Compte rendu de l'Atelier Professionnel MediaTekFormations

Aina Joy RABARIJAONA

L'atelier professionnel porte sur une application web Symfony dédiée à la consultation et à la gestion de formations et de playlists (projet MediaTekFormations). L'objectif est de prendre en main un existant fourni (cahier des charges, document Missions, dépôt ou archive), de stabiliser et d'enrichir le code, de mettre en place un back-office complet avec authentification, de tester et documenter le produit, puis de le déployer en production avec sauvegarde de base et intégration continue. Le travail s'est déroulé dans un cadre pédagogique avec suivi par Kanban (issues GitHub), branches Git par lot de fonctionnalités et livrables attendus (plan de tests, PV de recette, documentation technique, vidéo utilisateur, compte rendu).

Technologies et outils utilisés

Domaine	**Technologies et outils**
Framework et langage	PHP, Symfony 6
Base de données	MySQL, Doctrine ORM, DQL
Front et gabarits	Twig, Bootstrap
Qualité et tests	SonarLint (NetBeans), PHPUnit
Sécurité	Symfony Security, authentification, CSRF, hachage des mots de passe
Documentation	PhpDoc, phpDocumentor
Outils de développement	NetBeans, Git, GitHub, GitHub Actions, GitHub Projects (Kanban)
Environnement local	WampServer ou stack équivalente
Hébergement et déploiement	Infomaniak (FTP/FTPS), lftp, workflow de déploiement continu
Sauvegarde	mysqldump, script shell, cron

Sommaire

1. Mission 0 : Préparer l'environnement de travail

Durée indicative : 2 h. Installation et configuration de NetBeans avec le plugin SonarLint ; environnement Symfony (PHP, Composer, Git) et serveur de développement type WampServer (ou stack équivalente) ; récupération du cahier des charges, du document Missions et de l'application (dépôt ou archive) ; lecture du README et prise en main du projet ; création de la base de données et import du script SQL fourni ; tests des fonctionnalités existantes du front office ; création du dépôt GitHub personnel et d'un projet Kanban (issues en colonnes To do, In progress, Done) pour suivre les tâches des missions suivantes.

Ce que j'ai appris

Mise en route d'une stack PHP/Symfony locale, alignement des versions PHP et des extensions avec Symfony 6, configuration de DATABASE_URL et prise en main de GitHub Projects pour le suivi en colonnes.

Problèmes rencontrés et traitements

Alignement des versions PHP et des extensions avec Symfony 6 ; première configuration de DATABASE_URL et de l'environnement local ; prise en main de GitHub Projects ou équivalent pour le suivi Kanban.

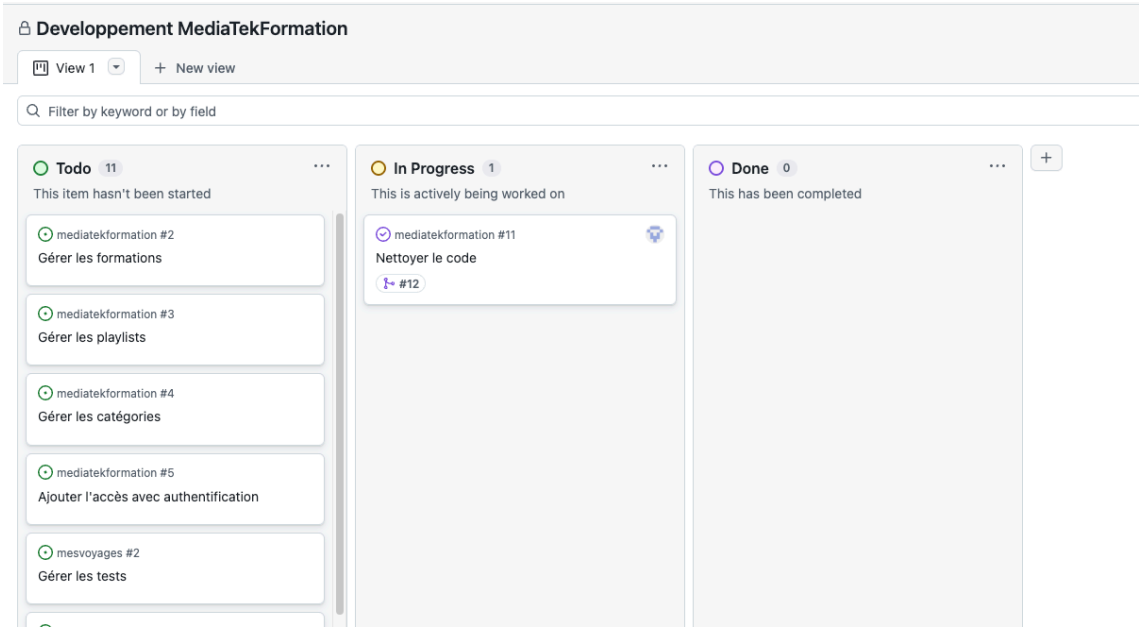
2. Mission 1 : Nettoyer et optimiser le code existant

2.1 Tâche 1 : Nettoyer le code

Nettoyage du code selon SonarLint, sans modifier les fichiers imposés par Symfony. Suppressions des chaînes en dur pour limiter les duplications ; constantes en majuscules ; fusion des tests imbriqués inutiles ; attribut alt sur toutes les images ; descriptions sur toutes les tables. Temps estimé 2 h, temps réel 2 h 30.

