

Atelier professionnel Mediatek Formation

BTS SIO OPTION SLAM
ESTELLE FRIEDT

TABLE DES MATIERES

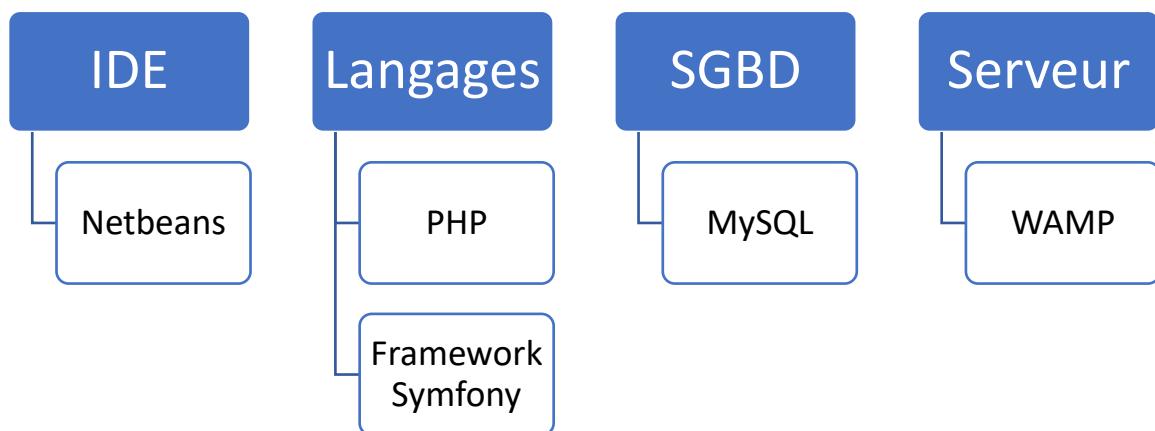
Mission 1 : Nettoyer et optimiser le code existant.....	3
Tâche 1 : Nettoyer le code	3
Tâche 2 : Ajouter une fonctionnalité	11
Mission 2 : Coder la partie back-office	14
Tâche 1 : Gérer les formations	14
Tâche 2 : Gérer les playlists.....	26
Tâche 3 : Gérer les catégories	29
Tâche 4 : Ajouter l'accès avec authentification	30
Mission 3 : Tester et documenter	34
Tâche 1 : Gérer les tests.....	34
Tâche 2 : Créer la documentation technique	36
Tâche 3 : Créer la documentation utilisateur	37
Mission 4 : déployer le site et gérer le déploiement continu	38
Tâche 1 : Déployer le site	38
Tâche 2 : Gérer la sauvegarde et la restauration de la BDD	39
Tâche 3 : Mettre en place le déploiement continu.....	43
BILAN	45
ANNEXE Plan de tests	46
Tests unitaires.....	46
Tests d'intégration	46
Tests fonctionnels	48
Tests de compatibilité.....	50

Contexte

Le réseau des médiathèques de la Vienne, MediaTek86, a pour mission de centraliser les prêts de livres, DVD et CD, tout en développant les services numériques pour l'ensemble des médiathèques du département. Dans le but d'augmenter l'attractivité des médiathèques, MediaTek86 souhaite se développer autour de deux axes principaux :

- Enrichir ses services en permettant aux adhérents d'emprunter des films en VOD.
- Compléter l'offre traditionnelle des médiathèques en proposant des formations sur les outils numériques et des modules d'autoformation en ligne.

Langages et technologies utilisées



MISSION 1 : NETTOYER ET OPTIMISER LE CODE EXISTANT



TACHE 1 : NETTOYER LE CODE

Par convention il est nécessaire de nommer les constantes en majuscules :

✗ *Code avant correction :*

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

Intitulé de l'erreur Sonarlint :



✓ *Code après correction :*

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

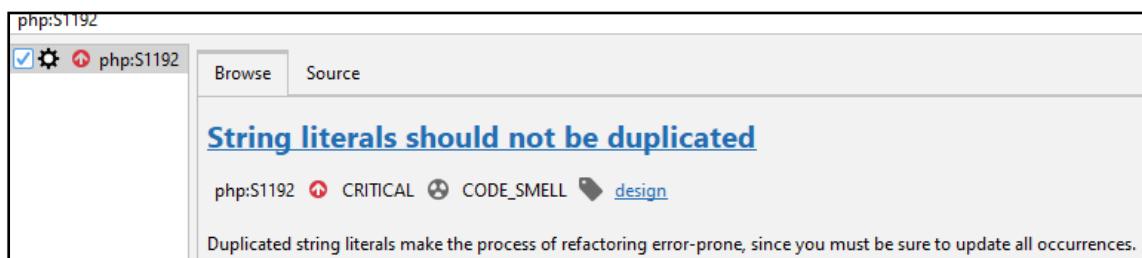
```
public function getMiniature(): ?string {  
    return self::CHEMINIMAGE . $this->videoId . "/default.jpg";  
}  
  
public function getPicture(): ?string {  
    return self::CHEMINIMAGE . $this->videoId . "/hqdefault.jpg";  
}
```

Dans `FormationController.php` :

✗ *Code avant correction :*

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

Erreur relevée par Sonarlint :



Il est donc recommandé de définir le chemin comme une constante afin de n'avoir à le modifier qu'une seule fois si nécessaire :

✓ *Code après correction :*

```
class FormationsController extends AbstractController {
    const PAGES_FORMATIONS = "pages/formations.html.twig";
```

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

Dans PlaylistsController.php :

✗ Code avant correction :

```
49     #[Route('/playlists', name: 'playlists')]
50     public function index(): Response{
51         $playlists = $this->playlistRepository->findAllOrderBy('ASC');
52         $categories = $this->categorieRepository->findAll();
53         return $this->render("pages/playlists.html.twig", [
54             'playlists' => $playlists,
55             'categories' => $categories
56         ]);
57     }
```

Erreurs relevées par Sonarlint :



✓ Code après correction :

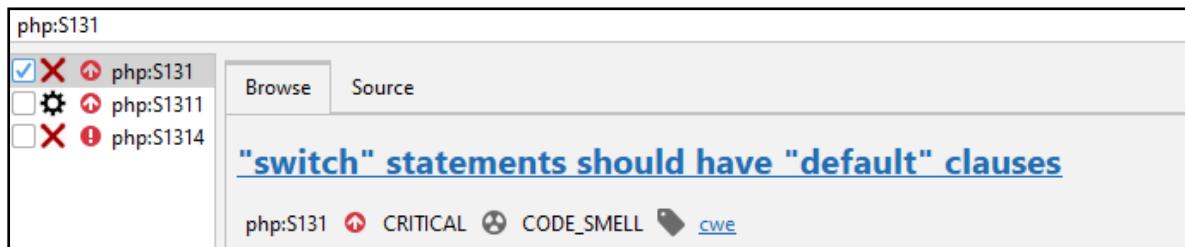
```
1 class PlaylistsController extends AbstractController {
2
3     const PAGES_PLAYLISTS = "pages/playlists.html.twig";
4 }
```

```
1 /**
2  * @Route("/playlists", name="playlists")
3  * @return Response
4  */
5
6     #[Route('/playlists', name: 'playlists')]
7     public function index(): Response{
8         $playlists = $this->playlistRepository->findAllOrderBy('ASC');
9         $categories = $this->categorieRepository->findAll();
10        return $this->render(self::PAGES_PLAYLISTS, [
11            'playlists' => $playlists,
12            'categories' => $categories
13        ]);
14    }
```

✗ Code avant correction :

```
61     #[Route('/playlists/tri/{champ}/{ordre}', name: 'playlists.sort')]
62     public function sort($champ, $ordre): Response{
63         switch($champ){
64             case "name":
65                 $playlists = $this->playlistRepository->findAllOrderBy($ordre);
66                 break;
67         }
68     }
```

Erreur relevée par Sonarlint :

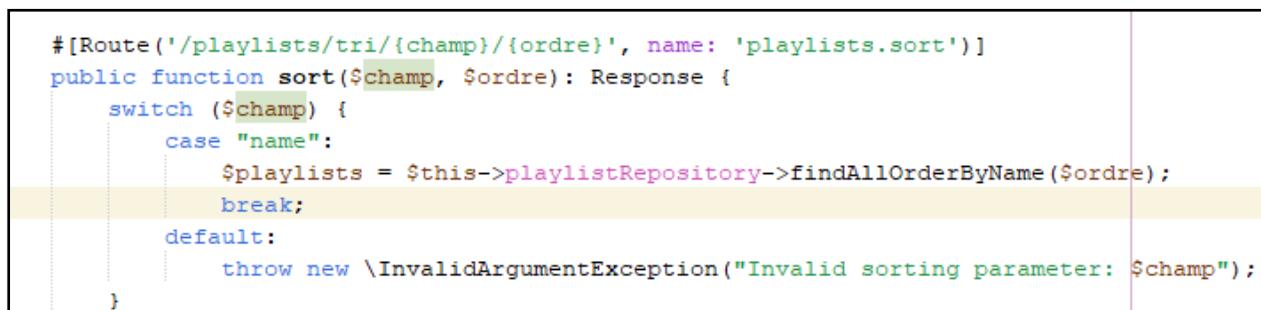


The screenshot shows the Sonarlint interface with the following details:

- Top left: php:S131
- Top right: Browse, Source buttons
- Left sidebar: php:S131 (checked), php:S1311, php:S1314
- Middle section: **"switch" statements should have "default" clauses**
- Bottom: php:S131 CRITICAL CODE_SMELL CWE

Dans cette instruction switch, aucune clause default n'est présente. Cela signifie qu'aucune action n'est définie dans le cas où aucune des conditions spécifiées n'est remplie. Il est donc nécessaire d'ajouter une clause default pour garantir un comportement défini si un champ invalide est fourni :

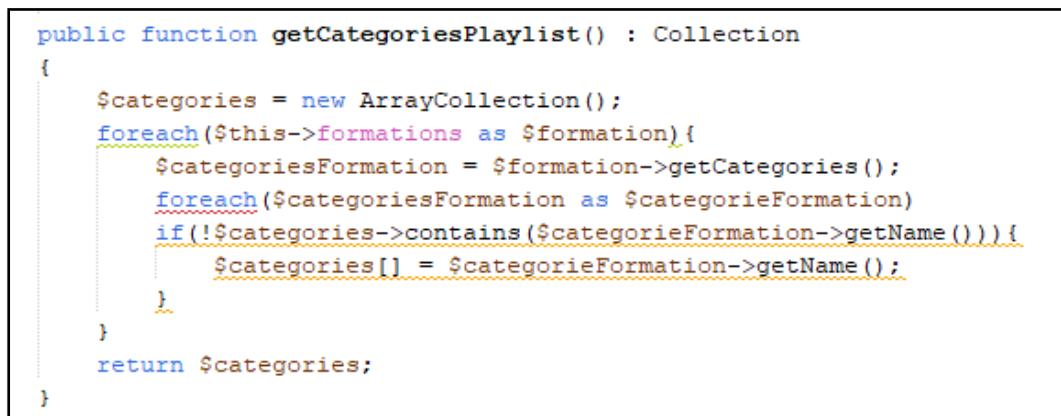
✓ Code après correction :



```
#[Route('/playlists/tri/{champ}/{ordre}', name: 'playlists.sort')]
public function sort($champ, $ordre): Response {
    switch ($champ) {
        case "name":
            $playlists = $this->playlistRepository->findAllOrderByname($ordre);
            break;
        default:
            throw new \InvalidArgumentException("Invalid sorting parameter: $champ");
    }
}
```

Playlist.php :

✗ Code avant correction :



```
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;
}
```

Erreur relevée par Sonarlint :



The screenshot shows the Sonarlint interface with the following details:

- Top left: php:S121 (checked)
- Top right: Browse, Source buttons
- Middle section: **Control structures should use curly braces**
- Bottom: php:S121 CRITICAL CODE_SMELL pitfall

Ici afin que le code soit plus clair il faut rajouter une accolade à la structure du foreach :

✓ *Code après correction :*

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

✗ *Code avant correction :*

```
76     public function removeFormation(Formation $formation): static {  
77         if ($this->formations->removeElement($formation)) {  
78             // set the owning side to null (unless already changed)  
79             if ($formation->getPlaylist() === $this) {  
80                 $formation->setPlaylist(null);  
81             }  
82         }  
83     }  
84     return $this;  
}
```

Erreur relevée par Sonarlint :



Collapsible "if" statements should be merged

php:S1066 MAJOR CODE_SMELL clumsy

L'erreur signalée ici indique que deux instructions if imbriquées peuvent être combinées.

