

Table des matières

MEDIATEKFORMATION	2
Mission 1 : Nettoyer et optimiser le code existant	3
Tache 1 : nettoyer le code.....	3
Problèmes rencontrées et solutions	3
Tache 2	14
Mission 2 : coder la partie back-office	17
Tâche 1 : gérer les formations.....	17
Tâche 2 : gérer les playlists.....	22
Tache 3 : gérer les catégories.....	28
Tâche 4 : ajouter l'accès avec authentification	31
Mission 3 : tester et documenter	36
Tache 1 : gérer les tests.....	36
Tâche 2 : créer la documentation technique	38
Tâche 3 : créer la documentation utilisateur	41
Mission 4 : déployer le site et gérer le déploiement continu	42
Tâche 1 : déployer le site.....	42
Tâche 2 : gérer la sauvegarde et la restauration de la BDD	44
Tâche 3 : mettre en place le déploiement continu	45
Bilan final de l'application	47

MEDIATEKFORMATION

Contexte de la mission

Le chef de projet nous a mis en charge de corriger les erreurs du premier développeur en charge de la création de l'application (corriger les manquements aux bonnes pratiques de codage et l'oublie d'ajout de fonctionnalité présente dans le cahier des charges initial) et de finaliser l'application dans sa partie back-office afin de déployer le site.

Langages utilisés

- PHP
- HTML
- CSS

Technologies utilisées

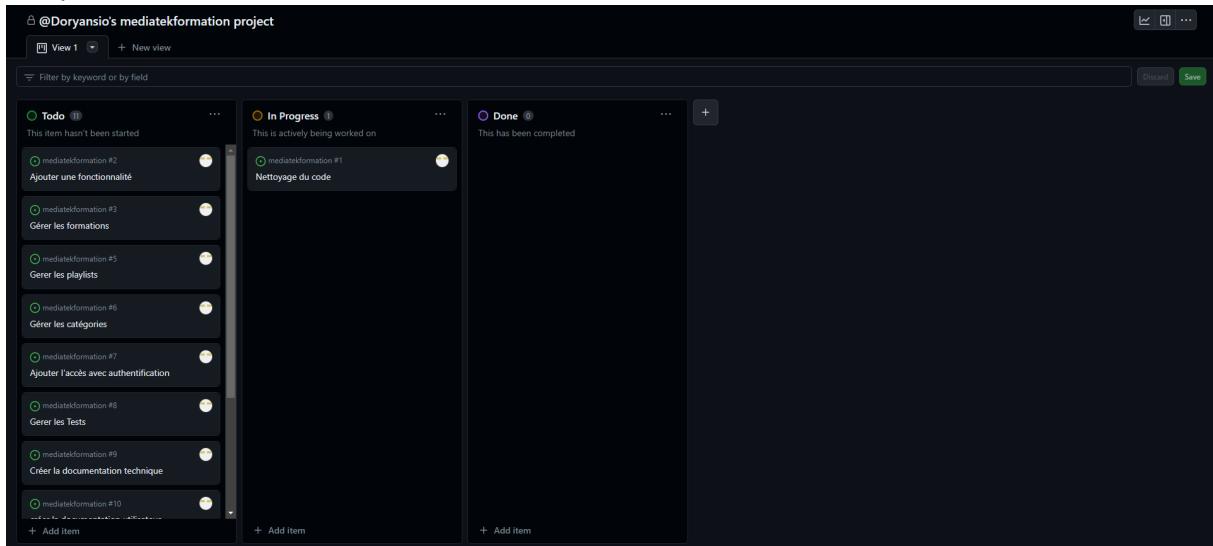
- Symphony
- Doctrine
- PhpMyAdmin
- Twig
- Bootstrap
- Keycloak

Mission 1 : Nettoyer et optimiser le code existant

Tache 1 : nettoyer le code

Temps estimé : 2 heures

Temps de réalisation : 1 heures



Nettoyer le code en suivant les indications de Sonarlint (ne nettoyer que les fichiers créés par le développeur donc trier les « Action items » de Sonarlint par « Location » et s'arrêter au premier fichier dans « vendor »)

En rappel :

- Eviter les chaines « en dur » (pour éliminer les « strings littéraux duplicated »)
- Nommer les constantes en Majuscules
- Fusionner certains tests imbriqués inutilement.
- Ajouter l'attribut « alt » à toutes les images.
- Ajouter l'attribut « description » à toutes les tables.

Normalement, seul l'item demandant d'ajouter un header à une table, doit rester.

Problèmes rencontrés et solutions

Lors de l'analyse du code avec sonarLint plusieurs erreurs de code ont été retrouvée :

Dans les fichiers FormationsController.php, FormationsRepository.php, playlistRepository et dans le fichier PlaylistController.php

```
Source History Issues Search Tools Help
26     * @var CategorieRepository
27     */
28     private $categorieRepository;
29
30     /**
31      * @var FormationsRepository
32      */
33     private $formationsRepository;
34
35     /**
36      * @var EtapeRepository
37      */
38     private $etapeRepository;
39
40     /**
41      * @var FormationRepository
42      */
43     private $formationRepository;
44
45     /**
46      * @var PlaylistRepository
47      */
48     private $playlistRepository;
49
50     /**
51      * @var UserRepository
52      */
53     private $userRepository;
54
55     /**
56      * @var Session
57      */
58     private $session;
59
60     /**
61      * @var EntityManager
62      */
63     private $em;
64
65     /**
66      * @var Router
67      */
68     private $router;
69
70     /**
71      * @var Paginator
72      */
73     private $paginator;
74
75     /**
76      * @var Formations
77      */
78     private $formations;
79
80     /**
81      * @var Etapes
82      */
83     private $etapes;
84
85     /**
86      * @var Formations
87      */
88     private $formations;
89
90     /**
91      * @var Etapes
92      */
93     private $etapes;
94
95     /**
96      * @var Formations
97      */
98     private $formations;
99
100    /**
101     * @var EntityManager
102     */
103    private $entityManager;
```

General Analysis Glance X

FormationsController.php X FormationsController.php X

Issues

0 Analyse dans 1 seconde

0 critical (0 total)

0 phpS102: String literals should not be duplicated. (1)

```

    /**
     * Exige que tous les champs contiennent une valeur
     * ou que tous les champs soient vides
     * Spécie type :Champs
     * Spécie type :Familles
     * Spécie type :Tables
     * Spécie type :Tables dans une autre table
     */
    public function findMyContentValue($champs, $valeurs, $table="") : array{
        if($valeurs=="")
            return $this->findMyContentValue();
        if($table=="")
            return $this->createQueryBuilder("f")
                ->where("f.champs = :champs AND f.valeurs = :valeurs")
                ->setParameter("f.champs", "S")
                ->setParameter("f.valeurs", "S")
                ->getQueryBuilder()
                ->getResult();
        else{
            return $this->createQueryBuilder("f")
                ->select("f.$table")
                ->where("f.$table = :table")
                ->setParameter("f.$table", "S")
                ->setParameter("valeurs", "S")
                ->getQueryBuilder()
                ->getResult();
        }
    }

```

Sonar Rule Details Window | SonarLint Analyzer Window

Nodes

- 0 critical (0 issues)
- 0 info (0 issues)
- 0 php:S111: Constant names should comply with a naming convention (1)
- 0 19:18 Formation.php
- 0 70:30 FormationRepository.php
- 0 70:30 FormationRepository.php

```

    /**
     * Recupere toutes les playlists triées sur le nom de la playlist
     * Spécie type :Champs
     * Spécie type :Familles
     * Spécie type :Tables
     */
    public function findAllOrderByName($ordre): array{
        return $this->createQueryBuilder("p")
            ->select("p.nom")
            ->orderBy("p.nom", $ordre)
            ->getQueryBuilder()
            ->getResult();
    }

    /**
     * Exige que tous les champs contiennent une valeur
     * ou que tous les champs soient vides
     * Spécie type :Champs
     * Spécie type :Familles
     * Spécie type :Tables
     * Spécie type :Tables dans une autre table
     */
    public function findMyContentValue($champs, $valeurs, $table="") : array{
        if($valeurs=="")
            return $this->findAllOrderByName("ASC");
        if($table=="")
            return $this->createQueryBuilder("p")

```

Sonar Rule Details Window | SonarLint Analyzer Window

Nodes

- 0 critical (0 issues)
- 0 info (0 issues)
- 0 php:S111: Constant names should comply with a naming convention (1)
- 0 19:18 Formation.php
- 0 70:30 FormationRepository.php
- 0 70:30 FormationController.php
- 0 70:30 PlaylistRepository.php

```

    /**
     * Fournit FormationRepository
     */
    private $formationRepository;

    /**
     * Fournit CategorieRepository
     */
    private $categorieRepository;

    /**
     * Fournit PlaylistRepository
     */
    private $playlistRepository;

    function __construct(PlaylistRepository $playlistRepository,
                        CategorieRepository $categorieRepository,
                        FormationRepository $formationRepository)
    {
        $this->playlistRepository = $playlistRepository;
        $this->categorieRepository = $categorieRepository;
        $this->formationRepository = $formationRepository;
    }

    /**
     * @Route("/playlists", name="playlists")
     */
    public function index(): Response{
        $form = $this->createForm();
        $formAction = $this->createQueryBuilder()->findAllOrderByName("ASC");
        $formAction = $this->createQueryBuilder()->findAll();
        return $this->render('app/playlists.html.twig', [
            'playlists' => $playlists,
            'categories' => $categories,
        ]);
    }

```

Sonar Rule Details Window | SonarLint Analyzer Window

Nodes

- 0 critical (0 issues)
- 0 info (0 issues)
- 0 php:S111: Constant names should comply with a naming convention (1)
- 0 19:18 Formation.php
- 0 70:30 FormationRepository.php
- 0 70:30 FormationController.php
- 0 70:30 PlaylistRepository.php
- 0 70:30 PlaylistController.php

Nous avons le code d'erreur php: S1192: "String litterals should not be duplicated". Ici sonar Lint nous informe que certaines parcelles de code ce répètent ce qui pourraient nuire à la bonne lisibilité du code du a la répétition. Ce qui serait judicieux de faire pour éviter les répétitions ce serait de créer des variables constantes concernant les valeurs qui se répètent plusieurs fois. En plus de permettre une meilleure lisibilité du code si des changements sont à faire nous n'aurons pas à changer toutes les valeurs mais seulement le contenu de la constante

Prenons l'exemple ci-dessous comme correction la logique appliquée est applicable est valable pour les autres problèmes de ce type relevé par Sonarlint.

The screenshot shows the SonarLint IDE interface with the 'Source' tab selected. The code editor displays a PHP file with annotations. A specific line of code is highlighted in blue: `return $this->render('formations', [PAGE_FORMATIONS => $formations, 'categories' => $categories]);`. Below the code editor, the status bar indicates 'Analyze done, 0 issue found'.

```

19     private const PAGE_FORMATIONS = "pages/formations.html.twig";
20
21     /**
22      * @var FormationRepository
23      */
24     private $formationRepository;
25
26     /**
27      * @var CategorieRepository
28      */
29     private $categorieRepository;
30
31     public function __construct(FormationRepository $formationRepository, CategorieRepository $categorieRepository)
32     {
33         $this->formationRepository = $formationRepository;
34         $this->categorieRepository = $categorieRepository;
35     }
36
37     /**
38      * @Route("/formations", name="formations")
39      * @return Response
40      */
41     public function index(): Response
42     {
43         $formations = $this->formationRepository->findAll();
44         $categories = $this->categorieRepository->findAll();
45         return $this->render('formations', [
46             'formations' => $formations,
47             'categories' => $categories
48         ]);
49     }

```

Nous pouvons voir que la valeur originale renvoyée : « pages/formations.html.twig »

A été placé dans la constante « PAGE_FORMATIONS ». Cela facilite la lecture de code et permet une modification plus rapide si par exemple la valeur été amenée il faudrait juste changer la valeur de la constante.

The screenshot shows the SonarLint IDE interface with the 'Source' tab selected. The code editor displays a PHP file with annotations. Two specific lines of code are highlighted in blue: `private const PFORMATION = 'p.formations';` and `private const PNAME = 'p.name';`. Below the code editor, the status bar indicates 'Analyze done, 0 issue found'.

```

16     /**
17      * @var PlaylistRepository
18      */
19
20     private const PFORMATION = 'p.formations';
21     private const PNAME = 'p.name';
22
23     public function __construct(ManagerRegistry $registry)
24     {
25         parent::__construct($registry, Playlist::class);
26     }
27
28     public function add(Playlist $entity, bool $flush = false): void
29     {
30         $this->getEntityManager()->persist($entity);
31
32         if ($flush) {
33             $this->getEntityManager()->flush();
34         }
35     }
36
37     public function remove(Playlist $entity, bool $flush = false): void
38     {
39         $this->getEntityManager()->remove($entity);
40
41         if ($flush) {
42             $this->getEntityManager()->flush();
43         }
44     }

```

Ici deux constantes ont été ajoutées -> PFORMATION ET PNAME.

Dans le fichier FormationsRepository c'est la constante FPUBLISHED qui a été ajouté avec comme valeur « f.publishedAt »

The screenshot shows the SonarLint IDE interface. The main window displays the code for `FormationRepository`. A status bar at the bottom indicates "Analyze done, 0 issue found".

```

17 class FormationRepository extends ServiceEntityRepository
18 {
19     private const PUBLISHED = 'f.publishedAt';
20
21     public function __construct(EntityManager $entityManager)
22     {
23         parent::__construct($entityManager, Formation::class);
24     }
25
26     public function add(Formation $entity, bool $flush = false): void
27     {
28         $this->getEntityManager()->persist($entity);
29
30         if ($flush) {
31             $this->getEntityManager()->flush();
32         }
33     }
34
35     public function remove(Formation $entity, bool $flush = false): void
36     {
37         $this->getEntityManager()->remove($entity);
38
39         if ($flush) {
40             $this->getEntityManager()->flush();
41         }
42     }
43
44     /**
45      * @return routes les formations triées sur un champ
46     */

```

Enfin dans le fichier `PlaylistController.php` à la manière du premier exemple le chemin vers le page des playlists a été mise dans une constante.