Actions réalisées

Utilisation du plugin SonarLint dans NetBeans : repérer chaque règle, corriger, vérifier la disparition de l'alerte. Dans src : corrections Critical (php:S115 conventions de constantes ; php:S1192 chaînes dupliquées extraites vers des constantes dans PlaylistsController et FormationsController ; php:S121 accolades sur les structures de contrôle ; php:S131 clause default sur les switch). Major : php:S1066 fusion d'if imbriqués. Minor : php:S1301 switch à moins de trois cas réécrit en if ; php:S1784 visibilité du constructeur ; php:S1808 formatage. Dans les templates : Web:TableWithoutCaptionCheck avec description des tableaux (accueil, formations, playlists) ; ajustements de formatage sur plusieurs fichiers.



Preuve d'avancement (SonarLint).

src/Entity/Formation.php +3 -3

```
@@ -15,7 +15,7 @@ class Formation
15 15      /**
16 16       * Début de chemin vers les images
17 17       */
18    -     private const cheminImage = "https://i.ytimg.com/vi/";
   18 +     private const CHEMIN_IMAGE = "https://i.ytimg.com/vi/";
19 19
20 20      #[ORM\Id]
21 21      #[ORM\GeneratedValue]
@@ -110,12 +110,12 @@ public function setVideoId(?string $videoId):
    static
110 110
111 111      public function getMiniature(): ?string
112 112      {
113    -         return self::cheminImage.$this->videoId."/default.jpg";
   113 +         return self::CHEMIN_IMAGE.$this->videoId."/default.jpg";
114 114      }
115 115
116 116      public function getPicture(): ?string
117 117      {
118    -         return self::cheminImage.$this->videoId."/hqdefault.jpg";
   118 +         return self::CHEMIN_IMAGE.$this->videoId."/hqdefault.jpg";
119 119      }
120 120
121 121      public function getPlaylist(): ?playlist
```

php:S115, constantes.

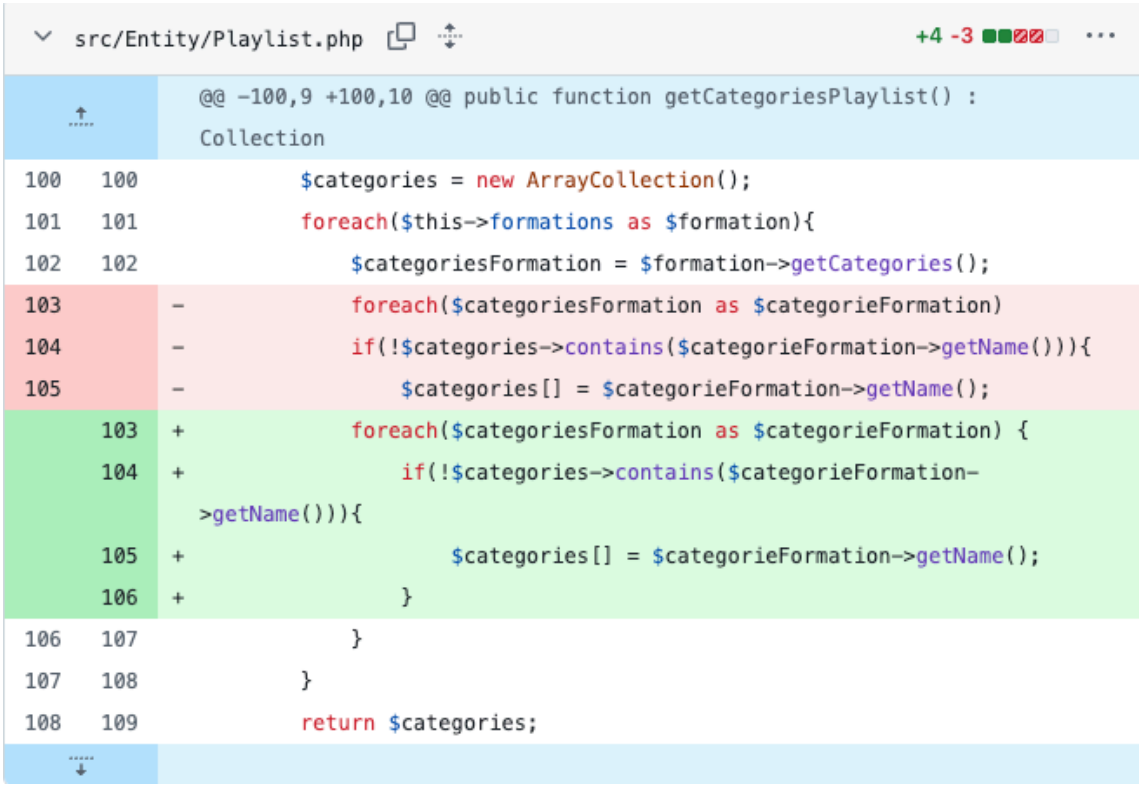
…troller/FormationsController.php +5 -3

```
@@ -15,6 +15,8 @@
 */
class FormationsController extends AbstractController {

+    private const FORMATIONS_TEMPLATE = 'pages/formations.html.twig';
+
    /**
     *
     * @var FormationRepository
@@ -36,7 +38,7 @@ function __construct(FormationRepository
$formationRepository, CategorieReposito
    public function index(): Response{
        $formations = $this->formationRepository->findAll();
        $categories = $this->categorieRepository->findAll();
-        return $this->render("pages/formations.html.twig", [
+        return $this->render(self::FORMATIONS_TEMPLATE, [
            'formations' => $formations,
            'categories' => $categories
        ]);
@@ -46,7 +48,7 @@ public function index(): Response{
    public function sort($champ, $ordre, $table=""): Response{
        $formations = $this->formationRepository->findAllOrderBy($champ,
$ordre, $table);
        $categories = $this->categorieRepository->findAll();
-        return $this->render("pages/formations.html.twig", [
+        return $this->render(self::FORMATIONS_TEMPLATE, [
            'formations' => $formations,
            'categories' => $categories
        ]);
@@ -57,7 +59,7 @@ public function findAllContain($champ, Request
$request, $table=""): Response{
        $formations = $this->formationRepository->
>findByContainValue($champ, $valeur, $table);
        $categories = $this->categorieRepository->findAll();
-        return $this->render("pages/formations.html.twig", [
+        return $this->render(self::FORMATIONS_TEMPLATE, [
            'formations' => $formations,
            'categories' => $categories,
            'valeur' => $valeur,
```

