

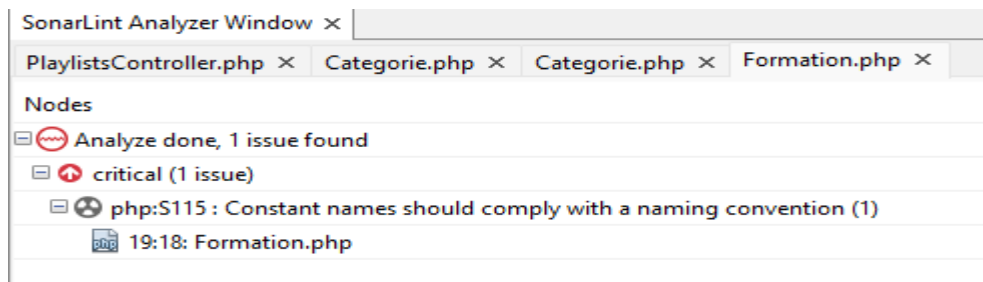
## Nouvelle analyse SonarLint





Les analyses du folder **Controller** étant terminées et corrigées, nous passons au folder **Entity**.

### Résultat SonarLint pour la classe php **Formation**



```
/**
 * Début de chemin vers les images
 */
private const cheminImage = "https://i.ytimg.com/vi/";

public function getMiniature(): ?string
{
    return self::cheminImage.$this->videoId."/default.jpg";
}

public function getPicture(): ?string
{
    return self::cheminImage.$this->videoId."/hqdefault.jpg";
}
```

### Modifications apportées

La constante **cheminImage** ne respecte pas la convention d'écriture pour une constante, celle-ci doit être écrite en lettres majuscules.

### Extraits de code corrigés

```
/**
 * Début de chemin vers les images
 */
private const CHEMINIMAGE = "https://i.ytimg.com/vi/";

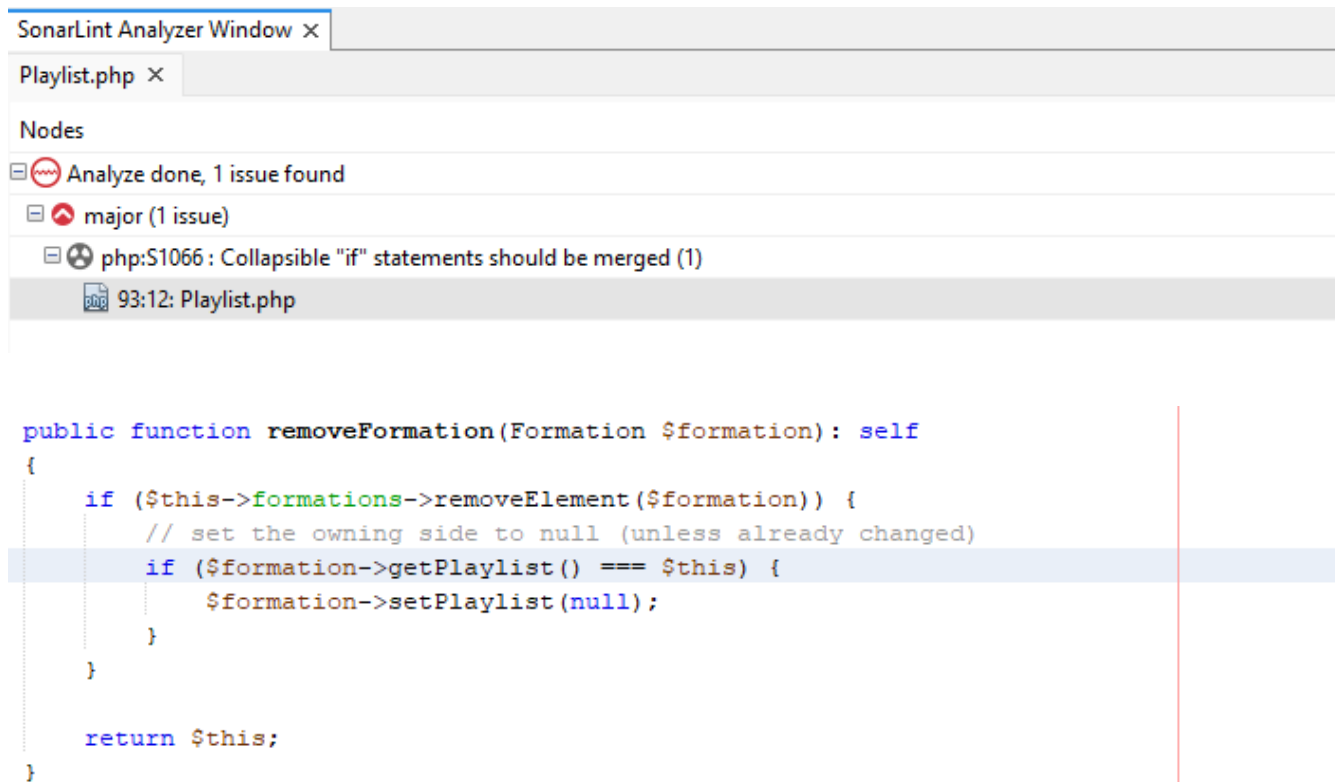
public function getMiniature(): ?string
{
    return self::CHEMINIMAGE.$this->videoId."/default.jpg";
}

public function getPicture(): ?string
{
    return self::CHEMINIMAGE.$this->videoId."/hqdefault.jpg";
}
```

### Nouvelle analyse SonarLint



## Résultat SonarLint pour la classe php **Playlist**



The screenshot shows the SonarLint Analyzer Window for the file Playlist.php. It indicates that the analysis is complete and one issue was found. The issue is a major one, labeled 'php:S1066: Collapsible "if" statements should be merged (1)'. The code snippet below the issue shows a function `removeFormation` with two nested `if` statements that can be merged.

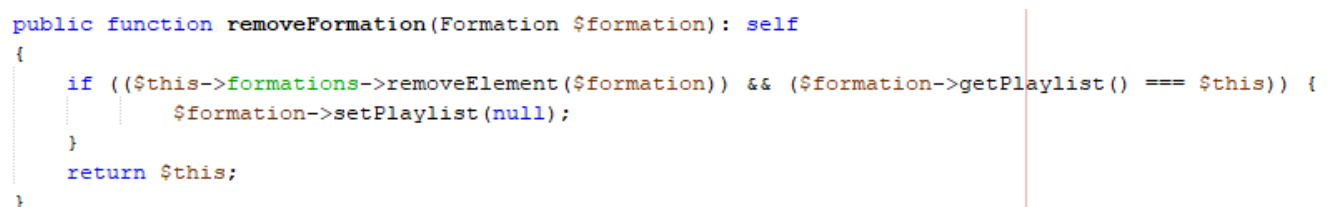
```
public function removeFormation(Formation $formation): self
{
    if ($this->formations->removeElement($formation)) {
        // set the owning side to null (unless already changed)
        if ($formation->getPlaylist() === $this) {
            $formation->setPlaylist(null);
        }
    }

    return $this;
}
```

### Modifications apportées

Afin d'éviter une répétition de la structure conditionnelle **if**, on merge les deux instructions afin de rendre le code plus lisible.

### Extraits de code corrigés



The screenshot shows the corrected code snippet where the two `if` statements have been merged into a single one using the `&&` operator.

```
public function removeFormation(Formation $formation): self
{
    if (($this->formations->removeElement($formation)) && ($formation->getPlaylist() === $this)) {
        $formation->setPlaylist(null);
    }
    return $this;
}
```

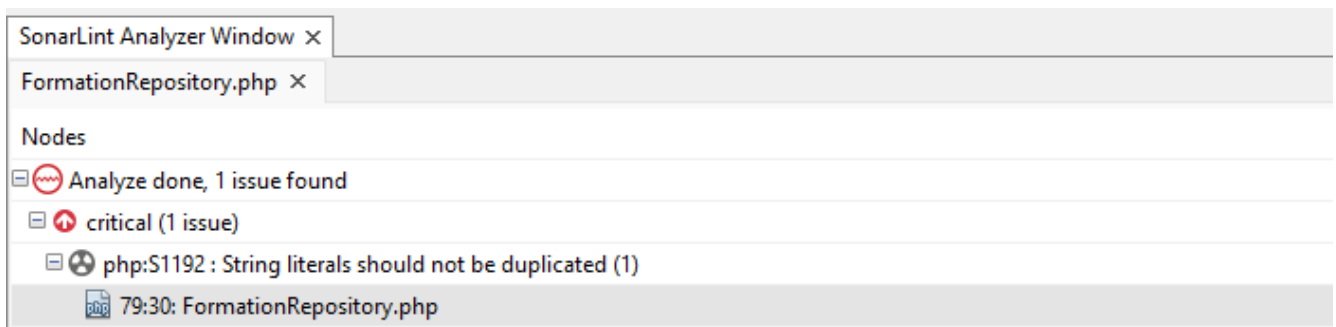
### Nouvelle analyse SonarLint



The screenshot shows the SonarLint Analyzer Window for the file Playlist.php after a new analysis. It indicates that the analysis is complete and no issues were found.

Nous passons ensuite au folder **Repository**.

## Résultat SonarLint sur **FormationRepository**



The screenshot shows the SonarLint Analyzer Window for the file FormationRepository.php. It indicates that the analysis is complete and one issue has been found. The issue is categorized as 'critical' and is a 'php:S1192' rule violation, stating 'String literals should not be duplicated (1)'. The issue is located at line 79, column 30 in the file FormationRepository.php.

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

```

/**
 * Retourne les n formations les plus récentes
 * @param type $nb
 * @return Formation[]
 */
public function findAllLasted($nb) : array {
    return $this->createQueryBuilder('f')
        ->orderBy('f.publishedAt', 'DESC')
        ->setMaxResults($nb)
        ->getQuery()
        ->getResult();
}

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

```

### Modification apportée

Afin d'éviter la duplication du paramètre ***f.publishedAt***, on écrit la variable privée ***\$publishedAt*** en lui assignant le paramètre cité.

### Extraits de code corrigés

```

/**
 *
 * @var type String
 */
private $publishedAt = 'f.publishedAt';

```

On implémente cette nouvelle variable dans les instructions de la clause ***orderBy()*** dans les fonctions ***findByContainValue()***, ***findAllLasted()*** et ***findAllForOnePlaylist()***.

```

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

```

```

/**
 * Retourne les n formations les plus récentes
 * @param type $nb
 * @return Formation[]
 */
public function findAllLasted($nb) : array {
    return $this->createQueryBuilder('f')
        ->orderBy($this->publishedAt, 'DESC')
        ->setMaxResults($nb)
        ->getQuery()
        ->getResult();
}

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

```

## Nouvelle analyse SonarLint



## Résultat SonarLint pour la classe php **PlaylistRepository**

SonarLint Analyzer Window ×

PlaylistRepository.php ×

Nodes

- [-] Analyze done, 6 issues found
  - [-] critical (6 issues)
    - [-] php:S1192: String literals should not be duplicated (6)
      - 50:25: PlaylistRepository.php
      - 51:28: PlaylistRepository.php
      - 52:28: PlaylistRepository.php
      - 53:27: PlaylistRepository.php
      - 54:27: PlaylistRepository.php
      - 56:29: PlaylistRepository.php

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

(suite page 15)

```

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

```

### Modifications apportées

Afin d'éviter ces nouvelles duplications de type String dans le code, nous appliquons la même méthode que pour la classe php **FormationRepository**.

### Écriture et assignation des variables

```

// Propriétés privées de la classe PlaylistRepository
private $id = 'p.id id';
private $name = 'p.name name';
private $nameCategory = 'c.name categorienome';
private $formations = 'p.formations';
private $categories = 'f.categories';
private $nameCategories = 'c.name';

```

## Ainsi que leur implémentation

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

(suite page 17)



```

/**
 * 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 $champ dans une autre table
 * @return Playlist[]
 */
public function findByContainValue($champ, $valeur, $table=""): array{
    if($valeur==""){
        return $this->findAllOrderBy('name', 'ASC');
    }
    if($table==""){
        return $this->createQueryBuilder('p')
            ->select($this->id)
            ->addSelect($this->name)
            ->addSelect($this->nameCategory)
            ->leftjoin($this->formations, 'f')
            ->leftjoin($this->categories, 'c')
            ->where('p.'.$champ.' LIKE :valeur')
            ->setParameter('valeur', '%'.$valeur.'%')
            ->groupBy('p.id')
            ->addGroupBy($this->nameCategories)
            ->orderBy('p.name', 'ASC')
            ->addOrderBy($this->nameCategories)
            ->getQuery()
            ->getResult();
    }else{
        return $this->createQueryBuilder('p')
            ->select($this->id)
            ->addSelect($this->name)
            ->addSelect($this->nameCategory)
            ->leftjoin($this->formations, 'f')
            ->leftjoin($this->categories, 'c')
            ->where('c.'.$champ.' LIKE :valeur')
            ->setParameter('valeur', '%'.$valeur.'%')
            ->groupBy('p.id')
            ->addGroupBy($this->nameCategories)
            ->orderBy('p.name', 'ASC')
            ->addOrderBy($this->nameCategories)
            ->getQuery()
            ->getResult();
    }
}

```

## Nouvelle analyse SonarLint

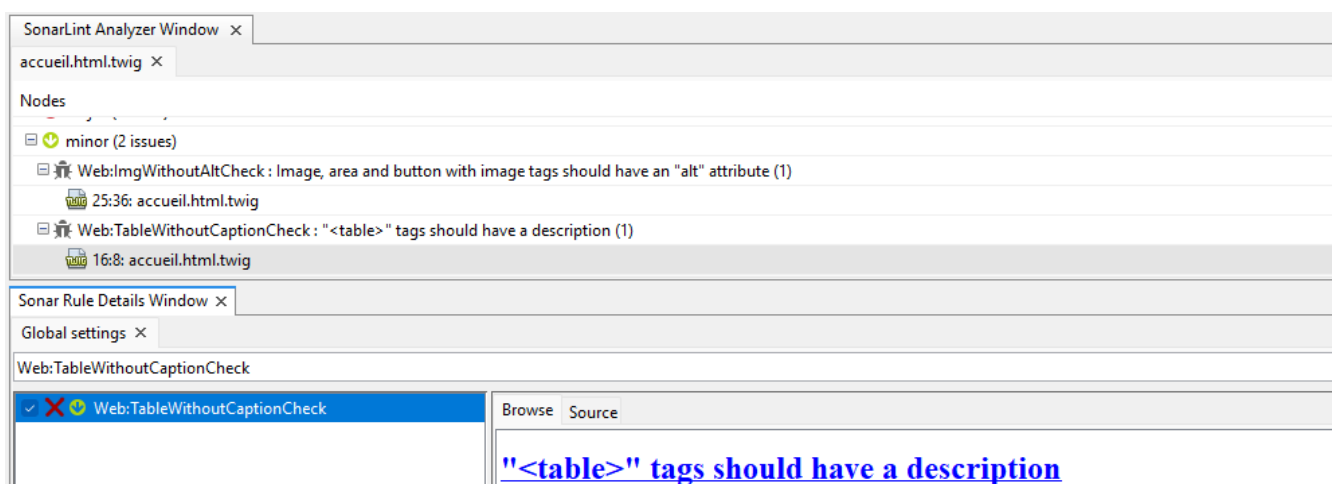


## Analyse SonarLint finale du dossier **src**



Notre travail portera ensuite sur le folder **template**.

## Résultat SonarLint pour **accueil.html.twig**



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

```
<table class="table">
```

## Modifications apportées

Ajout d'un élément ***“description”***.

```
<p Voici les deux dernières formations ajoutées au catalogue :</p>
<table class="table" description="deuxdernièresformations">
```

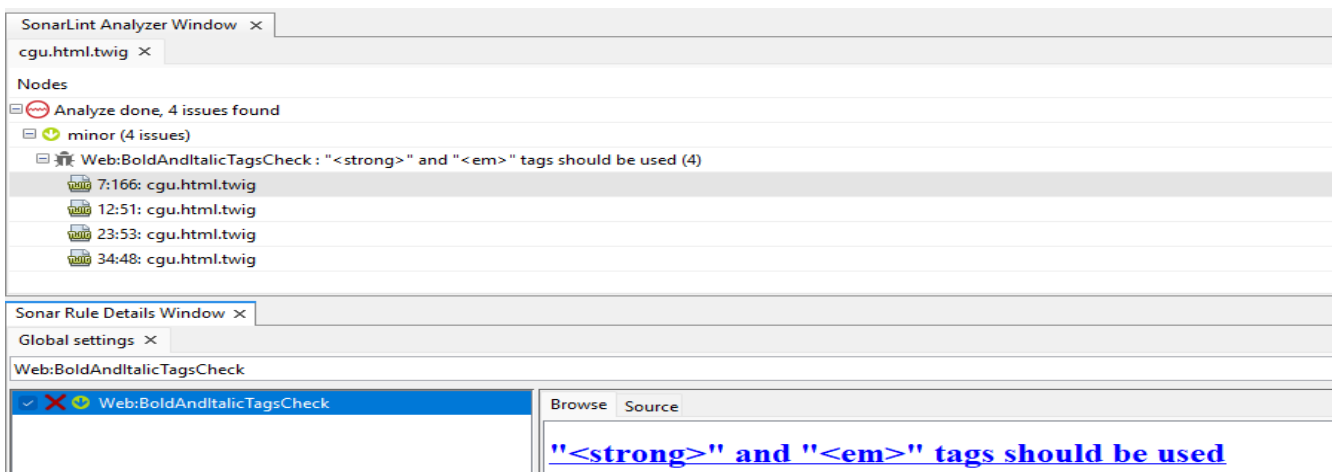
Ajout d'un attribut ***“alt”*** afin de donner une alternative textuelle à une image.

```
<!-- emplacement photo -->
{% if formation.picture %}
    <a href="{{ path('formations.showone', {id:formation.id}) }}">
        
    </a>
{% endif %}
```

## Nouvelle analyse SonarLint



## Résultat SonarLint pour ***cgu.html.twig***



(ci-après "*i>*le site*</i>").*

```

<summary>A. Éditeur du site (ci-après "<i>l'Éditeur</i>")</summary>
<summary>B. Hébergeur du site (ci-après "<i>l'Hébergeur</i>")</summary>
<summary>C. Utilisateurs (ci-après "<i>les Utilisateurs</i>")</summary>

```

### Modifications apportées

Refactorisation de l'attribut “*i*” par “**em**”. En effet, cet attribut n'est pas compatible avec certains dispositifs d'affichage (version de navigateur, lecture HTML adaptée à certains handicaps).

### Extraits de code corrigés

```

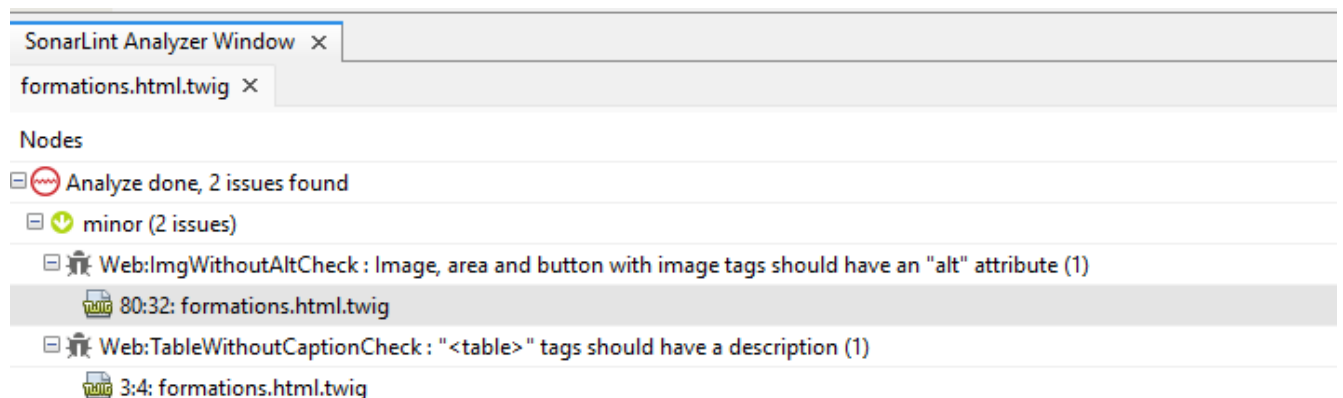
(ci-après "<em>le site</em>").
<summary>A. Éditeur du site (ci-après "<em>l'Éditeur</em>")</summary>
<summary>B. Hébergeur du site (ci-après "<em>l'Hébergeur</em>")</summary>
<summary>C. Utilisateurs (ci-après "<em>les Utilisateurs</em>")</summary>

```

### Nouvelle analyse SonarLint



### Résultat SonarLint pour formations.html.twig



```

<td class="text-center">
    {% if formation.miniature %}
        <a href="{{ path('formations.showone', {id:formation.id}) }}">
            
        </a>
    {% endif %}
</td>

{% extends "basefront.html.twig" %}
{% block body %}
    <table class="table table-striped">

```

### Modifications apportées

Ajout d'un attribut **"alt"** afin de donner une alternative textuelle à une image.