The screenshot shows the SonarLint IDE interface. The main window displays the code for `PlaylistController`. A status bar at the bottom indicates "Analyze done, 0 issue found".

```

15 /**
16  * @author Cendre
17 */
18 class PlaylistController extends AbstractController
19 {
20     private const PAGE_PLAYLIST = "pages/playlists.html.twig";
21
22     /**
23      * @var PlaylistRepository
24     */
25     private $playlistRepository;
26
27     /**
28      * @var FormationRepository
29     */
30     private $formationRepository;
31
32     /**
33      * @var CategorieRepository
34     */
35     private $categorieRepository;
36
37     /**
38      * @param PlaylistRepository $playlistRepository
39      * @param CategorieRepository $categorieRepository
40      * @param FormationRepository $formationRepository
41      */
42     public function __construct(PlaylistRepository $playlistRepository,
43                               CategorieRepository $categorieRepository,
44                               FormationRepository $formationRepository)
45     {
46         $this->playlistRepository = $playlistRepository;
47         $this->categorieRepository = $categorieRepository;
48         $this->formationRepository = $formationRepository;
49     }

```

Le code d'erreur php: S115: Constant names should comply with a naming convention.

Cette information sur les règles de nommage nous signale que dans le fichier `formation.php` où nous trouvons la variable constante `Cheminimage` montré ci-dessous

```

1 <?php
2
3 namespace App\Entity;
4
5 use App\Repository\FormationRepository;
6 use DateTimeInterface;
7 use Doctrine\Common\Collections\ArrayCollection;
8 use Doctrine\Common\Collections\Collection;
9 use Doctrine\ORM\Mapping as ORM;
10
11 /**
12 * @ORM\Entity(repositoryClass=FormationRepository::class)
13 */
14 class Formation
15 {
16     /**
17      * Début de chemin vers les images
18      */
19     private const CHEMINIMAGE = "https://i.ytimg.com/vi/";
20
21     /**
22      * @ORM\Id
23      * @ORM\GeneratedValue
24      * @ORM\Column(type="integer")
25      */
26     private $id;
27
28     /**
29      * @ORM\Column(type="datetime", nullable=true)
30      */
31     private $publishedAt;
32 }
33

```

Sonar Rule Details Window | SonarLint Analyzer Window X

src X

Nodes

- Analyze done, 6 issues found
- critical (4 issues)
- phpS115: Constant names should comply with a naming convention! (1)

19:18: Formation.php

Les règles de nomenclature précise que le nom des variables constantes doivent être écrites en majuscules. Il nous suffit juste de changer le nom de la variable comme ceci (en oubliant pas de changer les endroits où cette variable est appelée) :

```

1 <?php
2
3 namespace App\Entity;
4
5 use App\Repository\FormationRepository;
6 use DateTimeInterface;
7 use Doctrine\Common\Collections\ArrayCollection;
8 use Doctrine\Common\Collections\Collection;
9 use Doctrine\ORM\Mapping as ORM;
10
11 /**
12 * @ORM\Entity(repositoryClass=FormationRepository::class)
13 */
14 class Formation
15 {
16     /**
17      * Début de chemin vers les images
18      */
19     private const CHEMINIMAGE = "https://i.ytimg.com/vi/";
20
21     /**
22      * @ORM\Id
23      * @ORM\GeneratedValue
24      * @ORM\Column(type="integer")
25      */
26     private $id;
27
28     /**
29      * @ORM\Column(type="datetime", nullable=true)
30      */
31     private $publishedAt;
32 }
33

```

Sonar Rule Details Window | SonarLint Analyzer Window X

src X Formation.php X

Nodes

- Analyze done, 0 issue found

Le code d'erreur php: S121: Control structure should use curly braces.

Nous trouvons cette erreur dans le fichier playlist.php en effet une boucle for each n'est pas à l'intérieur de parenthèse {}

The screenshot shows an IDE interface with a code editor and a SonarLint analysis window.

Code Editor:

```
95     }
96 }
97     return $this;
98 }
99 }
100 /**
101 * @return Collection<int, string>
102 */
103 public function getCategoriesPlaylist() : Collection
104 {
105     $categories = new ArrayCollection();
106     foreach($this->formations as $formation){
107         $categoriesFormation = $formation->getCategories();
108         foreach($categoriesFormation as $categorieFormation)
109             if(!$categories->contains($categorieFormation->getName())){
110                 $categories[] = $categorieFormation->getName();
111             }
112         }
113     }
114     return $categories;
115 }
116 }
117 }
118 }
```

SonarLint Analyzer Window:

- Analyze done, 5 issues found
- critical (3 issues)
- php:S121: Control structures should use curly braces (1)

Pour remédier à ce problème il suffit juste de mettre la condition présente dans le foreach entre parenthèse {}

```

97     return $this;
98 }
99
100 /**
101 * @return Collection<int, string>
102 */
103
104 public function getCategoriesPlaylist() : Collection
105 {
106     $categories = new ArrayCollection();
107     foreach($this->formations as $formation){
108         $categoriesFormation = $formation->getCategories();
109         foreach($categoriesFormation as $categorieFormation){
110             if(!$categories->contains($categorieFormation->getName())){
111                 $categories[] = $categorieFormation->getName();
112             }
113         }
114     }
115     return $categories;
116 }
117
118 }
119

```

Sonar Rule Details Window SonarLint Analyzer Window

L'erreur "collapsible if statement should be merged."

Dans le fichier playlist.php nous pouvons effectuer quelques optimisations de code pour le rendre plus lisible au niveau de la fonction removeFormation ou nous voyons deux test « if » à la suite.

```

79     public function addFormation(Formation $formation): self
80     {
81         if (! $this->formations->contains($formation)) {
82             $this->formations[] = $formation;
83             $formation->setPlaylist($this);
84         }
85
86         return $this;
87     }
88
89     public function removeFormation(Formation $formation): self
90     {
91         if ($this->formations->removeElement($formation)) {
92             // set the owning side to null (unless already changed)
93             if ($formation->getPlaylist() === $this) {
94                 $formation->setPlaylist(null);
95             }
96         }
97
98         return $this;
99     }
100
101 /**
102 * @return Collection<int, string>
103 */
104 public function getCategoriesPlaylist() : Collection
105 {
106     $categories = new ArrayCollection();
107     foreach($this->formations as $formation){
108         $categoriesFormation = $formation->getCategories();
109         foreach($categoriesFormation as $categorieFormation){
110             if(!$categories->contains($categorieFormation->getName())){
111                 $categories[] = $categorieFormation->getName();
112             }
113         }
114     }
115     return $categories;
116 }
117
118 }
119

```

Sonar Rule Details Window SonarLint Analyzer Window

src X

Nodes

- Analyze done, 11 issues found
- critical (0 issues)
- major (1 issue)
- phpS1066: Collapsible "if" statements should be merged (1)
 - 9312: Playlist.php
- minor (1 issue)

Nous pouvons optimiser le code en regroupant les deux conditions dans un seul if comme dans l'image ci-dessous

```

76     return $this->formations;
77 }
78
79 public function addFormation(Formation $formation): self
80 {
81     if (!$this->formations->contains($formation)) {
82         $this->formations[] = $formation;
83         $formation->setPlaylist($this);
84     }
85
86     return $this;
87 }
88
89 public function removeFormation(Formation $formation): self
90 {
91     if ($this->formations->removeElement($formation) || ($formation->getPlaylist() === $this)) {
92         // set the owning side to null (unless already changed)
93         $formation->setPlaylist(null);
94     }
95     return $this;
96 }
97
98 /**
99 * @return Collection<int, string>
100 */
101
102 /**
103 * @return Collection<int, string>
104 */

```

Sonar Rule Details Window | SonarLint Analyzer Window X
src X Formation.php X src X Playlist.php X Playlist.php X
Nodes
Analyze done, 0 issue found

Php: S1301: « switch » statements should have default clauses & switch statement should have at least 3 case clauses.

```

59 /**
60 * @Route("/playlists/tri/{champ}/{ordre}", name="playlists.sort")
61 * @param type $champ
62 * @param type $ordre
63 * @return Response
64 */
65 public function sort($champ, $ordre): Response{
66     switch($champ){
67         case "name":
68             $playlists = $this->playlistRepository->findAllOrderBy($ordre);
69             break;
70     }
71     $categories = $this->categorieRepository->findAll();
72     return $this->render(self::PAGES_PLAYLIST, [
73         'playlists' => $playlists,
74         'categories' => $categories
75     ]);
76 }
77
78 /**
79 * @Route("/playlists/recherche/{champ}/{table}", name="playlists.findAllContain")
80 * @param type $champ
81 * @param Request $request
82 * @param type $table
83 * @return Response
84 */
85 public function findAllContain($champ, Request $request, $table=""): Response{
86     $valeur = $request->get("recherche");
87 }

```

Sonar Rule Details Window | SonarLint Analyzer Window X

Dans ce cas-là l'erreur réside dans une mauvaise utilisation de la fonction switch. En effet nous devons a priori analyser qu'un seul cas pour le moment donc le switch n'est pas nécessaire.

The screenshot shows the NetBeans IDE interface. The code editor displays PHP code for a controller, specifically the `PlaylistsController.php`. The SonarLint Analyzer Window at the bottom shows a single issue: "Analyze done, 0 issue found".

```

56     'categories' => $categories
57   ];
58 }
59 /**
60 * @Route("/playlists/tri/{champ}/{ordre}", name="playlists.sort")
61 * @param type $champ
62 * @param type $ordre
63 * @return Response
64 */
65 public function sort($champ, $ordre): Response{
66   $playlists = $this->playlistRepository->findAllOrderByName($champ,$ordre);
67   $categories = $this->categorieRepository->findAll();
68   return $this->render(self::PAGES_PLAYLIST, [
69     'playlists' => $playlists,
70     'categories' => $categories,
71   ]);
72 }
73
74 /**
75 * @Route("/playlists/recherche/{champ}/{table}", name="playlists.findallcontain")
76 * @param type $champ
77 * @param Request $request
78 * @param type $table
79 * @return Response
80 */
81
82
83
84

```

Au niveau des fichier html sonar Lint nous renvoie aussi des manières d'optimiser le code et de respecter les bonnes pratiques.

Web: imgWithoutAltCheck: image area and button with image should have an “alt” attribute

l'erreur que nous renvoie sonarLint nous informe que bien que mineur une image devra avoir une balise « alt » qui permettrait à l'utilisateur d'avoir une alternative textuelle dans le cas où l'image ne se chargerai pas au moment de l'affichage de la page web. Ce problème se trouve dans les fichiers `acceuil.html.twig`, `basefront.html.twig` `formations.html.twig` et `playlist.html.twig`

The screenshot shows the NetBeans IDE interface. The code editor displays Twig template code for `basefront.html.twig`. The SonarLint Analyzer Window at the bottom shows two issues: "Analyze done, 2 issues found" and "Web:BoldAndItalicTagsCheck: and tags should be used ()".

```

1  {% extends "base.html.twig" %}
2
3  {% block title %}{% endblock %}
4  {% block stylesheets %}{% endblock %}
5  {% block top %}
6    <div class="container">
7      <!-- titre -->
8      <div class="text-left">
9          
10     </div>
11     <!-- menu -->
12     <nav class="navbar navbar-expand-lg navbar-light bg-light">
13         <div class="collapse navbar-collapse" id="navbarSupportedContent">
14             <ul class="navbar-nav mr-auto">
15                 <li class="nav-item">
16                     <a class="nav-link" href="{{ path('accueil') }}>Accueil</a>
17                 </li>
18                 <li class="nav-item">
19                     <a class="nav-link" href="{{ path('formations') }}>Formations</a>
20                 </li>
21                 <li class="nav-item">
22                     <a class="nav-link" href="{{ path('playlists') }}>Playlists</a>
23                 </li>
24             </ul>
25         </div>
26     </nav>
27     <div>
28     {% endblock %}
29     {% block body %}{% endblock %}

```

Voici un exemple de correction qui pourrait être similaire à toutes les erreurs de ce type.

```

1  {% extends "base.html.twig" %}
2
3  {# block title #}{% endblock %}
4  {# block stylesheets #}{% endblock %}
5  {# block top #}
6  <div class="container">
7      <!-- titre -->
8      <div class="text-left">
9          
10     </div>
11     <!-- menu -->
12     <nav class="navbar navbar-expand-lg navbar-light bg-light">
13         <div class="collapse navbar-collapse" id="navbarSupportedContent">
14             <ul class="navbar-nav mr-auto">
15                 <li class="nav-item">
16                     <a class="nav-link" href="{{ path('accueil') }}>Accueil</a>
17                 </li>
18                 <li class="nav-item">
19                     <a class="nav-link" href="{{ path('formations') }}>Formations</a>
20                 </li>
21                 <li class="nav-item">
22                     <a class="nav-link" href="{{ path('playlists') }}>Playlists</a>
23                 </li>
24             </ul>
25         </div>
26     </nav>
27 </div>
28 {# endblock #}
29 {# block body #}{% endblock %}

```

Certaines balises ne sont pas adaptées selon sonarLint nous le voyons avec le code d'erreur Web :BoldandItalicTagsCheck : `` and `` tags should be used

Cette erreur mineure nous suggère de remplacer les balises `<i>` par les balises `` et `` ce qui permettrait de mettre ce qui est entre ces balise plus en évidence pour le lecteur.