…ntroller/PlaylistsController.php +6 -4

```
@@ -15,7 +15,9 @@
 * @author emds
 */
class PlaylistsController extends AbstractController {
-
+    private const PLAYLISTS_TEMPLATE = 'pages/playlists.html.twig';
+
    /**
     *
     * @var PlaylistRepository
@@ -36,7 +38,7 @@ function __construct(PlaylistRepository
$playlistRepository, CategorieReposito
    public function index(): Response{
        $playlists = $this->playlistRepository->findAll();
        $categories = $this->categorieRepository->findAll();
-        return $this->render("pages/playlists.html.twig", [
+        return $this->render(self::PLAYLISTS_TEMPLATE, [
            'playlists' => $playlists,
            'categories' => $categories
        ]);
@@ -46,7 +48,7 @@ public function index(): Response{
    public function sort($champ, $ordre, $table=""): Response{
        $playlists = $this->playlistRepository->findAllOrderBy($champ,
$ordre, $table);
        $categories = $this->categorieRepository->findAll();
-        return $this->render("pages/playlists.html.twig", [
+        return $this->render(self::PLAYLISTS_TEMPLATE, [
            'playlists' => $playlists,
            'categories' => $categories
        ]);
@@ -57,7 +59,7 @@ public function findAllContain($champ, Request
$request, $table=""): Response{
        $playlists = $this->playlistRepository->
>findByContainValue($champ, $valeur, $table);
        $categories = $this->categorieRepository->findAll();
-        return $this->render("pages/playlists.html.twig", [
+        return $this->render(self::PLAYLISTS_TEMPLATE, [
            'playlists' => $playlists,
            'categories' => $categories,
            'valeur' => $valeur,
```

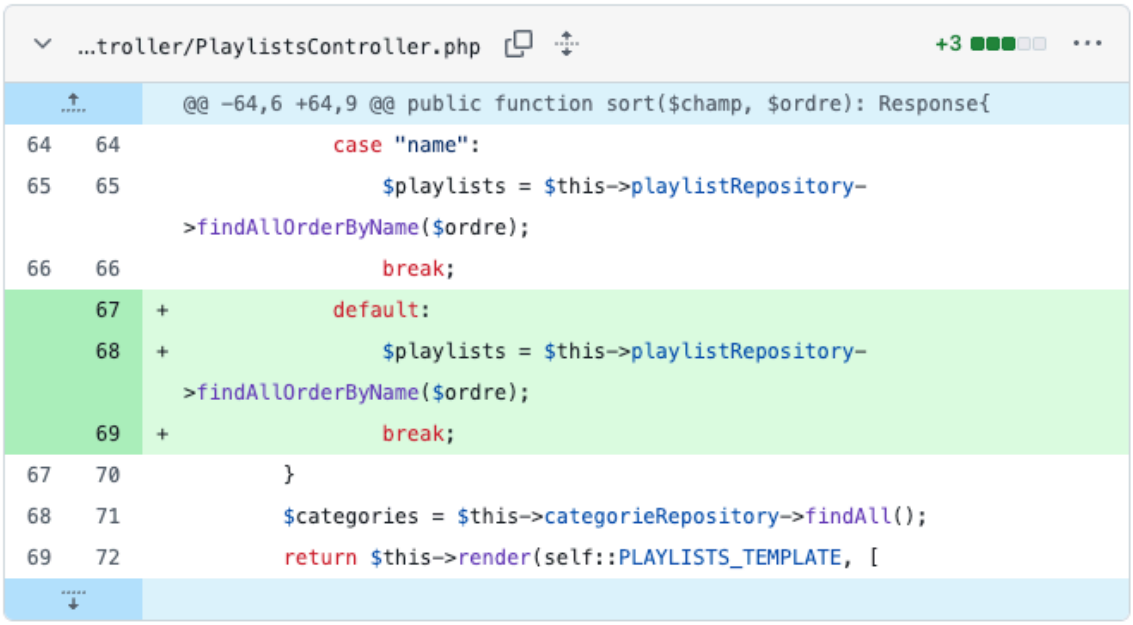
php:S1192, chaînes dupliquées.