Une modification du code est donc nécessaire en supprimant les 2^{ème} if et en rajoutant &&.

✓ *Code après correction :*

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

Dossier templates :

✗ *Code avant correction :*

```
<td class="text-center">
  {% if formation.miniature %}
    <a href="{{ path('formations.showone', {id:formation.id}) }}>
      ` doit comporter un attribut `alt`. Il est donc nécessaire de modifier les fichiers concernés :

✓ *Code après correction :*

Dans `formations.html.twig` :

```
<td class="text-center">
 {% if formation.miniature %}


```

Dans `accueil.html.twig` :

```



```

Dans `playlists.html.twig` :

```



```

Dans **basefront.html.twig** :

```

```

```

```

✗ *Code avant correction :*

Dans **cgu.html.twig** :

```
(ci-après "les Utilisateurs")
```

```
(ci-après "le site").
```

Dans **basefront.html.twig** :

```
<p><small><i> Consultez nos Conditions Générales d'Utilisation </i></small></p>
```

*Erreur relevée par Sonarlint :*

**"<strong>" and "<em>" tags should be used**

Web:BoldAndItalicTagsCheck ✅ MINOR ⚙ BUG 🔍 [accessibility](#)

Cette erreur indique que les balises strong et em doivent être utilisées pour marquer le texte qui doit être mis en avant.

✓ *Code après correction :*

Dans **cgu.html.twig** :

Les balises *<i>* (italique) sont remplacées par les balises *<em>* :

```
(ci-après "les Utilisateurs")
```

```
(ci-après "le site").
```

Dans **basefront.html.twig** :

```
<p><small> Consultez nos Conditions Générales d'Utilisation </small></p>
```

✗ *Code avant correction :*

Dans `accueil.html.twig` :

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

Dans `playlists.html.twig` :

```
<table class="table table-striped">
```

Dans `formations.html.twig` :

```
<table class="table table-striped">
```

*Erreur relevée par Sonarlint :*

**"<table>" tags should have a description**

Web:TableWithoutCaptionCheck ⚡ MINOR 🐛 BUG 🛡️ [accessibility](#), [wcag2-a](#)

Cette erreur indique qu'il n'y a pas d'attributs pour décrire le contenu du tableau

Il faut donc rajouter la balise `<caption>` afin d'améliorer l'accessibilité et fournir une explication du contenu du tableau :

✓ *Code après correction :*

Dans `accueil.html.twig` :

```
<table class="table">
 <caption> table pour les deux dernières formations ajoutées au catalogue</caption>
```

Dans `playlists.html.twig` :

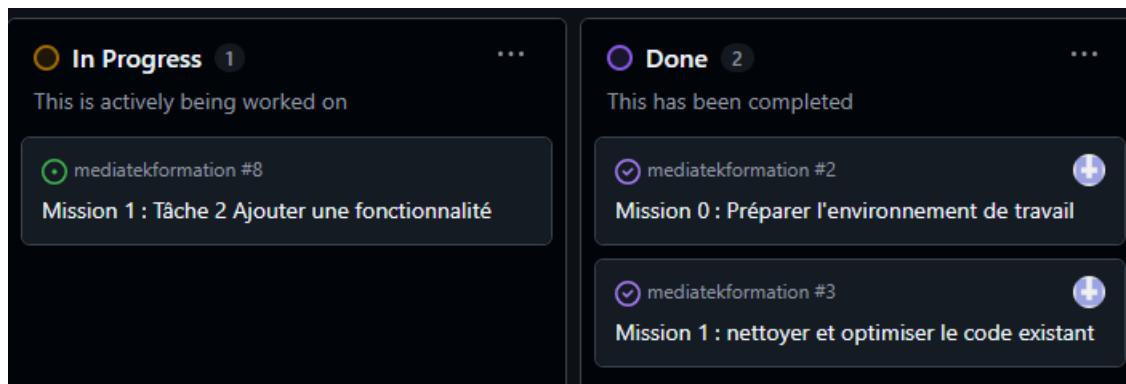
```
<table class="table table-striped">
 <caption>Liste des playlists</caption>
```

Dans `formations.html.twig` :

```
<table class="table table-striped">
 <caption>Liste des formations</caption>
 <thead>
```

## TACHE 2 : AJOUTER UNE FONCTIONNALITE

»» 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.



- Ajout dans `PlaylistRepository` d'une méthode qui permet de récupérer toutes les playlists triées en fonction du nombre total de formations associées à chaque playlist :
- Définition du paramètre `$ordre` pour spécifier le tri
  - Construction de la requête avec Doctrine :
    - Utilisation de `createQueryBuilder('p')` pour interroger l'entity `Playlist` puis ajout d'une jointure avec l'entité `formations` à l'aide de `leftJoin('p.formations','f')` pour récupérer les formations associées et grouper les résultats par l'ID de la playlist.
    - Utilisation de la méthode `orderBy` pour trier les playlists selon le nombre total de formations en intégrant la fonction `COUNT(f.id)` pour compter le nombre de formations.
    - Finalisation de la requête avec l'appel de `getQuery()` et `getResult()` afin d'obtenir les résultats.

```
/**
 * Retourne les playlists triées en fonction du nombre total de formations
 * @param type $ordre
 * @return array
 */
public function findAllOrderByTotalNb($ordre): array{
 return $this->createQueryBuilder('p')
 ->leftJoin('p.formations', 'f')
 ->groupBy('p.id')
 ->orderBy('COUNT(f.id)', $ordre)
 ->getQuery()
 ->getResult();
}
```

➤ Dans [PlaylistsController](#) :

Ajout d'un nouveau cas de tri dans le switch de la méthode sort. Ce tri permet de choisir la méthode de récupération des playlists en fonction du critère de l'utilisateur (name, ou nb formations) et de l'ordre (ASC ou DESC). Les playlists triées et les catégories sont ensuite envoyées à la vue pour affichage.

```
#Route('/playlists/tri/{champ}/{ordre}', name: 'playlists.sort')
public function sort($champ, $ordre): Response {
 switch ($champ) {
 case "name":
 $playlists = $this->playlistRepository->findAllOrderByNome($ordre);
 break;
 case "nbFormations":
 $playlists = $this->playlistRepository->findAllOrderByTotalNb($ordre);
 break;
 default:
 throw new \InvalidArgumentException("Invalid sorting parameter: $champ");
 }
 $categories = $this->categorieRepository->findAll();
 return $this->render(self::PAGES_PLAYLISTS, [
 'playlists' => $playlists,
 'categories' => $categories
]);
}
```

➤ Gestion de la vue : mise à jour du fichier `playlist.html` :

Ajout de l'information du nombre de formation par playlist avec l'utilisation de `playlist[k].formations|length`

```
</td>
<td class="text-center">
 {{ playlists[k].formations|length }}
```

Afin de permettre le tri croissant ou décroissant sur le nombre de formations présente dans la playlist on ajoute les boutons avec le path correspondant :

```
<th class="text-center align-top" scope="col">
 Nombre de formations

 ASC
 DESC
```

Résultat avec la colonne ajoutée dans la page qui liste les playlists :

playlist	catégories	Nombre de formations
<a href="#">Base Test</a>		0
<a href="#">Bases de la programmation (C#)</a>	C# POO	74
<a href="#">Compléments Android (programmation mobile)</a>	Android	13

Il était également demandé d'afficher l'information dans la page d'une playlist :

- Ajout dans `playlist.html.twig` :