Summary: C. Utilisateurs (ci-après "Utilisateurs")

Issue: Article 2 - Confidentialité

Details: Le site ne collecte pas de données.

Code Snippet:

```

<summary>C. Utilisateurs (ci-après "Utilisateurs")</summary>
<p>Sont considérés comme utilisateurs tous les internautes qui naviguent, l:</p>
</details>
<div class="bd-example">
<h4>Article 2 - Confidentialité</h4>
<p>Le Site est par principe accessible aux Utilisateurs 24/24h et 7/7j, sauf interrup</p>
<p>En cas d'impossibilité d'accès au Site, celui-ci s'engage à faire son maximum af</p>
<h4>Article 4 - loi applicable et juridiction</h4>
<p>Les présentes Mentions Légales sont régies par la loi française. En cas de différe</p>
<h4>Article 5 - contact</h4>
<p>Pour tout signalement de contenus ou d'activités illicites, l'Utilisateur peut co</p>
<p>Le site mediatekformation vous souhaite une excellente navigation !</p>
</div>

```

Analyzer done, 9 issues found:

- major (1 issue)
- minor (8 issues)
- Web-BoldAndItalicTagsCheck: '`` and `` tags should be used (5)

Issues:

- 34:22: basefront.html.twig
- 7:166: cgu.html.twig
- 12:51: cgu.html.twig
- 23:53: cgu.html.twig
- 34:48: cgu.html.twig

```

20     </address>
21     </details>
22     <details>
23       <summary>B. Hébergeur du site (ci-après "<strong><em>l'Hébergeur</strong></em>")</summary>
24       <address>
25         <strong>Planethoster</strong>
26         <br>4416 Louis-B.-Mayer
27         <br>Laval, Québec
28         <br>H7P 0G1 Canada
29         <br>Téléphone:&nbsp;0102030405
30         <br>Courriel:&nbsp;contact@planethoster.com
31     </address>
32   </details>
33   <details>
34     <summary>C. Utilisateurs (ci-après "<strong><em>les Utilisateurs</strong></em>")</summary>
35     <p>Sont considérés comme utilisateurs tous les internautes qui naviguent, lisent, visionnent
36   </details>
37 </div>
38 💡 Article 2 - Confidentialité
39 <div class="bd-example">
40   Le site ne collecte pas de données.
41 </div>
42 💡 Article 3 - accessibilité
43 <p>Le Site est par principe accessible aux Utilisateurs 24/24h et 7/7j, sauf interruption, programmée
44 <p>En cas d'impossibilité d'accès au Site, celui-ci s'engage à faire son maximum afin d'en rétablir l
45 💡 Article 4 - loi applicable et juridiction
46 <p>Les présentes Mentions Légales sont régies par la loi française. En cas de différend et à défaut d
47 💡 Article 5 - contact
48 <p>Pour tout signalement de contenus ou d'activités illicites, l'Utilisateur peut contacter l'Editeur

```

Web:TableWithoutCaptionCheck : "<table>" tags should have à description

Ici Sonarlint nous informe qu'il serait plus judicieux de rajouter une description à nos tableaux dans les fichiers accueil.html.twig, formations.html.twig et dans le fichier playlists.html.twig. cela permettrait en effet à l'utilisateur une plus grande compréhension du site sur lequel il se trouve.

Par exemple nous pourrons ajouter une description ou un titre au tableau en question.

```

</p>
<p>Voici les <strong>deux dernières formations</strong> ajoutées au catalogue :</p>



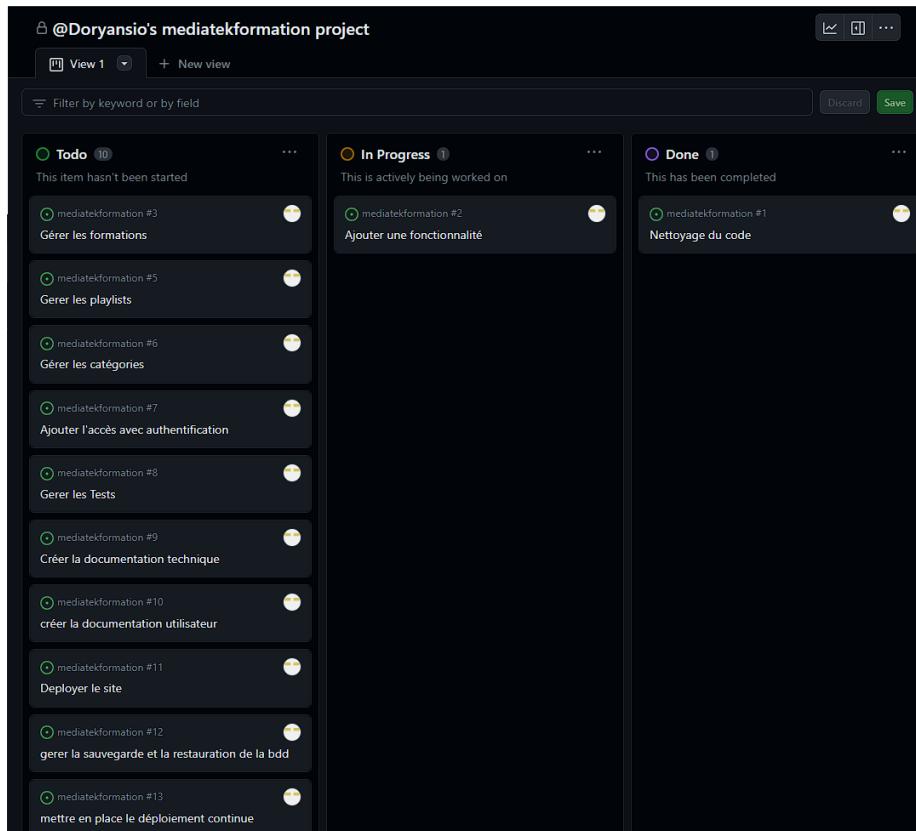


```

Tache 2

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.

Ajouter une fonctionnalité



Temps estimé : 2 heures

Temps réel : 2 heures 30 minutes.

playlist	nombre de formations	catégories
Bases de la programmation (C#)	74	C# POO
Compléments Android (programmation mobile)	13	Android
Cours Composant logiciel	2	Cours
Cours Curseurs	2	SQL Cours POO
Cours de programmation objet	1	POO Cours
Cours Informatique embarquée	1	Cours
Cours MCD MLD MPD	2	MCD Cours
Cours MCD vs Diagramme de classes	2	MCD Cours
Cours Merise/2	1	MCD Cours
Cours Modèle relationnel et MCD	1	MCD Cours
Cours Transactions et verrou	3	SQL Cours

Bases de la programmation (C#)

catégories : C# POO

Nombre de formations : 74

description :

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

1ère partie : programmation procédurale en mode console (non graphique)
n°1 à 30 : procédural, notions élémentaires (variables, saisie/affichage, affectations/calculs, alternatives (if/switch), itérations (while/do-while/for))
n°31 à 42 : procédural, tableaux (1 et 2 dimensions, manipulations, tris, recherches)
n°43 à 59 : procédural, modules et paramètres (procédures et fonctions)

2ème partie : événementiel (en mode graphique)
n°60 à 67 : événementiel (programmation graphique)

3ème partie : initiation à l'objet
n°68 à 74 : notions de base en programmation objet sur des classes "métier"

Bases de la programmation n°1 - procédural : premier exemple
Bases de la programmation n°2 - procédural : exercice1 (affichage)
Bases de la programmation n°3 - procédural : exercice2 (saisie)
Bases de la programmation n°4 - procédural : exercice3 (calcul)
Bases de la programmation n°5 - procédural : exercice4 (calcul dans affichage)
Bases de la programmation n°6 - procédural : exercice5 (condition)
Bases de la programmation n°7 - procédural : exercice6 (conditions imbriquées)
Bases de la programmation n°8 - procédural : exercice7 (boucle)

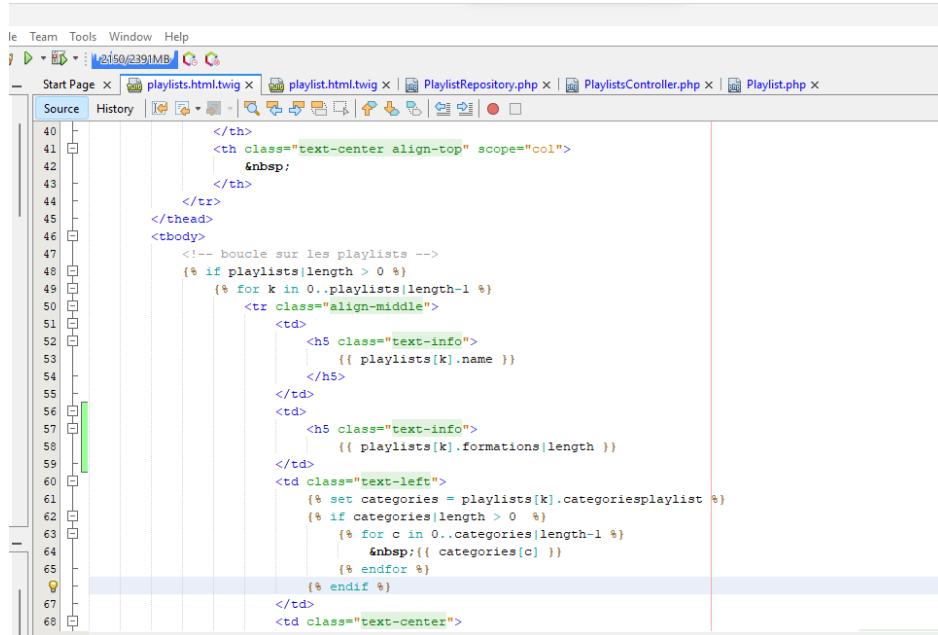
Lors de cette tâche il fallait que nous ajoutions une fonctionnalité qui nous permettait de voir le nombre de formations par playlists et de pouvoir les trier par ordre croissant et décroissant dans le tableau des playlists.

Pour le premier cas, soit afficher le nombre de formations par playlists nous avons déjà une petite indication dans le code au niveau du fichier playlists.html.twig dans la boucle des playlists.

En effet nous avons la syntaxe pour avoir la taille d'un élément, ici les tables de la BDD (nous prendrons l'exemple de la ligne 48 : playlists|length).

Comme la table formations en relation avec la table playlists nous pouvons y accéder comme un élément de la table playlist.

Nous avons aussi une boucle déterministe avec la clé [k] se rapportant à chaque itération de la boucle. Nous n'avons plus qu'à demander au programme que dans chaque playlist[k] il nous renvoie le nombre d'élément (formations) qui la compose. D'où l'ajout du code : playlists[k].formations|length.



```
<thead>
    <tr>
        <th></th>
        <th class="text-center align-top" scope="col">&ampnbsp</th>
    </tr>
</thead>
<tbody>
    <!-- boucle sur les playlists -->
    {# if playlists.length > 0 #}
    {# for k in 0..playlists.length-1 #}
        <tr class="align-middle">
            <td>
                <h5 class="text-info">{{ playlists[k].name }}</h5>
            </td>
            <td>
                <h5 class="text-info">{{ playlists[k].formations|length }}</h5>
            </td>
            <td class="text-left">
                {# set categories = playlists[k].categoriesplaylist #}
                {# if categories.length > 0 #}
                {# for c in 0..categories.length-1 #}
                    &ampnbsp{{ categories[c] }}<br />
                {# endfor #}
                {# endif #}
            </td>
            <td class="text-center">
```

La méthode est similaire pour afficher le nombre de formations dans la page de cette même formation.

Nous pouvons déjà voir que dans le fichier playlist.html.twig que nous pouvons boucler sur playlistformation qui est en réalité une méthode de formation Repository findAllForOnePlaylist qui prend en paramètre l'id de la playlist et qui renvoie un Array.

Nous avons donc juste à demander au programme de nous donner la taille de cette liste.



```
<div class="col">
    <h4 class="text-info mt-5">{{ playlist.name }}</h4>
    <strong>catégories : </strong>
        <!-- boucle pour afficher les catégories -->
        {# set anccategorie = '' #}
        {# for playlist in playlistcategories #}
            {{ playlist.name }}&ampnbsp
        {# endfor #}
        <br /> <br />
    <strong>Nombre de formations : </strong>
        {{ playlistformations|length }}

    <br /><br />
    <strong>description :</strong><br />
```

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.

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.