src/Entity/Playlist.php +4 -3

```
@@ -100,9 +100,10 @@ public function getCategoriesPlaylist() : Collection
100 100         $categories = new ArrayCollection();
101 101         foreach($this->formations as $formation){
102 102             $categoriesFormation = $formation->getCategories();
103     -           foreach($categoriesFormation as $categorieFormation)
104     -           if(!$categories->contains($categorieFormation->getName())){
105     -               $categories[] = $categorieFormation->getName();
    103 +           foreach($categoriesFormation as $categorieFormation) {
    104 +               if(!$categories->contains($categorieFormation->getName())){
    105 +                   $categories[] = $categorieFormation->getName();
    106 +               }
106 107             }
107 108         }
108 109         return $categories;
```

php:S121, accolades sur les structures de contrôle.



…troller/PlaylistsController.php +3

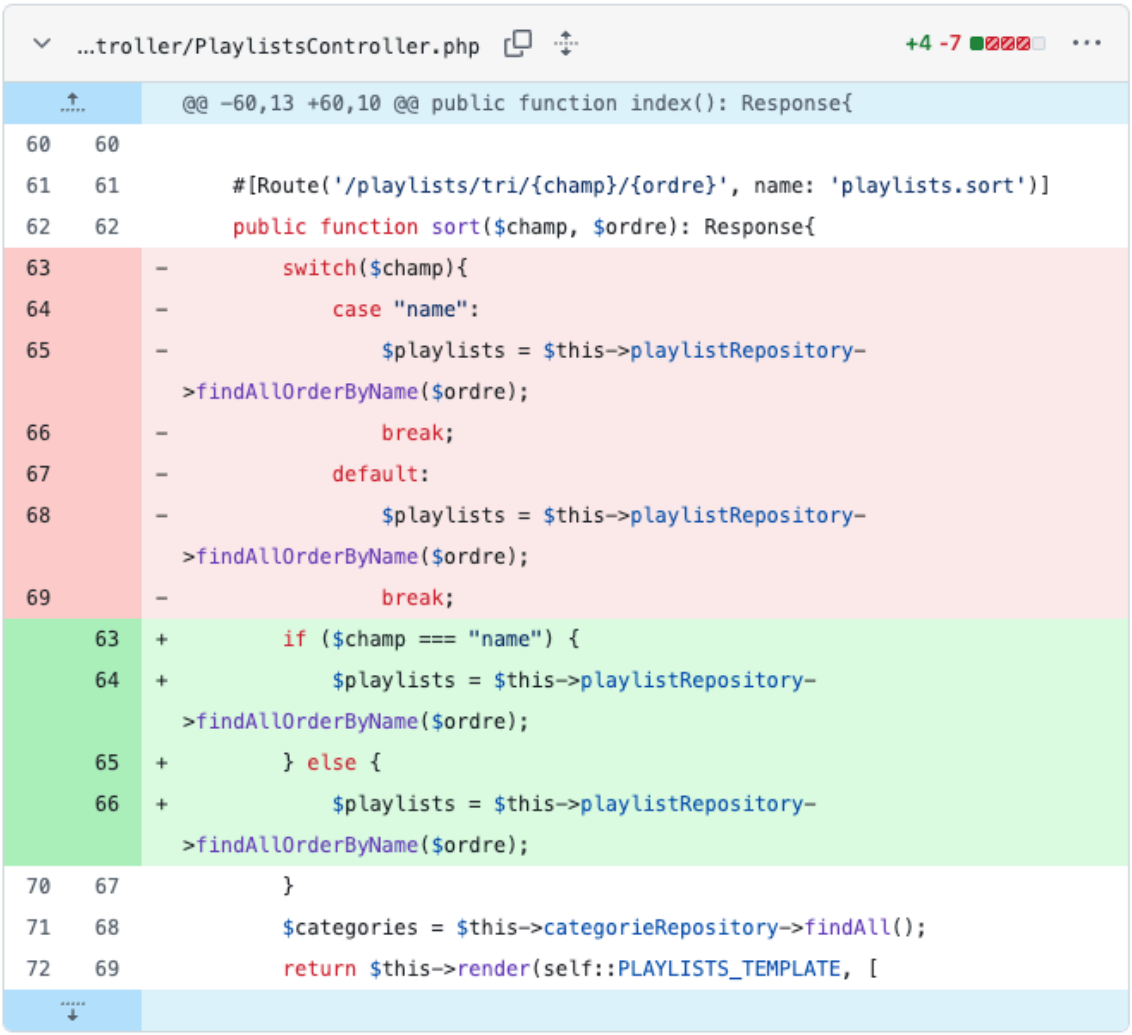
```
@@ -64,6 +64,9 @@ public function sort($champ, $ordre): Response{
64 64              case "name":
65 65                  $playlists = $this->playlistRepository->findAllOrderByName($ordre);
66 66                  break;
   67 +            default:
   68 +                $playlists = $this->playlistRepository->findAllOrderByName($ordre);
   69 +                break;
67 70          }
68 71          $categories = $this->categorieRepository->findAll();
69 72          return $this->render(self::PLAYLISTS_TEMPLATE, [
```

php:S131, clause default sur les switch.