```
Nombre de formations présentes dans la playlist :
{{ playlist.formations|length }}
```

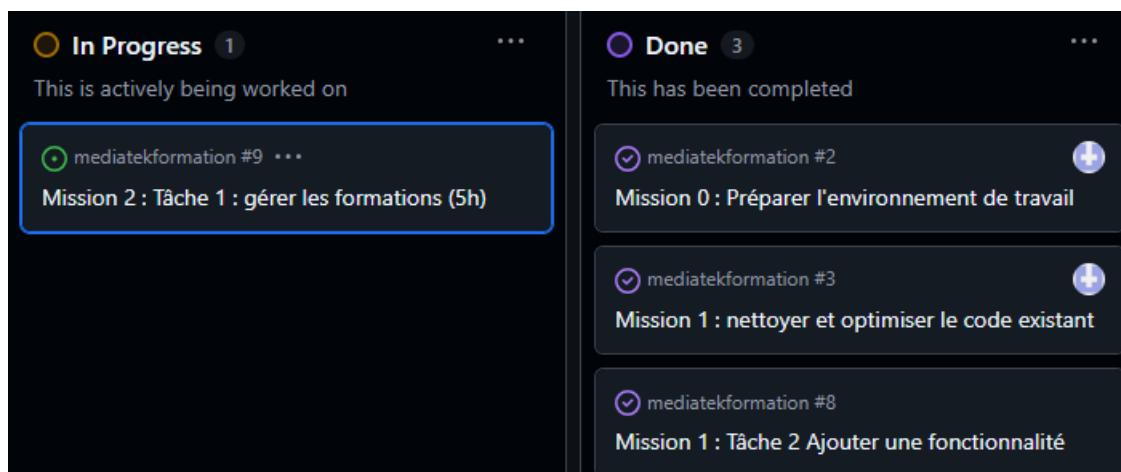
Résultat avec la colonne ajoutée sur la page d'une playlist en particulier :

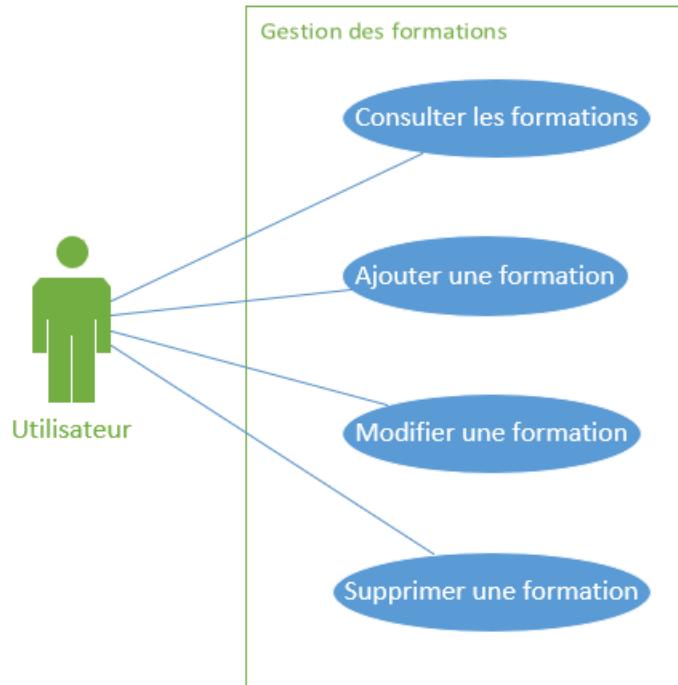
<b>Eclipse et Java</b>		Eclipse n°1 : installation de l'IDE
<b>catégories :</b> Java UML		
<b>Nombre de formations présentes dans la playlist :</b> 7		Eclipse n°2 : rétroconception avec ObjectAid
<b>description :</b> Utilisation de l'IDE Eclipse et développement en Java.		Eclipse n°3 : GitHub et Eclipse

## MISSION 2 : CODER LA PARTIE BACK-OFFICE

### TACHE 1 : GERER LES FORMATIONS

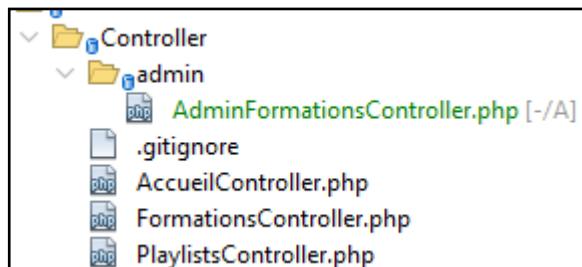
- »» Une page doit permettre de lister les formations et, pour chaque formation, afficher un bouton permettant de la supprimer (après confirmation) et un bouton permettant de la modifier. Si une formation est supprimée, il faut aussi l'enlever de la playlist où elle se trouvait.
- »» Les mêmes tris et filtres présents dans le front office doivent être présents dans le back office.
- »» Un bouton doit permettre d'accéder au formulaire d'ajout d'une formation. Les saisies doivent être contrôlées. Seul le champ "description" n'est pas obligatoire ainsi que la sélection de catégories (une formation peut n'avoir aucune catégorie). La playlist et la ou les catégories doivent être sélectionnées dans une liste (une seule playlist par formation, plusieurs catégories possibles par formation). La date ne doit pas être saisie mais sélectionnée. Elle ne doit pas être postérieure à la date du jour
- »» Le clic sur le bouton permettant de modifier une formation doit amener sur le même formulaire, mais cette fois prérempli.



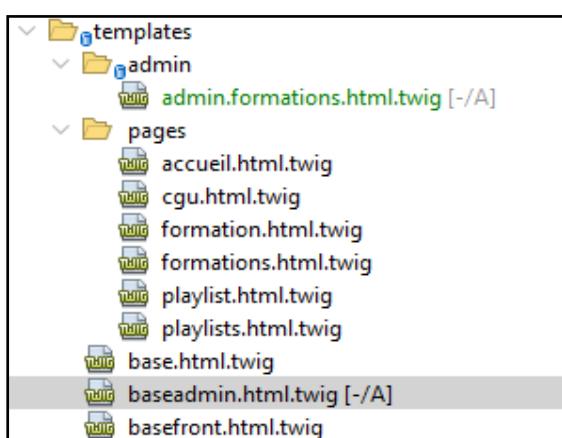


Création des dossiers et fichiers nécessaires :

- Ajout du dossier admin dans Controller et création du fichier [AdminFormationsController](#)



- Ajout d'un dossier admin dans templates et création du fichier [admin.formations.html.twig](#) et de [baseadmin.html.twig](#)



➤ Création de la classe [AdminFormationsController](#) :

- Création du constructeur pour accéder aux données des formations et des catégories.
- Création de la méthode index pour récupérer les enregistrements et les envoyer à la vue : [admin.formations.html.twig](#)

```
class AdminFormationsController extends AbstractController {

 const PAGES_FORMATIONS_ADMIN = "admin/admin.formations.html.twig";

 /**
 * Accès aux formations
 * @var FormationRepository
 */
 private $formationRepository;

 /**
 * Accès aux catégories
 * @var CategorieRepository
 */
 private $categorieRepository;

 /**
 * Crédit du constructeur
 * @param FormationRepository $formationRepository
 * @param CategorieRepository $categorieRepository
 */
 function __construct(FormationRepository $formationRepository, CategorieRepository $categorieRepository) {
 $this->formationRepository = $formationRepository;
 $this->categorieRepository = $categorieRepository;
 }

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

➤ Dans la page `admin.formations.html.twig` :

Boucle for pour parcourir toutes les formations et affichage des détails de celles-ci : tire, playlist, catégories, date de publication.

```
{% for formation in formations %}
 <tr class="align-middle">
 <td>
 <h5 class="text-info">
 {{ formation.title }}
 </h5>
 </td>
 <td class="text-left">
 {{ formation.playlist.name }}
 </td>
 <td class="text-left">
 {% for categorie in formation.categories %}
 {{ categorie.name }}

 {% endfor %}
 </td>
 <td class="text-center">
 {{ formation.publishedatstring }}
 </td>
 <td class="text-center">
 {% if formation.miniature %}
 Modifier

 {% endif %}
 </td>
 </tr>
{% endfor %}
```

Création des boutons permettant supprimer et modifier la formation avec les path correspondants :

```
Modifier
Supprimer
```

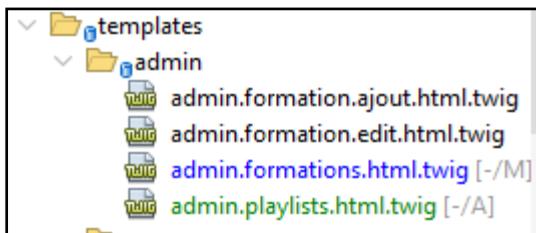
Création d'une nouvelle route dans [AdminFormationsController](#) :

Cette méthode permet de supprimer une formation.

Si la formation est associée à une playlist, la méthode `removeFormation()` est utilisée pour la retirer de cette playlist avant de supprimer la formation de la base de données. Un message de confirmation est ensuite affiché et l'utilisateur est redirigé vers la page des formations

```
/**
 * Suppression d'une formation
 * @param Formation $formation
 * @return Response
 */
#[Route('/admin/suppr/{id}', name: 'admin.formations.suppr')]
public function suppr(Formation $formation): Response {
 if ($formation) {
 //récupération de la playlist associée à la formation
 $playlist = $formation->getPlaylist();
 //si la formation est liée à une playlist retrait de celle ci
 if ($playlist) {
 $playlist->removeFormation($formation);
 }
 //Suppression dans la bdd
 $this->formationRepository->remove($formation);
 }
 $this->addFlash('danger', 'La formation' . $formation->getTitle() . ' a été supprimée avec succès.');
 return $this->redirectToRoute('admin.formations');
}
```

Création des nouveaux fichiers twig : `admin.formation.edit` et `admin.formation.ajout`:



Création du formulaire :

Dans une fenêtre de commande dans le dossier du projet je tape la commande :

Php bin/console make :form

```
Administrator : Invite de commandes
Microsoft Windows [version 10.0.22631.4317]
(c) Microsoft Corporation. Tous droits réservés.

C:\Windows\System32>cd C:\wamp64\www\mediatekformation
C:\wamp64\www\mediatekformation>php bin/console make:form

The name of the form class (e.g. DeliciousPuppyType):
> FormationType

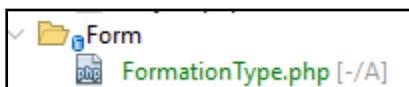
The name of Entity or fully qualified model class name that the new form will be bound to (empty for none)
:
> Formation

created: src/Form/FormationType.php

Success!

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

C:\wamp64\www\mediatekformation>
```



Le formulaire va permettre la modification ou l'ajout d'une nouvelle formation avec le renseignement de plusieurs données qui vont également être contrôlées :

- La date de publication :

- o On utilise html 5 => true afin d'activer l'utilisation du sélecteur de date permettant ainsi à l'utilisateur de choisir une date via un calendrier interactif.
- o **required => true** garanti que la date doit obligatoirement être renseignée avant de soumettre le formulaire.
- o Pour s'assurer que la date de publication n'est pas postérieure à aujourd'hui, une contrainte de validation est ajoutée dans l'entité Formation.

```
#[ORM\Column(type: Types::DATETIME_MUTABLE, nullable: true)]
#[Assert\LessThanOrEqual('today', message:"La date de publication ne peut pas être postérieure à aujourd'hui")]
private ?\DateTimeInterface $publishedAt = null;
```

- o On ajoute également un placeholder pour définir un texte d'exemple dans le champs de saisie ainsi qu'un onkeydown=> return false qui empêche l'utilisateur de saisir la date manuellement et limite la saisie au sélecteur de date.

- Seuls les champs description et la sélection de catégories ne sont pas obligatoires c'est pourquoi on utilise ici required=> false.

```
$builder
 ->add('publishedAt', DateTimeType::class, [
 'label' => 'Date de publication',
 'widget' => 'single_text',
 'data' => isset($options['data']) &&
 $options['data']->getPublishedAt() != null ? $options['data']->getPublishedAt() : new DateTime('now'),
 'html5' => true, // Utiliser le sélecteur de date natif HTML5
 'required' => true,
 'attr' => [
 'placeholder' => 'jj/mm/aaaa',
 'onkeydown' => 'return false;', // Empêche la saisie de texte
],
])
 ->add('title', TextType::class, [
 'label' => 'Titre',
 'required' => true,
])
 ->add('description', TextareaType::class, [
 'required'=> false,
])
 ->add('categories', EntityType::class, [
 'class' => Categorie::class,
 'choice_label' => 'name',
 'multiple' => true,
 'expanded' => true,
 'required' => false,
])
 ->add('videoId', TextType::class, [
 'required' => false,
])
 ->add('playlist', EntityType::class, [
 'class' => Playlist::class,
 'choice_label' => 'name',
 'multiple' => false,
 'required' => true,
])
 ->add('submit', SubmitType::class, [
```

➤ Création d'une nouvelle route dans [AdminFormationController](#) :

- Lorsqu'un utilisateur clique sur le bouton Modifier, symfony récupère l'ID de la formation à partir de l'URL.  
L'ID est alors utilisé pour récupérer l'objet Formation correspondant en base de données.
- Génération du formulaire avec `createForm(FormationType :class, $formation)` qui permet de pré remplir les champs avec les données existantes de la formation.
- Le formulaire est ensuite envoyé à la vue. Lors de la soumission ce celui-ci Symfony récupère et vérifie les données. Si celles-ci sont valides, la méthode `add` du repository est appelée pour enregistrer les modifications en base de données.
- Un message de confirmation est affiché à l'utilisateur et il est redirigé vers la liste des formations.

```
/**
 * Modification d'une formation
 * @param Formation $formation
 * @param Request $request
 * @return Response
 */
#[Route('/admin/edit/{id}', name: 'admin.formation.edit')]
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);
 $this->addFlash('success', 'La modification de la formation ' . $formation->getTitle() . ' a bien été prise en compte.');
 return $this->redirectToRoute('admin.formations');
 }

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

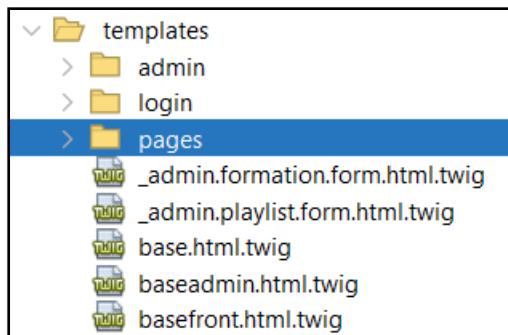
Pour l'ajout d'une formation :