Ajout d'un élément **"description"**.

### Extraits de code corrigés

```

<td class="text-center">
    {% if formation.miniature %}
        <a href="{{ path('formations.showone', {id:formation.id}) }}">
            
        </a>
    {% endif %}
</td>

```

```

{% block body %}
    <table class="table table-striped" description='Page des formations'>

```

### Nouvelle analyse SonarLint



## Résultat SonarLint sur `playlists.html.twig`



SonarLint Analyzer Window x

playlists.html.twig x

Nodes

- Analyze done, 1 issue found
- minor (1 issue)
- Web:TableWithoutCaptionCheck : "<table>" tags should have a description (1)

3:4: playlists.html.twig

```
{% extends "basefront.html.twig" %}
{% block body %}
    <table class="table table-striped">
```

## Modification apportée

Ajout d'un élément ***"description"***.

## Extrait de code corrigé

```
{% extends "basefront.html.twig" %}
{% block body %}
    <table class="table table-striped" description='Page des playlists'>
```

## Nouvelle analyse SonarLint



SonarLint Analyzer Window x

playlists.html.twig x

Nodes

- Analyze done, 0 issue found

## Résultat SonarLint pour `playlist.html.twig`



SonarLint Analyzer Window x

playlist.html.twig x

Nodes

- Analyze done, 1 issue found
- minor (1 issue)
- Web:ImgWithoutAltCheck : Image, area and button with image tags should have an "alt" attribute (1)

23:32: playlist.html.twig

Sonar Rule Details Window x

Global settings x

Web:ImgWithoutAltCheck

Web:ImgWithoutAltCheck Browse Source

**Image, area and button with image tags should have an "alt" attribute**

```

<!-- boucle sur l'affichage des formations -->
{% for formation in playlistformations %}
    <div class="row mt-1">
        <div class="col-md-auto">
            {% if formation.miniature %}
                <a href="{{ path('formations.showone', {id:formation.id}) }}">
                    
                </a>
            {% endif %}

```

## Modification apportée

Création d'un attribut **"alt"**.

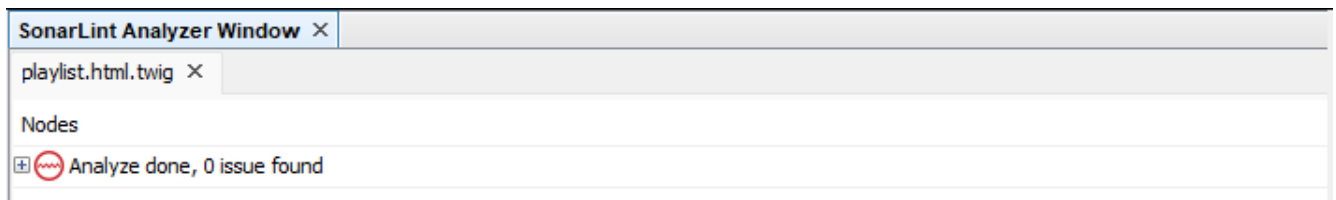
## Extrait de code

```

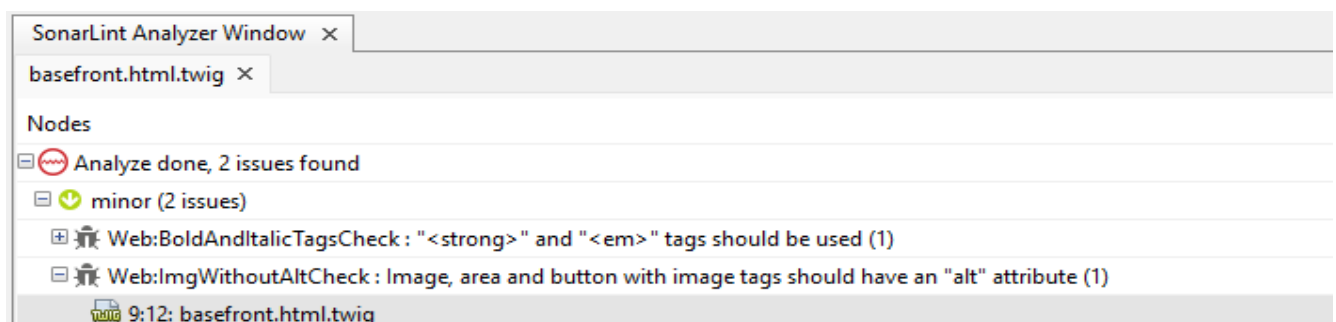
<!-- boucle sur l'affichage des formations -->
{% for formation in playlistformations %}
    <div class="row mt-1">
        <div class="col-md-auto">
            {% if formation.miniature %}
                <a href="{{ path('formations.showone', {id:formation.id}) }}">
                    
                </a>
            {% endif %}

```

## Nouvelle analyse SonarLint



## Résultat SonarLint pour **basefront.html.twig**



```

<p><small><i>
  Consultez nos <a class="link-secondary" href="{{ path('cgu') }}">Conditions Générales d'Utilisation</a>
</i></small></p>

<div class="text-left">
  
</div>

```

## Modifications apportées

Refactoring de l'attribut **"i"** en attribut **"em"**.  
Création d'un attribut **"alt"**.

## Extraits de code corrigés

```

<p><small><em>
  Consultez nos <a class="link-secondary" href="{{ path('cgu') }}">Conditions Générales d'Utilisation</a>
</em></small></p>

<div class="text-left">
  
</div>

```

## Nouvelle analyse SonarLint



## Ainsi que pour le folder templates





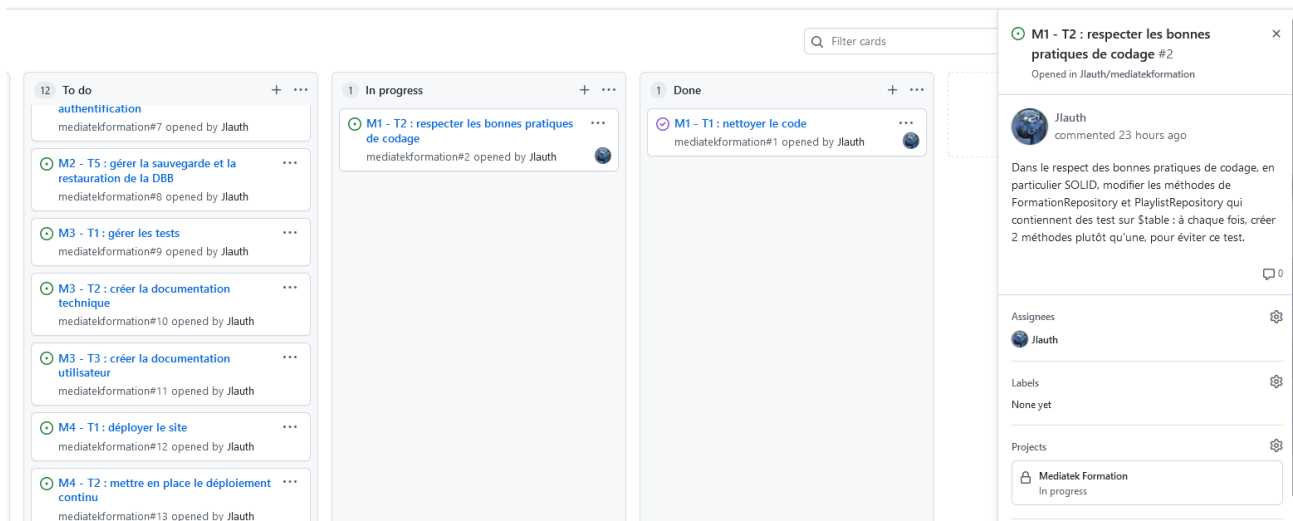
## Tâche 2 : respecter les bonnes pratiques de codage.

Dans le respect des bonnes pratiques de codage, en particulier du SOLID (ici le S pour “Single responsibility”), modifier les méthodes de `FormationRepository` et `PlaylistRepository` qui contiennent des tests sur `$table`. Créer à chaque fois deux méthodes plutôt qu’une afin d’éviter ce test. Le reste du code doit être adapté pour exploiter ces nouvelles méthodes.

Temps de travail estimé  
réel  
2 heures

Temps de travail  
  
3 heures

### Kanban de la tâche actuelle “In Progress”



The screenshot displays a Kanban board with three columns: 'To do', 'In progress', and 'Done'. The 'In progress' column contains one task: 'M1 - T2 : respecter les bonnes pratiques de codage'. A detailed view of this task is shown on the right, including a description, assignee (Jlauth), and project (Mediatek Formation).

**Task Details:**

- Task:** M1 - T2 : respecter les bonnes pratiques de codage #2
- Opened in:** Jlauth/mediatekformation
- Assignee:** Jlauth
- Labels:** None yet
- Projects:** Mediatek Formation

**Description:** Dans le respect des bonnes pratiques de codage, en particulier SOLID, modifier les méthodes de `FormationRepository` et `PlaylistRepository` qui contiennent des test sur `$table` : à chaque fois, créer 2 méthodes plutôt qu’une, pour éviter ce test.

Nous commençons par la classe **FormationRepository**.

La fonction ***findAllOrderBy()*** avant modification.

```
/**
 * Retourne toutes les formations triées sur un champ
 * @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();
    }
}
```

### Modifications apportées

Le but étant d'alléger le code, et de respecter le "Single Responsibility" en référence au S l'acronyme SOLID, nous dupliquons la fonction ***findAllOrderBy()*** en deux fonctions distinctes.

La première garde le même nom, ***findAllOrderBy()***, et retourne les informations triées sur le champ avec ***\$champ*** présent dans l'entité **Formation**.

---

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

La seconde fonction est nommée ***findAllOrderByTable()***. Cette fonction retourne également les informations triées sur un champ mais en y ajoutant un critère, ***\$table***, afin que cette fonction s'exécute en dehors des variables de l'entité **Formation**.

```
/**
 * Retourne toutes les informations triées sur un champ
 * Avec $champ présent dans une autre entité
 * @param type $champ
 * @param type $ordre
 * @param type $table la table correspondant au $champ recherché
 * @return Formation[]
 */
public function findAllOrderByTable($champ, $ordre, $table): array {
    return $this->createQueryBuilder('f')
        ->join('f.' . $table, 't')
        ->orderBy('t.' . $champ, $ordre)
        ->getQuery()
        ->getResult();
}
```

La fonction ***findByContainValue()*** avant modification.

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

## Modifications apportées

On scinde également la fonction en deux afin que chacune n'ait qu'une seule responsabilité.

---

```
/**
 * Retourne la valeur renseignée en fonction du champ
 * Ou tous les enregistrements si valeur est null
 * @param type $champ
 * @param type $valeur
 * @return Formation[]
 */
public function findByContainValue($champ, $valeur): array {
    if ($valeur == "") {
        return $this->findAll();
    } else {
        return $this->createQueryBuilder('f')
            ->where('f.' . $champ . ' LIKE :valeur')
            ->orderBy($this->publishedAt, 'DESC')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->getQuery()
            ->getResult();
    }
}

/**
 * Retourne la valeur renseignée en fonction du champ
 * Ou tous les enregistrements si la valeur est null
 * Avec $champ présent dans une autre entité
 * @param type $champ
 * @param type $valeur
 * @param type $table la table correspondant au $champ recherché
 * @return Formation[]
 */
public function findByContainValueTable($champ, $valeur, $table): array {
    if ($valeur == "") {
        return $this->findAll();
    } else {
        return $this->createQueryBuilder('f')
            ->join('f.' . $table, 't')
            ->where('t.' . $champ . ' LIKE :valeur')
            ->orderBy($this->publishedAt, 'DESC')
            ->addOrderBy('f.id', 'DESC')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->getQuery()
            ->getResult();
    }
}
```

---

On effectue les modifications requises dans **FormationsController**, pour les fonctions **sort()** et **findAllContain()**.

### Modifications apportées

Initialisation de **\$table** à **null** et vérification dans la première ligne de la fonction **sort()** grâce à la structure de test **"if"**.

Si table n'est pas égale à **null**, la première fonction **findAllOrderByTable()** est appelée avec la table en paramètre afin de pouvoir l'utiliser ultérieurement dans twig grâce à la route de la fonction **sort()**.

Sinon la seconde fonction **findAllOrderBy()** est appelée sans le paramètre **\$table**.

```
/**
 * @Route("/formations/tri/{champ}/{ordre}/{table}", name="formations.sort")
 * @param type $champ
 * @param type $ordre
 * @param type $table
 * @return Response
 */
public function sort($champ, $ordre, $table = ""): Response {
    if ($table != "") {
        $formations = $this->formationRepository->findAllOrderByTable($champ, $ordre, $table);
    } else {
        $formations = $this->formationRepository->findAllOrderBy($champ, $ordre);
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesFormations, [
        'formations' => $formations,
        'categories' => $categories
    ]);
}
```

On effectue la même démarche pour la fonction **findAllContain()** en oubliant pas au préalable d'initialiser **\$valeur** en fonction de la requête sur **"recherche"**.

```
/**
 * @Route("/formations/recherche/{champ}/{table}", name="formations.findallcontain")
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
public function findAllContain($champ, Request $request, $table = ""): Response {
    $valeur = $request->get("recherche");
    if ($table != "") {
        $formations = $this->formationRepository->findByContainValueTable($champ, $valeur, $table);
    } else {
        $formations = $this->formationRepository->findByContainValue($champ, $valeur);
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesFormations, [
        'formations' => $formations,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}
```

Concernant la classe **PlaylistRepository**, voici la fonction **findByContainValue()** avant modification.

```
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 categoriename')
            ->leftjoin('p.formations', 'f')
            ->leftjoin('f.categories', 'c')
            ->where('p.' . $champ . ' LIKE :valeur')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->groupBy('p.id')
            ->addGroupBy('c.name')
            ->orderBy('p.name', 'ASC')
            ->addOrderBy('c.name')
            ->getQuery()
            ->getResult();
    }else{
        return $this->createQueryBuilder('p')
            ->select('p.id id')
            ->addSelect('p.name name')
            ->addSelect('c.name categoriename')
            ->leftjoin('p.formations', 'f')
            ->leftjoin('f.categories', 'c')
            ->where('c.' . $champ . ' LIKE :valeur')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->groupBy('p.id')
            ->addGroupBy('c.name')
            ->orderBy('p.name', 'ASC')
            ->addOrderBy('c.name')
            ->getQuery()
            ->getResult();
    }
}
```

### Modifications apportées

La démarche effectuée dans **FormationRepository** sera également appliquée pour cette fonction, **findByContainValue()**.

(suite page 31)

---

```

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

```

---

```

/**
 * Retourne la valeur renseignée en fonction du champ
 * Ou tous les enregistrements si la valeur est null
 * Avec $champ présent dans une autre entité
 * @param type $champ
 * @param type $valeur
 * @param type $table si $champ dans une autre table
 * @return array
 */
public function findByContainValueTable($champ, $valeur, $table): array {
    if ($valeur == "") {
        return $this->findAll();
    } else {
        return $this->createQueryBuilder('p')
            ->leftjoin($this->formations, 'f')
            ->leftjoin($this->categories, 'c')
            ->where('c.' . $champ . ' LIKE :valeur')
            ->setParameter('valeur', '%' . $valeur . '%')
            ->groupBy($this->idPlaylist)
            ->orderBy($this->namePlaylist, 'ASC')
            ->getQuery()
            ->getResult();
    }
}

```

---

\* Vous remarquerez que **\$table** n'est jamais initialisé dans cette deuxième fonction. Cela sera expliqué lors de la Mission 3.

On effectue les modifications requises dans **PlaylistsController**, pour la fonction **findAllContain()**.

### Modifications apportées

Afin d'implémenter ces nouvelles fonctions, on initialise la condition **"if"** après avoir récupéré la valeur de **\$valeur** et ce afin de vérifier quelle fonction appeler, en fonction de la valeur de **\$table**.

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

*\*Depuis la Mission 3, cette fonction **findAllContain()** est écrite de cette façon.*

```
/**
 * @Route("/playlists/recherche/{champ}", name="playlists.findallcontain")
 * @param type $champ
 * @param Request $request
 * @return Response
 */
public function findAllContain($champ, Request $request): Response {
    if ($this->isCsrfTokenValid('filtre_' . $champ, $request->get('_token'))){
        $valeur = $request->get("recherche");
        $playlists = $this->playlistRepository->findByContainValue($champ, $valeur);
        $categories = $this->categorieRepository->findAll();
        return $this->render($this->pagesPlaylists, [
            'playlists' => $playlists,
            'categories' => $categories,
            'valeur' => $valeur
        ]);
    }
    return $this->redirectToRoute("playlists");
}
```



### Tâche 3 : ajouter une fonctionnalité.

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

Temps de travail estimé  
réel  
2 heures

Temps de travail  
  
10 heures

### Kanban de la tâche actuelle "In Progress"

The screenshot displays a Jira Kanban board with three columns: 'To do', 'In progress', and 'Done'. The 'To do' column contains 11 cards, the 'In progress' column contains 1 card, and the 'Done' column contains 2 cards. The card in the 'In progress' column is titled 'M1 - T3 : ajouter une fonctionnalité' and is opened by 'Jlauth'. A detailed view of this card is shown on the right, displaying the task description, assignees, labels, projects, and milestones.

**11 To do**

- M2 - T1 : gérer les formations  
mediatekformation#4 opened by Jlauth
- M2 - T2 : gérer les playlists  
mediatekformation#5 opened by Jlauth
- M2 - T3 : gérer les catégories  
mediatekformation#6 opened by Jlauth
- M2 - T4 : ajouter l'accès avec authentification  
mediatekformation#7 opened by Jlauth
- M2 - T5 : gérer la sauvegarde et la restauration de la DB8  
mediatekformation#8 opened by Jlauth
- M3 - T1 : gérer les tests  
mediatekformation#9 opened by Jlauth
- M3 - T2 : créer la documentation technique  
mediatekformation#10 opened by Jlauth
- M3 - T3 : créer la documentation utilisateur  
mediatekformation#11 opened by Jlauth
- M4 - T1 : déployer le site  
mediatekformation#12 opened by Jlauth

**1 In progress**

- M1 - T3 : ajouter une fonctionnalité  
mediatekformation#3 opened by Jlauth

**2 Done**

- M1 - T1 : nettoyer le code  
mediatekformation#1 opened by Jlauth
- M1 - T2 : respecter les bonnes pratiques de codage  
mediatekformation#2 opened by Jlauth

**M1 - T3 : ajouter une fonctionnalité #3**  
Opened in Jlauth/mediatekformation

Jlauth commented 2 days ago

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.

Assignees  
Jlauth

Labels  
None yet

Projects  
Mediatek Formation  
In progress

Milestone  
No milestone

[Go to issue for full details](#)

[Close issue](#)

## Process

Suite à de nombreuses tentatives, la solution a été trouvée grâce à l'intervention de Madame Martins Da Silva.

Le tri sur le nombre de formations par playlist ne pouvant s'effectuer correctement avec une seule fonction car les différents types de catégories perturbent le rendu d'une fonction unique avec un tri sur ces mêmes catégories. On a donc ajouté une fonction dans la partie **Playlist** afin d'obtenir une collection contenant les noms de catégories ne retournant qu'une seule fois chaque type.

```
/**
 * @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;
}
```

Dans **PlaylistRepository**, écrire de la première fonction **findAllOrderByName()** permettant un tri sur le nom d'une playlist.

```
/**
 * Tri sur le nom d'une playlist
 * @param type $ordre
 * @return Playlist[]
 */
public function findAllOrderByName($ordre): array {
    return $this->createQueryBuilder('p')
        ->leftjoin($this->formations, 'f')
        ->groupBy($this->idPlaylist)
        ->orderBy($this->namePlaylist, $ordre)
        ->getQuery()
        ->getResult();
}
```

Puis écriture d'une seconde fonction appelée ***findAllOrderByNbFormation()*** qui effectuera cette fois un tri sur le nombre de formation(s).