src/Entity/Playlist.php +2 -5

```
@@ -82,11 +82,8 @@ public function addFormation(Formation $formation): static
82  82
83  83      public function removeFormation(Formation $formation): static
84  84      {
85      -       if ($this->formations->removeElement($formation)) {
86      -           // set the owning side to null (unless already changed)
87      -           if ($formation->getPlaylist() === $this) {
88      -               $formation->setPlaylist(null);
89      -           }
    85  +       if ($this->formations->removeElement($formation) && $formation->getPlaylist() === $this) {
    86  +           $formation->setPlaylist(null);
90  87          }
91  88
92  89          return $this;
```

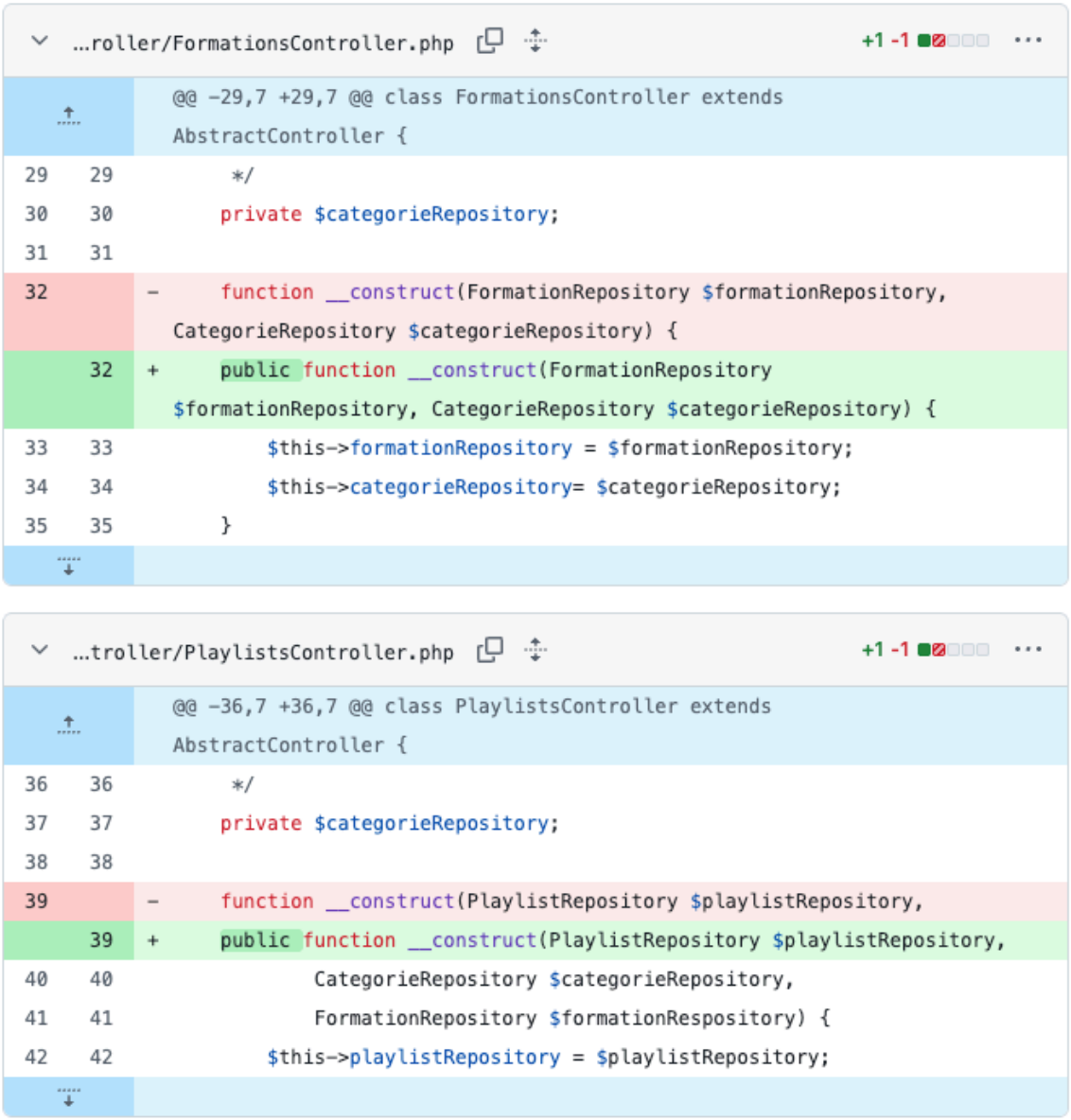
php:S1066, fusion d'if imbriqués.