```
/**
 * Ajout d'une formation
 * @param Request $request
 * @return Response
 */
#[Route('/admin/ajout', name: 'admin.formation.ajout')]
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);
 $this->addFlash('success', 'La formation ' . $formation->getTitle() . ' a été ajoutée au catalogue.');
 return $this->redirectToRoute('admin.formations');
 }

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

Création d'un nouveau fichier `_admin.formation.form.html.twig` qui contient la structure du formulaire. Ce fichier sera réutilisé à la fois pour la vue de modification et l'ajout de formations.



```
/**
 * Tri des formations en fonction du champs et ordre spécifié
 * @param type $champ
 * @param type $ordre
 * @param type $table
 * @return Response
 */
#[Route('/admin/tri/{champ}/{ordre}/{table}', name: 'admin.formation.sort')]
public function sort($champ, $ordre, $table = ""): Response {
 $formations = $this->formationRepository->findAllOrderBy($champ, $ordre, $table);
 $categories = $this->categorieRepository->findAll();
 return $this->render(self::PAGES_FORMATIONS_ADMIN, [
 'formations' => $formations,
 'categories' => $categories
]);
}
```

Il est ensuite inclus dans la vue de modification de formation

`admin.formation.edit.html.twig`.

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

Ajouter les mêmes tris et filtres que dans le front office :

Création de nouvelles routes dans le controller des formations :

- `Admin.formation.sort` : tri des formations en fonction d'un champ et d'un ordre croissant ou décroissant

- `Admin.formations.findallcontain` : permet de trier les formations selon un critère saisi par l'utilisateur ( titre, playlist, catégorie)

Ajout des boutons de tri dans la vue :

Tri par titre ASC / DESC

```


```

Tri par playlist

```


```

Tri par date

```


```

Ajout des formulaires :

Filtre par titre de formation avec ajout d'une protection CSRF et génération d'un token unique

```
<form class="form-inline mt-1" method="POST" action="{{ path('admin.formations.findallcontain', {champ:'title'}) }}">
 <div class="form-group mr-1 mb-2">
 <input type="text" class="sm" name="recherche"
 value="{% if valeur|default and not table|default %}{{ valeur }}{% endif %}">
 <input type="hidden" name="_token" value="{{ csrf_token('filtre_title') }}">
 <button type="submit" class="btn btn-info mb-2 btn-sm">filtrer</button>
 </div>
</form>
```

```
/**
 * Filtre les formations en fonction de la valeur saisie par l'utilisateur
 * @param type $champ
 * @param Request $request
 * @param type $table
 * @return Response
 */
#[Route('/admin/recherche/{champ}/{table}', name: 'admin.formations.findallcontain')]
public function findAllContain($champ, Request $request, $table = ""): Response {
 $valeur = $request->get("recherche");
 $formations = $this->formationRepository->findByContainValue($champ, $valeur, $table);
 $categories = $this->categorieRepository->findAll();
 return $this->render(self::PAGES_FORMATIONS_ADMIN, [
 'formations' => $formations,
 'categories' => $categories,
 'valeur' => $valeur,
 'table' => $table
]);
}
```

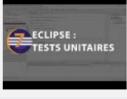
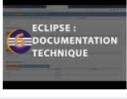
## Filtre par nom de playlist

```
<form class="form-inline mt-1" method="POST" action="{{ path('admin.formations.findallcontain', {champ:'name', table:'playlist'}) }}">
 <div class="form-group mr-1 mb-2">
 <input type="text" class="sm" name="recherche"
 value="<% if valeur|default and table|default and table=='playlist' %>{ valeur }%<% endif %>">
 <input type="hidden" name="_token" value="{{ csrf_token('filtre_name') }}>
 <button type="submit" class="btn btn-info mb-2 btn-sm">filtrer</button>
 </div>
</form>
```

## Filtre par catégorie avec choix dans une liste

```
<form class="form-inline mt-1" method="POST" action="{{ path('admin.formations.findallcontain', {champ:'id', table:'categories'}) }}">
 <select class="form-select form-select-sm" name="recherche" id="recherche" onchange="this.form.submit()>
 <option value=""></option>
```

## Page de gestion des formations :

Gestion des formations					Ajouter une nouvelle formation
formation	playlist	catégories	date	action	
<a href="#">Eclipse n°7 : Tests unitaires</a>	Eclipse et Java	Java	02/01/2021		<a href="#">Modifier</a> <a href="#">Supprimer</a>
<a href="#">Eclipse n°6 : Documentation technique</a>	Eclipse et Java	Java	30/12/2020		<a href="#">Modifier</a> <a href="#">Supprimer</a>
<a href="#">Eclipse n°5 : Refactoring</a>	Eclipse et Java	Java	29/12/2020		<a href="#">Modifier</a> <a href="#">Supprimer</a>

## Page de modification d'une formation avec le formulaire prérempli :

### Détail de la formation

Date de publication

Video id

Titre

Playlist

Categories

Java  
 UML  
 C#  
 Python  
 MCD  
 Android  
 POO  
 SQL  
 Cours

Description

Intégration de JUnit dans l'application et création de tests unitaires.  
00:07 : rappel sur le principe du test unitaire

## Page d'ajout d'une nouvelle formation avec le formulaire à renseigner :

### Ajout d'une nouvelle formation

Date de publication

31/01/2025

Titre

Playlist

Categories

Java  
 UML  
 C#  
 Python  
 MCD  
 Android  
 POO  
 SQL  
 Cours

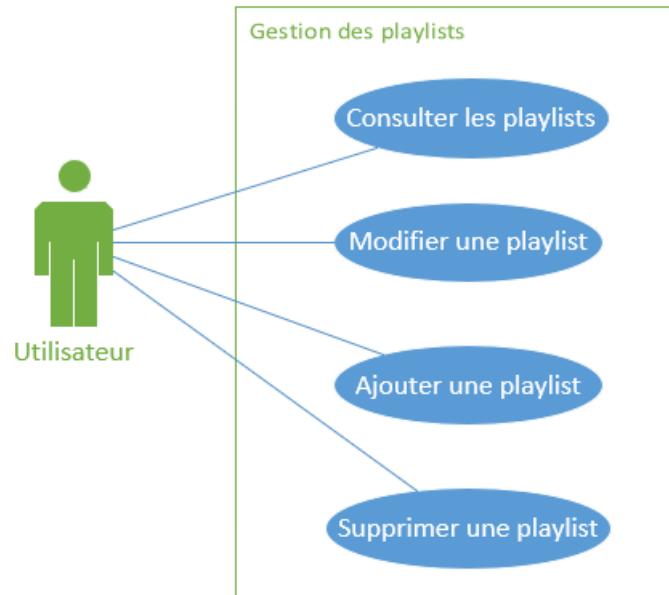
Description

## TACHE 2 : GERER 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. La suppression d'une playlist n'est possible que si aucune formation n'est rattachée à elle.
- »» Les mêmes tris et filtres présents dans le front office doivent être présents dans le back office.
- »» Un bouton doit permettre d'accéder au formulaire d'ajout d'une playlist. Les saisies doivent être contrôlées. L'ajout d'une playlist consiste juste à saisir son nom et sa description. Seul le champ name est obligatoire.
- »» Le clic sur le bouton permettant de modifier une playlist doit amener sur le même formulaire, mais cette fois prérempli. Cette fois, la liste des formations de la playlist doit apparaître, mais il ne doit pas être possible d'ajouter ou de supprimer une formation : ce n'est que dans le formulaire de la formation qu'il est possible de préciser sa playlist de rattachement.

The screenshot shows a digital workspace interface with two main columns: 'In Progress' on the left and 'Done' on the right. Both columns have a header with a status icon, a count, and three dots for more options.

- In Progress:** Status: **In Progress** (1). Description: This is actively being worked on. A card is highlighted with a blue border: **mediatekformation #10** ... **Mission 2 : tâche 2 : Gérer les playlists (5h)**.
- Done:** Status: **Done** (4). Description: This has been completed. It lists four completed missions with checkmarks and a plus icon:
  - mediatekformation #2** ... **Mission 0 : Préparer l'environnement de travail**
  - mediatekformation #3** ... **Mission 1 : nettoyer et optimiser le code existant**
  - mediatekformation #8** ... **Mission 1 : Tâche 2 Ajouter une fonctionnalité**
  - mediatekformation #9** ... **Mission 2 : Tâche 1 : gérer les formations (5h)**



Pour cette tâche, j'ai suivi un processus similaire à celui utilisé pour la gestion des formations. Une page est créée pour afficher la liste des playlists, avec des boutons permettant de les supprimer (après confirmation) ou de les modifier

Page de la gestion des playlists :

Gestion des playlists

La playlist Base Test a été effectuée avec succès.

Ajouter une nouvelle playlist

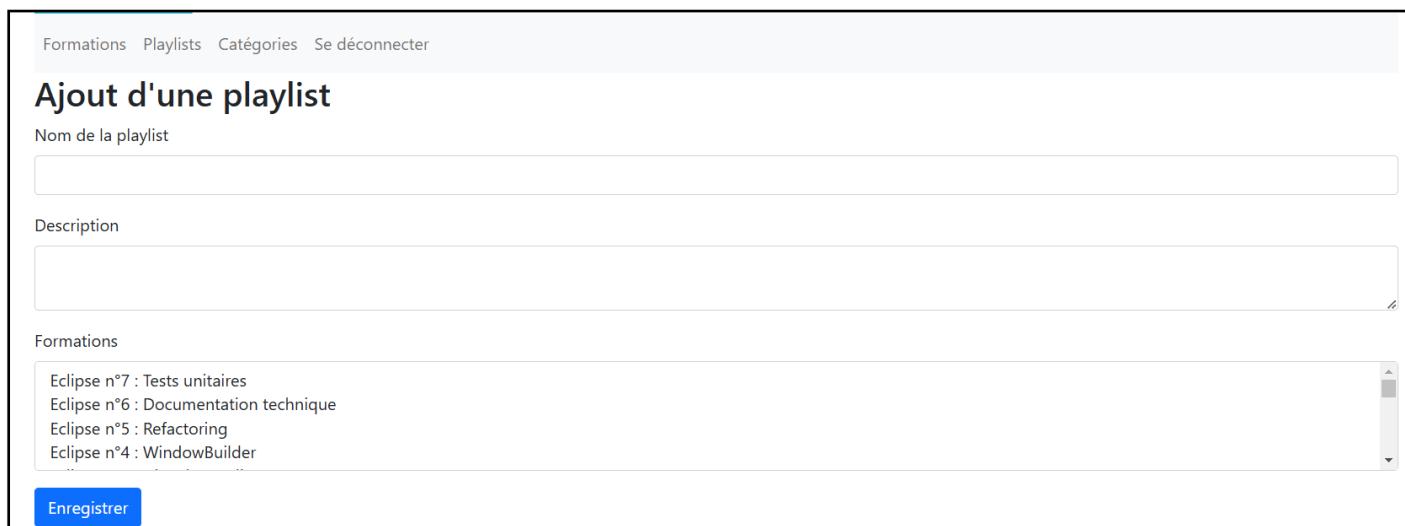
Playlist	Catégories	Nombre de formations
Base Test < > <input type="text"/> filtrer	Catégories < >	0 Modifier Supprimer
Bases de la programmation (C#)	C# POO	74 Modifier
Compléments Android (programmation mobile)	Android	13 Modifier
Cours Composant logiciel	Cours	2 Modifier
Cours Curseurs	SQL Cours POO	2 Modifier

La suppression d'une playlist n'est possible que si aucune formation ne lui est rattachée. Pour garantir cette condition, le bouton "Supprimer" ne s'affiche que lorsque la playlist ne contient aucune formation.

```
{% if playlists[k].formations|length == 0 %}
 <a href="{{ path('admin.playlist.suppr', {id:playlists[k].id}) }}" class="btn btn-danger" onclick="return confirm('Etes-vous sûr de vouloir supprimer
{% endif %}
```

Le bouton "Modifier", quant à lui, est toujours visible et permet d'accéder au formulaire de modification de la playlist.