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

Afin d'actualiser **PlaylistsController** avec ces nouvelles fonctions, nous écrivons la fonction ***sortOnNbFormation()*** qui nous permet d'implémenter ***findAllOrderByNbFormations()*** et d'y créer une route distincte de **"playlists.sort"**.

```
/**
 * @Route("/playlists/tri/{ordre}", name="playlists.sortonnbformation")
 * @param type $ordre
 * @return Response
 */
public function sortOnNbFormation($ordre): Response {
    $playlists = $this->playlistRepository->findAllOrderByNbFormation($ordre);
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesPlaylists, [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}
```

Ainsi, dans la fonction ***sort()***, on utilise la structure conditionnelle **"switch/case"** afin de sélectionner quelle fonction appeler suivant la valeur de la variable ***\$playlists***.

(suite page 36)

```

/**
 * @Route("/playlists/tri/{champ}/{ordre}", name="playlists.sort")
 * @param type $champ
 * @param type $ordre
 * @return Response
 */
public function sort($champ, $ordre): Response {
    switch ($champ) {
        case "name":
            $playlists = $this->playlistRepository->findAllOrderByName($ordre);
            break;
        case "nbformations":
            $playlists = $this->playlistRepository->findAllOrderByNbFormation($ordre);
            break;
    }
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesPlaylists, [
        'playlists' => $playlists,
        'categories' => $categories
    ]);
}

```

Une fois **PlaylistController** configuré, nous passons à **playlists.html.twig** afin de créer ce nouvel élément de tri sur le nombre de formations avec le bon path dans le header.

```

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

```

Dans le body de cette même page, on initialise donc le nombre et la taille des formations en fonction des playlists.

```

{% 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">
            {{ playlists[k].formations|length }}
        </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>
    </tr>
{% endfor %}

```

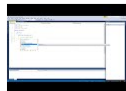



Le rendu du front avec un évènement tri effectué sur le champ “*Nombre de formations*”.

Accueil Formations Playlists			
Playlist	Nombre de formations	Catégories	
<input type="text"/> <input type="button" value="filtrer"/>	<input type="button" value="&lt;"/> <input type="button" value="&gt;"/>	<input type="text"/>	
Bases de la programmation (C#)	74	C# POO	<input type="button" value="Voir détail"/>
Programmation sous Python	19	Python POO	<input type="button" value="Voir détail"/>
MCD : exercices progressifs	18	MCD	<input type="button" value="Voir détail"/>
TP Android (programmation mobile)	18	Android SQL Java	<input type="button" value="Voir détail"/>
Compléments Android (programmation mobile)	13	Android	<input type="button" value="Voir détail"/>
Visual Studio 2019 et C#	11	C# POO	<input type="button" value="Voir détail"/>
Cours UML	10	UML Cours	<input type="button" value="Voir détail"/>
Eclipse et Java	8	Java UML	<input type="button" value="Voir détail"/>
MCD exercices d'examen (sujets EDC BTS SIO)	8	MCD	<input type="button" value="Voir détail"/>
Exercices objet (sujets EDC BTS SIO)	8	POO	<input type="button" value="Voir détail"/>

Concernant le comptage du nombre de formations par playlists dans ***playlist.html.twig***, on utilise affiche directement la taille.

```
<strong>Nombre de formations disponibles</strong><br />
    {{ playlistformations|length }}
<br /><br />
```

Le compte rendu final du côté du front lors de l'affichage du détail d'une playlist.

<p><b>Bases de la programmation (C#)</b></p> <p><b>Catégories</b> C# POO</p> <p><b>Nombre de formations disponibles</b> 74</p> <p><b>Description</b> Exemples progressifs de programmes en procédural, événementiel et objet sous Visual Studio (version Entreprise 2017). Prérequis : aucun</p>	 <p>Bases de la programmation n°1 - procédural : premier exemple</p>  <p>Bases de la programmation n°2 - procédural : exercice1 (affichage)</p>  <p>Bases de la programmation n°3 - procédural : exercice2 (saisie)</p>  <p>Bases de la programmation n°4 - procédural : exercice3 (calculs)</p>
--	--

## Mission 2 : coder la partie back-office

### Tâche 1 : gérer les formations.

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.

Temps de travail estimé  
réel  
5 heures

Temps de travail  
10 heures

### Kanban de la tâche actuelle "In Progress"

The screenshot displays a Jira Kanban board with three columns: 'To do' (10 cards), 'In progress' (1 card), and 'Done' (3 cards). The 'In progress' column contains a card titled 'M2 - T1 : gérer les formations' with the subtitle 'mediatekformation#4 opened by Jlauth'. A sidebar on the right provides details for this card, including a comment from 'Jlauth' stating 'Permettre l'ajout, la modification et la suppression d'une formation.' and a 'Close issue' button.

Column	Card Title	Subtitle
To do	M2 - T2 : gérer les playlists	mediatekformation#5 opened by Jlauth
To do	M2 - T3 : gérer les catégories	mediatekformation#6 opened by Jlauth
To do	M2 - T4 : ajouter l'accès avec authentification	mediatekformation#7 opened by Jlauth
To do	M2 - T5 : gérer la sauvegarde et la restauration de la DB	mediatekformation#8 opened by Jlauth
To do	M3 - T1 : gérer les tests	mediatekformation#9 opened by Jlauth
To do	M3 - T2 : créer la documentation technique	mediatekformation#10 opened by Jlauth
To do	M3 - T3 : créer la documentation utilisateur	mediatekformation#11 opened by Jlauth
To do	M4 - T1 : déployer le site	mediatekformation#12 opened by Jlauth
To do	M4 - T2 : mettre en place le déploiement continu	
In progress	M2 - T1 : gérer les formations	mediatekformation#4 opened by Jlauth
Done	M1 - T1 : nettoyer le code	mediatekformation#1 opened by Jlauth
Done	M1 - T2 : respecter les bonnes pratiques de codage	mediatekformation#2 opened by Jlauth
Done	M1 - T3 : ajouter une fonctionnalité	mediatekformation#3 opened by Jlauth

## Process

Création en base de **src** du dossier **Form** et la classe **FormationType** dans celui-ci.  
Les use correspondants ont été importés et les données demandées ajoutées comme suit.

```
<?php

namespace App\Form;

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

/** Description of FormationType ...5 lines */
class FormationType extends AbstractType {
```

La fonction **buildForm()** permet via le paramètre **\$builder** d'ajouter des éléments au formulaire.

```
public function buildForm(FormBuilderInterface $builder, array $options): void {

    $builder
```

Le premier add correspond à un encadré de type **TextType**, une label et une option required à true afin que ce champ soit obligatoirement renseigné.

```
->add('title', TextType::class, [
    'label' => 'Intitulé de la nouvelle formation',
    'required' => true
])
```

Le second add correspond au choix d'une playlist en visant l'entité **Playlist** depuis **EntityType**, avec le choix du label sur le nom de la playlist, un choix multiple à false et required à true.

```
->add('playlist', EntityType::class, [
    'class' => Playlist::class,
    'choice_label' => 'name',
    'multiple' => false,
    'required' => true,
])
```

Le troisième élément correspond au choix d'une ou plusieurs catégories, choix non obligatoire afin de valider le formulaire ultérieurement.

```
->add('categories', EntityType::class, [
    'class' => Catégorie::class,
    'label' => 'Catégorie',
    'choice_label' => 'name',
    'multiple' => true,
    'required' => false
])
```

Le quatrième élément de type **DateType** permet d'afficher un widget date en **single\_text** (une seule entrée de type date), initialisée au jour même par défaut, requise afin de valider le formulaire.

```
->add('publishedAt', DateType::class, [
    'widget' => 'single_text',
    'data' => isset($options['data']) && $options['data']->getPublishedAt() != null ?
        $options['data']->getPublishedAt() : new DateTime('now'),
    'label' => 'Date',
    'required' => true
])
```

Les deux derniers éléments sont le renseignement d'une description non obligatoire ainsi qu'un bouton de type **SubmitType** permettant de déclencher l'action de validation du formulaire.

```
->add('description', TextareaType::class, [
    'label' => 'Description de la nouvelle formation',
    'required' => false
])
->add('submit', SubmitType::class, [
    'label' => 'Ajouter'
])
```

Nous définissons également la fonction **configureOptions()** afin que l'array par défaut soit celui de la classe **Formation**.

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

Une fois le formulaire configuré, création d'un dossier complémentaire **admin** en source de **Controller** ainsi qu'une nouvelle classe **AdminFormationsController** en y reprenant les mêmes



fonctions que pour la classe **FormationsController** afin de répondre à la demande de rendu identique avec la partie front, en prenant soin de modifier les routes des fonctions.

### **index()**

```
* @Route("/admin", name="admin. formations")
```

### **sort()**

```
* @Route("/admin/tri/{champ}/{ordre}/{table}", name="admin. formations. sort")
```

Concernant **findAllContain()**, on implémente à la première ligne de code une test afin de vérifier que le token csrf est valide. Si celui-ci l'est le code de la fonction est exécuté. En revanche, si le token n'est pas valide l'utilisateur est redirigé sur la page sans que le filtre de recherche ne soit initialisé. On prend également soin de modifier la route de cette fonction.

```
/**
 * @Route("/admin/formations/recherche/{champ}/{table}", name="admin. formations. findAllContain")
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
public function findAllContain($champ, Request $request, $table = ""): Response {
    if ($this->isCsrfTokenValid('filtre_' . $champ, $request->get('_token'))) {
        $valeur = $request->get("recherche");
        if ($table != "") {
            $formations = $this->formationRepository->findByContainValueTable($champ, $valeur, $table);
        } else {
            $formations = $this->formationRepository->findByContainValue($champ, $valeur);
        }
        $categories = $this->categorieRepository->findAll();
        return $this->render($this->pagesFormationsAdmin, [
            'formations' => $formations,
            'categories' => $categories,
            'valeur' => $valeur,
            'table' => $table
        ]);
    }
    return $this->redirectToRoute($this->redirectToAF);
}
```

Puis on écrit trois nouvelles fonctions, pour supprimer la formation sélectionnée.

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

Éditer la formation sélectionnée. Ici on crée la variable **\$formFormation** qui prend en valeur la création du formulaire depuis notre **FormationType** créé précédemment.

```
/**
 * @Route("/admin/formations/edit/{id}", name="admin.formation.edit")
 * @param Formation $formation
 * @param Request $request
 * @return Response
 */
public function edit(Formation $formation, Request $request): Response {
    $formFormation = $this->createForm(FormationType::class, $formation);
    $formFormation->handleRequest($request);
    if ($formFormation->isSubmitted() && $formFormation->isValid()) {
        $this->formationRepository->add($formation, true);
        return $this->redirectToRoute($this->redirectToAF);
    }
    return $this->render("admin/admin.formation.edit.html.twig", [
        'formation' => $formation,
        'formFormation' => $formFormation->createView()
    ]);
}
```

Ajouter une formation.

```
/**
 * @Route("/admin/formations/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 $this->redirectToRoute($this->redirectToAF);
    }
    return $this->render("admin/admin.formation.ajout.html.twig", [
        'formation' => $formation,
        'formFormation' => $formFormation->createView()
    ]);
}
```

Concernant le tri des catégories dans le champ de recherche, les catégories ne sont plus prises en compte. Malgré de nombreux essais afin d'intégrer un token aux catégories, la solution trouvée consiste à créer une fonction complémentaire **findAllContainCategories()** permettant d'effectuer une recherche via le champ catégories uniquement et donc sans csrf.

```

/**
 * @Route("/admin/formations/rechercher/{champ}/{table}", name="admin.formations.findallcontaincategories")
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
public function findAllContainCategories($champ, Request $request, $table): Response {
    $valeur = $request->get("recherche");
    $formations = $this->formationRepository->findByContainValueTable($champ, $valeur, $table);
    $categories = $this->categorieRepository->findAll();
    return $this->render($this->pagesFormationsAdmin, [
        'formations' => $formations,
        'categories' => $categories,
        'valeur' => $valeur,
        'table' => $table
    ]);
}

```

Nous créons à la suite et en source du dossier **templates** deux nouvelles pages.

La page **baseadmin.html.twig** héritant de **base.html.twig**.

```

{% extends "base.html.twig" %}

{% block title %}{% endblock %}
{% block stylesheets %}{% endblock %}
{% block top %}
    <div class="container">
        <div class="text-end">
            <a href="{{ path('logout') }}">Déconnexion</a>
        </div>
        <!-- titre -->
        <div class="text-center">
            <h1>
                <strong>Page d'administration de MediaTek86</strong>
            </h1>
        </div>
        <!-- menu -->
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{{ path('admin.formations') }}">Formations</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ path('admin.playlists') }}">Playlists</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ path('admin.categories') }}">Catégories</a>
                    </li>
                </ul>
            </div>
        </nav>
    </div>
{% endblock %}
{% block body %}{% endblock %}
{% block javascripts %}{% endblock %}

```

Ainsi que `_admin.formation.form.html.twig` afin d'initialiser le formulaire et qui hérite de `baseadmin.html`.

```
{% extends "base.html.twig" %}
{% block body %}
    {% form_theme formFormation 'bootstrap_4_layout.html.twig' %}
    {{ form_start(formFormation) }}
    {{ form_end(formFormation) }}
{% endblock %}
```

Un thème bootstrap 4 a été utilisé afin d'uniformiser le rendu du et des futurs formulaires. Ce nouveau `form_themes` a été formulé dans le package `twig.yaml`.

```
twig:
    default_path: '%kernel.project_dir%/templates'

when@test:
    twig:
        strict_variables: true
        form_themes: ['bootstrap_4_layout.html.twig']
```

Nous créons alors un nouveau dossier `admin` en base de template ainsi qu'une nouvelle page `admin.formations.html.twig`. La structure est la même que dans `formations.html.twig`, c'est pourquoi nous allons voir uniquement les changements opérés.

Les routes ont été modifiées pour le tri des formations, playlists et dates.

```
Formations enregistrées<br />
<a href="{{ path('admin.formations.sort', {champ:'title', ordre:'ASC'}) }}"
Playlist<br />
<a href="{{ path('admin.formations.sort', {table:'playlist', champ:'name', ordre:'ASC'}) }}"
Date<br />
<a href="{{ path('admin.formations.sort', {champ:'publishedAt', ordre:'ASC'}) }}"
```

Les CSRF token sont présents pour les inputs des formations et playlists en relation avec le champ de recherche identifié dans la méthode POST.

```
<form class="form-inline mt-1" method="POST" action="{{ path('admin.formations.findallcontain', {champ:'title'}) }}">
<input type="hidden" name="_token" value="{{ csrf_token('filtre_title') }}">
<form class="form-inline mt-1" method="POST" action="{{ path('admin.formations.findallcontain', {champ:'name'}) }}">
<input type="hidden" name="_token" value="{{ csrf_token('filtre_name') }}">
```

Ainsi que la recherche sur catégories liées à sa fonction spécifique dans `AdminFormationsController`.

```
Catégories<br/>
<form class="form-inline mt-1" method="POST" action="{{ path('admin.formations.findallcontaincategories',
```

Sont également codés les nouveaux boutons permettant les actions demandées.

L'ajout d'une nouvelle formation.

```
<th class="text-center align-top" scope="col">
    <a href="{{ path('admin.formation.ajout') }}" class="btn btn-primary btn-lg">Ajouter</a>
</th>
```

La modification d'une formation existante.

```
<td class="text-end">
    <a href="{{ path('admin.formation.edit', {id:formation.id}) }}" class="btn btn-secondary"
        onclick="return confirm('Modifier {{formation.title}} ?');">Modifier</a>
</td>
```

La suppression d'une formation existante.

```
<td class="text-end">
    <a href="{{ path('admin.formation.suppr', {id:formation.id}) }}" class="btn btn-danger"
        onclick="return confirm('Supprimer {{formation.title}} ?');">Supprimer</a>
</td>
```

En lien avec ces nouvelles actions disponibles, nous créons les pages correspondantes. Ces deux pages ont en commun l'include **\_admin.formation.form.html.twig** qui initialise le formulaire ainsi que le thème.

La page d'ajout d'une nouvelle formation.

```
{% extends "baseadmin.html.twig" %}
{% block title %}{% endblock %}
{% block body %}
    <br/>
    <h2 class='text-info'>Ajout d'une nouvelle formation</h2>
    {{ include ('_admin.formation.form.html.twig') }}
{% endblock %}
```

La page d'édition d'une formation.

```
{% extends "baseadmin.html.twig" %}
{% block body %}
    <br/>
    <h2>Détail de la formation</h2>
    <br/>
    {{ include ('_admin.formation.form.html.twig') }}
{% endblock %}
```

Concernant la sécurité du code ajouté, nous avons vu l'utilisation des tokens pour contrer le CSRF. Les requêtes paramétrées afin d'éviter les injections SQL sont présentes dans la classe **FormationRepository**, respectivement dans les fonctions **findByContainValue()** et **findByContainValueTable()**.

```
return $this->createQueryBuilder('f')
    ->where('f.' . $champ . ' LIKE :valeur')
```

Certains contrôles de saisie ont été présentés lors du codage de **FormType**. Afin renforcer leur sécurité, l'entité **Formation** a reçu plusieurs assertions qui permettent également dans le cas du titre une customisation du champ.

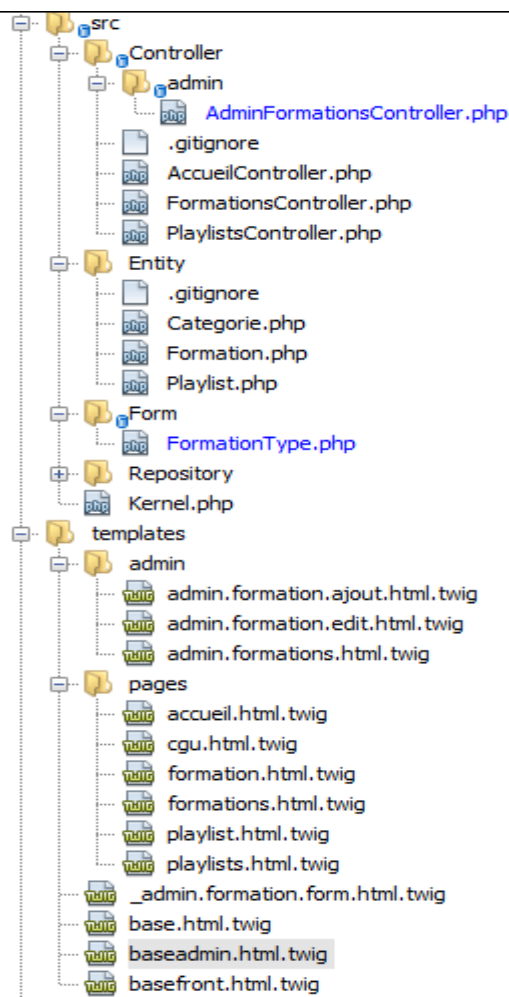
```
/**
 * @var datetime $publishedAt
 *
 * @ORM\Column(type="datetime", nullable=false)
 * @Assert\NotBlank()
 * @Assert\LessThanOrEqual("now")
 */
private $publishedAt;

/**
 * @var string $title
 *
 * @ORM\Column(name="title", type="string", nullable=false)
 * @Assert\NotBlank()
 * @Assert\Length(min=4, max=30, minMessage="Minimum {{ limit }} caractères", maxMessage="Maximum {{ limit }} caractères.")
 */
private $title;

/**
 * @var text $description
 *
 * @ORM\Column(name="description", type="text", nullable=true)
 * @Assert\Length(min=6, max=255)
 */
private $description;
```

(suite page 47)

Avant de passer au rendu utilisateur, voici la nouvelle arborescence du projet.