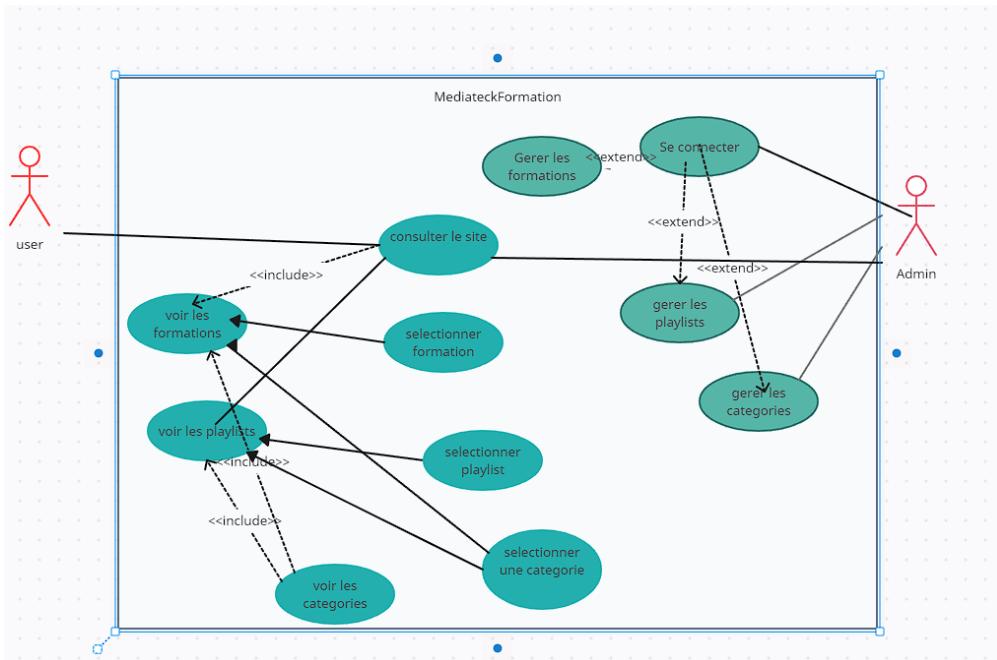
The screenshot shows a GitHub project board titled '@Doryansio's mediatekformation project'. The board is organized into three columns: 'Todo' (9 items), 'In Progress' (1 item), and 'Done' (2 items). Each item is represented by a card with a circular progress indicator (green for Todo, yellow for In Progress, purple for Done) and a small profile picture. The cards contain the following information:

- Todo (9 items):**
 - mediatekformation #5: Gérer les playlists
 - mediatekformation #6: Gérer les catégories
 - mediatekformation #7: Ajouter l'accès avec authentification
 - mediatekformation #8: Gérer les Tests
 - mediatekformation #9: (empty)
 - mediatekformation #10: (empty)
 - mediatekformation #11: (empty)
 - mediatekformation #12: (empty)
 - mediatekformation #13: (empty)
- In Progress (1 item):**
 - mediatekformation #3: Gérer les formations
- Done (2 items):**
 - mediatekformation #1: Nettoyage du code
 - mediatekformation #2: Ajouter une fonctionnalité

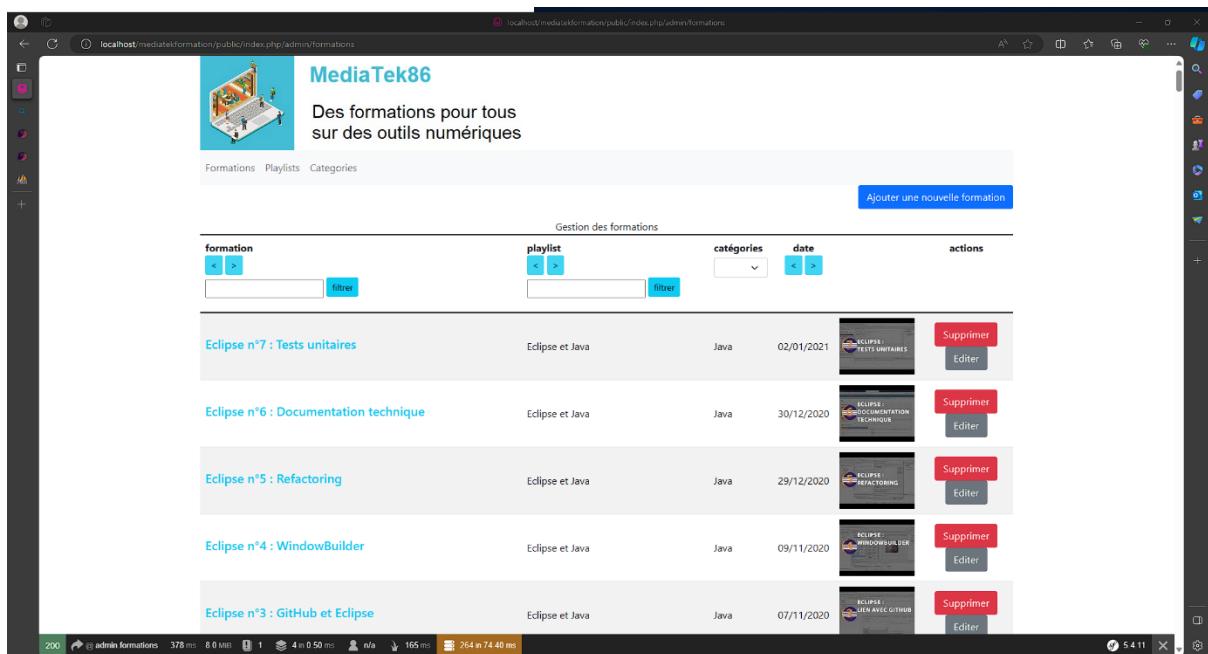
Temps requis -> 5 heures

Temps réel -> 5h30

Voici le diagramme de cas d'utilisations du site.



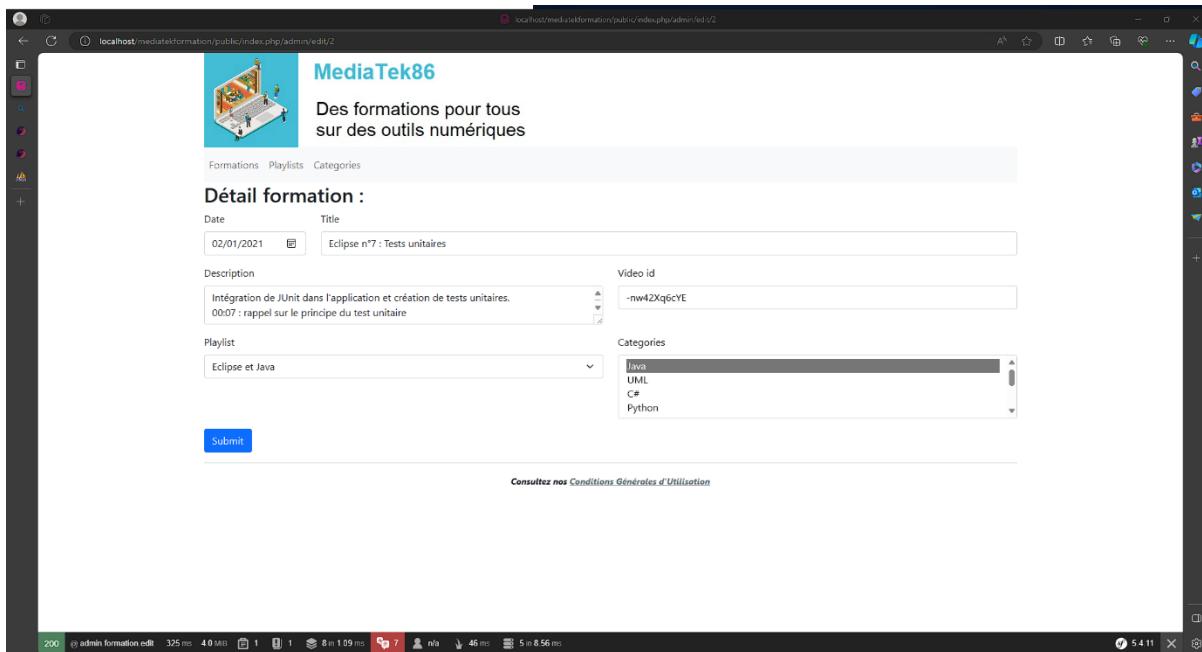
Cette tache consistait à créer une partie back office similaire a la partie front au niveau de la chartre graphique utilisé mais permettant également d'ajouter, d'éditer et de supprimer des formations.



Pour se faire il a fallu apporter des modifications dans chaque packages de l'application.

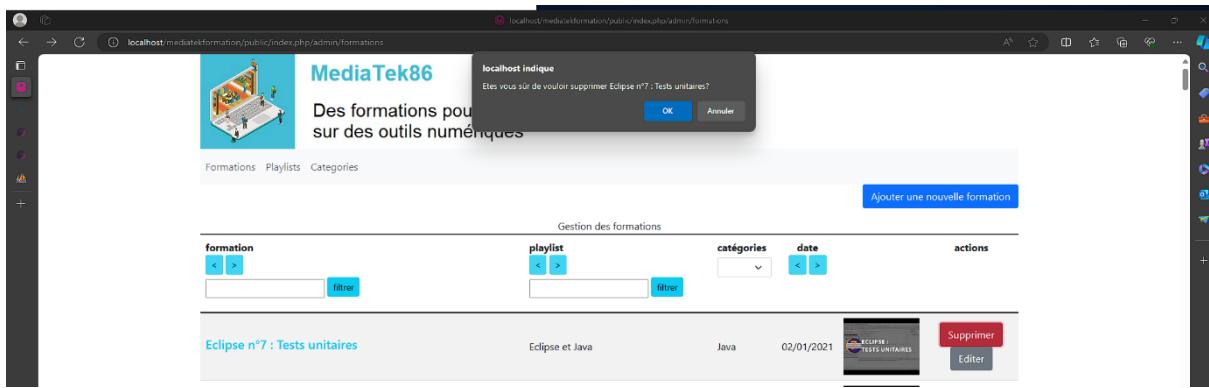
Au niveau de la vue, nous avons donc ajouter des boutons « supprimer » « éditer » et ajouter une nouvelle formation ». ces derniers réagissent avec le Controller admin des formations

AdminFormationsController.php ou nous avons créé deux méthodes ; edit et ajout qui nous permettent d'interagir directement avec le repository des formations où se trouve les méthodes « add » et « remove » qui permettent d'ajouter ou de supprimer un élément de la table ici la table formations dans la base de données mediatekformation. Ce qui fait que lors d'un ajout, d'une modification ou d'une suppression cette dernière affectera aussi la base de données et les autres tables. Supprimer , ajouter ou éditer une formation générera aussi des modifications dans la playlist dans laquelle elle est rattachée

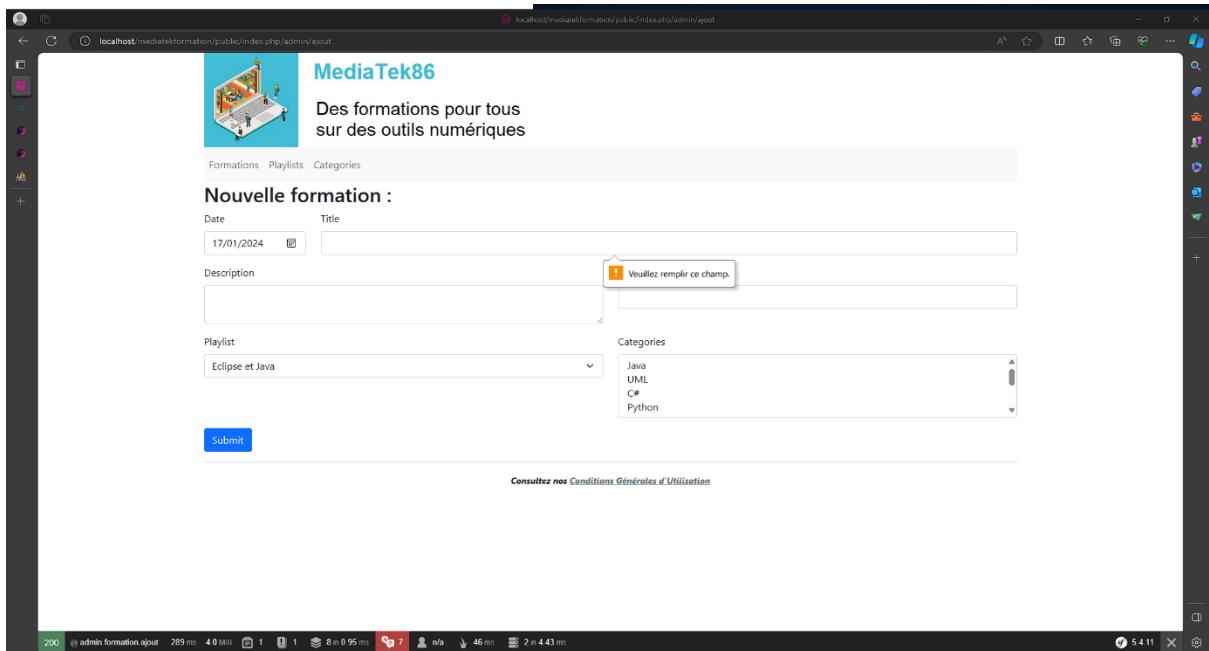


Pour accéder à la page d'éditions il faut d'abord la créer et renvoyer la route correspondante dans le Controller. Une fois dans la page d'éditions nous avons un formulaire qui nous donne toutes les données de la formation sélectionnée ainsi qu'un bouton « submit ». L'édition de la formation utilise la méthode « add » du repository. Cependant à la validation du formulaire nous devons retourner sur la même page mais avec les modifications que l'on a faites. Pour se faire la méthode édition doit en plus d'avoir en paramètre l'entité formation et la méthode de requête, avoir l'id de la formation à modifier afin de pouvoir effectuer une redirection vers la même page mais cette fois-ci modifiée à la validation du formulaire.

Pour permettre l'affichage du formulaire et notamment de la playlist et des catégories. Nous avons dû créer une méthode dans leur entité respective qui nous permettait d'y accéder au format de chaîne de caractère afin de pouvoir les insérer dans notre formulaire.