Les options de tri et de filtre présentes dans le front office sont également implémentées dans le back office. Un bouton permet d'accéder à un formulaire d'ajout de playlist, où les saisies sont validées.



Formations Playlists Catégories Se déconnecter

### Ajout d'une playlist

Nom de la playlist

Description

Formations

- Eclipse n°7 : Tests unitaires
- Eclipse n°6 : Documentation technique
- Eclipse n°5 : Refactoring
- Eclipse n°4 : WindowBuilder

Enregistrer

L'ajout nécessite la saisie du nom (obligatoire) et de la description.

Lors de la modification d'une playlist, le formulaire est prérempli et affiche la liste des formations associées, sans possibilité d'ajouter ou de supprimer des formations, cette gestion étant réservée à l'édition de la formation elle-même.



Formations Playlists Catégories Se déconnecter

### Détail de la playlist

Nom de la playlist

Description

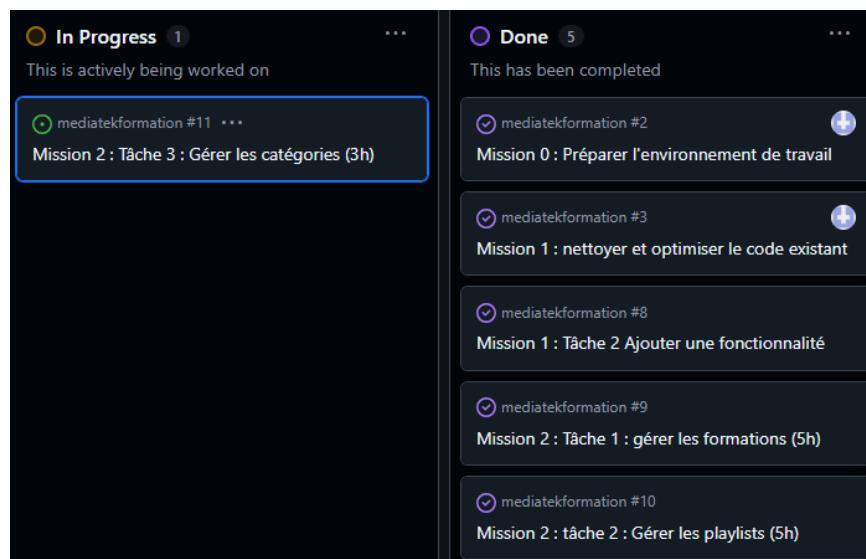
Formations

- Bases de la programmation n°2 - procédural : exercice1 (affichage)
- Bases de la programmation n°1 - procédural : premier exemple
- Exercice triggers, sql et correctifs (correction sql sujet EDC cas aeroplan 2014 BTS SIO)
- Exercice triggers, sql et correctifs (correction sql sujet EDC cas supmaster 2014 BTS SIO)

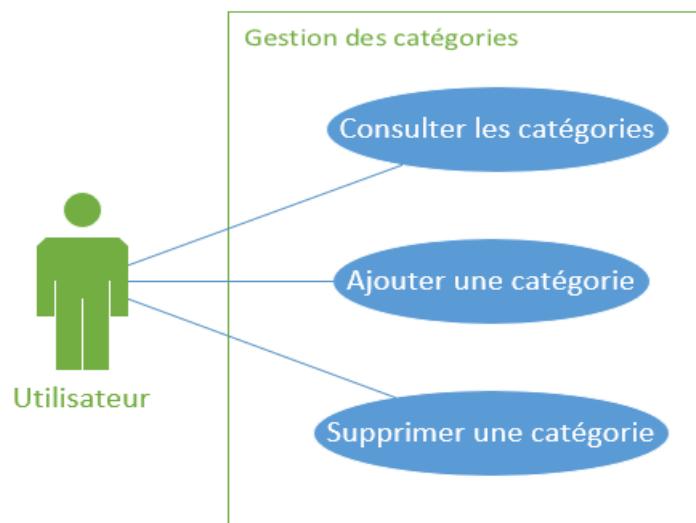
Enregistrer

## TACHE 3 : GERER LES CATEGORIES

- »» 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
- »» Dans la même page, un mini formulaire doit permettre de saisir et d'ajouter directement une nouvelle catégorie, à condition que le nom de la catégorie n'existe pas déjà.



The image shows two side-by-side screenshots of a digital workspace interface. The left screenshot shows a 'In Progress' section with one item: 'mediatekformation #11 ... Mission 2 : Tâche 3 : Gérer les catégories (3h)'. The right screenshot shows a 'Done' section with five items: 'mediatekformation #2 Mission 0 : Préparer l'environnement de travail', 'mediatekformation #3 Mission 1 : nettoyer et optimiser le code existant', 'mediatekformation #8 Mission 1 : Tâche 2 Ajouter une fonctionnalité', 'mediatekformation #9 Mission 2 : Tâche 1 : gérer les formations (5h)', and 'mediatekformation #10 Mission 2 : tâche 2 : Gérer les playlists (5h)'. Each item has a small circular icon with a checkmark or a plus sign next to it.



Pour cette tâche, j'ai suivi un processus similaire à celui utilisé pour la gestion des formations et des playlists. Une page est créée pour afficher la liste des catégories, avec des boutons permettant de les supprimer (après confirmation). Un mini formulaire est également ajouté pour permettre de saisir directement une nouvelle catégorie si son nom n'existe pas.

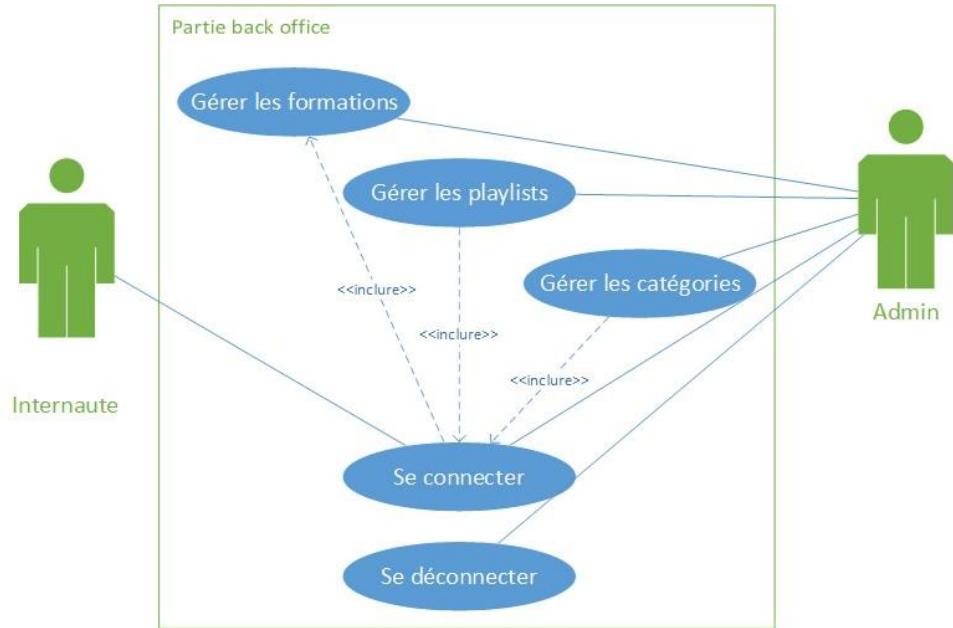
Page de gestion des catégories :

Catégories	Formations	Suppression
Java	Eclipse n°7 : Tests unitaires Eclipse n°6 : Documentation technique Eclipse n°5 : Refactoring Eclipse n°4 : WindowBuilder Eclipse n°3 : GitHub et Eclipse Eclipse n°2 : rétroconception avec ObjectAid Eclipse n°1 : installation de l'IDE TP Android n°5 : code du contrôleur et JavaDoc POO TP Java n°6 : polymorphisme POO TP Java n°5 : événements et contrôleur POO TP Java n°4 : démarrage sur le contrôleur, construction du modèle POO TP Java n°3 : interface graphique POO TP Java n°2 : MVC POO TP Java n°1 : configuration d'Eclipse	<a href="#">Delete</a>

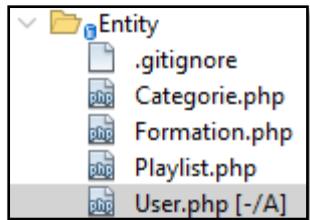
## TACHE 4 : AJOUTER L'ACCES AVEC AUTHENTIFICATION

- »» Le back office ne doit être accessible qu'après authentification : un seul profil administrateur doit avoir le droit d'accès.
- »» Il doit être possible de se déconnecter, sur toutes les pages (avec un lien de déconnexion).

<b>In Progress</b> 1	<b>Done</b> 6
This is actively being worked on	This has been completed
<b>mediatekformation #12</b> Mission 2 : Tâche 4 : Ajouter l'accès avec authentification	<b>mediatekformation #11</b> Mission 2 : Tâche 3 : Gérer les catégories (3h)
	<b>mediatekformation #10</b> Mission 2 : tâche 2 : Gérer les playlists (5h)



Le bundle Security étant déjà installé, une classe User est créée et enregistrée en base données afin de mémoriser les utilisateurs qui auront des autorisations d'accès.