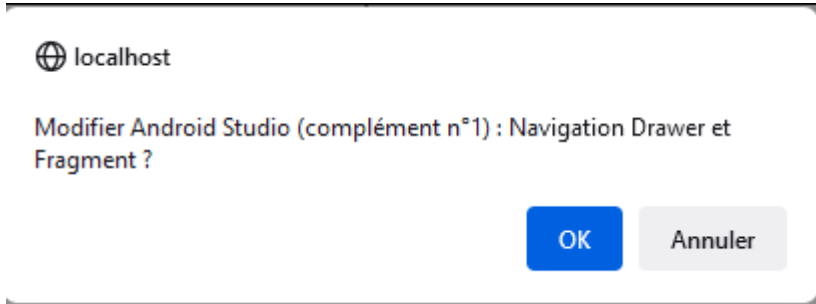


## Rendu utilisateur

### Page d'administration de MediaTek86

Formations		Playlists			
Formations enregistrées		Playlist	Catégories	Date	Ajouter
<div>&lt; &gt;</div> <div><input type="text"/> Filtrer</div>		<div>&lt; &gt;</div> <div><input type="text"/> filtrer</div>	<div>▼</div>	<div>&lt; &gt;</div>	
Android Studio (complément n°1) : Navigation Drawer et Fragment		Compléments Android (programmation mobile)	Android	09/07/2018	<div>Modifier</div> <div>Supprimer</div>
Android Studio (complément n°10) : Ajout icone dans menu		Compléments Android (programmation mobile)	Android	18/10/2018	<div>Modifier</div> <div>Supprimer</div>
Android Studio (complément n°11) : Transformer une image en texte		Compléments Android (programmation mobile)	Android	18/12/2018	<div>Modifier</div> <div>Supprimer</div>
Android Studio (complément n°12) : Positionner texte sur photo		Compléments Android (programmation mobile)	Android	17/09/2019	<div>Modifier</div> <div>Supprimer</div>
Android Studio (complément n°13) : Permissions		Compléments Android (programmation mobile)	Android	29/09/2019	<div>Modifier</div> <div>Supprimer</div>

Message de confirmation avant la modification d'une formation.



Page de modification d'une formation.

### Détail de la formation

Intitulé de la nouvelle formation  
Eclipse n°8 : Déploiement

Playlist  
Eclipse et Java

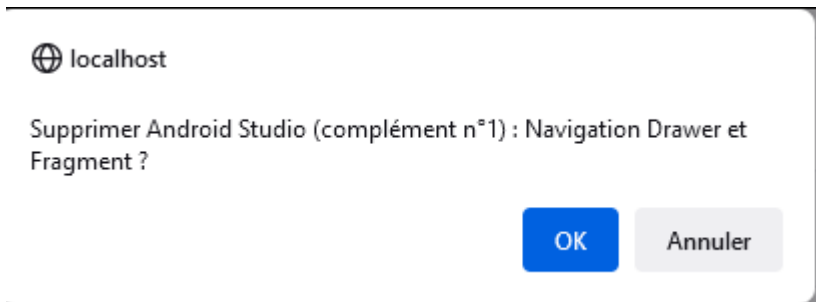
Catégorie  
Android  
C#  
Cours  
Java

Date  
04 / 01 / 2021

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

Ajouter

Message de confirmation avant la suppression d'une formation.



Page d'ajout d'une nouvelle formation avec test sur la taille du titre.



## Ajout d'une nouvelle formation

Intitulé de la nouvelle formation

Minimum 4 caractères

Tes



Playlist

Eclipse et Java

Catégorie

Android

C#

Cours

Java

Date

10 / 12 / 2022

Description de la nouvelle formation

Ajouter

Titre de la formation non indiqué.

Intitulé de la nouvelle formation

Catégorie

Veuillez compléter ce champ.

Java

Date non valide.

Date

This value should be less than or equal to Nov 16, 2022, 8:51 PM.

19 / 11 / 2022



Les données sont bien enregistrées dans la page d'administration des formations.

## Page d'administration de MediaTek86

Formations Playlists

Formations enregistrées



Filtrer

Playlist



filtrer

Catégories



Date



Ajouter

Formation Symphony

Cours de programmation objet

POO

Cours

16/11/2022

Modifier

Supprimer

Eclipse n°8 : Déploiement

Eclipse et Java

Java

04/01/2021

Modifier

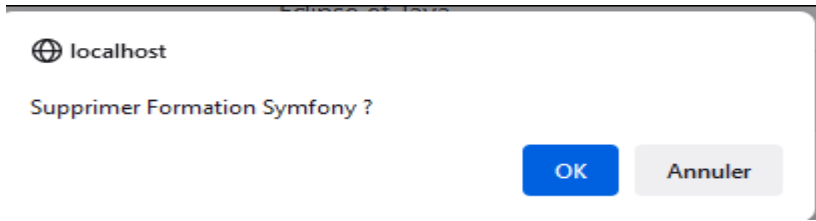
Supprimer

Ainsi qu'en base de données.

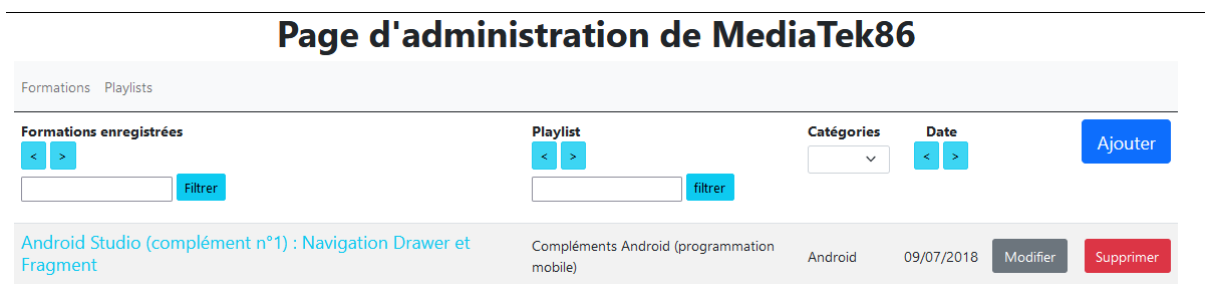
Options

			id	published_at	1	title	description	video_id	playlist_id
<input type="checkbox"/>	Éditer	Copier	Supprimer	255	2022-11-16 00:00:00	Formation Symphony	Ceci est une description complète.	NULL	27
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	2021-01-04 17:00:12	Eclipse n°8 : Déploiement	Exécution de l'application en dehors de l'IDE, en ...	Z4yTTXka958	1

Exécution de la suppression.



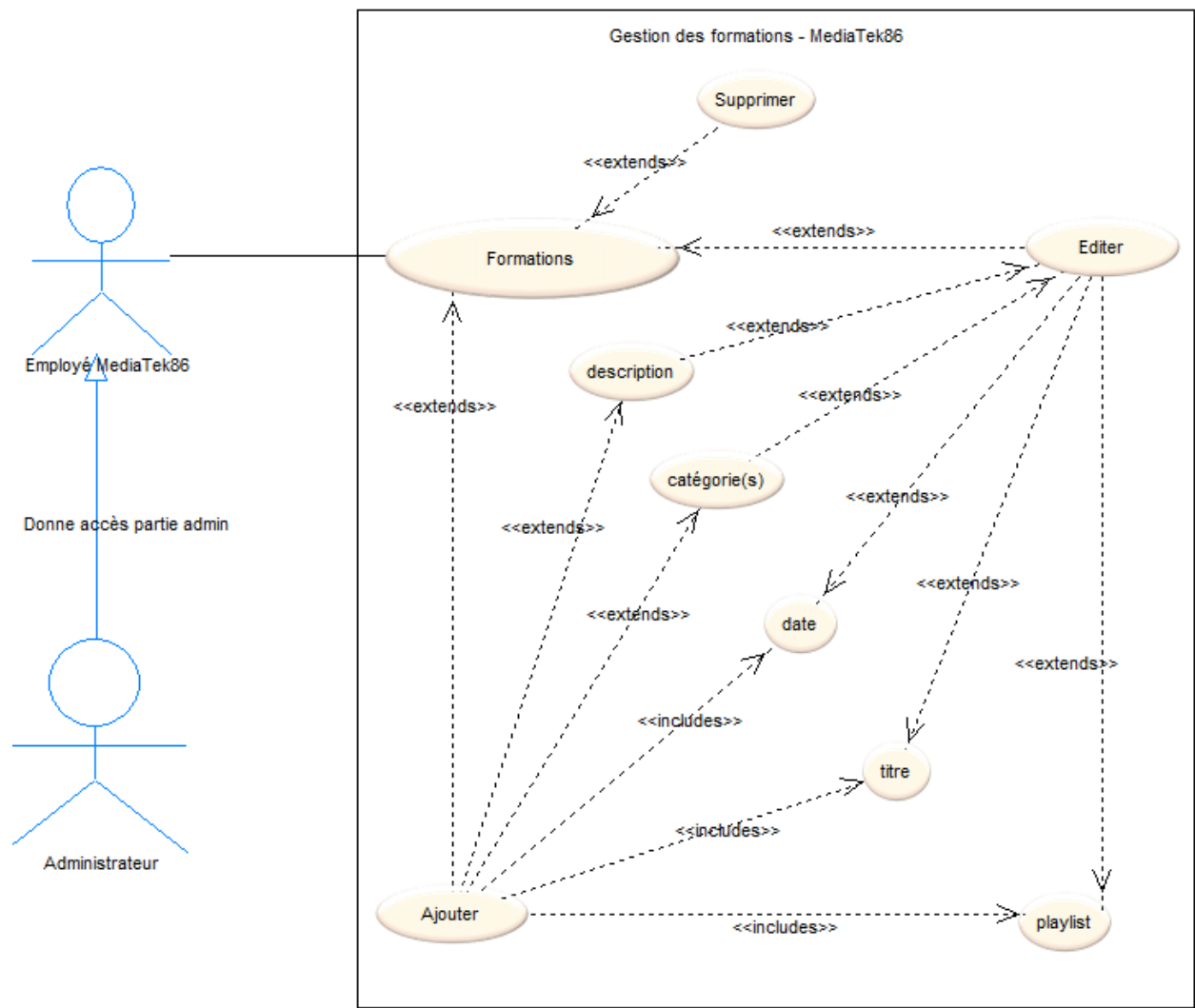
La page du site actualisée avec la ligne supprimée.



Ainsi qu'en base de données.

Options					
	id	published_at	title	description	video_id
<input type="checkbox"/>	1	2021-01-04 17:00:12	Eclipse n°8 : Déploiement	Exécution de l'application en dehors de l'IDE, en ...	Z4yTTXka958

Diagramme de cas d'utilisation



## Tâche 2 : gérer les playlists.

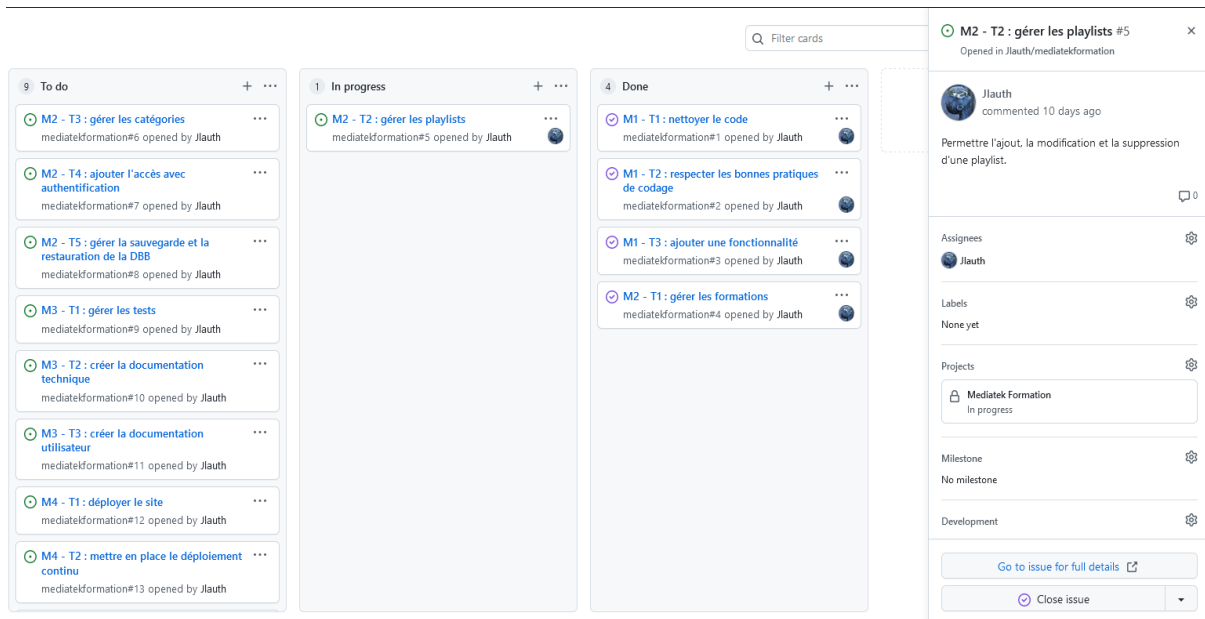
Une page doit permettre de lister les playlists et, pour chaque playlist, afficher un bouton permettant de la supprimer (après confirmation) et un bouton permettant de la modifier.

Temps de travail estimé  
réel  
5 heures  
heures

Temps de travail

10

Kanban de la tâche actuelle "In Progress"



## Process

Création en source de **Form** des classes **PlaylistType** et **PlaylistTypeAdd**.

Les use correspondants ont été importés comme suit.

```
<?php

namespace App\Form;

use App\Entity\Formation;
use App\Entity\Playlist;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
```

La construction d'un premier formulaire, **PlaylistType**, est dédié uniquement à l'édition d'une playlist.

```
class PlaylistType extends AbstractType {

    public function buildForm(FormBuilderInterface $builder, array $options): void {
        $builder
            ->add('name', TextType::class, [
                'label' => 'Intitulé de la playlist',
                'required' => true
            ])
            ->add('formations', EntityType::class, [
                'class' => Formation::class,
                'label' => 'Formation(s) liée(s) à la playlist',
                'choice_label' => 'title',
                'multiple' => true,
                'disabled' => true
            ])
            ->add('description', TextareaType::class, [
                'label' => 'Description de la nouvelle playlist',
                'required' => false
            ])
            ->add('submit', SubmitType::class, [
                'label' => 'Ajouter'
            ])
        ;
    }
}
```

Ce second formulaire, **PlaylistTypeAdd**, permet d'afficher la création d'une playlist avec seulement les trois champs requis.

```
class PlaylistTypeAdd extends AbstractType {

    public function buildForm(FormBuilderInterface $builder, array $options): void {
        $builder
            ->add('name', TextType::class, [
                'label' => 'Intitulé de la playlist',
                'required' => true
            ])
            ->add('description', TextareaType::class, [
                'label' => 'Description de la nouvelle playlist',
                'required' => false
            ])
            ->add('submit', SubmitType::class, [
                'label' => 'Ajouter'
            ])
        ;
    }
}
```

Pour ces deux nouvelles classes, la fonction ***configureOptions()*** est la même.

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

Ces deux classes reprennent l'option **required** à true ou false en fonction de la demande.

Création de la classe **AdminPlaylistsController** en y reprenant les mêmes fonctions que pour la classe **PlaylistsController** afin de répondre à la demande de rendu identique avec la partie front, en prenant soin de modifier les routes des fonctions.

#### ***index()***

```
/**
 * @Route("/admin/playlists", name="admin.playlists")
 */
```

#### ***sort()***

```
/**
 * @Route("/admin/playlists/tri/{champ}/{ordre}", name="admin.playlists.sort")
 */
```

#### ***findAllContain()***

```
/**
 * @Route("/admin/playlists/recherche/{champ}", name="admin.playlists.findAllContain")
 */
```

#### ***findAllContainCategories()***

```
/**
 * @Route("/admin/playlists/recherche/{champ}/{table}", name="admin.playlists.findAllContainCategories")
 */
```

La création des trois nouvelles fonctions, afin de créer une nouvelle playlist.

```

/**
 * @Route("/admin/playlists/ajout", name="admin.playlist.ajout")
 * @param Request $request
 * @return Response
 */
public function ajout(Request $request): Response {
    $playlist = new Playlist();
    $formPlaylistAdd = $this->createForm(PlaylistTypeAdd::class, $playlist);
    $formPlaylistAdd->handleRequest($request);
    if ($formPlaylistAdd->isSubmitted() && $formPlaylistAdd->isValid()) {
        $this->playlistRepository->add($playlist, true);
        return $this->redirectToRoute($this->redirectToAP);
    }
    return $this->render("admin/admin.playlist.ajout.html.twig", [
        'playlist' => $playlist,
        'formPlaylistAdd' => $formPlaylistAdd->createView()
    ]);
}

```

Editer une playlist existante.

```

/**
 * @Route("/admin/playlists/edit/{id}", name="admin.playlist.edit")
 * @param Playlist $playlist
 * @param Request $request
 * @return Response
 */
public function edit(Playlist $playlist, Request $request): Response {
    $formPlaylist = $this->createForm(PlaylistType::class, $playlist);
    $formPlaylist->handleRequest($request);
    if ($formPlaylist->isSubmitted() && $formPlaylist->isValid()) {
        $this->playlistRepository->add($playlist, true);
        return $this->redirectToRoute($this->redirectToAP);
    }
    return $this->render("admin/admin.playlist.edit.html.twig", [
        'playlist' => $playlist,
        'formPlaylist' => $formPlaylist->createView()
    ]);
}

```

Et supprimer une playlist.

```
/**
 * @Route("/admin/playlists/suppr/{id}", name="admin.playlist.suppr")
 * @param Formation $playlist
 * @return Response
 */
public function suppr(Playlist $playlist): Response {
    $this->playlistRepository->remove($playlist, true);
    return $this->redirectToRoute($this->redirectToAP);
}
```

### Configuration côté twig

**\_admin.playlists.form.html.twig** n'étant pas nécessaire dans le cas des playlists étant donné que celles-ci n'ont pas le même formulaire. On crée ainsi directement les trois pages **admin.playlists** avec **admin** en source.

Les routes ont été modifiées pour le tri des playlists et du nombre de formation(s).

```
Playlist<br />
<a href="{{ path('admin.playlists.sort', {champ:'name', ordre:'ASC'}) }}"

Nombre de formations<br />
<a href="{{ path('admin.playlists.sortonnbformation', {ordre:'ASC'}) }}"
```

Les CSRF token sont présents pour les inputs des playlists en relation avec le champ de recherche identifié dans la méthode POST.

```
<form class="form-inline mt-1" method="POST" action="{{ path('admin.playlists.findallcontain', {champ:'name'}) }}">
<input type="hidden" name="_token" value="{{ csrf_token('filtre_name') }}">
```

Ainsi que le `findallcontain` spécifique pour les catégories.

```
<form class="form-inline mt-1" method="POST" action="{{ path('admin.playlists.findallcontaincategories', (c
```

Le codage des boutons afin de répondre à la demande d'ajout d'une nouvelle playlist.

```
<th class="text-center align-top" scope="col">
    <a href="{{ path('admin.playlist.ajout') }}" class="btn btn-primary btn-lg">Ajouter</a>
</th>
```

De modification.

```
<td class="text-end">
    <a href="{{ path('admin.playlist.edit', {id:playlists[k].id}) }}" class="btn btn-secondary"
        onclick="return confirm('Modifier {{playlists[k].name}} ?')">Modifier</a>
</td>
```



Et de suppression. Comme demandé, on effectue un test afin de vérifier si la playlist sélectionnée ne contient pas de formation(s). Si celle-ci est vide, la suppression est possible. Petit ajout afin que le message soit cohérent en terme de syntaxe si la playlist contient une formation ou plusieurs.

```
{% if playlists[k].formations|length == 1 %}
    <a href="{{ path('admin.playlist.suppr', {id:playlists[k].id}) }}" class="btn btn-danger"
        onclick="alert('Impossible de supprimer une playlist contenant {{playlists[k].formations|length}} formation');
        return false;">Supprimer</a>
{% elseif playlists[k].formations|length > 1 %}
    <a href="{{ path('admin.playlist.suppr', {id:playlists[k].id}) }}" class="btn btn-danger"
        onclick="alert('Impossible de supprimer une playlist contenant {{playlists[k].formations|length}} formations');
        return false;">Supprimer</a>
{% else %}
    <a href="{{ path('admin.playlist.suppr', {id:playlists[k].id}) }}" class="btn btn-danger"
        onclick="return confirm('Supprimer {{playlists[k].name}} ?')">Supprimer</a>
{% endif %}
```

Afin d'accéder à l'affichage de ces deux nouvelles pages, on crée dans le dossier **admin**, **admin.playlist.edit.html.twig** afin d'éditer la playlist sélectionnée et en incluant le thème bootstrap présenté lors de la tâche précédente.

```
{% extends "baseadmin.html.twig" %}
{% block title %}{% endblock %}
{% block body %}
    {% form_theme formPlaylist 'bootstrap_4_layout.html.twig' %}
    <br/>
    <h2>Détail de la playlist</h2>
    <br/>
    {{ form_start(formPlaylist) }}
    {{ form_end(formPlaylist) }}
{% endblock %}
```

Puis la page **admin.playlist.ajout.html.twig**

```
{% extends "baseadmin.html.twig" %}
{% block title %}{% endblock %}
{% block body %}
    {% form_theme formPlaylistAdd 'bootstrap_4_layout.html.twig' %}
    <br/>
    <h2 class='text-info'>Ajout d'une nouvelle playlist</h2>
    {{ form_start(formPlaylistAdd) }}
    {{ form_end(formPlaylistAdd) }}
{% endblock %}
```

En termes de sécurité, les requêtes paramétrées afin d'éviter les injections SQL sont présentes dans la classe **PlaylistRepository**, respectivement dans les fonctions **findByContainValue()** et **findByContainValueTable()**.

```
return $this->createQueryBuilder('f')
    ->where('f.' . $champ . ' LIKE :valeur')
```

Nous trouvons également certains contrôles de saisie complémentaires dans l'entité **Playlist**, dont l'utilisation du validator `UniqueEntity` afin de vérifier qu'une playlist n'a qu'une seule fois le même nom.