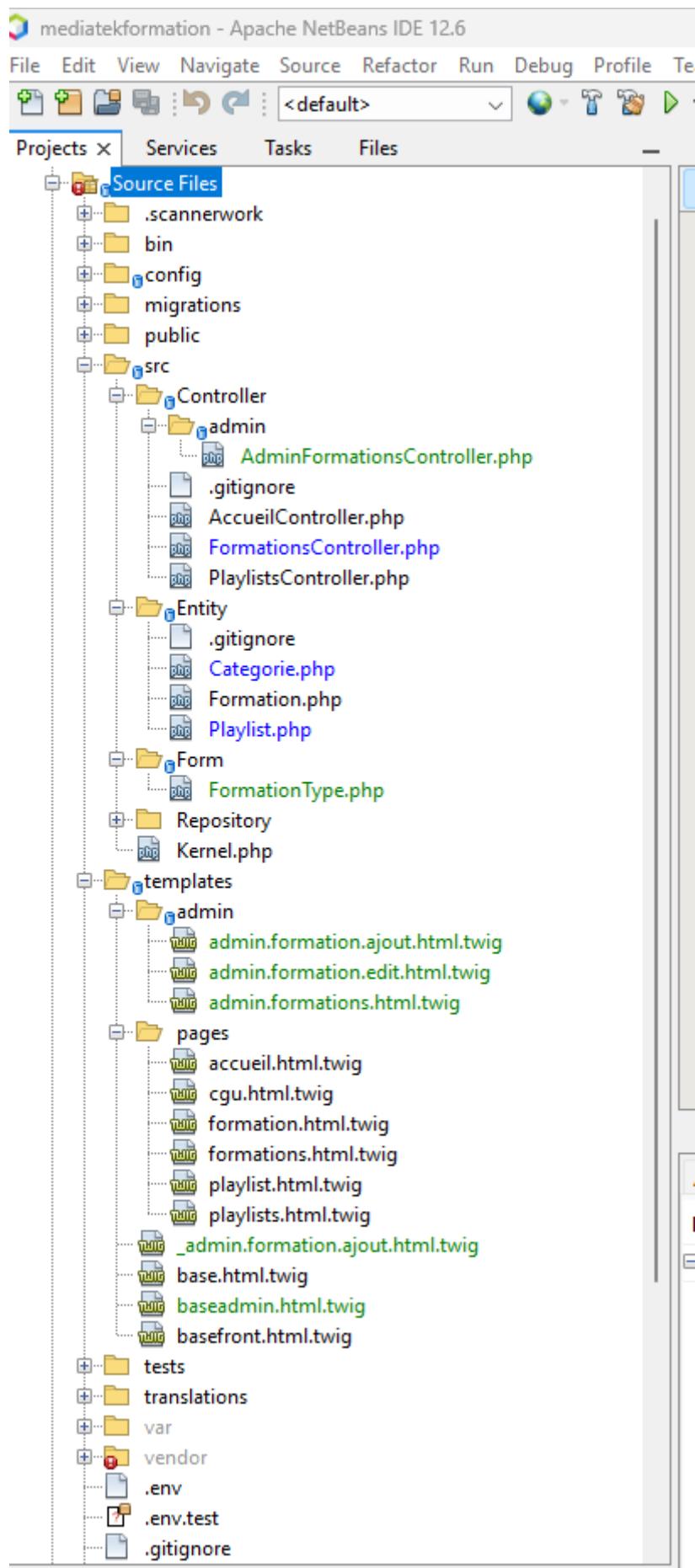


Pour supprimer une formation le principe est similaire à l'édition de formations dans le fond : les différentes couches de programmes s'échangent des informations afin de répondre aux demandes de l'utilisateur dans le cas échéant la suppression d'une formation. La suppression étant irréversible à partir du moment où l'utilisateur clique sur le bouton de suppression et de ce fait souhaite déclencher un événement. Un alerte apparaît pour lui demander s'il est sûr de son choix avec le nom de la formation en question.



Abordons maintenant l'ajout de formation. Nous nous retrouvons avec le même formulaire que celui de modification (les deux reposent sur la même architecture) à l'exception que celui-ci est vide pour certains champs et que le titre est différent.

Si nous regardons de plus près nous voyons que la date est remplie à celle du jour. Si nous créons une formation sa date de publication ne pourra pas être antérieure à la date où elle a été publiée. De plus certains champs sont déjà remplis. Les deux listes « playlist » et « catégories ». Pour les playlists il n'a pas de possibilité de ne pas en avoir car formation fait partie d'une playlist alors qu'une formation peut être au sein d'une, plusieurs ou aucune catégorie. Aussi attribuer un nom à la formation est obligatoire.



Enfin voici l'arborescence de fichier de l'application, avec en bleu les fichiers modifiés et en vert les fichiers créés.

Pour les modifications dans les fichiers catégories et playlist ce sont les méthodes convertissant les collections en chaîne de caractères pour pouvoir y accéder dans le formulaire.

Le fichier AdminFormationsController est très similaire au Controller du fichier front mais les routes renvoient toutes vers des pages admin et dans le Controller admin il y a des méthodes que l'on peut exploiter que du côté administrateur. Edit, ajout et suppr. La méthode ajout est particulière car à l'intérieur de cette méthode on génère une nouvelle instance de l'objet formation.

Nous avons dû créer une classe formulaire dans l'invite de commande PHP de l'application et le rattacher à la bonne table pour avoir les éléments de la table au sein du formulaire.

Le fichier _admin.formation.ajout est une sorte de Template dont vont se servir les fichiers admin formation edit et ajout. Ce Template contient la mise en forme du formulaire les informations s'afficheront ou pas selon là l'évènement déclencher par l'utilisateur. Si l'utilisateur souhaite modifier une formation les informations déjà présente seront affiché s'il souhaite créer une formation le formulaire sera vierge.

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.

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.

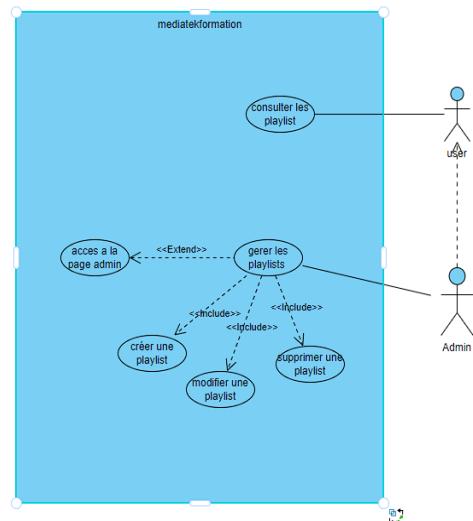
Un bouton doit permettre d'accéder au formulaire d'ajout d'une playlist. Les saisies doivent être contrôlées. L'ajout d'une playlist consiste juste à saisir son nom et sa description. Seul le champ Name est obligatoire.

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.

Temps estimé : 5h

Temps réel 3h

Diagramme de cas d'utilisation de ce cas :



Cette tache consistait à créer une page côté administrateur pour gérer les playlists avec les mêmes filtres que la page front office des playlists mais avec l'ajout des boutons d'éditions et de suppressions

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

Le bouton de suppression permet de retirer la playlist de la page ainsi que de la base de données. Cependant cette suppression ne peut se faire seulement si la playlist ne contient aucune formation. Sans quoi la suppression ne se fait pas et nous sommes renvoyés sur la page admin des playlists avec une petit message d'alerte nous indiquant que ce n'est pas possible.

playlist	nombre de formations	catégories
Bases de la programmation (C#)	74	C# POO
Compléments Android (programmation mobile)	13	Android
Cours Composant logiciel	2	Cours
Cours Curseurs	2	SQL Cours POO
Cours de programmation objet	1	POO Cours
Cours Informatique embarquée	1	Cours
Cours MCD MLD MPD	2	MCD Cours

Comme pour la partie gestion des formations, nous avons la possibilité d'éditer les playlists de modifier le nom et les catégories qui sont dans la playlist. En ce qui concerne les formations nous avons la liste des formations qui composent la playlist mais ce n'est pas possible de rajouter une formation directement via cette page.

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

MediaTek86

Des formations pour tous sur des outils numériques

Formations Playlists Catégories

Detail de la playlist:

Name: Bases de la programmation

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

Enregistrer

Bases de la programmation n°74 - POO : collections

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

Bases de la programmation n°72 - POO : héritage

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

Bases de la programmation n°70 - POO : encapsulation

Enfin le bouton pour créer une nouvelle formation nous redirige vers un formulaire pour créer une playlist ou la saisie est contrôlée. Pour créer une playlist nous avons obligatoirement besoin d'un nom pour qu'elle soit valable et enregistrer dans la base de données.

localhost/mediatedformation/public/index.php/admin/playlist

MediaTek86

Des formations pour tous sur des outils numériques

Formations Playlists Catégories

Nouvelle playlist :

Name: Description:
! Veuillez remplir ce champ.

Enregistrer

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

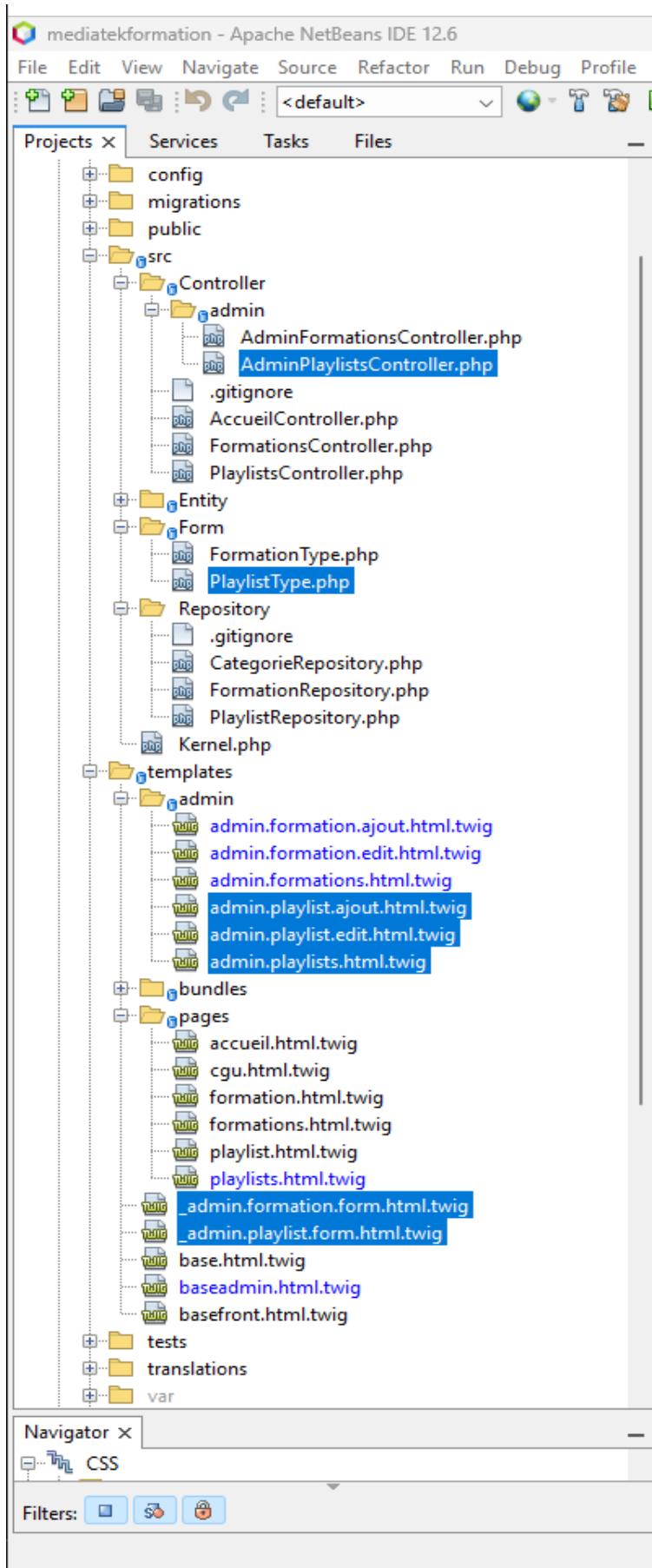
Une fois la playlist créée nous la voyons dans la page des playlists et aussi dans la base de données

Eclipse et Java	4	Java UML	4	Editer Supprimer
Exercices objet (sujets EDC BTS SIO)	8	POO	8	Editer Supprimer
Exercices triggers, sql et correctifs (sujets EDC BTS SIO)	4	SQL	4	Editer Supprimer
MCD : exercices progressifs	18	MCD	18	Editer Supprimer
MCD exercices d'examen (sujets EDC BTS SIO)	8	MCD	8	Editer Supprimer
POO TP Java	6	Java POO	6	Editer Supprimer
Programmation sous Python	19	Python POO	19	Editer Supprimer
Sujet E5 SLAM 2017 métropole : cas AHM-23	4		4	Editer Supprimer
Sujet E5 SLAM 2017 Nouméa: cas DANE	4		4	Editer Supprimer
Sujet E5 SLAM 2018 métropole : cas WEBCAISSE	4	POO SQL	4	Editer Supprimer
Sujet E5 SLAM 2018 Nouméa : cas LOCALUX	3	POO SQL	3	Editer Supprimer
Sujet E5 SLAM 2019 métropole : cas RESTILOC	4	Android SQL POO	4	Editer Supprimer
test	0		0	Editer Supprimer
TP Android (programmation mobile)	18	Android SQL Java	18	Editer Supprimer
Visual Studio 2019 et C#	11	C# POO	11	Editer Supprimer

Tableau des playlists

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

Voici l'arborescence de fichier avec les éléments ajouté à l'application.