…troller/PlaylistsController.php +4 -7

```
@@ -60,13 +60,10 @@ public function index(): Response{
60  60
61  61      #[Route('/playlists/tri/{champ}/{ordre}', name: 'playlists.sort')]
62  62      public function sort($champ, $ordre): Response{
63      -       switch($champ){
64      -           case "name":
65      -               $playlists = $this->playlistRepository->findAllOrderByName($ordre);
66      -               break;
67      -           default:
68      -               $playlists = $this->playlistRepository->findAllOrderByName($ordre);
69      -               break;
    63  +       if ($champ === "name") {
    64  +           $playlists = $this->playlistRepository->findAllOrderByName($ordre);
    65  +       } else {
    66  +           $playlists = $this->playlistRepository->findAllOrderByName($ordre);
70  67            }
71  68            $categories = $this->categorieRepository->findAll();
72  69            return $this->render(self::PLAYLISTS_TEMPLATE, [
```

php:S1301, switch réécrit en if.



…roller/FormationsController.php +1 -1

```
@@ -29,7 +29,7 @@ class FormationsController extends AbstractController {
29 29          */
30 30         private $categorieRepository;
31 31
32    -      function __construct(FormationRepository $formationRepository, CategorieRepository $categorieRepository) {
   32 +      public function __construct(FormationRepository $formationRepository, CategorieRepository $categorieRepository) {
33 33             $this->formationRepository = $formationRepository;
34 34             $this->categorieRepository= $categorieRepository;
35 35         }
```

…troller/PlaylistsController.php +1 -1

```
@@ -36,7 +36,7 @@ class PlaylistsController extends AbstractController {
36 36          */
37 37         private $categorieRepository;
38 38
39    -      function __construct(PlaylistRepository $playlistRepository,
   39 +      public function __construct(PlaylistRepository $playlistRepository,
40 40                 CategorieRepository $categorieRepository,
41 41                 FormationRepository $formationRespository) {
42 42             $this->playlistRepository = $playlistRepository;
```

php:S1784, visibilité du constructeur.

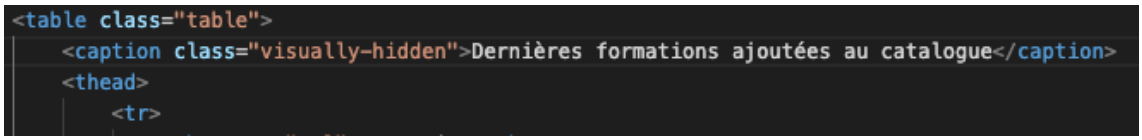
…ontroller/AccueilController.php +6 -4

```
@@ -11,8 +11,8 @@
 *
 * @author emds
 */
- class AccueilController extends AbstractController{
-
+ class AccueilController extends AbstractController
+ {
     /**
      * @var FormationRepository
      */
@@ -27,15 +27,17 @@ public function __construct(FormationRepository
$repository) {
     }

     #[Route('/', name: 'accueil')]
-    public function index(): Response{
+    public function index(): Response
+    {
         $formations = $this->repository->findAllLasted(2);
         return $this->render("pages/accueil.html.twig", [
             'formations' => $formations
         ]);
     }

     #[Route('/cgu', name: 'cgu')]
-    public function cgu(): Response{
+    public function cgu(): Response
+    {
         return $this->render("pages/cgu.html.twig");
     }
 }
```

php:S1808, formatage.

```
...roller/FormationsController.php                                  +12 -7
@@ -13,8 +13,8 @@
13  13   *
14  14   * @author emds
15  15   */
16      - class FormationsController extends AbstractController {
17      -
        + class FormationsController extends AbstractController
        + {
18  18       private const FORMATIONS_TEMPLATE = 'pages/formations.html.twig';
19  19
20  20       /**
@@ -29,13 +29,15 @@ class FormationsController extends
AbstractController {
29  29        */
30  30       private $categorieRepository;
31  31
32      -     public function __construct(FormationRepository
$formationRepository, CategorieRepository $categorieRepository) {
        +     public function __construct(FormationRepository
$formationRepository, CategorieRepository $categorieRepository)
        +     {
33  34           $this->formationRepository = $formationRepository;
34  35           $this->categorieRepository= $categorieRepository;
35  36       }
36  37
37  38       #[Route('/formations', name: 'formations')]
38      -     public function index(): Response{
        +     public function index(): Response
        +     {
39  41           $formations = $this->formationRepository->findAll();
40  42           $categories = $this->categorieRepository->findAll();
41  43           return $this->render(self::FORMATIONS_TEMPLATE, [
@@ -45,7 +47,8 @@ public function index(): Response{
```

php:S1808, formatage (suite).



```
<table class="table">
    <caption class="visually-hidden">Dernières formations ajoutées au catalogue</caption>
    <thead>
        <tr>
```

Web:TableWithoutCaptionCheck, descriptions des tableaux Twig.