En suivant la documentation de Symfony (partie sécurité) on effectue ensuite les modifications nécessaire dans le fichier « `security.yaml` » afin de paramétrier l'authentification :

On précise ici que pour accéder au chemin « `/admin` », il faut avoir le « `ROLE_ADMIN` ».

```

access control:
 - { path: ^/admin, roles: ROLE_ADMIN }

```

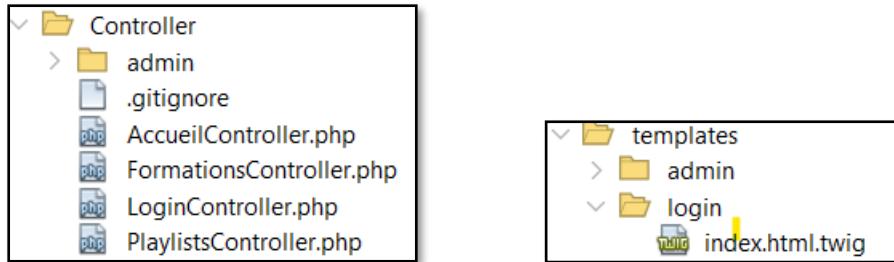
Création d'un formulaire d'authentification en utilisant la commande suivante :

```

Php bin/console
make :controller Login

```

Création de `LoginController.php` dans le dossier Controller et de la page `index.html.twig` dans le dossier `templates>login` :



Au niveau du contrôleur la route permettant d'afficher le formulaire de connexion est définie comme suit :

```

class LoginController extends AbstractController
{
 #[Route('/login', name: 'app_login')]

```

Activation de l'authentification par formulaire :

Dans un premier temps, les paramètres d'authentification par formulaire doivent être activés dans le fichier « security.yaml ». Cette configuration fait référence à la route définie dans le contrôleur pour l'affichage du formulaire de connexion :

```

form_login:
 #app login : nom de la route définie dans le contrôleur
 login_path: app_login
 check_path: app_login
 default_target_path: admin.formations
 enable_csrf: true

```

- **Login\_path** définit la route vers laquelle l'utilisateur est dirigé pour accéder au formulaire de connexion. Elle correspond à la route définie dans le contrôleur, ici **app\_login**.
- **Check\_path** est la route où les informations du formulaire sont envoyées pour être vérifiées.
- **Default\_target\_path** indique la route vers laquelle l'utilisateur est redirigé après une authentification réussie, ici la page de gestion des formations (**admin.formations**).
- **Enable\_csrf** active la protection CSRF pour sécuriser le formulaire. Il est donc nécessaire d'ajouter un champ de token en masqué dans la vue.

Modification du contrôleur pour gérer les erreurs d'authentification :

Dans un second temps, la modification du contrôleur est nécessaire afin d'afficher les éventuelles erreurs si l'utilisateur ne saisit pas des informations correctes et réafficher le dernier login.

Pour cela on utilise **AuthenticationUtils** fourni par Symfony afin de récupérer les informations liées à l'authentification :

- getLastAuthenticationError() : permet d'obtenir l'erreur d'authentification la plus récente, par exemple en cas de saisie d'un mot de passe ou login incorrect.
- getLastUsername() : récupère le dernier nom d'utilisateur saisi lors de la dernière tentative de connexion afin de le réafficher dans le formulaire.

```
public function index(AuthenticationUtils $authenticationUtils): Response
{
 //recupération si erreur
 $error = $authenticationUtils->getLastAuthenticationError();
 //recupération éventuelle du dernier nom de login utilisé
 $lastUsername = $authenticationUtils->getLastUsername();
 return $this->render('login/index.html.twig', [
 'last_username'=> $lastUsername,
 'error'=>$error
]);
}
```

### Gestion de la vue [index.html.twig](#) :

- Le formulaire vérifie d'abord si des erreurs sont à afficher suite à une soumission incorrecte (comme défini dans le contrôleur).
- La valeur {{ last\_username }} permet de préremplir la zone saisie avec le nom d'utilisateur lors d'une précédente soumission.

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

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

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

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

 <button class="btn btn-lg btn-primary" type="submit"> Se connecter</button>
 </form>

 {% endblock %}
```

## ➤ Gestion de la déconnexion :

Pour gérer la déconnexion il est nécessaire d'ajouter une méthode qui récupère la route dans le contrôleur :

```
//Gérer la déconnexion
#[Route('/logout', name:'logout')]
public function logout()
```

Une fois la route fixée il faut configurer logout dans le firewall du fichier [security](#) afin que Symfony gère automatiquement la déconnexion de l'utilisateur :

```
logout:
 path: logout
```

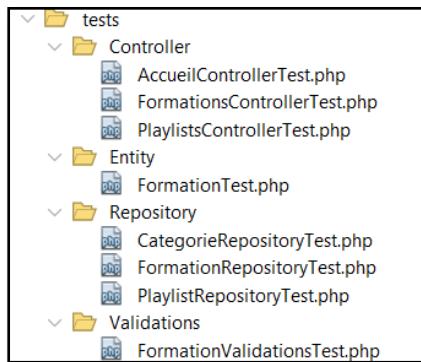
## MISSION 3 : TESTER ET DOCUMENTER

### TACHE 1 : GERER LES TESTS

The screenshot shows a task management interface with two main sections:

- In Progress (1):** This task is actively being worked on. It was created by mediatekformation #13 and is titled "Mission 3 : Tâche 1 : Gérer les tests (7h)".
- Done (7):** This task has been completed. It was created by mediatekformation #12 and is titled "Mission 2 : Tâche 4 : Ajouter l'accès avec authentification".

Pour réaliser les différents tests, j'ai mis en place un dossier tests contenant plusieurs sous-dossiers :



**Controller** : regroupe les tests fonctionnels. Les classes de test héritent de `WebTestCase`.

**Entity** : contient un test unitaire pour vérifier le bon fonctionnement de la méthode qui retourne la date de parution sous forme de chaîne. Ici, la classe de test hérite de `TestCase`.

**Repository** : regroupe les tests d'intégration pour les méthodes des repositories Catégorie, Formation et Playlist. Ces tests nécessitant un accès au noyau de l'application, ils utilisent la classe KernelTestCase.

**Validations** : inclut un test d'intégration pour vérifier que lors de l'ajout ou de la modification d'une formation, la date de publication ne peut pas être postérieure à aujourd'hui. Ce test utilise également KernelTestCase.

Afin de tester la compatibilité avec Chrome et Firefox j'ai utilisé Selenium qui est un Framework de test permettant d'enregistrer un scénario et de le rejouer.

The screenshot shows a Selenium IDE test script titled "Untitled". The script consists of the following steps:

Step	Action	Target
2	✓ set window size	1936x1048
3	✓ click	linkText=Formations
4	✓ run script	window.scrollTo(0,138)
5	✓ click	css= align-middle:nth-child(1) img
6	✓ click	linkText=Playlists
7	✓ click	linkText=Voir détail
8	✓ mouse over	linkText=Bases de la programmation n°3 - procédural : exercice2 (saisie)
9	✓ click	linkText=Accueil

Below the steps, there are four input fields: Command, Target, Value, and Description. At the bottom, there are tabs for Log and Reference, and a status message: "Test1\* completed successfully".

## TACHE 2 : CREER LA DOCUMENTATION TECHNIQUE

The screenshot shows a task management interface with two columns: 'In Progress' and 'Done'. The 'In Progress' column contains one task: 'Mission 3 : Tâche 2 : Créer la documentation technique (1h)'. The 'Done' column contains one task: 'Mission 3 : Tâche 1 : Gérer les tests (7h)'. Each task has a status icon (yellow for In Progress, purple for Done), a user name ('mediatekformation'), a task ID ('#14' for In Progress, '#13' for Done), and a '...' button.

Afin de créer la documentation technique et ne prendre en compte que les classes ajoutées par le développeur j'ouvre une fenêtre de commande en mode admin :