Le fichier Controller est celui qui permet de gérer l'ajout, la modification et la suppression des playlist
le tri des playlists

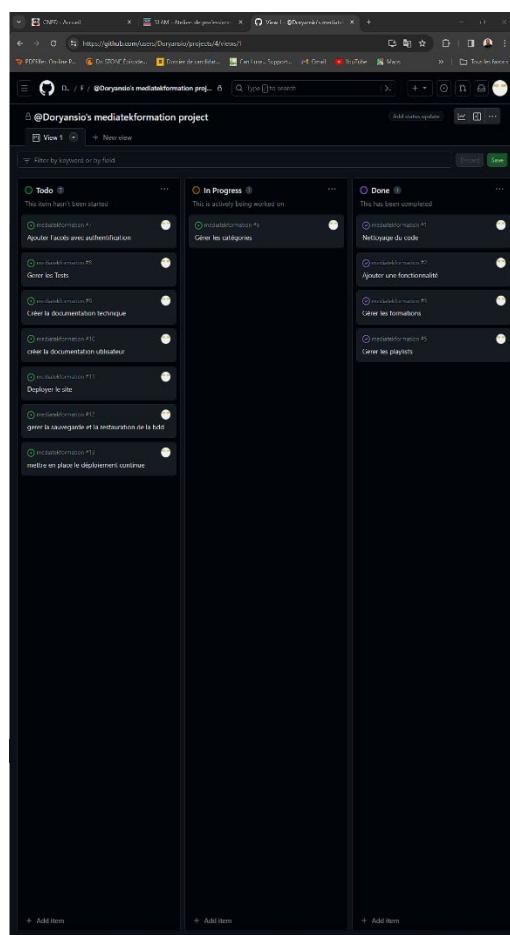
Le fichier de formulaire a été créé en ligne de commande car c'est un fichier qui va reprendre les éléments de l'entité à qui elle correspond pour générer un formulaire avec les mêmes éléments que l'entité

Enfin les dossiers .Twig représente la vue.

Tache 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.

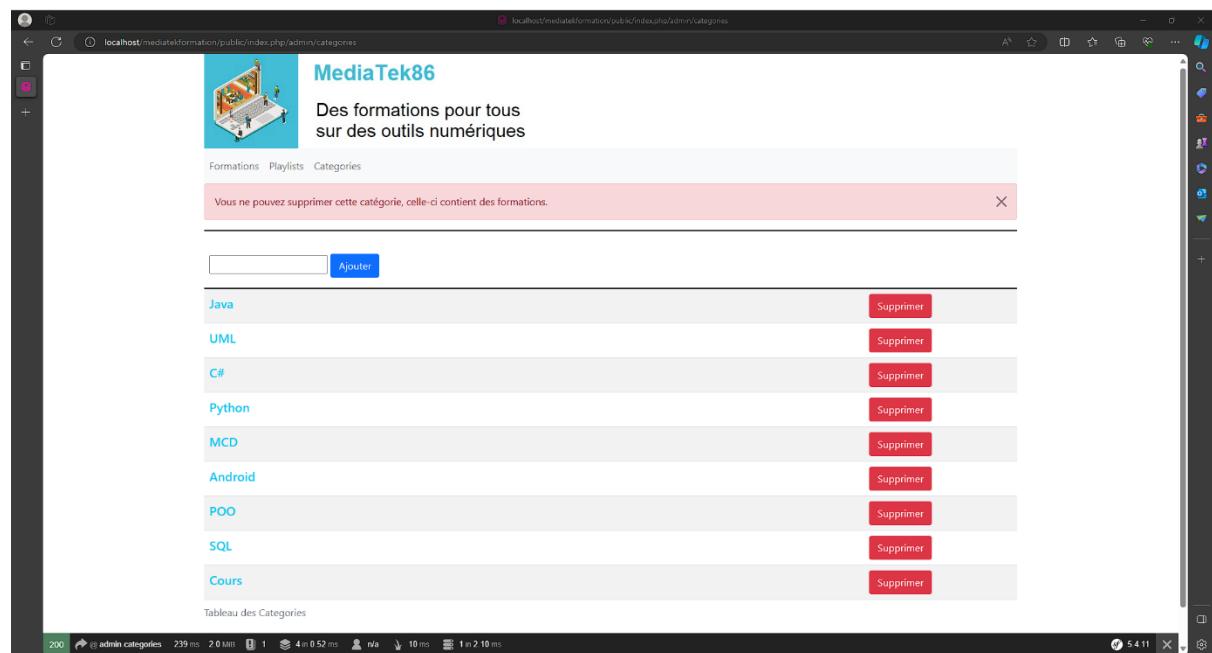
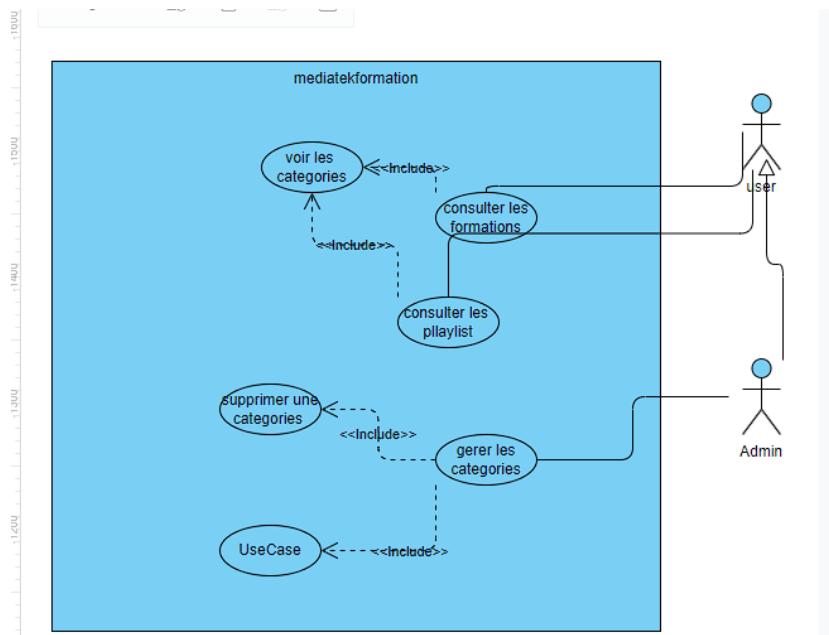
Dans la même page, un mini formulaire doit permettre de saisir et d'ajouter directement une nouvelle catégorie, à condition que le nom de la catégorie n'existe pas déjà.



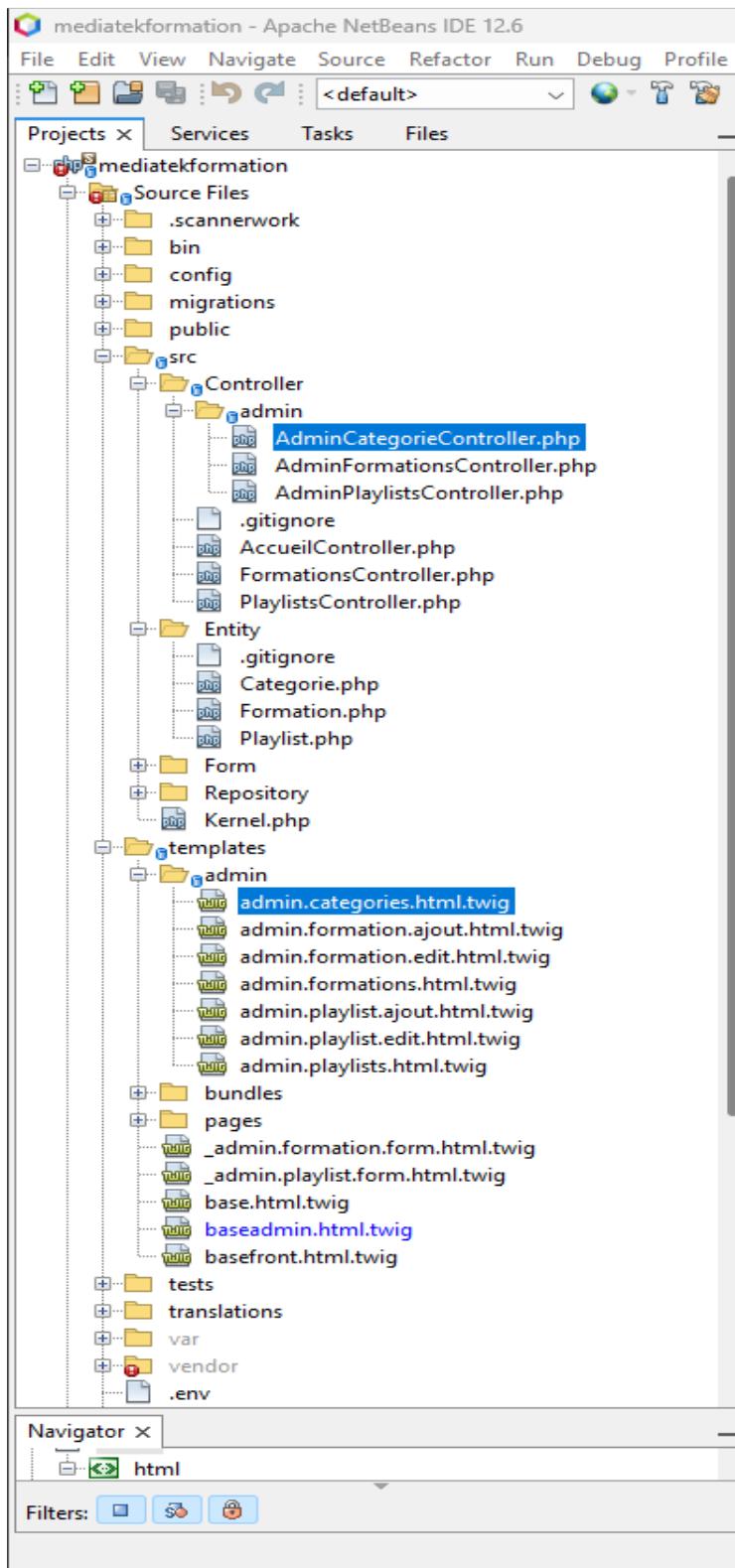
Temps estimé 3h

Temps réel 2h

Voici le diagramme de cas d'utilisation de cette tache



Cette mission consistait à créer une page administrateur pour la gestion des catégories entre autres la suppression de ces derniers cependant les catégories ne peuvent pas être supprimées si elles sont rattachées à une formation et deux catégories ne peuvent pas porter le même nom.

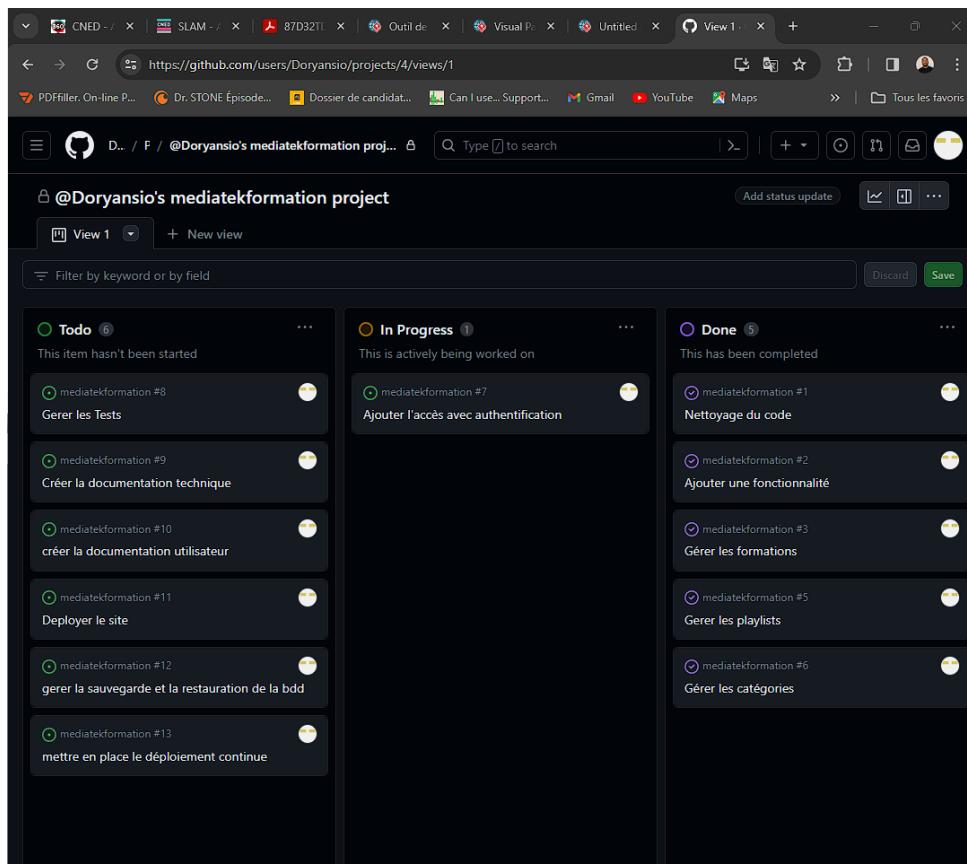


ici le fichier AdminCategorieController.php nous permette de gérer les méthodes d'ajout et de suppressions en mettant des tests dans les méthodes. Ce fichier utilise les repository des formations mais aussi des catégories pour permettre de voir si les catégories que l'on souhaite supprimer sont vides ou bien si ces derniers contiennent des formations.