AccueilController.php src/Controller

```
21 hidden lines
     *
     * @param FormationRepository $repository
     */
    public function __construct(FormationRepository $repository) {
    public function __construct(FormationRepository $repository)
    {
        $this->repository = $repository;
    }

16 hidden lines
```

PlaylistsController.php src/Controller

```
35 hidden lines
     */
    private $categorieRepository;

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

php:S1808, formatage AccueilController, Playlist, PlaylistsController.

Playlist.php src/Entity

```
97 hidden lines
        foreach ($this->formations as $formation) {
            $categoriesFormation = $formation->getCategories();
            foreach ($categoriesFormation as $categorieFormation) {
                if (!$categories->contains($categorieFormation->getName())){
                if (!$categories->contains($categorieFormation->getName())) {
                    $categories[] = $categorieFormation->getName();
                }
            }
```

php:S1808, formatage (complément).

Bilan

Toutes les issues signalées sur les fichiers concernés sont corrigées. Le code respecte les conventions de nommage, de structure et de formatage recommandées par SonarLint.

Ce que j'ai appris

Lecture ciblée des règles SonarLint (php:S115, S1192, S121, S131, S1066, S1301, S1784, S1808) et application aux contrôleurs PHP et aux gabarits Twig ; discipline PSR-12 et distinction entre code applicatif et fichiers imposés par le framework.

Références

Règles SonarLint citées dans le bilan (php:S115, S1192, S121, S131, S1066, S1301, S1784, S1808) ; contrôle Web:TableWithoutCaptionCheck sur les tableaux Twig.

Obstacles

Volume important de règles à traiter fichier par fichier (PHP et Twig), ce qui a dépassé le temps estimé. Distinction des fichiers imposés par Symfony pour ne pas toucher au cœur du framework. Alerte sur l'en-tête de tableau de la page d'accueil laissée en dernier selon l'énoncé. Harmonisation du formatage sans régression fonctionnelle.