```
"C:\wamp64\bin\php\php8.2.13\php.exe"
"C:\wamp64\bin\php\php8.2.13\ext\phpDocumentor.phar" "run" "--ansi" "--directory"
"C:/wamp64/www/mediatekformation/src" "--target"
"C:/wamp64/www/mediatekformation_doc" "--title" "mediatekformation"
```

mediatekformation

### Namespaces

- App
- Controller
- Entity
- Form
- Repository

### Packages

- Application

### Reports

- Deprecated
- Errors
- Markers

### Indices

- Files

## Application

### Table of Contents

### Classes

- C [AccueilController](#)**  
*Description of AccueilController*
- C [AdminCategoriesController](#)**  
*Description of AdminCategoriesController*
- C [AdminFormationsController](#)**  
*Controleur page admin Formations*
- C [AdminPlaylistsController](#)**  
*Controleur page admin Playlists*
- C [FormationsController](#)**  
*Controleur des formations*

## TACHE 3 : CREER LA DOCUMENTATION UTILISATEUR

<span style="color: orange;">●</span> In Progress <span style="color: orange;">1</span>	<span style="color: purple;">●</span> Done <span style="color: purple;">9</span>
This is actively being worked on	This has been completed
<span style="color: green;">●</span> mediatekformation #15 Mission 3 : Tâche 3 : Créer la documentation utilisateur (2h)	<span style="color: purple;">●</span> mediatekformation #14 ... Mission 3 : Tâche 2 : Créer la documentation technique (1h)

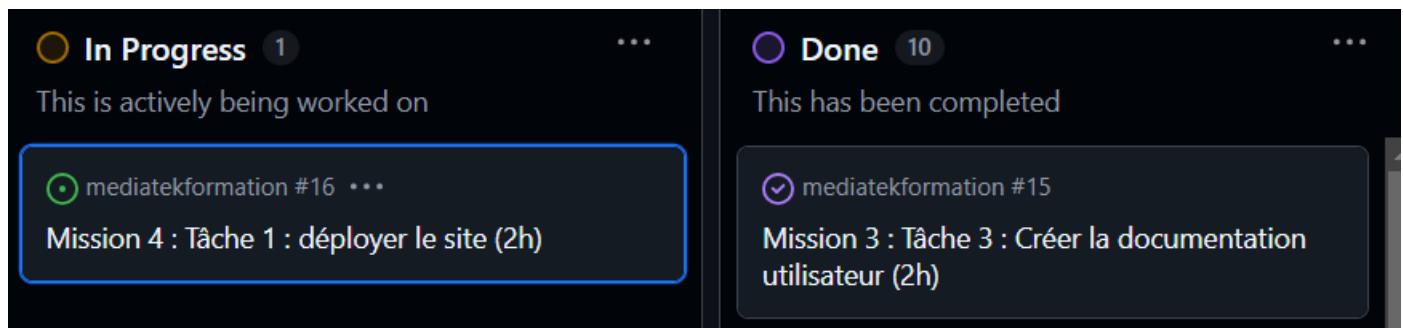
The screenshot shows the homepage of the **MediaTek86** website. The header includes a logo of a person working on a computer, the text "Des formations pour tous sur des outils numériques", and navigation links for Accueil, Formations, Playlists, and Administration. Below the header, a welcome message and a brief description of the site's purpose are displayed. A section titled "Bienvenue sur le site de MediaTek86 consacré aux formations en ligne" provides information on how to use the site, including links to Formations and Playlists. A "Recent additions" section shows two video thumbnails for "ECLIPSE : DÉPLOIEMENT" and "ECLIPSE : TESTS UNITAIRES", each with a timestamp (14/11/2024 and 02/01/2021), a playlist (Eclipse et Java), and a category (Java). At the bottom, a video player displays a presentation titled "Présentation Mediatekformation" with a play button and a progress bar from 00:00:00 to 00:05:01.

La vidéo de présentation de l'application est disponible sur la page dédiée à l'atelier professionnel du portfolio.

# MISSION 4 : DEPLOYER LE SITE ET GERER LE DEPLOIEMENT CONTINU

## TACHE 1 : DEPLOYER LE SITE

- »» 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.



Pour le déploiement du site je me suis connectée au panel CloudPanel :

Qu'est-ce que **CloudPanel** ? C'est un panneau de contrôle d'hébergement web spécialement conçu pour faciliter la gestion des sites web, des applications et des serveurs dans un environnement cloud. Il s'agit d'un solution complète qui combine les fonctions de serveur, base de données, administration de domaine.

1. On commence par exporter la base de donnée locale « mediatek formation » au format SQL pour l'importe vers l'hébergeur. Un utilisateur de base donnée est également créé pour celle-ci.
2. Sur le panel qabox, on créé une nouvelle base de données puis on ajoute un utilisateur associé à celle-ci avec les permissions nécessaires.

The screenshot shows the 'Base de données' section of the CloudPanel interface. It includes a table with columns for 'IDENTIFIANT', 'BASE DE DONNÉES', 'PHPMYADMIN', 'PERMISSIONS', and 'ACTION'. There is also a section for 'Utilisateurs de base de données' with a table showing 'IDENTIFIANT', 'BASE DE DONNÉES', 'PERMISSIONS', and 'ACTION'.

IDENTIFIANT	BASE DE DONNÉES	PHPMYADMIN	PERMISSIONS	ACTION
adminmediatek	mediatekformation	Gérer	Lecture et écriture	Supprimer

3. Configuration ftp : Afin de transférer les fichiers du site, on configure un accès FTP qui implique la création d'un compte utilisateur FTP ainsi que l'attribution d'un mot de passe pour celui-ci.

Utilisateurs FTP			<a href="#">Ajouter utilisateur</a>
IDENTIFIANT	RÉPERTOIRE DE BASE	ACTION	
ftp-mediatekformation	/home/qabox-mediatekformation/htdocs/mediatekformation.qabox.fr/		<a href="#">Supprimer</a>

4. Transfert des fichiers via filezilla : On utilise le logiciel FileZilla pour transférer tous les fichiers du site depuis l'environnement local vers le serveur.

## TACHE 2 : GERER LA SAUVEGARDE ET LA RESTAURATION DE LA BDD

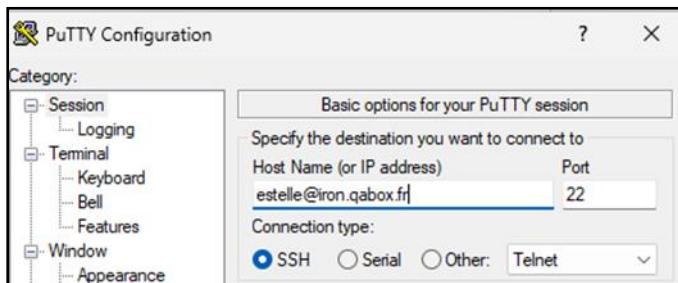
- »» Une sauvegarde journalière automatisée doit être programmée pour la BDD
- »» La restauration pourra se faire manuellement, en exécutant le script de sauvegarde.

The screenshot shows a task management interface with two columns. The left column, labeled 'In Progress' with a count of 1, contains the text 'This is actively being worked on' and a list item 'mediatekformation #17'. The right column, labeled 'Done' with a count of 11, contains the text 'This has been completed' and a list item 'mediatekformation #16'. The list item 'mediatekformation #16' is highlighted with a blue border.

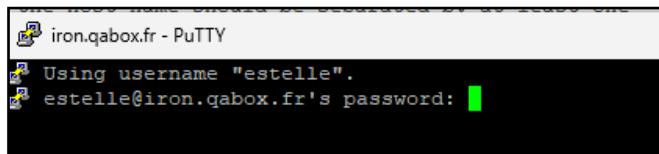
1. Créer un accès SSH via le panel Cloud Panel :

The screenshot shows the Cloud Panel interface for managing a domain. At the top, it displays the domain 'mediatekformation.qabox.fr', the site 'qabox-mediatekformation', and the IP address '163.172.229.248'. Below this, a navigation bar includes 'Paramètres', 'Vhost', 'Base de données', 'Varnish Cache', 'SSL/TLS', 'Sécurité' (which is highlighted with a red box), 'SSH/FTP' (which is also highlighted with a red box), 'Explorateur de fichiers', and 'Tâches cron'. A 'Journaux' section is also present. At the bottom, there is a table for 'Utilisateurs SSH' with a blue 'Ajouter utilisateur' button highlighted with a red box.

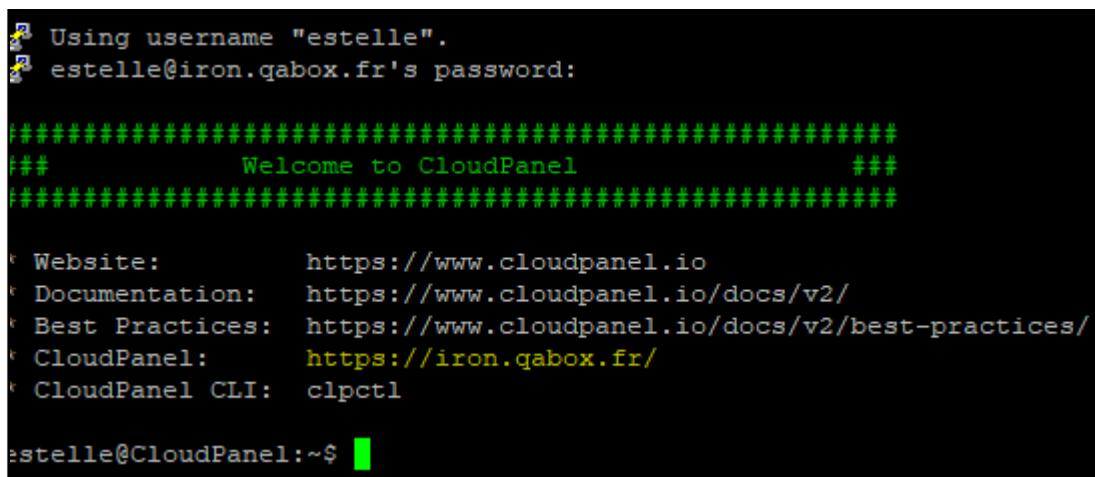
2. Télécharger le logiciel Putty qui est un client SSH pour pouvoir se connecter au serveur.
3. Dans « Host name » renseigner le nom d'hôte du serveur avec l'utilisateur : [estelle@iron.qabox.fr](mailto:estelle@iron.qabox.fr)



4. Renseigner le mot de passe indiqué lors de la création de l'utilisateur :



Nous sommes maintenant connectés au serveur et à notre compte en SSH :



5. Créer le fichier à l'aide de l'éditeur de texte « Nano » : nano backup.sh



Puis copier le script de backup :

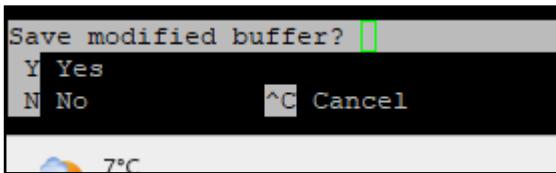
```
#!/bin/sh DATE=`date -l`
find /home/estelle/savebdd/bdd* -mtime -1 -exec rm {} \; clpctl db:export
--databaseName=mediatekformation
--file=/home/estelle/savebdd/bddbackup_${DATE}.sql.gz
```

Ici, nous n'utilisons pas mysqldump, mais la commande « clpctl » comme indiqué dans la documentation du panel :

<https://www.cloudpanel.io/docs/v2/cloudpanel-cli/site-user-commands/>

Le panel dispose de son propre outil de backup.

Une fois le script copié, il faut enregistrer le fichier :



6. Le script enregistre le backup de la base de donnée dans le dossier « **savebdd** », mais ce fichier n'existe pas, il faut donc le créer à l'aide de la commande : « **mkdir savebdd** ».

```
estelle@CloudPanel:~$ sh backup.sh
find: '/home/estelle/savebdd/bdd*': No such file or directory
Database mediatekformation has been exported.
estelle@CloudPanel:~$
```

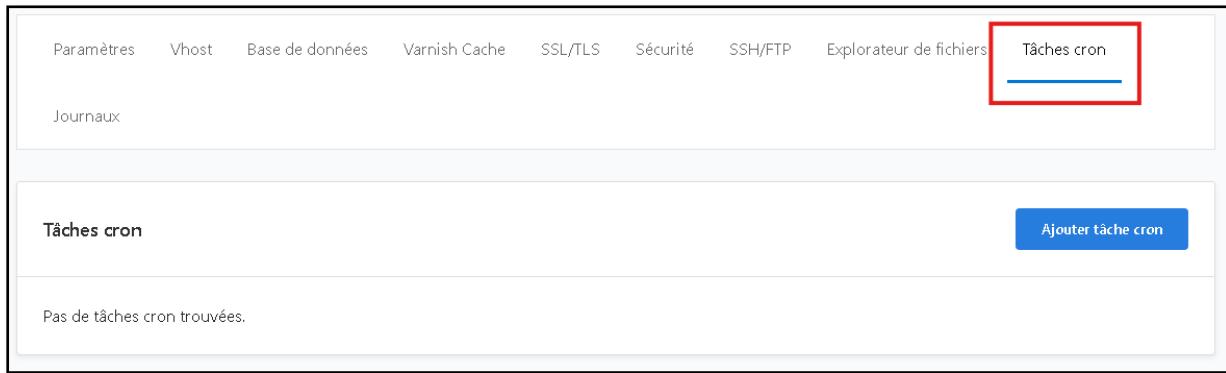
7. Ensuite le script peut être lancé à l'aide de la commande « **sh backup.sh** » :

8. Afin de vérifier que le backup a bien fonctionné, on se rends dans le dossier « **savebdd** » avec la commande « **cd savebdd** » . Ensuite on liste les fichiers avec la commande « **ls** » :