Tâche 4 : ajouter l'accès avec authentification

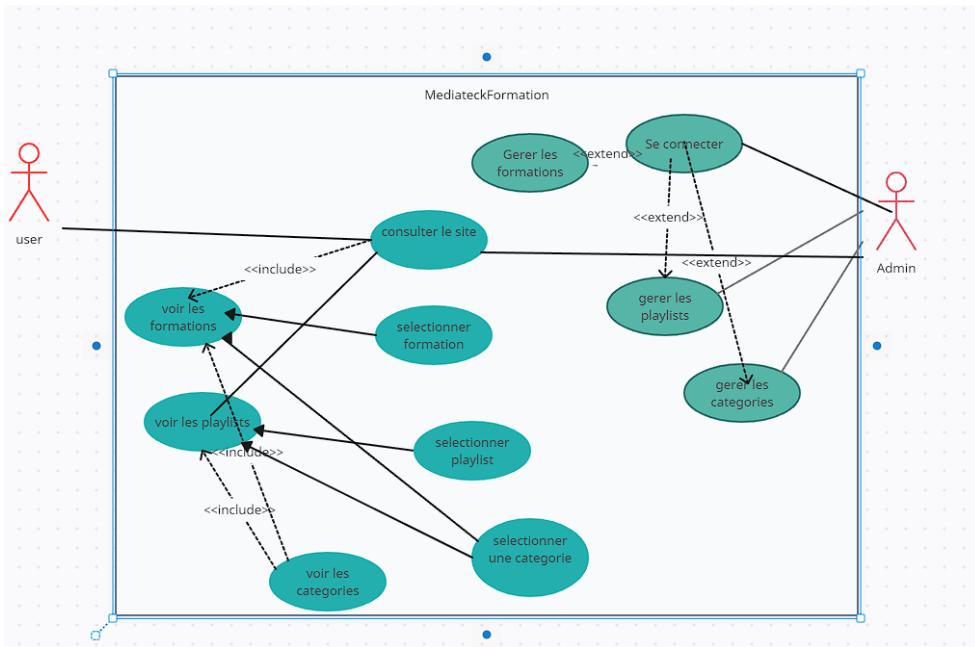
Le back office ne doit être accessible qu'après authentification : un seul profil administrateur doit avoir le droit d'accès. Pour gérer l'authentification, utiliser Keycloak.

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



Temps estimé 4h

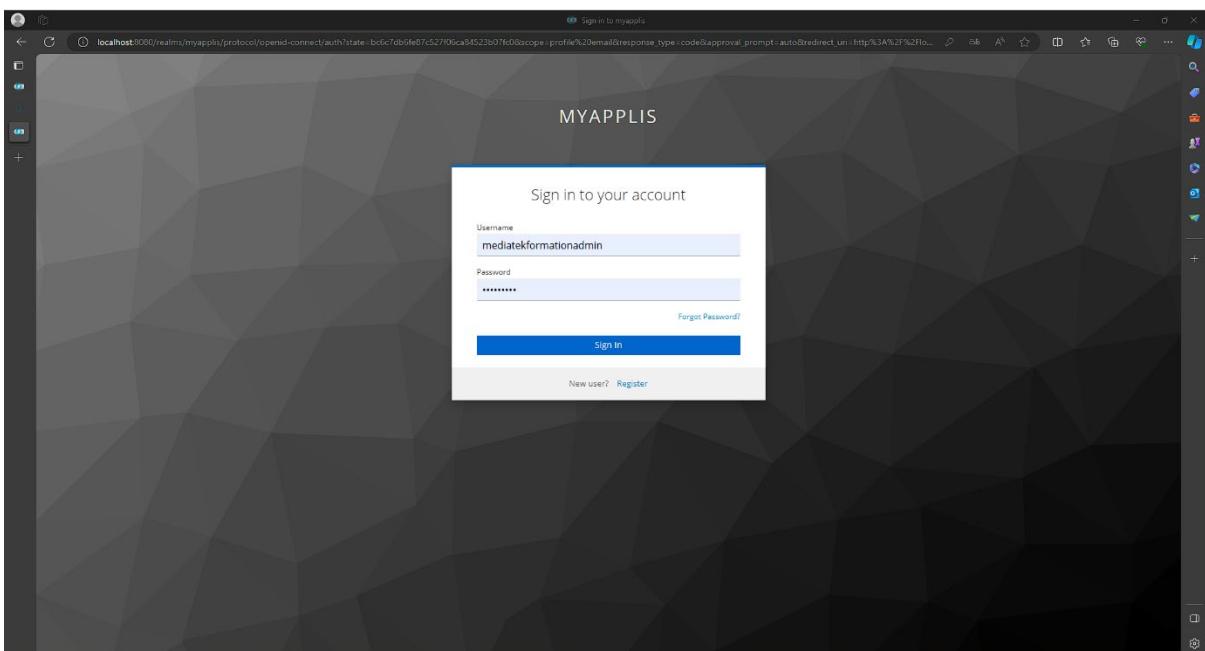
Temps réel 2h



Dans cette mission nous devions ajouter un accès identifier pour accéder à la partie administrateur de notre site. Nous avons utilisé Keycloak. Pour se faire nous avons eu besoin de 2 bundles supplémentaire pour faire le lien entre notre serveur local Keycloak et notre applications php. Plusieurs fichiers ont été modifier directement par la console lorsque nous avons télécharger les bundles.

Il nous a fallu mettre à jour la base de données pour y enregistrer une nouvelle table qui concerne les utilisateurs. Cette nouvelle table a pour attribut un id un mail un rôle et un mot de passe. Mais également un id Keycloak.

Le serveur et l'application étant lié l'application doit avoir en mémoire les informations de l'utilisateur (mots de passe mail etc..) afin de les comparer avec les user référencé sur Keycloak.

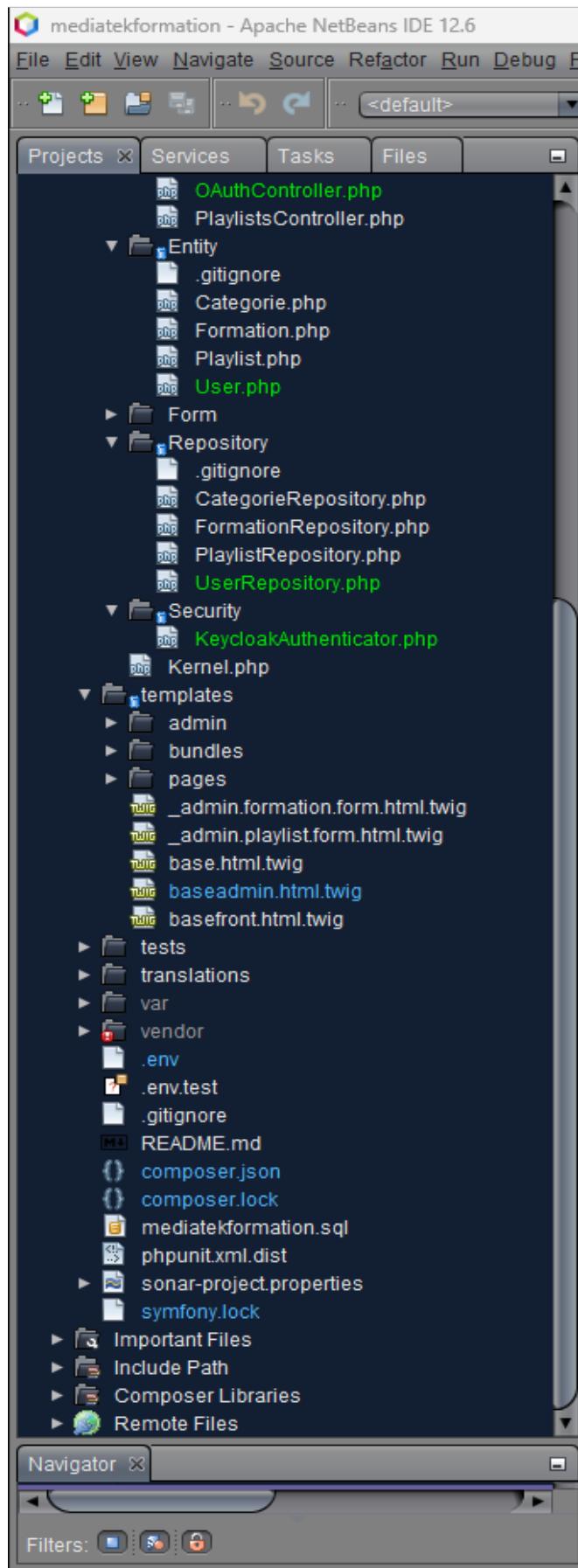


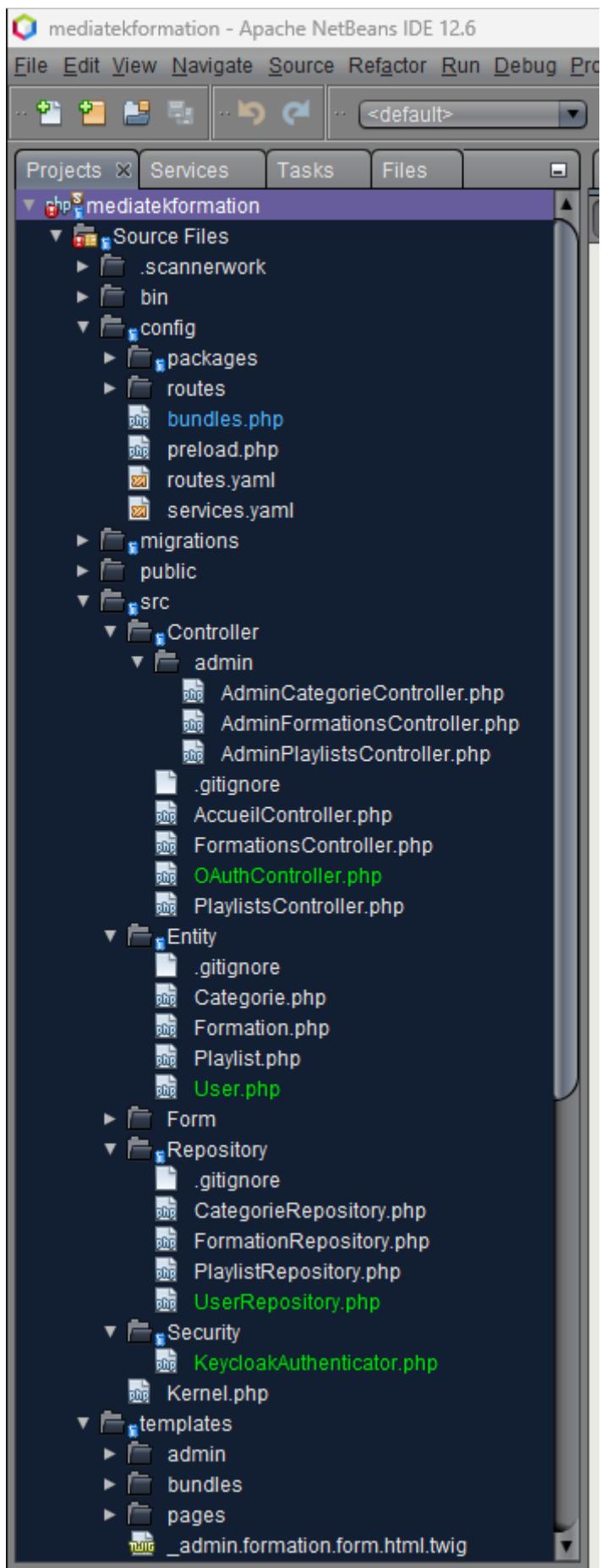
La connexion en mode admin ne se fait que par le biais de l'url ou nous sommes redirigés vers Keycloak en local afin d'entrer nos identifiants de connections, cependant une fois sur la page administrateur il est possible de se déconnecter.

formation	playlist	catégories	date	actions
Eclipse n°4 : WindowBuilder	Eclipse et Java	Java	09/11/2020	Supprimer Editer
Eclipse n°3 : GitHub et Eclipse	Eclipse et Java	Java	07/11/2020	Supprimer Editer
Eclipse n°2 : rétroconception avec ObjectAid	Eclipse et Java	Java UML	05/11/2020	Supprimer Editer
Eclipse n°1 : Installation de l'IDE	Eclipse et Java	Java	03/11/2020	Supprimer Editer

Une fois cela fait nous allons voir que nous ne sommes plus enregistrés en tant qu'utilisateur administrateur dans la barre de gestions Symfony

nous sommes redirigés vers la page d'accueil de la partie front end.





Mission 3 : tester et documenter

Tache 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 :

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

Tests d'intégration sur les Repository :

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

Tests fonctionnels :

Contrôler que la page d'accueil est accessible.

Dans chaque page contenant des listes :

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

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

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

Tests de compatibilité :

Créer un scénario avec Sélénum, sur la partie front office, et le jouer sur plusieurs navigateurs pour tester la compatibilité du site.