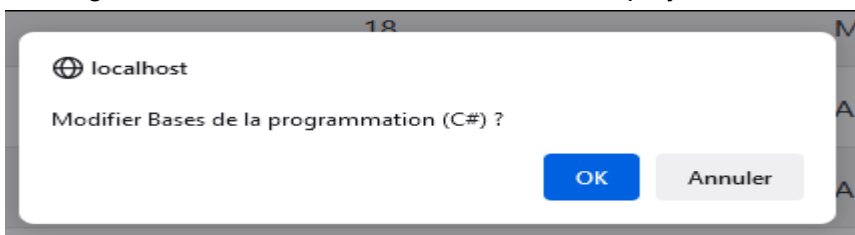
```
/**
 * @ORM\Table()
 * @ORM\Entity(repositoryClass=PlaylistRepository::class)
 * @UniqueEntity(fields="name", message="Ce nom de playlist existe déjà")
 */
class Playlist
{
    /**
     * @var integer $id
     *
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(name="id", type="integer")
     */
    private $id;

    /**
     * @var string $name
     *
     * @ORM\Column(name="name", type="string", nullable=false, unique=true)
     * @Assert\Length(min=4, max=20, minMessage="Minimum {{ limit }} caractères", maxMessage="Maximum {{ limit }} caractères. C'est déjà pas mal non?")
     */
    private $name;
```

## Rendu utilisateur

Page d'administration de MediaTek86			
Formations Playlists Catégories			
<b>Playlist</b> <input type="text"/> <input type="button" value="filtrer"/>	<b>Nombre de formations</b> <input type="text"/> <input type="text"/>	<b>Catégories</b> <input type="text"/>	<input type="button" value="Ajouter"/>
Bases de la programmation (C#)	74	C# POO	<input type="button" value="Modifier"/> <input type="button" value="Supprimer"/>
Programmation sous Python	19	Python POO	<input type="button" value="Modifier"/> <input type="button" value="Supprimer"/>

Message de confirmation avant de modifier une playlist.



Page de modification d'une playlist.

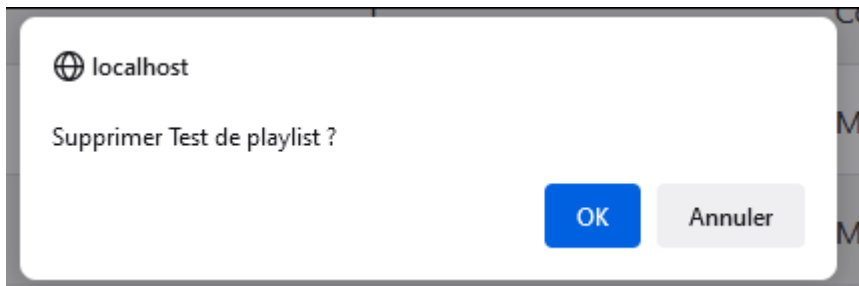
### Détail de la playlist

Intitulé de la playlist

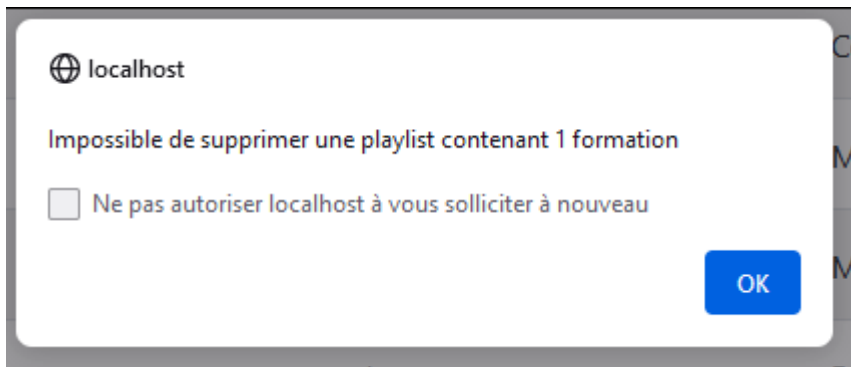
Formation(s) liée(s) à la playlist

Description de la nouvelle playlist  
  
 Prérequis : aucun

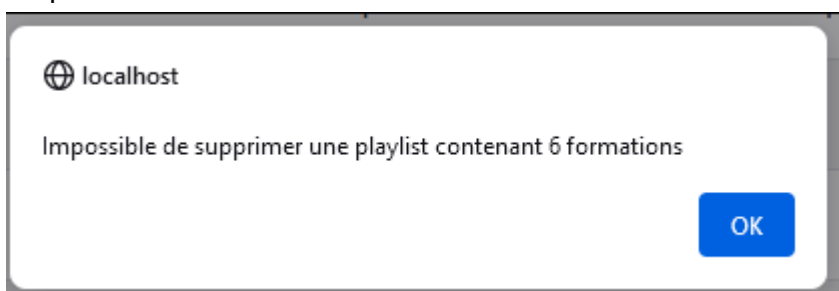
Message de confirmation avant la suppression d'une playlist.



Message d'erreur si la playlist contient une formation.



Ou plusieurs formations.



La page d'ajout d'une playlist.

---

### Ajout d'une nouvelle playlist

Intitulé de la playlist

Description de la nouvelle playlist

Ajouter

Le message d'erreur si le nom de playlist n'est pas unique.

## Ajout d'une nouvelle playlist

Intitulé de la playlist

Ce nom de playlist existe déjà

Bases de la programmation (C#)

Le message d'erreur si le champ playlist est vide.

## Ajout d'une nouvelle playlist

Intitulé de la playlist

Description de la nouvelle playlist  
Veuillez compléter ce champ.

Ajouter

Ou dans le cas où la taille de l'intitulé de la playlist ne correspond pas aux critères.

## Ajout d'une nouvelle playlist

Intitulé de la playlist

Minimum 4 caractères

Tes

## Ajout d'une nouvelle playlist ,

Intitulé de la playlist

Maximum 30 caractères.







Test

La création d'une nouvelle playlist est bien présente dans le front.

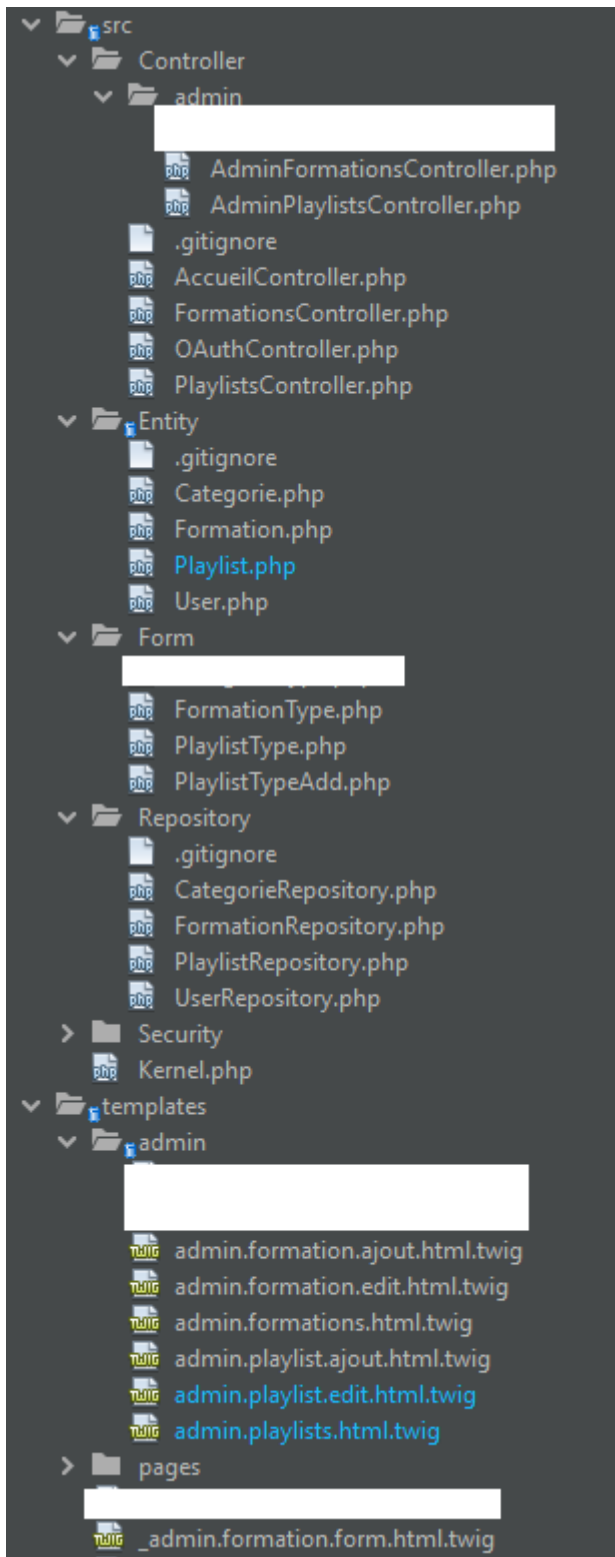
Playlist	Nombre de formations	Catégories	Ajouter
<div><div>&lt; &gt;</div><div><input type="text"/></div><div>filtrer</div></div>	<div><div>&lt; &gt;</div></div>	<div><div></div></div>	
Test de playlist	0		<div>Modifier Supprimer</div>

Ainsi qu'en base de données.

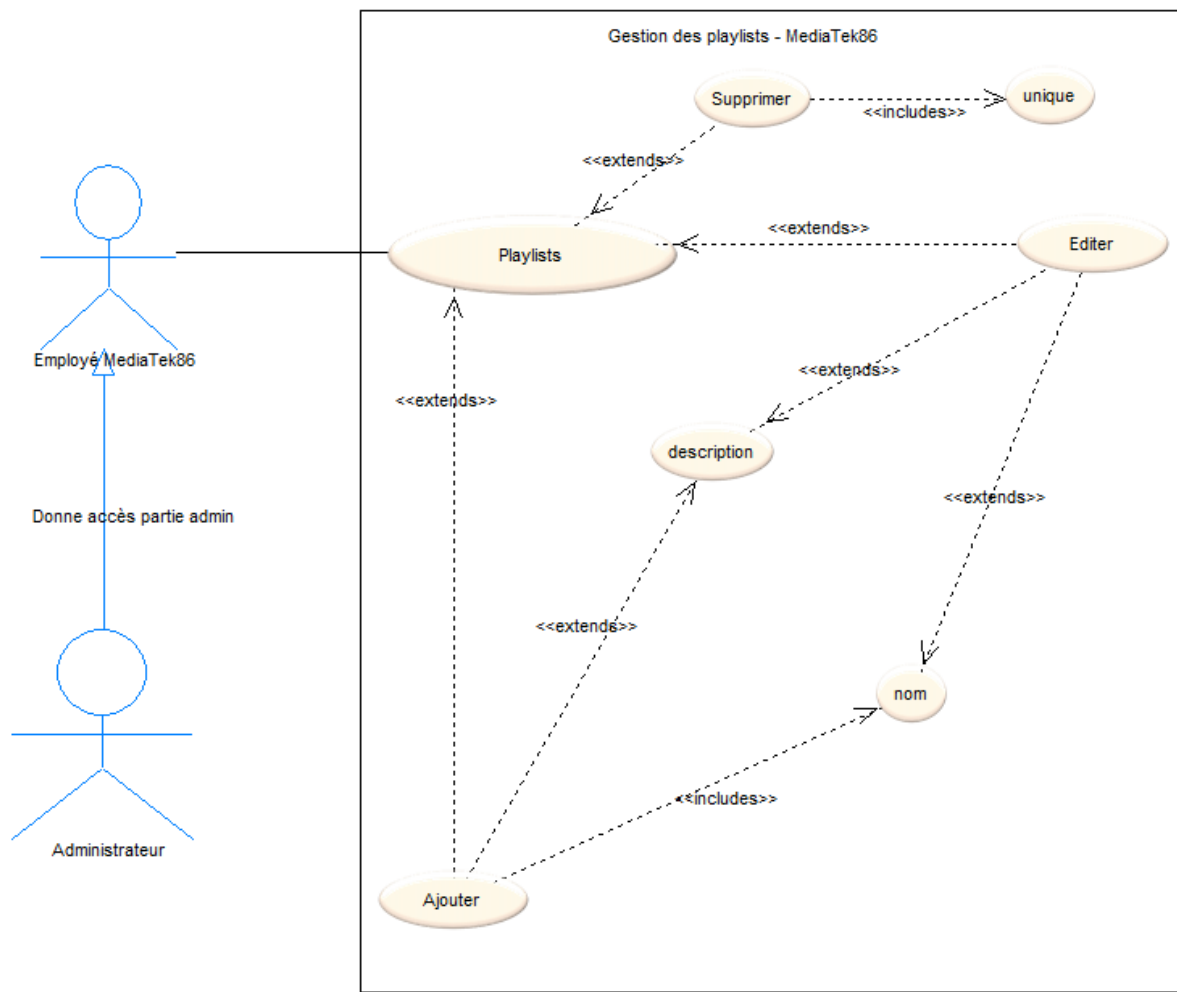
Options

			id	name	description	
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	35	Test	NULL
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	32	Test de playlist	Je suis la description du test d'ajout de playlist...

## Arborescence du projet



## Diagramme de cas d'utilisation



### Tâche 3 : gérer les catégories.

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.

Temps de travail estimé  
réel  
3 heures

Temps de travail  
5 heures

### Kanban de la tâche actuelle "In Progress"

The screenshot displays a Jira Kanban board with three columns: 'To do' (8 items), 'In progress' (1 item), and 'Done' (5 items). The 'In progress' column contains the task 'M2 - T3 : gérer les catégories' (mediatékformation#6). A detailed view of this task is shown on the right, indicating it was opened in Jira by 'Jlauth' and has a comment from 'Jlauth' stating 'Permettre l'ajout et la suppression d'une catégorie.' The task is currently in the 'In progress' state, with no assignees, labels, or projects. The 'To do' column lists tasks such as 'M2 - T4 : ajouter l'accès avec authentification' and 'M3 - T1 : gérer les tests'. The 'Done' column lists tasks like 'M1 - T1 : nettoyer le code' and 'M2 - T1 : gérer les formations'.

## Process

Création dans le dossier **Form** de la classe **CategorieType**. Ici seul le champ permettant de renseigner le nom d'une catégorie est demandé et présent.

```
<?php

namespace App\Form;

use App\Entity\Categorie;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

/**
 * Description of CategorieType
 *
 * @author Jean
 */
class CategorieType extends AbstractType {

    public function buildForm(FormBuilderInterface $builder, array $options) {

        $builder
            ->add('name', TextType::class, [
                'label' => 'Intitulé de la catégorie',
                'required' => true
            ])
            ->add('submit', SubmitType::class, [
                'label' => 'Ajouter'
            ])
        ;
    }

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



Création de **AdminCategoriesController** dans **Controller\admin**.

```
<?php

namespace App\Controller\admin;

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

/**
 * Description of AdminCategoriesController
 *
 * @author Jean
 */
class AdminCategoriesController extends AbstractController {

    /**
     *
     * @var type String
     */
    private $pageCategoriesAdmin = "admin/admin.categories.html.twig";

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

    /**
     *
     * @param CategorieRepository $categorieRepository
     */
    public function __construct(CategorieRepository $categorieRepository) {
        $this->categorieRepository = $categorieRepository;
    }
}
```

Les fonctions **index()** et **sort()** sont raccourcies par rapport aux fonctions des deux autres classes créées précédemment dans **admin**, étant donné que l'on ne veut qu'afficher la liste des catégories et effectuer un tri unique sur leur nom. Les routes sont également créées afin de correspondre aux besoins futurs.

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

/**
 * @Route("/admin/categories/tri/{champ}/{ordre}", name="admin.categories.sort")
 * @return Response
 */
public function sort($champ, $ordre): Response{
    $categories = $this->categorieRepository->findAllOrderBy($champ, $ordre);
    return $this->render($this->pageCategoriesAdmin, [
        'categories' => $categories
    ]);
}
```

La création de la fonction permettant la suppression d'une catégorie en fonction de son id.

```
/**
 * @Route("/admin/categories/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');
}
```

(suite page 66)

La fonction permettant l'ajout d'une nouvelle catégorie, faisant appel à **CategorieType** afin de savoir comment créer le formulaire requis.

```
/**
 * @Route("/admin/categories/ajout", name="admin.categorie.ajout")
 * @param Request $request
 * @return Response
 */
public function ajout(Request $request): Response{
    $categorie = new Categorie();
    $formCategorie = $this->createForm(CategorieType::class, $categorie);
    $formCategorie->handleRequest($request);
    if($formCategorie->isSubmitted() && $formCategorie->isValid()){
        $this->categorieRepository->add($categorie, true);
        return $this->redirectToRoute('admin.categories');
    }
    return $this->render("admin/admin.categorie.ajout.html.twig", [
        'categorie' => $categorie,
        'formCategorie' => $formCategorie->createView()
    ]);
}
```

Du côté **templates** on crée à sa source la page **\_admin.categorie.form.html.twig**. Celle-ci initialise le thème bootstrap externe ainsi que le formulaire codé dans le controller.

```
{% extends "base.html.twig" %}
{% block body %}
    {% form_theme formCategorie 'bootstrap_4_layout.html.twig' %}
    {{ form_start(formCategorie) }}
    {{ form_end(formCategorie) }}
{% endblock %}
```

Dans **templates\admin**, écriture de la page **admin.categories.html.twig** avec un champ de tri des catégories et un bouton permettant d'ajouter une nouvelle catégorie.

```
{% extends "baseadmin.html.twig" %}
{% block body %}
    <table class="table table-striped" description='Page des catégories'>
        <thead>
            <tr>
                <th class="text-left align-middle" scope="col">
                    Catégories<br />
                    <a href="{{ path('admin.categories.sort', {champ:'name', ordre:'ASC'}) }}" class="btn btn-
                    <a href="{{ path('admin.categories.sort', {champ:'name', ordre:'DESC'}) }}" class="btn btn-
                </th>
                <th class="text-end align-top" scope="col">
                    <a href="{{ path('admin.categorie.ajout') }}" class="btn btn-primary btn-lg">Ajouter</a>
                </th>
            </tr>
        </thead>
        <tbody>
```

L'affichage unique des catégories dans le body.

```
<tbody>
  {% for k in 0..categories|length-1 %}
    <tr class="align-middle">
      <td>
        <h5 class="text-info">
          {{ categories[k].name }}
        </h5>
      </td>
    </tr>
  {% endfor %}
</tbody>
```

Et la configuration de suppression d'une catégorie, impossible dans le cas où celle-ci contiendrait une ou plusieurs formations.

```
<td class="text-end">
  {% if categories[k].formations|length == 1 %}
    <a href="{{ path('admin.categorie.suppr', (id:categories[k].id)) }}" class="btn btn-danger"
      onclick="alert('Impossible de supprimer une catégorie contenant {{categories[k].formations|length}} formation');
      return false;">Supprimer</a>
  {% elseif categories[k].formations|length > 1 %}
    <a href="{{ path('admin.categorie.suppr', (id:categories[k].id)) }}" class="btn btn-danger"
      onclick="alert('Impossible de supprimer une catégorie contenant {{categories[k].formations|length}} formations'); return false;">Supprimer</a>
  {% else %}
    <a href="{{ path('admin.categorie.suppr', (id:categories[k].id)) }}" class="btn btn-danger"
      onclick="return confirm('Supprimer {{categories[k].name}} ?');">Supprimer</a>
  {% endif %}
</td>
```

Pour terminer, la page **admin.categorie.ajout.html.twig**. On aurait également pu se passer d'utiliser la passerelle d'accès **\_admin.categorie.form.html.twig**.

```
{% extends "baseadmin.html.twig" %}
{% block title %}{% endblock %}
{% block body %}
  <br/>
  <h2 class="text-info">Ajout d'une nouvelle categorie</h2>
  {{ include ('_admin.categorie.form.html.twig') }}
{% endblock %}
```

## Les différents rendus

La page d'administration des catégories.