```
estelle@CloudPanel:~$ cd savebdd
estelle@CloudPanel:~/savebdd$ ls
bddbackup_2025-02-02.sql.gz
estelle@CloudPanel:~/savebdd$
```

Automatisation de la tâche à l'aide d'un « **cron** » :

## 1. Se connecter au cloud panel et aller dans le menu « tâches cron » :



## 2. Cliquer sur ajouter une tâche cron :



Modèle Crontab → Choisir la fréquence d'exécution de la tâche.

Commande à exécuter → Indiquer le chemin du script précédemment créé.



## TACHE 3 : METTRE EN PLACE LE DEPLOIEMENT CONTINU

- »»» Configurer le dépôt Github pour que le site en ligne soit mis à jour à chaque push reçu dans le dépôt.

The image shows a GitHub project board with two columns: 'In Progress' and 'Done'. The 'In Progress' column has 1 item, and the 'Done' column has 12 items. The 'In Progress' item is a card for 'Mission 4 : Tâche 3 : Mettre en place le déploiement continu (1h)'. The 'Done' item is a card for 'Mission 4 : Tâche 2 : Gérer la sauvegarde et la restauration de la BDD (1h)'.

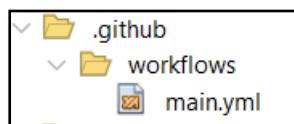
Afin de garantir que chaque push vers GitHub entraîne une mise à jour automatique du site en ligne, j'ai suivi plusieurs étapes :

1. Création du fichier yml d'automatisation : Sur GitHub dans l'onglet « Actions » → Set up a workflow yourself je créé le script de déploiement continu :

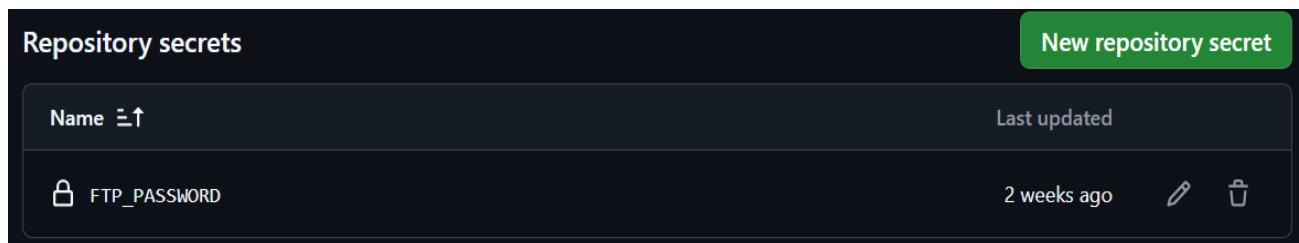
```
1 on: push
2 name: Deploy website on push
3 jobs:
4 web-deploy:
5 name: Deploy
6 runs-on: ubuntu-latest
7 steps:
8 - name: Get latest code
9 uses: actions/checkout@v3
10
11 - name: Sync files
12 uses: SamKirkland/FTP-Deploy-Action@v4.3.4
13 with:
14 server: iron.qabox.fr
15 server-dir:
16 username: ftp-mEDIATEKformation
17 password: ${{ secrets.ftp_password }}
18 timeout: 360000
19 protocol: ftp
20 port: 21
```

Après avoir rencontré plusieurs difficultés avec l'adresse du « server-dir », j'ai d'abord tenté d'utiliser l'adresse du répertoire de base du site. Cependant, cela entraînait la création d'un nouveau dossier contenant tous les fichiers au mauvais emplacement. Après plusieurs essais et ajustements, j'ai constaté que pour modifier correctement les fichiers existants, il fallait laisser l'adresse du server-dir vide.

2. Une fois le script créé un « Commit changes » est effectué ce qui ajoute le dossier .github/workflows à la racine du dépôt, contenant le fichier YML nouvellement créé.

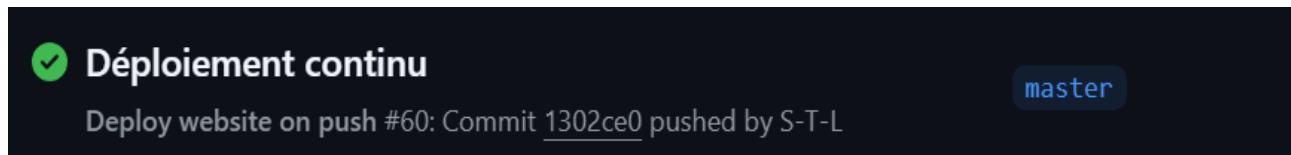


3. Dans le dépôt GitHub sous Secrets and variables → actions → new repository secret on renseigne le nom donné au fichier yml ainsi que la valeur correspondante, qui contient le mot de passe permettant d'accéder au ftp du site.

A screenshot of the "Repository secrets" page on GitHub. The page has a dark header with the title "Repository secrets" and a green button "New repository secret". Below the header, there is a table with two columns: "Name" and "Last updated". The first row shows a secret named "FTP\_PASSWORD" with a lock icon, updated "2 weeks ago". To the right of the table are two icons: a gear and a trash can. The entire screenshot is enclosed in a white box with a black border.

4. Test du déploiement continu : une modification mineure est effectuée sur une page HTML, suivie d'un commit et d'un push.

La modification est bien prise en compte sur le site en ligne, et dans l'onglet « workflow run » de GitHub on peut vérifier que le processus a été exécuté et validé avec succès.

A screenshot of the "Déploiement continu" section of the GitHub workflow run page. It shows a green checkmark icon followed by the text "Déploiement continu". Below that, it says "Deploy website on push #60: Commit 1302ce0 pushed by S-T-L". To the right, there is a blue button with the word "master". The entire screenshot is enclosed in a white box with a black border.

## BILAN

Cet atelier m'a permis d'approfondir mes connaissances avec Symfony en explorant de nouvelles syntaxes et en recherchant des informations complémentaires. J'ai particulièrement apprécié la possibilité de travailler simultanément sur les aspects front end et back end ce qui m'a offert une vision plus globale d'un projet web.

De plus ce travail m'a aidé à améliorer mon organisation de travail. En effet sur ce projet j'ai d'abord focalisé mon travail sur l'aspect pratique et le code en reportant la rédaction du bilan à la fin. Cette approche m'a malheureusement fait perdre beaucoup de temps car j'ai du revenir sur les premières missions pour retracer exactement les méthodes que j'avais mises en place.

J'ai rencontré quelques difficultés à mettre en place le déploiement continu. L'erreur provenait d'une adresse server dir incorrecte ce qui m'a conduit à de nombreux essais infructueux avant d'identifier et corriger le problème.

Contexte : MediaTek86

Situation professionnelle : Symfony

Application : mediatekformation (site de mise à disposition des auto-formations).

## ANNEXE PLAN DE TESTS

### TESTS UNITAIRES

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la méthode <code>getPublishedAtString()</code> de la classe <code>Formation</code> pour voir si elle retourne la bonne date au bon format.	Test unitaire lancé avec la date : 16/01/2025 09 : 20 : 00  Test avec une date nulle : null	La méthode doit retourner 16/01/2025  La méthode doit renvoyer une chaine vide	OK

### TESTS D'INTEGRATION

But du test	Action de contrôle	Résultat attendu	Bilan
Lors de l'ajout ou de la modification d'une formation, contrôler que la date n'est pas postérieure à aujourd'hui.	Test avec une date dans le futur (+1 year) avec une erreur attendue	La validation échoue car il y a bien une erreur avec la date dans le futur	OK

FormationRepositoryTest

Vérifier que le nombre d'enregistrements dans la table Formation est correct	Compter les enregistrements de la table Formation et le comparer avec le nombre attendu (237)	Nombre d'enregistrements présent en BDD : 237	OK
Vérifier que l'ajout d'une nouvelle formation avec add() mets bien à jour la base de donnée	Ajouter une formation et comparer le nombre total avant et après l'ajout	Le nombre total augmente de 1 après l'ajout de la formation	OK
Vérifier que la suppression d'une formation avec remove() met bien à jour la base de données	Ajouter une formation, puis la supprimer, et comparer le nombre total avant l'ajout et après la suppression	Le nombre total diminue de 1 après la suppression de la formation	OK
Vérifier le tri des formations par un champ en ordre ASC ou DESC	Appeler findAllOrderBy() avec un champ et un ordre, puis vérifier que le tri respecte l'ordre spécifié	ASC : "Android Studio (complément n°1) : Navigation Drawer et Fragment"  DESC : "UML : Diagramme de paquetages"	OK
Vérifier le filtrage des formations par une valeur spécifique dans un champ	Appeler findByContainValue() sur le champ title avec la valeur "Eclipse" et compter le nombre de résultats	Résultat = 9  "Eclipse n°8 : Déploiement"	OK
Vérifier la récupération des formations les plus récentes	Test de findAllLasted avec la date "2025/01/17 00:00:00 "	2025/01/17 00:00:00	OK
Vérifier la récupération des formations d'une playlist	Test de findAllForOnePlaylist() avec l'id 2	"C# : présentation des objets graphiques"	OK
Vérifier que le nombre d'enregistrements dans la table Playlist est correct	Compter les enregistrements de la table Playlist et le comparer avec le nombre attendu (27)	Nombre d'enregistrements présent en BDD : 27	OK

Vérifier que l'ajout d'une nouvelle playlist avec add() met bien à jour la base de données	Ajouter une playlist et comparer le nombre total avant et après l'ajout	Le nombre total augmente de 1 après l'ajout de la playlist	OK
Vérifier que la suppression d'une playlist avec remove() met bien à jour la base de données	Ajouter une playlist, puis la supprimer, et comparer le nombre total avant l'ajout et après la suppression	Le nombre total diminue de 1 après la suppression de la playlist	OK
Vérifier le tri des playlists par nom en ordre	Test de findAllOrderByNome() avec "DESC"	'Visual Studio 2019 et C#'	OK

## TESTS FONCTIONNELS

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler que la page d'accès est accessible	Test fonctionnel : assertResponseStatusCodeSame(Response::HTTP_OK)	HTTP_OK	OK
Contrôler que la page des formations est accessible	assertResponseStatusCodeSame(Response::HTTP_OK);	HTTP_OK	OK
PAGE FORMATIONS			

Vérifier le tri ASC selon le nom des playlists	Test avec un ordre ASC	Android Studio (complément n°1) : Navigation Drawer et Fragment'	OK
Vérifier le tri DESC selon le nom des playlists	Test avec un ordre DESC	C# : ListBox en couleur	OK
Vérifier le tri ASC sur le titre des formations	Test avec un ordre ASC	Android Studio (complément n°1) : Navigation Drawer et Fragment'	OK
Vérifier le tri DESC sur le titre des formations	Test avec un ordre DESC	UML : Diagramme de paquetages	OK
Vérifier le tri ASC sur la date de publication de la formation	Test avec un ordre ASC	Cours UML (8 à 11 / 33) : diagramme de classes	OK
Vérifier le tri DESC sur la date de publication de la formation	Test avec un ordre DESC	Test Formation	OK
Vérifier le filtre sur les formations	Test avec Python	Python n°18 : Décorateur singleton	OK
Vérifier le filtre sur les playlists	Test avec MCD	MCD exercice 18 : sujet 2006 (cas Credauto)	OK
Vérifier le filtre sur les catégories	Test avec id catégorie 2 (UML)	Eclipse n°2 : rétroconception avec ObjectAid'	OK
Vérifier que le lien vers les détails d'une formation fonctionne	Test du clic sur l'image de la première formation	/formations/formation/1	OK

Contrôler que la page des playlists est accessible	assertResponseStatusCodeSame(Response::HTTP_OK)	HTTP_OK	OK
Vérifier le tri Asc selon le nom des playlists	Test avec un ordre ASC	Bases de la programmation (C#)	OK
Vérifier le tri DESC selon le nom des playlists	Test avec un ordre DESC	Visual Studio 2019 et C#	OK
Vérifier le tri ASC selon le nombre de formations	Test avec un ordre ASC	Cours Informatique embarquée	OK
Vérifier le tri DESC selon le nombre de formations	Test avec un ordre DESC	Bases de la programmation (C#)	OK
Vérifier le filtre sur les playlists	Test avec programmation	Bases de la programmation (C#)	OK
Vérifier le filtre sur les catégories	Test avec la catégorie id 8 (SQL)	Cours Curseurs	OK
Vérifier que le lien vers le détail d'une playlist fonctionne	Test du clic sur le lien -> Voir le détail	/playlists/playlist/13	OK

## TESTS DE COMPATIBILITE

But du test	Action de contrôle	Résultat attendu	Bilan
Vérifier la compatibilité sur firefox et chrome	Création d'un scénario avec Selenium	Aucune erreur	OK