The screenshot shows a GitHub project board with the following tasks:

- Todo** (5 items):
 - Créer la documentation technique
 - créer la documentation utilisateur
 - Deployer le site
 - gerer la sauvegarde et la restauration de la bdd
 - mettre en place le déploiement continu
- In Progress** (1 item):
 - Gerer les Tests
- Done** (6 items):
 - Nettoyage du code
 - Ajouter une fonctionnalité
 - Gérer les formations
 - Gerer les playlists
 - Gérer les catégories
 - Ajouter l'accès avec authentification

Temps estimé : 7h

Temps réel : 5 heures

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

Plan de tests

Tests unitaires

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

Tests d'intégration

But du test	Action de contrôle	Résultat attendu	Bilan
Tester l'ajout et la suppression de nouvelles formations, nouvelles playlists, nouvelles catégories	Ne pas avoir la possibilité de supprimer une playlist si elle possède des formations Ne pas pouvoir ajouter une formation sans date de publication, titre	Message d'erreur lors des tests si l'une des conditions n'est pas remplie	Ok

Tests fonctionnels

But du test	Action de contrôle	Résultat attendu	Bilan
Tester toutes les méthodes des repository	Contrôle des tris sur les dates et sur l'ordre par rapport à la bdd Contrôle des recherches dans les formulaires	Les boutons de tri renvoient bien les playlists formations dans l'ordre croissant ou décroissant et les tris sur le nombre de formations et sur la date de parution aussi Les formulaires de recherches renvoient les info recherchées.	Ok

1

Tests de compatibilité

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la compatibilité du site en effectuant des scénarios de test sur plusieurs navigateurs	Exécuter des scénarios via selenium	Les actions faites sur chaque site sont viable et les tests sont ok	Ok

Tâche 2 : créer la documentation technique

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

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

The screenshot shows a GitHub project board titled "@Doryansio's mediatekformation project". The board is divided into three columns: "Todo" (4 items), "In Progress" (1 item), and "Done" (7 items). Each column has a header with a status indicator (green for Todo, orange for In Progress, purple for Done) and a count. Below each header is a brief description of the state.

- Todo** (4 items):
 - meditekformation #10 : créer la documentation utilisateur
 - meditekformation #11 : Deployer le site
 - meditekformation #12 : gerer la sauvegarde et la restauration de la bdd
 - meditekformation #13 : mettre en place le déploiement continu
- In Progress** (1 item):
 - meditekformation #9 : Créer la documentation technique
- Done** (7 items):
 - meditekformation #1 : Nettoyage du code
 - meditekformation #2 : Ajouter une fonctionnalité
 - meditekformation #3 : Gérer les formations
 - meditekformation #5 : Gérer les playlists
 - meditekformation #6 : Gérer les catégories
 - meditekformation #7 : Ajouter l'accès avec authentification
 - meditekformation #8 : Gérer les Tests

At the bottom of the board, there are buttons for "Add status update", "Discard", and "Save".

Temps estimé : 1h

Temps réel : 45min

Bing | CNED - / | CNED SLAM - / | CNED SLAM - / | Générate... | mediatekformation | View 1 | + | - | Tous les favoris

Fichier file:///D:/wamp64/www/mediatekformation_doc/classes/App-Entity-Categorie.html

PDFfiller. On-line P... Dr. STONE Episode... Dossier de candidat... Can I use... Support... Gmail YouTube Maps

mediatekformation

App \ Entity

Categorie

in package Application

[Categorie.php : 13](#)

Tags

ORM\Entity
(repositoryClass=CategorieRepository::class)

Table of Contents

- P [\\$formations](#) : mixed
- P [\\$id](#) : mixed
- P [\\$name](#) : mixed
- M [__construct\(\)](#) : mixed
- M [__toString\(\)](#) : mixed
- M [addFormation\(\)](#) : self
- M [getFormations\(\)](#) : Collection<int, Formation>
- M [getId\(\)](#) : int|null
- M [getName\(\)](#) : string|null
- M [removeFormation\(\)](#) : self
- M [setName\(\)](#) : self

Properties

\$formations

[Categorie.php : 30](#)

```
private mixed $formations
```

Tags

ORM\ManyToMany
(targetEntity=Formation::class, mappedBy="categories")

\$id

[Categorie.php : 20](#)

```
private mixed $id
```

Tags

ORM\Id
ORM\GeneratedValue
ORM\Column
(type="integer")



Tâche 3 : créer la documentation utilisateur

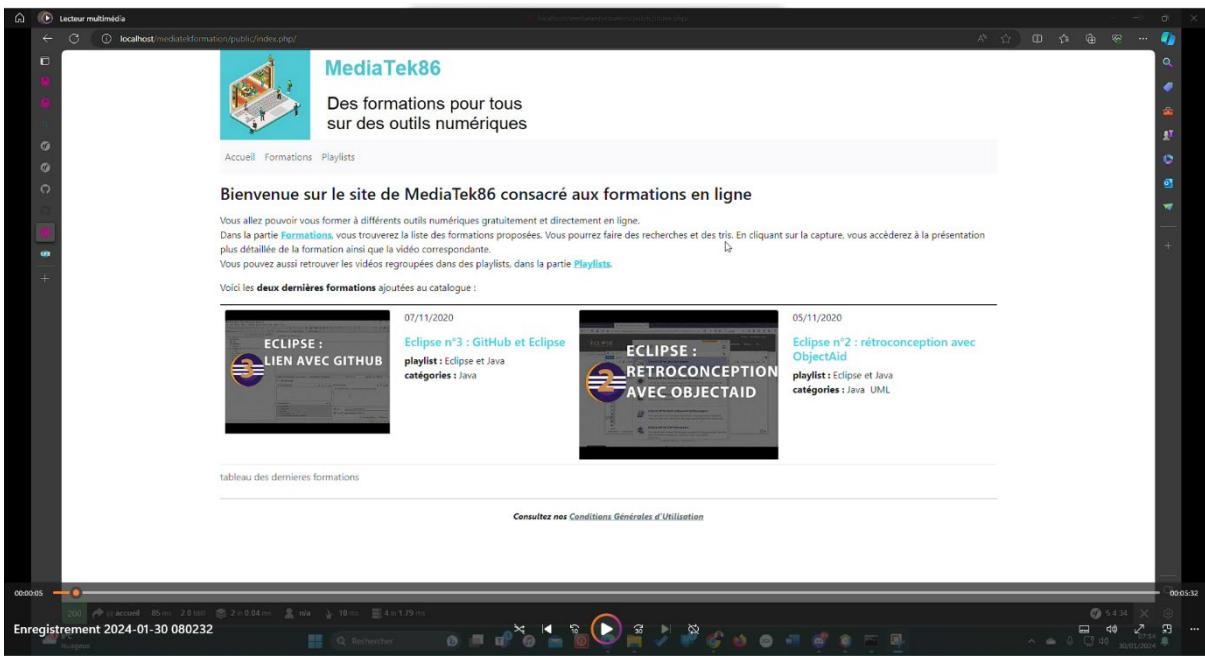
Créer en vidéo qui permet de montrer toutes les fonctionnalités du site (front et back office).
Cette vidéo ne doit pas dépasser les 5mn et doit présenter clairement toutes les fonctionnalités, en montrant les manipulations qui doivent être accompagnées d'explications orales.

Temps estimé 2 heures

Temps réel 1h 30

The screenshot shows a GitHub project board titled '@Doryansio's mediatekformation project'. The board is divided into three columns: 'Todo', 'In Progress', and 'Done'. Each column has a title and a brief description below it.

- Todo:** This item hasn't been started.
 - mediatekformation #11 Deployer le site
 - mediatekformation #12 gerer la sauvegarde et la restauration de la bdd
 - mediatekformation #13 mettre en place le déploiement continu
- In Progress:** This is actively being worked on.
 - mediatekformation #10 créer la documentation utilisateur
- Done:** This has been completed.
 - mediatekformation #1 Nettoyage du code
 - mediatekformation #2 Ajouter une fonctionnalité
 - mediatekformation #3 Gérer les formations
 - mediatekformation #5 Gerer les playlists
 - mediatekformation #6 Gérer les catégories
 - mediatekformation #7 Ajouter l'accès avec authentification
 - mediatekformation #8 Gerer les Tests
 - mediatekformation #9 Créer la documentation technique



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 (voir l'article "Keycloak en ligne et en HTTPS" dans le wiki du dépôt).

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

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

The screenshot shows a GitHub project board with the following structure:

- Todo** (Green circle): This item hasn't been started. One item listed: "mediatekformation #13 mettre en place le déploiement continu".
- In Progress** (Yellow circle): This is actively being worked on. One item listed: "mediatekformation #11 Deployer le site".
- Done** (Purple circle): This has been completed. Nine items listed, all with checkmarks:
 - mediatekformation #1 Nettoyage du code
 - mediatekformation #2 Ajouter une fonctionnalité
 - mediatekformation #3 Gérer les formations
 - mediatekformation #5 Gérer les playlists
 - mediatekformation #6 Gérer les catégories
 - mediatekformation #7 Ajouter l'accès avec authentification
 - mediatekformation #8 Gérer les Tests
 - mediatekformation #9 Gérer les documentations

Temps estimé 2h

Temps réel 1h30.

L'installation de Keycloak en https sur la Vm azure de Microsoft nécessite tout d'abord d'obtenir un certificat et donc d'avoir un serveur apache que l'on peut obtenir avec Xamp ou Wamp dans un premier temps et d'installer certbot. Après avoir fini l'installation et entrer en ligne de commande ouverte en mode administrateur « certbot certonly –webroot » et avoir renseigné les informations nécessaires : mail valide DNS de la Vm et le web root.

Nous obtenons une combinaison de deux fichiers créés « fullchain.pem et prikey.pem »

Ces fichiers vont nous servir lors du lancement de de Keycloak dans la console.

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

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

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

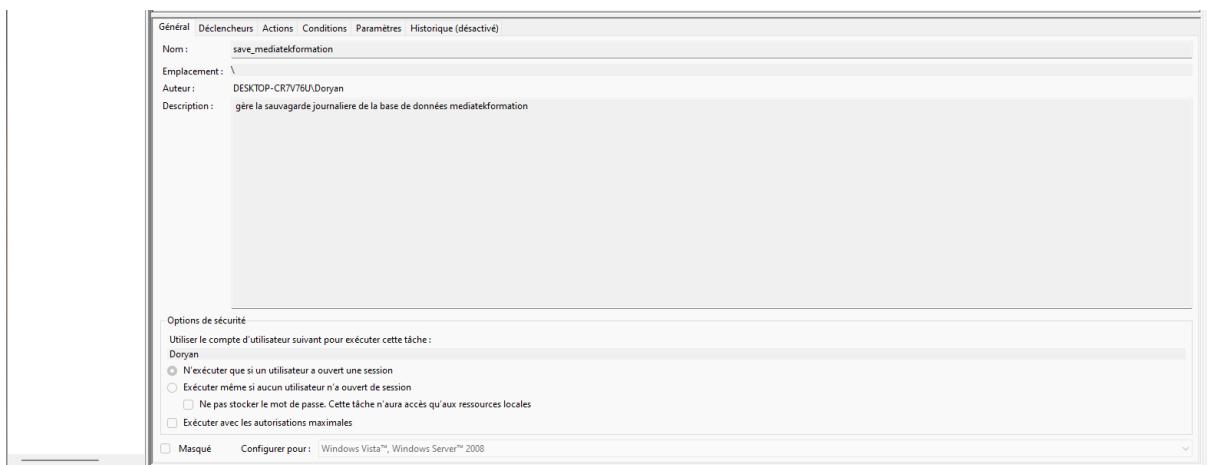
The screenshot shows a Trello board with the following structure:

- Todo:** 2 items
 - mediatekformation #11 Deployer le site
 - mediatekformation #13 mettre en place le déploiement continu
- In Progress:** 1 item
 - mediatekformation #12 gérer la sauvegarde et la restauration de la bdd
- Done:** 3 items
 - mediatekformation #1 Nettoyage du code
 - mediatekformation #2 Ajouter une fonctionnalité
 - mediatekformation #3

Temps estimé 1 heure

Temps réel 1 heure

Cette tâche consiste créer un script donnant accès à la commande SQL dump de Wamp server afin de sauvegarder la base de données mediatekformation puis d'automatiser l'exécution de ce script avec le planificateur de tache Windows.



L'organisation de la sauvegarde étant journalière, à condition que l'utilisateur allume son ordinateur permet d'avoir une sauvegarde la base de données en local mais aussi sur une machine virtuelle.

Pour restaurer la base de données il faudra juste récupérer la dernière sauvegarde et l'exporter dans le php admin local ou sur celui de l'hébergeur de notre site web.

Tâche 3 : mettre en place le déploiement continu

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

Temps estimé : 1 heure

Temps réel : 1 heure

Pour que le déploiement continu via GitHub se fasse, nous devons dans un premier temps créer un compte ftp sur notre hébergeur de site web. Puis sur le GitHub du dépôt nous devons créer un Workspace ou il nous faudra créer un fichier nommé MAIN.yml qui contiendra le script suivant :

```
on: push
name: Deploy website on push
jobs:
  web-deploy:
    name: Deploy
    runs-on: windows-latest
    steps:
      - name: Get latest code
        uses: actions/checkout@v2
```

```
- name: Sync files
uses: SamKirkland/FTP-Deploy-Action@4.3.0
with:
  server: ftp.mediatekformationsio.com
  server-dir: /public_html/mediatekformation/
  username: u536486208.doryan74
  password: ${{ secrets.ftp_password }}
```

une fois ce fichier créer et correctement configurée il nous faudra exécuter un pull pour avoir ce fichier en local dans notre projet

de retour sur le GitHub du dépôt il nous faudra mettre le mot de passe correspondant au compte ftp générer sur l'hébergeur. Pour que lorsque des modifications sont faites en local et pousser le vers le dépôt GitHub ces modifications s'applique au site mis en ligne directement.

Bilan final de l'application

Objectif atteint :

- Maintenance corrective de l'application en accord avec les bonnes pratiques de codage
- Elaboration des tests de l'application (test fonctionnels, test unitaires, test d'intégrations et test de comptabilité)
- Ajout de fonctionnalités correspondant au cahier des charges donné
- Création d'une documentation technique et d'une documentation utilisateur
- Création d'une partie administrateur de l'application permettant la gestion des formations des playlist et des catégories

Problèmes rencontrés :

- Le tri sur le nombre de formations
- Problème du nommage des variables
- Erreur de syntaxe dans le code