Catégories		Ajouter
< >		
Android		Supprimer
C#		Supprimer
Cours		Supprimer
Java		Supprimer
MCD		Supprimer
POO		Supprimer
Python		Supprimer
SQL		Supprimer
UML		Supprimer

La page d'ajout d'une nouvelle catégorie.

---

## Ajout d'une nouvelle categorie

Intitulé de la catégorie

Ajouter

Le refus d'ajout d'une catégorie portant un nom existant déjà en base de données.

---

## Ajout d'une nouvelle categorie

Intitulé de la catégorie

Ce nom de catégorie existe déjà

Android

Ajouter

Ainsi que la notification du renseignement d'un nom de catégorie indispensable.

---

## Ajout d'une nouvelle categorie

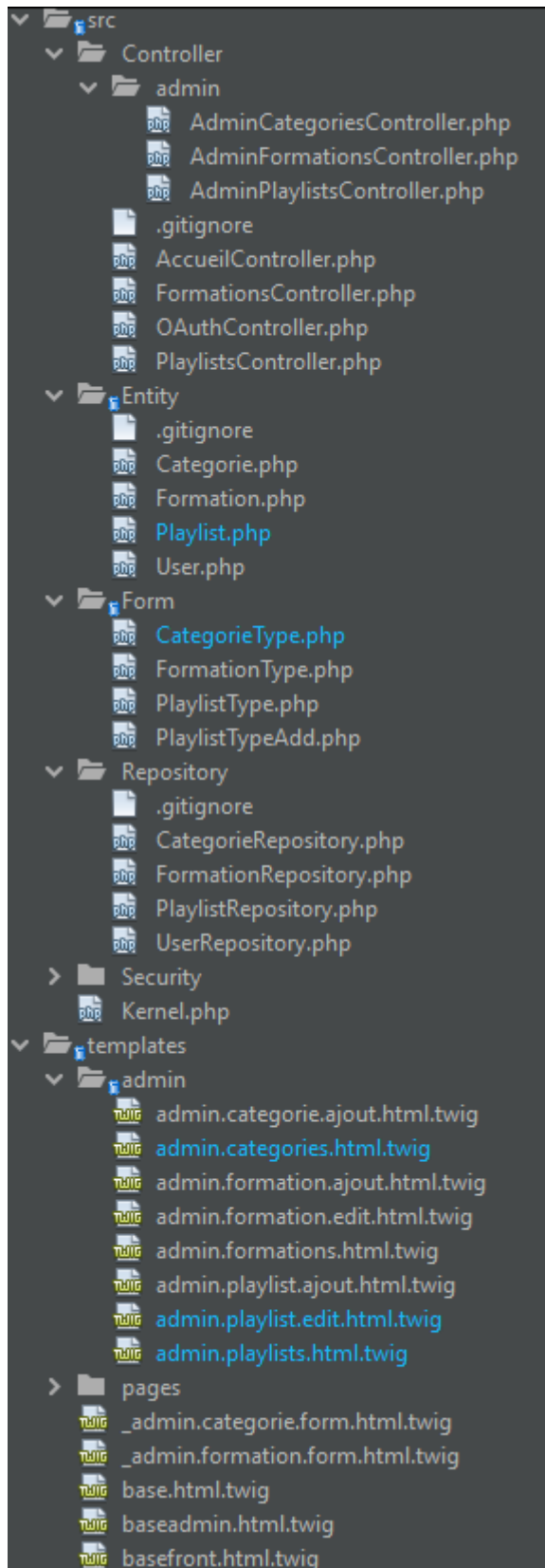
Intitulé de la catégorie

Ajouter

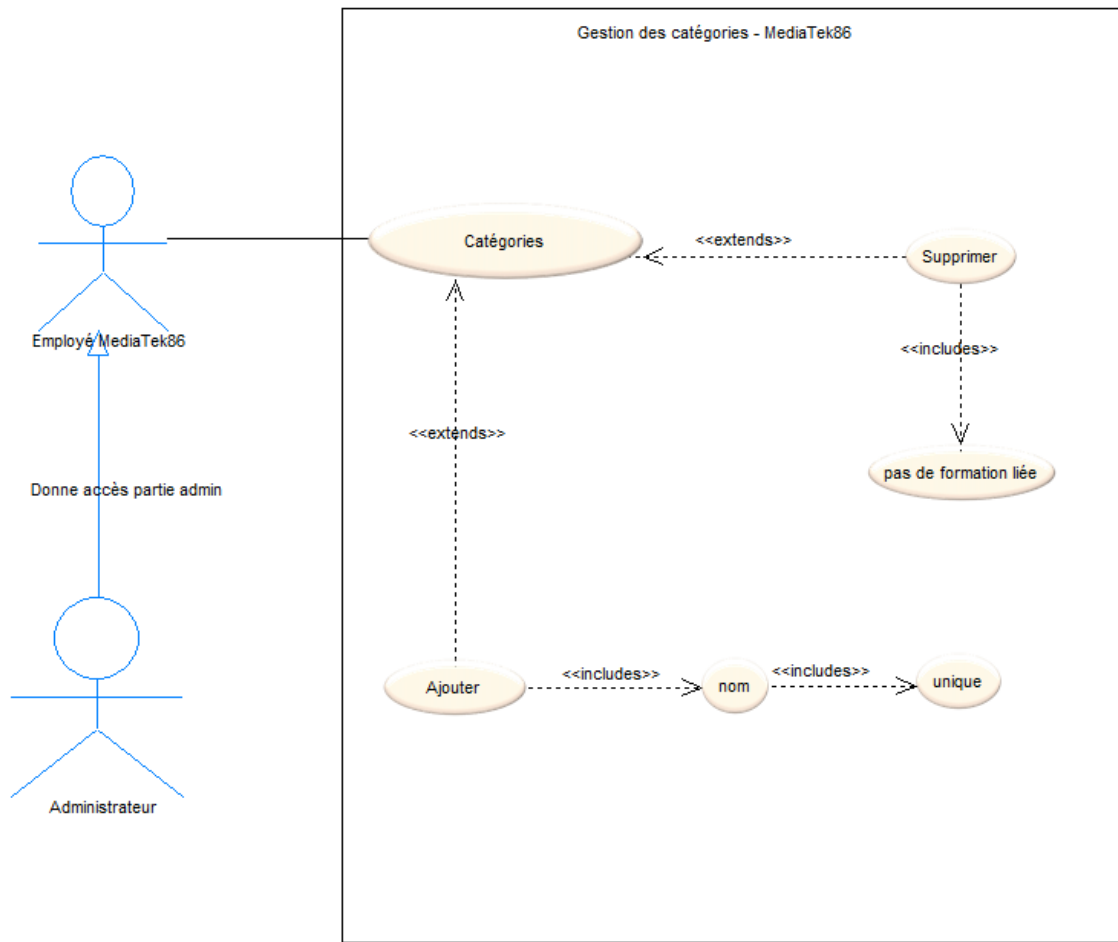
Veuillez compléter ce champ.

(suite page 69)

## La nouvelle arborescence du projet.



## Diagramme de cas d'utilisation



## Tâche 4 : ajouter l'accès avec les authentifications.

Le back office ne doit être accessible qu'après authentification : un seul profil administrateur doit avoir les droits d'accès.

Temps de travail estimé  
4 heures

Temps de travail réel  
6 heures

### Kanban de la tâche actuelle "In Progress"

The Kanban board displays tasks across three columns: To do, In progress, and Done. The 'In progress' column currently shows one task: 'M2 - T4 : ajouter l'accès avec authentification'. The 'To do' column lists six tasks, and the 'Done' column lists six tasks. A detailed view of the 'In progress' task is shown on the right, indicating it was opened in Jlauth/mediatekformation and includes a comment from Jlauth about adding authentication to the back office using Keycloak.

**To do**

- M2 - T5 : gérer la sauvegarde et la restauration de la DBB
- M3 - T1 : gérer les tests
- M3 - T2 : créer la documentation technique
- M3 - T3 : créer la documentation utilisateur
- M4 - T1 : déployer le site
- M4 - T2 : mettre en place le déploiement continu
- M5 : optimiser le référencement naturel

**In progress**

- M2 - T4 : ajouter l'accès avec authentification

**Done**

- M1 - T1 : nettoyer le code
- M1 - T2 : respecter les bonnes pratiques de codage
- M1 - T3 : ajouter une fonctionnalité
- M2 - T1 : gérer les formations
- M2 - T2 : gérer les playlists
- M2 - T3 : gérer les catégories

**M2 - T4 : ajouter l'accès avec authentification #7**

Opened in Jlauth/mediatekformation

Jlauth commented 19 days ago

Ajouter l'accès avec authentification à la partie back office en utilisant Keycloak.

Assignees: Jlauth

Labels: None yet

Projects: Mediatek Formation (In progress)

Milestone: No milestone

[Go to issue for full details](#)

[Close issue](#)



## Process

La route "JAVA\_HOME" a été configurée en amont ainsi que le path dans les variables d'environnement afin de pouvoir utiliser le jdk adéquat (ici jdk 18). Lancement du serveur Keycloak.

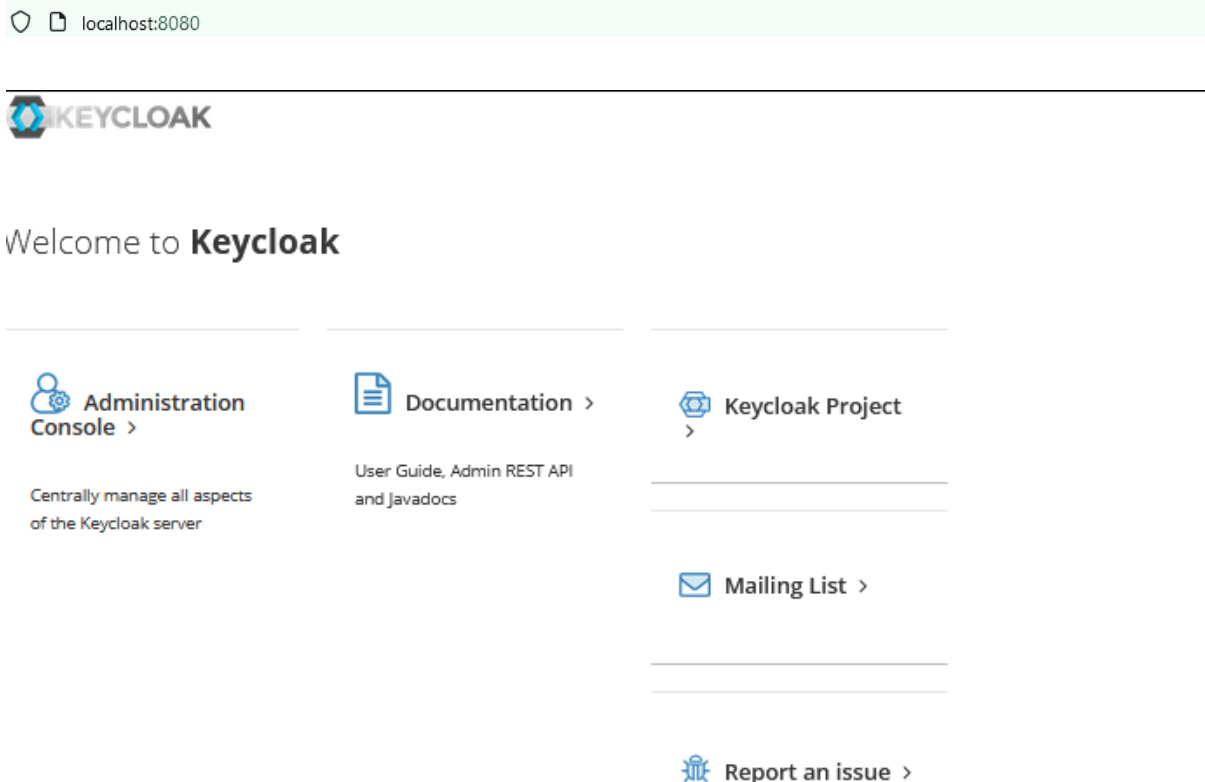
```
Microsoft Windows [version 10.0.22621.900]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Jean>cd c:\program files\keycloak\bin
```

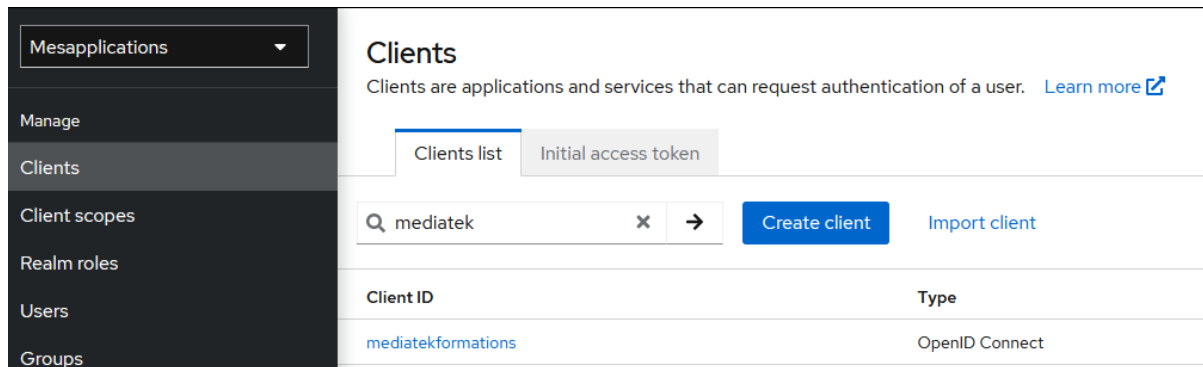
Connection effectuée sur le port 8080.

```
022-12-14 12:42:25,706 INFO [io.quarkus] (main) Profile dev activated.
022-12-14 12:42:25,706 INFO [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, jdbc-mariadb,
jdbc-mssql, jdbc-mysql, jdbc-oracle, jdbc-postgresql, keycloak, logging-gelf, narayana-jta, reactive-routes, resteasy,
resteasy-jackson, smallrye-context-propagation, smallrye-health, smallrye-metrics, vault, vertx]
022-12-14 12:42:25,711 WARN [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode.
DO NOT use this configuration in production.
```

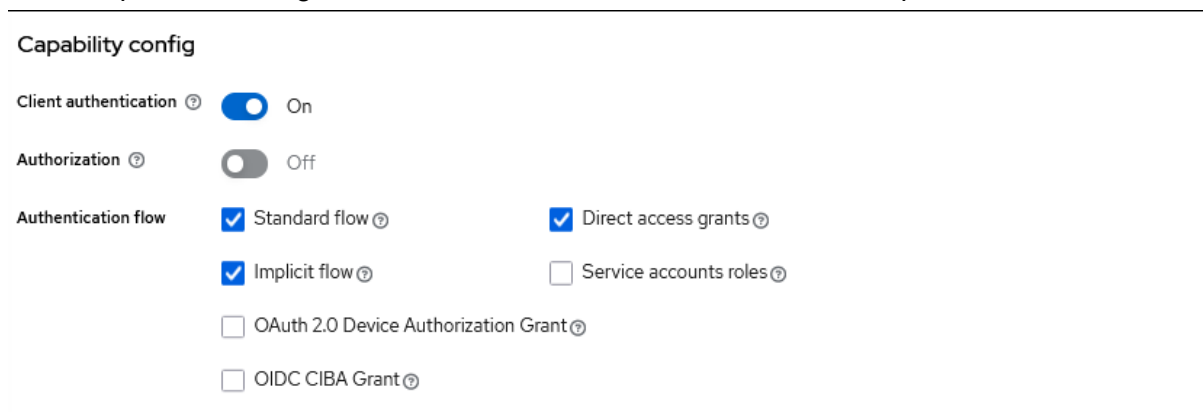
Accès à l'interface d'accès à la partie Administration Console. Un compte avait déjà été créé au préalable.



Le realm “Mes applications” est créé ainsi qu’un nouveau client : mediatekformations.



Les configurations spécifiques à ce nouveau client. Client authentication, standard flow, direct access grants et implicit flows ont été sélectionnés dans le but de permettre respectivement d’activer l’autorisation à ce client l’accès à une partie web privée, le second active l’authentification standard OpenID Connect avec un code d’autorisation, le troisième active la passerelle d’accès entre client et utilisateur dans le cadre de l’échange d’un token d’identification, et le quatrième active la prise de change d’authentification basée sur la redirection OpenID Connect



Dans le détail client, onglet Settings.

La valid redirect URL. Sans ajout d’un astérisque, les routes Keycloak ne sauraient pas vers où se diriger en source du projet.



Ainsi que les configurations suivantes.

(suite page 74)

**Login settings**

Login theme ⓘ Choose... ▼

Consent required ⓘ ☒ On

Display client on screen ⓘ ☐ Off

Client consent screen text ⓘ

**Logout settings**

Front channel logout ⓘ ☒ On

Front-channel logout URL ⓘ

Backchannel logout URL ⓘ

Backchannel logout session required ⓘ ☒ On

Backchannel logout revoke offline sessions ⓘ ☐ Off

Onglet credentials toujours côté client. Ici se trouve le client secret qui sera écrit directement dans notre code. Nous verrons son utilisation plus loin.

Clients > Client details

**mediatekformations** OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings Keys **Credentials** Roles Client scopes Sessions Advanced

Client Authenticator ⓘ Client Id and Secret ▼

Save

Client secret 

.....

☐ ☐ [Regenerate](#)

Puis création d'un utilisateur unique pour le client mediatekformations, avec mail d'authentification et mot de passe créé.

**Users**

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

User list Permissions

→ [Add user](#) [Delete user](#)

<input type="checkbox"/> Username	Email
<input type="checkbox"/> mtfadmin	mtfadmin@dom.com

mtfadmin

- Details
- Attributes
- Credentials
- Role mapping
- Groups
- Consents
- Identity provider links
- Sessions

ID \*

dab54170-005c-424f-8a0d-c2aba39cf255

Created at \*

11/27/2022, 8:45:17 PM

Username \*

mtfadmin

Email

mtfadmin@dom.com

Email verified ?

Off

First name

Last name

Enabled ?

On

Required user actions ?

Select action

Save

Revert

mtfadmin

- Details
- Attributes
- Credentials
- Role mapping
- Groups
- Consents
- Identity provider links
- Sessions

?	Type	User label
	Password	My password

mtfadmin

- Details
- Attributes
- Credentials
- Role mapping
- Groups
- Consents
- Identity provider links
- Sessions

Search session

→

Logout all sessions

Started	Last access	IP address	Clients
12/14/2022, 1:00:28 PM	12/14/2022, 1:00:28 PM	127.0.0.1	mediatekformations

Users > User details

## mtfadmin

Details
Attributes
Credentials
Role mapping
Groups
Consents
Identity provider links
Sessions

Client	Granted client scopes
mediatekformations	email, profile, roles

Realm settings

### Email settings

Email as username ⓘ
☐
Off

Login with email ⓘ
☒
On

Duplicate emails ⓘ
☐
On

Verify email ⓘ
☐
Off

### User info settings

Edit username ⓘ
☐
Off

La partie côté Keycloak terminée, accès à .env en racine du projet et écriture à la fin du fichier des informations requises afin de lier le projet à keycloak.

```
KEYCLOAK_SECRET=2lKI6DRFSDyWQ7GPUAn4X3lEnrCekb2G
KEYCLOAK_CLIENTID=mediatekformations
KEYCLOAK_APP_URL=http://localhost:8080
```

Ceci fait, nous créons une classe “user” et enregistrons en base de données afin de ne pas avoir à utiliser Keycloak à chaque connexion, avec les lignes de commande :

- **php bin/console make:user**
- **php bin/console make:entity User**
- **php bin/console make:migration**
- **php bin/console doctrine:migrations:migrate**

Côté phpMyAdmin, la table “user” correspondante a bien été créée.

The screenshot shows the phpMyAdmin interface for a database. At the top, a SQL query is displayed: `SELECT * FROM `user``. Below the query, there are several action links: `Profilage`, `[ Éditer en ligne ]`, `[ Éditer ]`, `[ Expliquer SQL ]`, `[ Créer le code source PHP ]`, and `[ Actualiser ]`. A control bar shows `Tout afficher`, `Nombre de lignes : 25`, and a search input `Filtrer les lignes: Chercher dans cette table`. Below this, there are `Options` and a table structure view. The table has columns: `id`, `email`, `roles`, `password`, and `keycloak_id`. A single row of data is visible with the following values: `1`, `mtfadmin@dom.com`, `["ROLE_ADMIN"]`, and `dab54170-005c-424f-8a0d-c2aba39cf255`. Action links for the row include `Éditer`, `Copier`, and `Supprimer`.

	id	email	roles	password	keycloak_id
	1	mtfadmin@dom.com	["ROLE_ADMIN"]		dab54170-005c-424f-8a0d-c2aba39cf255

Installations des bundle afin de créer le lien entre Symfony et Keycloak :

- ***composer require knpuniversity/oauth2-client-bundle 2.10***
- ***composer require stevenmaguire/oauth2-keycloak 3.1 --with-alldependencies***

Modification du fichier `knpu_oauth2_client`.

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

La configuration du firewall “security.yaml”.

```
form_login:
  login_path: oauth_login

access_control:
  - { path: ^/admin, roles: ROLE_ADMIN }
```

(suite page 78)

La création du controller, gestionnaire de l'authentification, par la ligne de commande **php bin/console make:controller OAuthController --no-template**.

La classe **OAuthController** automatiquement implémentée avec la définition des routes en lien avec les nouveaux besoins

```
<?php

namespace App\Controller;

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

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(){

    }
}
```

La création d'un dossier **Security** en base de **src** puis une classe dans ce nouveau dossier, **KeycloakAuthenticator** avec une écriture ou implémentation des use requis, ainsi que des différentes méthodes implémentées et réécrites.

```
class KeycloakAuthenticator extends OAuth2Authenticator implements AuthenticationEntryPointInterface {

    private $clientRegistry;
    private $entityManager;
    private $router;

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

```

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

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

```

```

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);
            // user existe recherche via son id Keycloak
            $existingUser = $this->entityManager
                ->getRepository(User::class)
                ->findOneBy(['keycloakId' => $keycloakUser->getId()]);
            if($existingUser){
                return $existingUser;
            }
            // user existe mais jamais connecté à 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;
            }
            // user n'existe pas encore dans la DB
            $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);
}

```



L'ajout du chemin dans le firewall **security.yaml** ainsi que la gestion de la déconnexion.

```
security:
    enable_authenticator_manager: true
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            entry_point: form_login
            form_login:
                login_path: oauth_login
            custom_authenticators:
                - App\Security\KeycloakAuthenticator
            logout:
                path: logout

    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
```

L'ajout de la route **'logout'** dans la maquette twig, créé dans **OAuthController**.

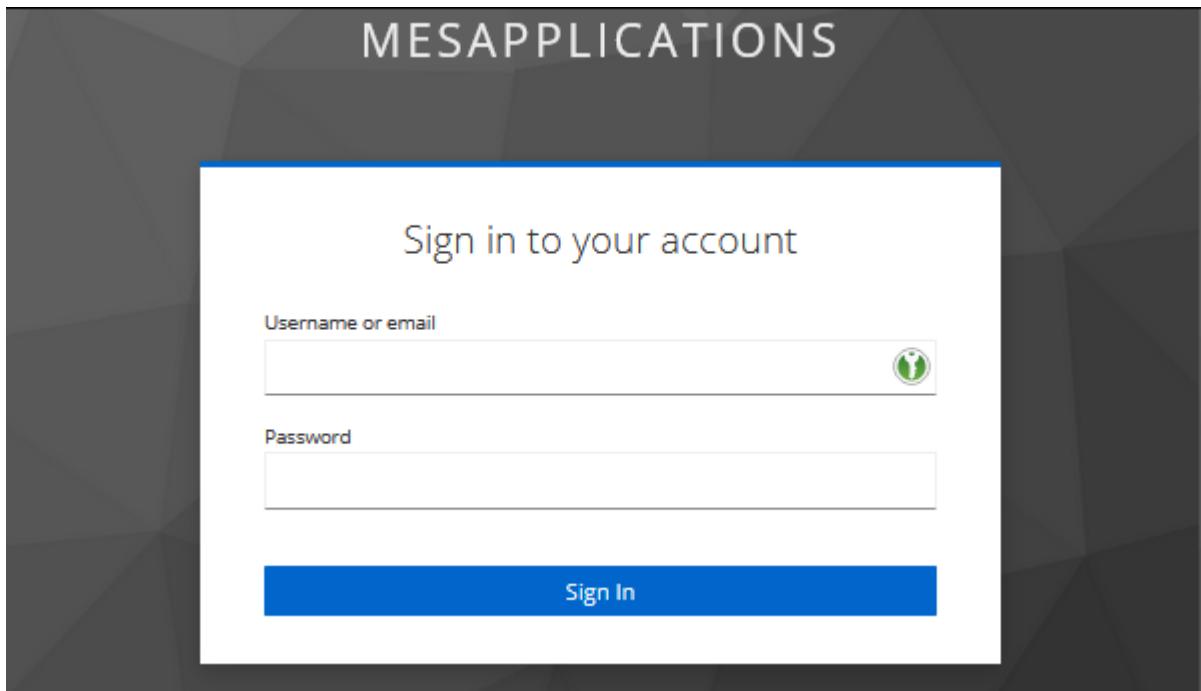
```
{% extends "base.html.twig" %}

{% block title %}{% endblock %}
{% block stylesheets %}{% endblock %}
{% block top %}
    <div class="container">
        <div class="text-end">
            <a href="{{ path('logout') }}">Déconnexion</a>
        </div>
    </div>
```

Afin de vérifier que tout fonctionne convenablement, on ajoute **/admin** à la fin de notre URL page côté front.

URL et formulaire de connexion à la partie admin.

  localhost:8080/realms/mesapplications/protocol/openid-connect/auth?state=f52bc127b40839e63.



## Rendu front

[Déconnexion](#)

### Page d'administration de MediaTek86

Formations Playlists Catégories

**Formations enregistrées**  
< >  
  
Filtrer

**Playlist**  
< >  
  
filtrer

**Catégories**  
v

**Date**  
< >

Ajouter

Eclipse n°8 : Déploiement	Eclipse et Java	Java	04/01/2021	Modifier	Supprimer
---------------------------	-----------------	------	------------	----------	-----------

[Déconnexion](#)

### Page d'administration de MediaTek86

Formations Playlists Catégories

**Catégories**  
< >  
Ajouter

Android	Supprimer
---------	-----------

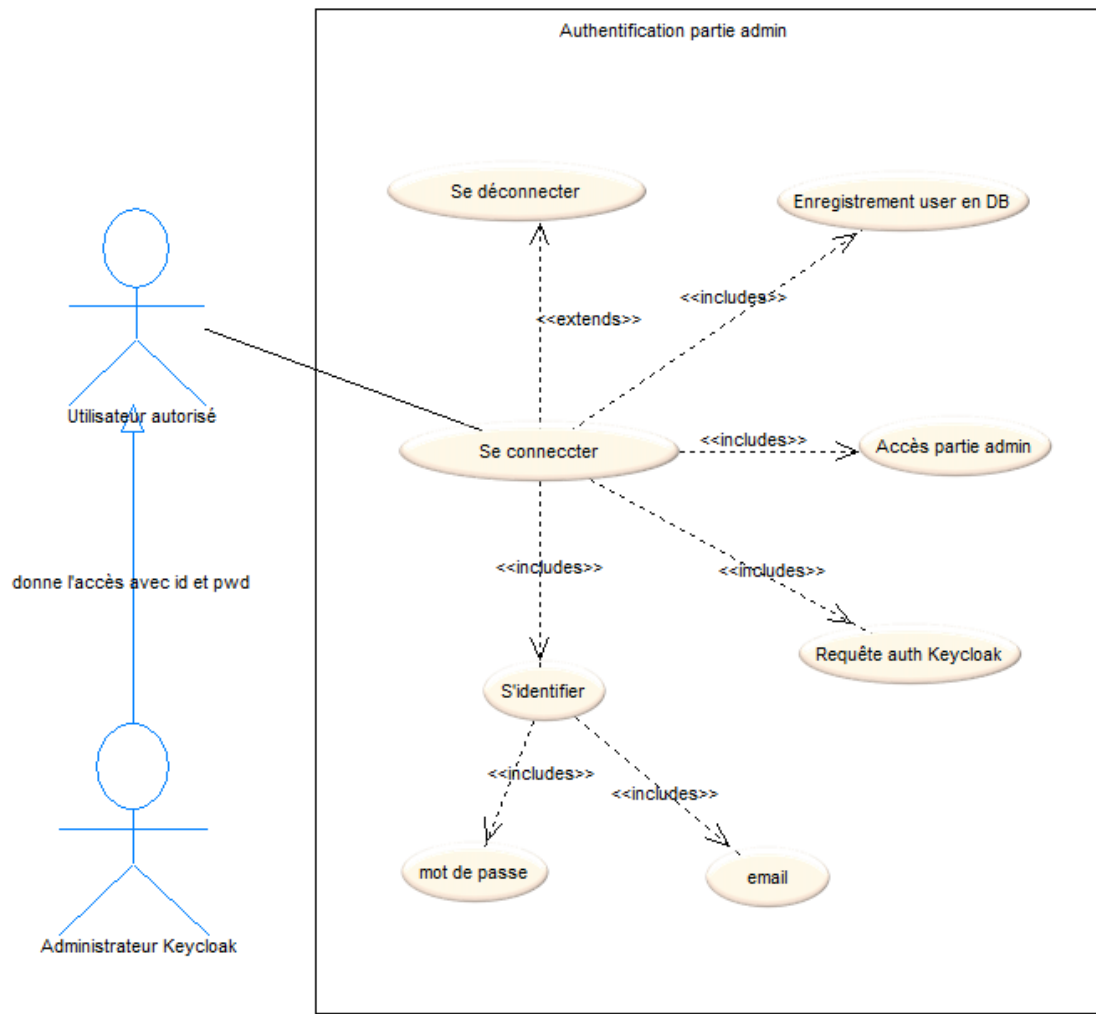
[Déconnexion](#)

### Page d'administration de MediaTek86

Formations Playlists Catégories

Ajout d'une nouvelle playlist

## Diagramme de cas d'utilisation



## Mission 3 : tester et documenter

### Tâche 1 : gérer les tests.

Tests unitaires : contrôler le fonctionnement de la méthode qui retourne la date de parution au format string.

Tests d'intégration sur les règles de validation : contrôler que la date n'est pas postérieure à aujourd'hui lors de l'ajout ou de la modification de cette dernière.

Tests d'intégration sur les Repository : contrôler toutes les méthodes ajoutées dans les classes Repository.

Tests fonctionnels : contrôler que la page d'accueil est accessible ainsi que les tris, filtres et clics des pages contenant des listes.

Tests de compatibilité : créer un scénario avec Selenium et le jouer sur plusieurs navigateurs.

Temps de travail estimé  
réel  
7 heures

Temps de travail  
  
15 heures

### Kanban de la tâche actuelle "In Progress"

The Kanban board displays three columns: 'To do', 'In progress', and 'Done'. The 'In progress' column contains the task 'M3 - T1 : gérer les tests'. A detailed view of this task is shown on the right, listing sub-tasks and a checklist of activities to be performed.

**Task: M3 - T1 : gérer les tests #9**  
Opened in Jlauth/mediatekformation

**Sub-tasks:**

- M1 - T1 : nettoyer le code (mediatekformation#1 opened by Jlauth)
- M1 - T2 : respecter les bonnes pratiques de codage (mediatekformation#2 opened by Jlauth)
- M1 - T3 : ajouter une fonctionnalité (mediatekformation#3 opened by Jlauth)
- M2 - T1 : gérer les formulaires (mediatekformation#4 opened by Jlauth)
- M2 - T2 : gérer les playbooks (mediatekformation#5 opened by Jlauth)
- M2 - T3 : gérer les catégories (mediatekformation#6 opened by Jlauth)
- M2 - T4 : ajouter l'accès à l'authentification (mediatekformation#7 opened by Jlauth)

**Réaliser les tests suivants :**

- 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 (avec Selenium)

**Assignees:** Jlauth

**Labels:** None yet

**Projects:** Mediatek Formation (In progress)

## Plan de tests

### Tests unitaires

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la méthode <b>getPublishedAtString()</b> de la classe Formation pour voir si elle retourne la bonne date au bon format.	Test unitaire lancé avec la date : 2022-11-29 22:00:13	29/11/2022	OK

### Tests d'intégration

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler, lors d'un ajout ou modification d'une formation, que la date n'est pas postérieure à aujourd'hui.	Test d'intégration lancé avec la date 2022-11-29 puis en ajoutant à la date 22:15:45	2022-11-29 2022-11-29 22:15:45	OK
Contrôler si une formation est ajoutée, dans la BDD test avec rollback (valable pour tous les tests d'intégration suivants).	Ajout d'une nouvelle formation.	Formation ajoutée.	OK
Contrôler si une formation est ajoutée puis supprimée.	Ajout d'une nouvelle formation puis suppression de celle-ci.	Formation ajoutée puis supprimée.	OK
Contrôler si une playlist est ajoutée.	Ajout d'une nouvelle playlist.	Playlist ajoutée.	OK
Contrôler si une playlist est ajoutée puis supprimée.	Ajout d'une nouvelle playlist puis suppression de celle-ci.	Playlist ajoutée puis supprimée.	OK
Contrôler si une catégorie est ajoutée.	Ajout d'une nouvelle catégorie.	Catégorie ajoutée.	OK

Contrôler si une catégorie est ajoutée puis supprimée.	Ajout d'une nouvelle catégorie puis suppression de celle-ci.	Catégorie ajoutée puis supprimée.	<b>OK</b>
Contrôler le tri par la fonction <b>testFindAllOrderByName</b> de PlaylistRepository.	Test d'intégration exécuté avec «ASC» en paramètre de la fonction et vérification du nombre de playlists (28) et du nom visé en première ligne (« Cours Curseurs »).	28 playlists  Cours Curseurs	<b>OK</b>
Contrôler le tri par la fonction <b>testFindAllOrderByNbFormations</b> de PlaylistRepository.	Test d'intégration exécuté avec «DESC» en paramètre de la fonction puis vérification du nombre de playlists (28) et du nom visé première ligne (« Bases de la programmation (C#) »).	28 playlists  Bases de la programmation (C#)	<b>OK</b>
Contrôler la recherche par la fonction <b>testFindAllByContainValue</b> de PlaylistRepository.	Test d'intégration exécuté avec «name» et «Eclipse et Java» en paramètres, vérification du nombre de playlists (1) et du nom visé en première ligne (« Eclipse et Java »).	Une playlist  Eclipse et Java	<b>OK</b>
Contrôler la recherche par la fonction <b>testFindAllByContainValue-Table</b> de PlaylistRepository.	Test d'intégration exécuté avec « name », « Cours » et « catégories » en paramètres de la fonction puis vérification du nombre de playlists (11) et du nom visé en première ligne (« Cours Composant logiciel »).	11 playlists  Cours Composant logiciel	<b>OK</b>
Contrôler le tri par la fonction <b>findAllOrderBy</b> de FormationRepository.	Test d'intégration exécuté avec « id » et « ASC » en paramètres de la fonction puis vérification du nombre de formations (238) et de l'id visé en première ligne (« Eclipse n°8 : Déploiement »).	238  Eclipse n°8 : Déploiement	<b>OK</b>
Contrôler le tri par la fonction <b>findAllOrderByTable</b> de FormationRepository.	Test d'intégration lancé avec « name », « DESC » et « catégories » en paramètres de la fonction puis vérification du nombre de catégories (224) et le nom visé en première ligne (« Eclipse n°2 : rétroconception avec ObjectAid »).	224  Eclipse n°2 : rétroconception avec ObjectAid	<b>OK</b>

Contrôler la recherche par la fonction <b><i>findByContainValue</i></b> de FormationRepository.	Test d'intégration exécuté avec « title » et « Eclipse » en paramètres de la fonction puis vérification du nombre de formations (9) et le nom visé en première ligne (« Eclipse n°8 : Déploiement »).	9  Eclipse n°8 : Déploiement	OK
Contrôler la recherche par la fonction <b><i>findByContainValueTable</i></b> de FormationRepository.	Test d'intégration exécuté avec « name », « DESC » et « catégories » en paramètres de la fonction puis vérification du nombre de catégories (224) et le nom visé en première ligne (« Eclipse n°2 : rétroconception avec ObjectAid »).	224  Eclipse n°2 : rétroconception avec ObjectAid	OK
Contrôler la recherche par la fonction <b><i>findAllLasted</i></b> de FormationRepository.	Test d'intégration exécuté avec « 1 » en paramètre de la fonction puis vérification du nombre de formations et la date visée en première ligne (« 04/01/2021 »).	1  04/01/2021	OK
Contrôler la recherche par la fonction <b><i>findAllForOnePlaylist</i></b> de FormationRepository.	Test d'intégration lancé avec « 3 » en paramètre de la fonction puis vérification du nombre de formations (18) pour cette playlist et la playlist visée en première ligne (« Eclipse n°1 : installation de l'IDE »).	19  Eclipse n°1 : installation de l'IDE	OK
Contrôler le tri par la fonction <b><i>findAllOrderBy</i></b> de CategorieRepository.	Test d'intégration lancé avec « id » et « DESC » en paramètres de la fonction puis vérification du nombre de catégories (10) et de la catégorie visée en première ligne (« Je suis une catégorie de test »).	10  Je suis une catégorie de test	OK
Contrôler la recherche par la fonction <b><i>findAllForOnePlaylist</i></b> de CategorieRepository.	Test d'intégration lancé avec « 1 » en paramètre de la fonction puis vérification du nombre de catégories (2) pour cette playlist et le nom de la catégorie visée en première ligne (« Java »).	2  Java	OK

## Tests fonctionnels

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler l'accès à la page "Accueil".	Test fonctionnel lancé par une requête en racine de la page web ('/').	Réponse HTTP_OK	<b>OK</b>
Contrôler le clic sur le lien "Formations" sur la page d'accueil.	Test fonctionnel sur le clic « Formations » et vérification de la validité du chemin vers la page '/formations'.	Réponse HTTP_OK Retour URL absolue	<b>OK</b>
Contrôler le clic sur le lien "Playlists" sur la page d'accueil.	Test fonctionnel sur le clic « Playlists » et vérification de la validité du chemin vers la page '/playlists'.	Réponse HTTP_OK Retour URL absolue	<b>OK</b>
Contrôler le clic sur le lien "cgu" sur la page d'accueil.	Test fonctionnel sur le clic « cgu » et vérification de la validité du chemin vers la page '/cgu'.	Réponse HTTP_OK Retour URL absolue	<b>OK</b>
Contrôler le clic sur le lien "Accueil" sur la page d'accueil.	Test fonctionnel sur le clic « Accueil » et vérification de la validité du chemin vers la page '/accueil'.	Réponse HTTP_OK Retour URL absolue	<b>OK</b>
Contrôler l'accès à la page "Formations".	Test fonctionnel lancé par une requête en racine de la page '/formations'.	Réponse HTTP_OK	<b>OK</b>
Contrôler le clic sur le tri des formations.	Test fonctionnel sur le clic tri des formations. Récupération de la validité de la route '/formations', du nombre de formations trouvées via la balise « th » et le titre en première ligne via la balise « h5 ».	Réponse HTTP_OK Retour URL absolue Nombre formations : 237 (test «th») Eclipse n°8 : Déploiement (1ère ligne «h5»)	<b>OK</b>
Contrôler le filtre de recherche d'une ou plusieurs formations.	Test fonctionnel pour une recherche de formation via le clic « filtrer » avec pour nom « recherche » et pour valeur « Android ». Comptage des résultats de recherche.	Champ « Android » trouvé et initialisé Nombre de formations Android : 32	<b>OK</b>



Contrôler le clic vers le lien d'une formation depuis "Formations".	Test fonctionnel sur le clic « Images des formations ». Vérification validité du chemin et ID sélectionnée via « REQUEST_URI ».	HTTP_OK Retour URL absolue avec ID = 1	<b>OK</b>
Contrôler le clic sur tri des playlists.	Test fonctionnel sur le clic tri des playlists. Récupération de la validité de la route '/playlists', du nombre de playlists trouvées via la balise « th » et le nom en première ligne via la balise « h5 ».	Réponse HTTP_OK Retour URL absolue Nombre playlists 27 (test « th ») Bases de la programmation (C#) (1ère ligne « h5 »)	<b>OK</b>
Contrôler le filtre de recherche d'une ou plusieurs playlists.	Test fonctionnel pour une recherche de playlist via le clic « filtrer » avec pour nom « recherche » et pour valeur « Cours ». Comptage des résultats de recherche.	Champ « Cours » trouvé et initialisé Nombre de playlists « Cours » : 11	<b>OK</b>
Contrôler le filtre de tri d'une catégorie dans "Playlists".	Test fonctionnel sur une recherche de catégorie via le clic « filtrer » avec pour nom « recherche » avec comme valeur « Android ». Comptage des résultats de recherche.	Champ « Android » trouvé et initialisé Nombre de catégories Android : 2 (logique suite à la création de la collection ne retournant qu'une seule fois chaque type)	<b>OK</b>
Contrôler le clic vers le lien d'une playlist depuis playlists.	Test fonctionnel sur le clic « Voir détail ». Vérification de la validité du chemin, de la validité de l'ID sélectionnée via « REQUEST_URI ».	Retour URL absolue avec ID = 13	<b>OK</b>

## Tests de compatibilité

But du test	Action de contrôle	Résultat attendu	Bilan
Scénario de compatibilité sur Firefox	Navigation aléatoire sur tous les éléments du site	Pas de problème de compatibilité	<b>OK</b>
Scénario de compatibilité sur Chrome	Utilisation du script Firefox et utilisé sous Chrome	Pas de problème de compatibilité	<b>OK</b>

## Tâche 2 : créer la documentation technique.

Contrôler les commentaires normalisés et générer la documentation technique du site complet.

Temps de travail estimé  
réel  
1 heure

Temps de travail  
  
3 heures

### Kanban de la tâche actuelle "In Progress"

Filter cards

**To do** (5 items)

- M4 - T2 : gérer la sauvegarde et la restauration de la DBB
- M3 - T3 : créer la documentation utilisateur
- M4 - T1 : déployer le site
- M4 - T3 : mettre en place le déploiement continu
- M5 : optimiser le référencement naturel

**In progress** (1 item)

- M3 - T2 : créer la documentation technique

**Done** (8 items)

- M1 - T1 : nettoyer le code
- M1 - T2 : respecter les bonnes pratiques de codage
- M1 - T3 : ajouter une fonctionnalité
- M2 - T1 : gérer les formations
- M2 - T2 : gérer les playlists
- M2 - T3 : gérer les catégories
- M2 - T4 : ajouter l'accès avec authentification
- M3 - T1 : gérer les tests

**M3 - T2 : créer la documentation technique #10**

Opened in J.lauth/mediatekformation

J.lauth commented on Nov 7, 2022

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

Assignees: No one—assign yourself

Labels: None yet

Projects: Mediatek Formation In progress

Milestone: No milestone

Development

Go to issue for full details

Close issue

### Page d'accueil de la documentation technique

Documentation technique MediaTekFormation

Search (Press "/" to focus)

**Namespaces**

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

**Packages**

- Application

**Reports**

- Deprecated
- Errors
- Markers

**Indices**

- Files

**Documentation**

**Packages**

- Application

**Namespaces**

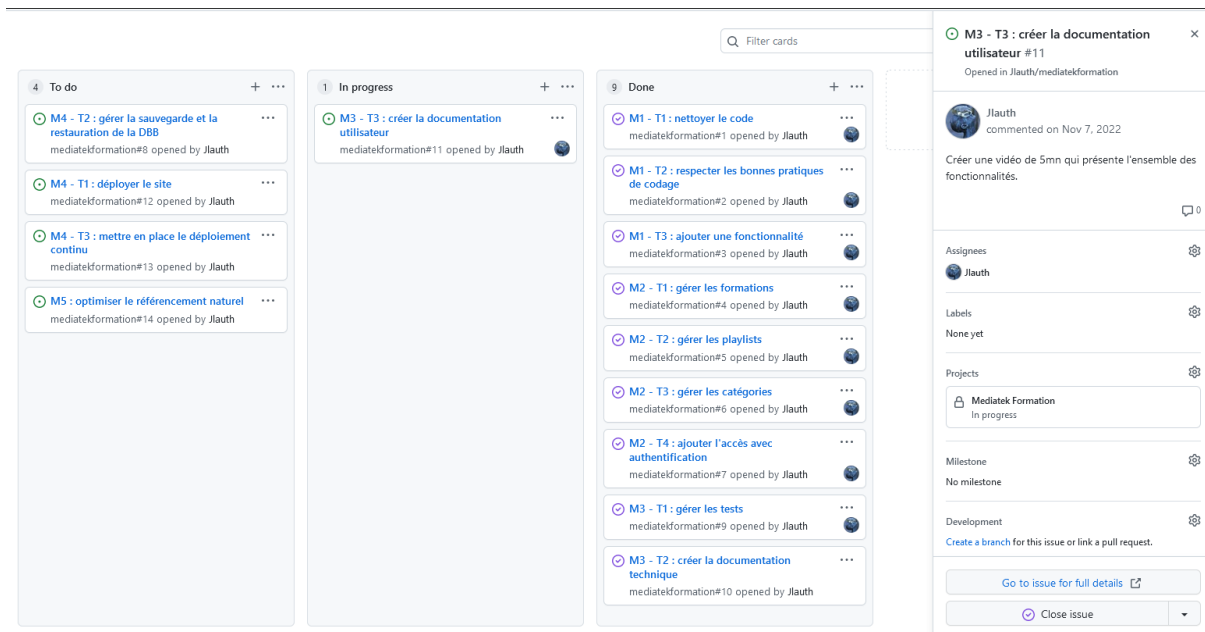
- App

### Tâche 3 : créer la documentation utilisateur.

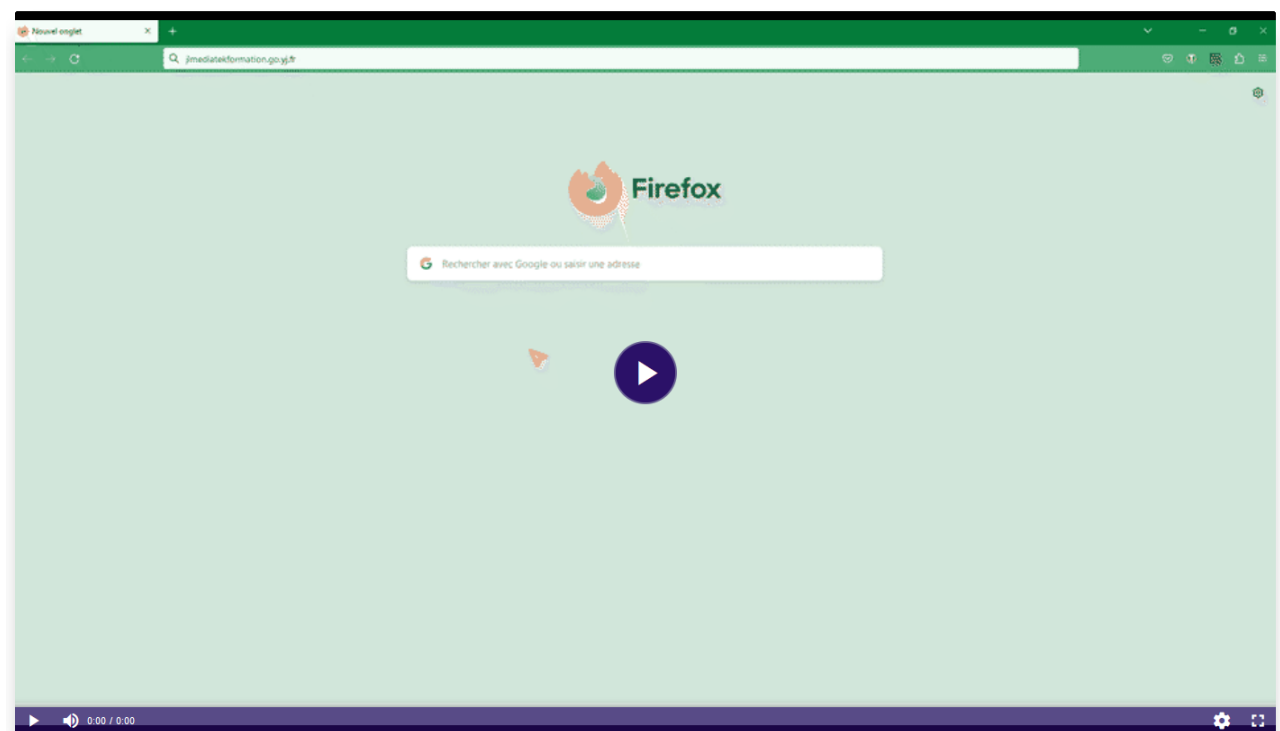
Temps de travail estimé  
réel  
2 heures

Temps de travail  
10 heures

#### Kanban de la tâche actuelle "In Progress"



#### Capture d'écran de la vidéo de présentation utilisateur



## Mission 4 : déployer le site et gérer le déploiement continu

### Tâche 1 : déployer le site.

Installer et configurer le serveur d'authentification Keycloak dans une VM en ligne.  
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.

Temps de travail estimé  
réel  
2 heures

Temps de travail  
10 heures

### Kanban de la tâche actuelle "In Progress"

The Kanban board displays tasks across three columns: To do, In progress, and Done.

- To do (3 items):**
  - M4 - T2 : gérer la sauvegarde et la restauration de la DBB (mediatekformation#8 opened by J.lauth)
  - M4 - T3 : mettre en place le déploiement continu (mediatekformation#13 opened by J.lauth)
  - M5 : optimiser le référencement naturel (mediatekformation#14 opened by J.lauth)
- In progress (1 item):**
  - M4 - T1 : déployer le site (mediatekformation#12 opened by J.lauth)
- Done (10 items):**
  - M1 - T2 : respecter les bonnes pratiques de codage (mediatekformation#2 opened by J.lauth)
  - M1 - T3 : ajouter une fonctionnalité (mediatekformation#3 opened by J.lauth)
  - M2 - T1 : gérer les formations (mediatekformation#4 opened by J.lauth)
  - M2 - T2 : gérer les playlists (mediatekformation#5 opened by J.lauth)
  - M2 - T3 : gérer les catégories (mediatekformation#6 opened by J.lauth)
  - M2 - T4 : ajouter l'accès avec authentification (mediatekformation#7 opened by J.lauth)
  - M3 - T1 : gérer les tests (mediatekformation#9 opened by J.lauth)
  - M3 - T2 : créer la documentation technique (mediatekformation#10 opened by J.lauth)
  - M3 - T3 : créer la documentation utilisateur (mediatekformation#11 opened by J.lauth)

**Task Details (M4 - T1 : déployer le site #12):**

- Opened in J.lauth/mediatekformation
- Commented on Nov 7, 2022: Déployer le site chez un hébergeur et le serveur d'authentification Keycloak dans une VM en ligne.
- Assignees: No one—assign yourself
- Labels: None yet
- Projects: Mediatek Formation (In progress)
- Milestone: No milestone
- Development: Create a branch for this issue or link a pull request.
- Buttons: Go to issue for full details, Close issue

## Process

Création d'une VM en ligne sur Microsoft Azure afin de pouvoir installer et configurer Keycloak en ligne et en HTTPS.

Système d'exploitation : Windows (Windows 10 Pro)  
Taille : Standard B1s (1 processeur virtuel, 1 Gio de mémoire)  
Adresse IP publique : [40.66.55.171](https://40.66.55.171)  
Réseau/sous-réseau virt... : [vmKeycloak\\_group-vnet/default](https://vmKeycloak_group-vnet/default)  
Nom DNS : [jlsiomonkeycloak.francecentral.cloudapp.azure.com](https://jlsiomonkeycloak.francecentral.cloudapp.azure.com)

Configuration de la mise en réseau via le port TCP 443 qui sera utilisé par le protocole HTTPS.

Règles des ports d'entrée	Règles des ports de sortie	Groupes de sécurité d'application	Équilibrage de charge			
Groupe de sécurité réseau <b>vmKeycloak-nsg</b> (attaché à l'interface réseau : <b>vmkeycloak549_z3</b> ) Impacts 0 sous-réseaux, 1 interfaces réseau				Ajouter une règle de port d'entrée		
Priorité	Nom	Port	Protocole	Source	Destination	Action
300	⚠ RDP	3389	TCP	N'importe lequel	N'importe lequel	🟢 Autoriser
320	HTTP	80	TCP	N'importe lequel	N'importe lequel	🟢 Autoriser
340	HTTPS	443	TCP	N'importe lequel	N'importe lequel	🟢 Autoriser

Prise de contrôle de la VM à distance par l'intermédiaire de l'outil Windows "Connexion Bureau à distance" de le but :

- de l'installation et configuration du prérequis OpenJDK afin de pouvoir utiliser Keycloak (configuration du path et création de la variable JAVA\_HOME dans les variables système).
- de l'installation et configuration Keycloak. On y crée un realm puis un client id en adéquation avec le nom renseigné dans le .env du projet sous KEYCLOAK\_CLIENTID et récupération du client secret dans l'onglet "Credentials". Création d'un utilisateur unique ayant tous les droits d'accès à la partie admin du site.
- de l'installation de Xampp. Une fois Xampp installé et Apache lancé, obtention du certificat SSL via Certbot.
- une fois toutes les étapes précédentes exécutées, lancement du serveur Keycloak en HTTPS. Afin de pouvoir utiliser ce Keycloak à partir d'une application, on configure dans le .env du projet le KEYCLOAK\_APP\_URL (le DNS de la VM), le KEYCLOAK\_SECRET (récupéré dans les "Credentials") et le KEYCLOAK\_CLIENTID (id du client créé dans Keycloak).

Concernant le déploiement du site nous utilisons la version gratuite de PlanetHoster. Une fois un compte FTP configuré, on upload le projet local via FileZilla. Le dossier vendor n'étant pas exporté automatiquement, il faut l'exporter dans un second temps.

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

Temps de travail estimé  
1 heure

Temps de travail réel  
4 heures

### Kanban de la tâche actuelle "In Progress"

The Kanban board displays the following tasks:

- In progress (1 card):**
  - M4 - T2 : gérer la sauvegarde et la restauration de la DBB (opened by Jlauth)
- Done (11 cards):**
  - M1 - T3 : ajouter une fonctionnalité (opened by Jlauth)
  - M2 - T1 : gérer les formations (opened by Jlauth)
  - M2 - T2 : gérer les playlists (opened by Jlauth)
  - M2 - T3 : gérer les catégories (opened by Jlauth)
  - M2 - T4 : ajouter l'accès avec authentification (opened by Jlauth)
  - M3 - T1 : gérer les tests (opened by Jlauth)
  - M3 - T2 : créer la documentation technique (opened by Jlauth)
  - M3 - T3 : créer la documentation utilisateur (opened by Jlauth)
  - M4 - T1 : déployer le site (opened by Jlauth)

**Task Details (M4 - T2 : gérer la sauvegarde et la restauration de la DBB #8):**

- Opened in Jlauth/mediatekformation
- Commented on Nov 7, 2022: Automatiser une tâche de sauvegarde quotidienne de la DBB.
- Assignees: Jlauth
- Labels: None yet
- Projects: Mediatek Formation (In progress)
- Milestone: No milestone
- Development
- Buttons: Go to issue for full details, Close issue

## Process

Création d'un script .sh permettant d'automatiser la sauvegarde de la base de données. Celui-ci se compose d'une variable permettant de mémoriser la date de création du backup, une ligne exécutant la suppression de l'ancien fichier de sauvegarde de la base de données et enfin de la ligne mysqldump qui récupère le script sql de la bdd et le sauvegarde.

Ce script est ensuite réécrit au format Linux et uploadé via FileZilla dans le dossier du projet adéquat sur PlanetHoster.

🏠 > savebdd	
Nom ↑	Taille
<input type="checkbox"/> backup.sh	248 B

Une fois le script importé, le paramétrage de la sauvegarde se fait via la configuration d'une tâche Cron dans le panel de gestion de PlanetHoster. Ici, on détermine une sauvegarde journalière et insérons la ligne de commande faisant appel à l'exécution de notre script.

### PARAMÈTRES COMMUNS

Une fois par jour(0 0 \* \* \*)

### MINUTE

0

0

### HEURE

0

0

### JOUR DU MOIS

\*

Chaque jour (\*)

### MOIS

\*

Chaque mois (\*)

### JOUR DE LA SEMAINE

\*

Chaque jour (\*)

### COMMANDE

/home/nacytbjg/savebdd/backup.sh

Notre backup est bien présent.

🏠 > savebdd	
Nom ↑	Taille
 backup.sh	276 B
 bddbbackup_2023-01-11.sql.gz	361 B

En cas de besoin de restauration de la base de données, il faut récupérer l'archive en local (ici bddbbackup\_2023-01-11.sql.gz), l'extraire et exécuter le script dans le phpMyAdmin PlanetHoster correspondant au projet afin de restaurer celle-ci.



### Tâche 3 : mettre en place le déploiement continu.

Temps de travail estimé  
1 heure

Temps de travail réel  
3 heures

#### Kanban de la tâche actuelle "In Progress"

The screenshot displays a Jira Kanban board with three columns: 'In progress', 'Done', and a third column with 12 items. The 'In progress' column contains one card: 'M4 - T3 : mettre en place le déploiement continu #13', opened by Jlauth. The 'Done' column contains 12 cards, each with a checkmark icon and a description. The third column contains 12 cards, each with a checkmark icon and a description. The card 'M4 - T3 : mettre en place le déploiement continu #13' is expanded, showing a comment from Jlauth dated Nov 7, 2022, and a description: 'Créer un script dans GitHub pour gérer le déploiement continu.' The right sidebar shows the issue details, including the title, description, assignees (Jlauth), labels (None yet), projects (Mediatek Formation), milestones (No milestone), and development status (Development). At the bottom, there are buttons for 'Go to issue for full details' and 'Close issue'.

1 In progress

12 Done

12

Filter cards

M4 - T3 : mettre en place le déploiement continu #13  
Opened in Jlauth/mediatekformation

Jlauth commented on Nov 7, 2022

Créer un script dans GitHub pour gérer le déploiement continu.

0

Assignees

Jlauth

Labels

None yet

Projects

Mediatek Formation  
In progress

Milestone

No milestone

Development

Go to issue for full details

Close issue

## Process

Dans le GitHub lié au projet, création d'un nouveau workflow et rédaction du script qui à chaque push mettre à jour le site web publié en ligne.




**Workflow file for this run**  
.github/workflows/main.yml at 71f239b

```
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: node123-eu.n0c.com
15            server-dir: /public_html/mediatekformations/public/
16            username: jlftp@jlmediatekformation.go.yj.fr
17            password: ${ secrets.ftp_password }
```

On commit ce nouveau fichier *main.yml* en racine du projet puis on le récupère en local avec un pull request dans l'IDE.


Le secret password est ensuite configuré directement dans GitHub.

**Repository secrets**

 **FTP\_PASSWORD** Updated 16 hours ago  

Le résultat et la validation du déploiement continu.

1 workflow run Event ▾ Status ▾ Branch ▾ Actor ▾

 **Update main.yml**  
Deploy website on push #3: Commit 71f239b pushed by Jlauth

main

16 hours ago  
1m 7s

...

## BILAN FINAL

C'était un défi important, mais j'ai travaillé dur et j'ai réussi à surmonter les obstacles techniques qui se sont présentés.

Cependant, le processus a été beaucoup plus difficile que prévu et les estimations du temps de travail alloué pour chaque tâche a été décuplé. Les problèmes rencontrés furent nombreux. La plupart étant liés au code lui-même, tandis que d'autres étaient dus à une mauvaise gestion du versioning et à une perte de données. Mais surtout, je reconnais que mes lacunes techniques et mon organisation du travail ont également été des facteurs clés dans la survenue de ces problèmes.

Malgré ces défis, je suis satisfait du travail accompli et je suis déterminé à apprendre de mes erreurs pour améliorer mes compétences techniques et mon organisation du travail pour mes futurs projets.