2.2 Tâche 2 : Ajouter une fonctionnalité

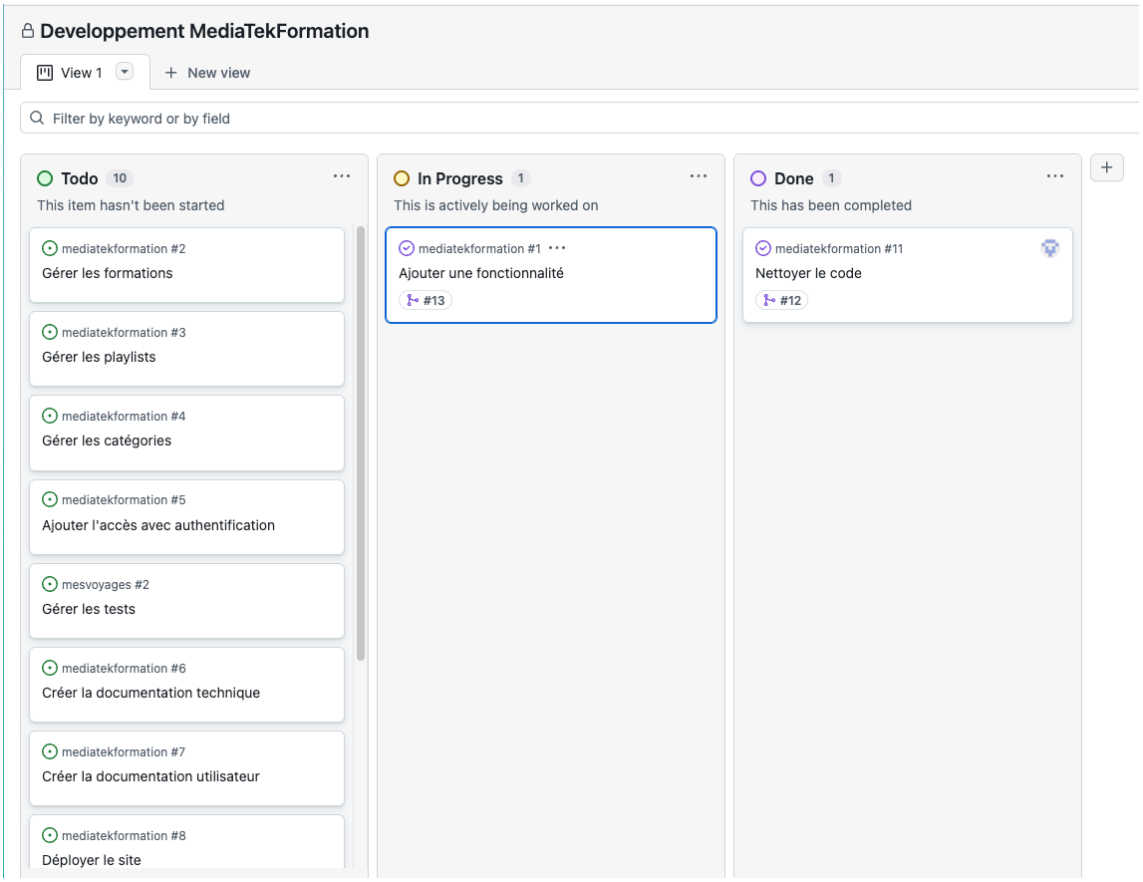
Affichage du nombre de formations par playlist et tri croissant ou décroissant sur ce critère, cumulable avec les filtres existants par nom et par catégorie. Temps estimé 2 h ; temps réel 3 h de développement et 30 min de rédaction du bilan.

Actions réalisées

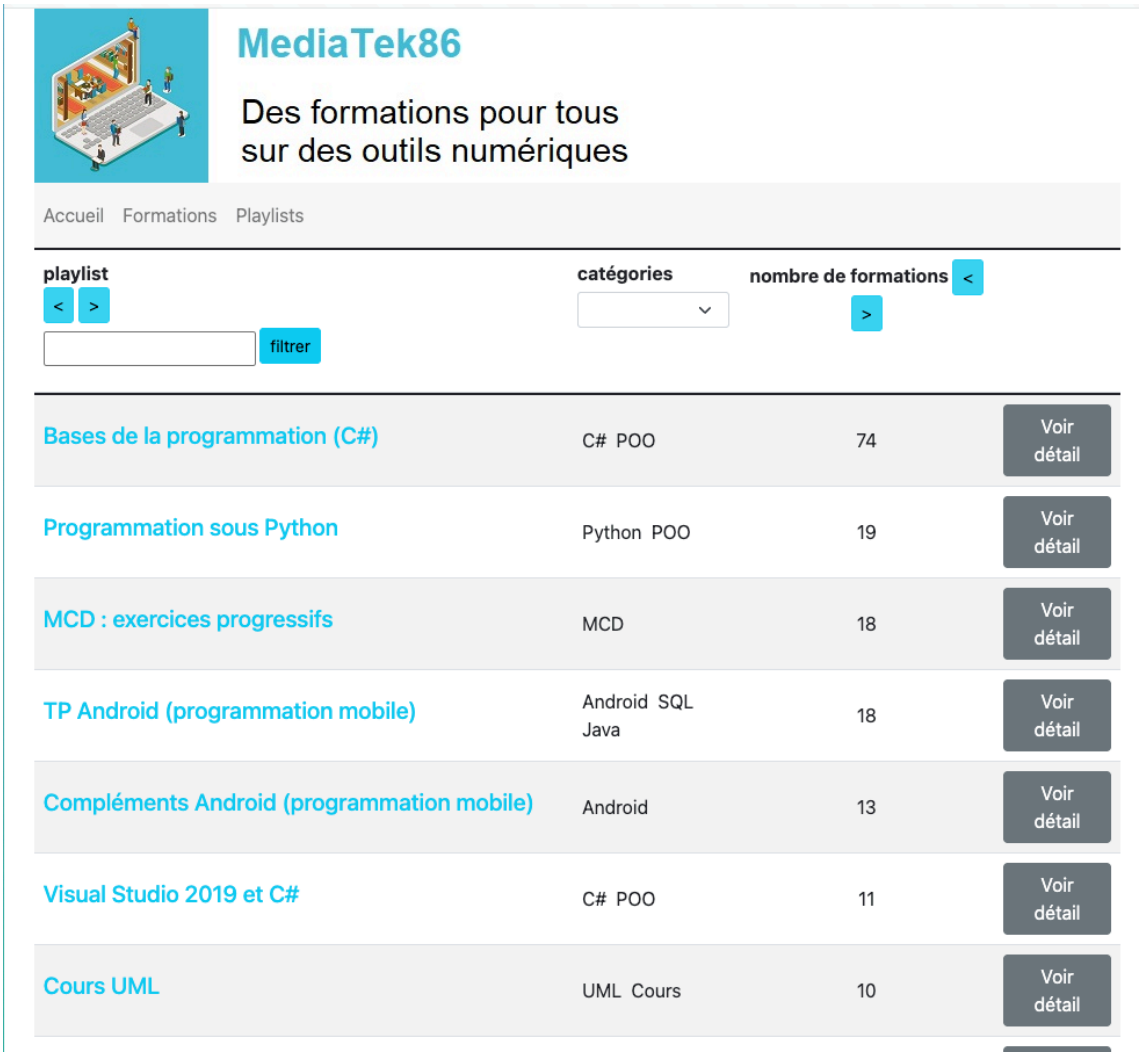
Dans PlaylistsController : méthode sortByNombreFormations(string ordre, Request request), route GET /playlists/tri/nombreformations/{ordre}, récupération des filtres en query string, appel repository dédié si filtre actif, sinon liste triée sur le compteur ; paramètres valeur, table et champ renvoyés à la vue pour conserver les filtres. Route déclarée avant la route générique de tri. Méthode sort adaptée pour cumuler filtre et tri par nom ; findAllContain ajustée pour transmettre champ à la vue.

Dans PlaylistRepository : findAllOrderByNombreFormations(ordre) avec leftJoin, groupBy sur l'identifiant, orderBy sur le nombre ; findByContainValueOrderByNombreFormations pour filtre + tri sur le compteur ; findByContainValue accepte un paramètre ordre pour le tri par nom.

Dans templates/pages/playlists.html.twig : colonne nombre de formations, liens ASC/DESC, conservation des paramètres recherche, champ et table dans l'URL ; affichage playlists[k].nombreformations. Dans playlist.html.twig : affichage du nombre sur la fiche playlist.



Preuve d'avancement.



Liste des playlists avec colonne et tri.

Preuve fonctionnelle complémentaire.

Bilan

L'utilisateur trie les playlists par nombre de formations en ordre croissant ou décroissant, peut filtrer puis changer de tri sans perdre le contexte, et voit le compteur en liste et en page de détail.

Ce que j'ai appris

Ordre des routes Symfony (route spécifique avant route générique), passage systématique des paramètres de requête à la vue et au repository, et requêtes DQL avec COUNT, LEFT JOIN, GROUP BY et ORDER BY sur agrégat (playlists sans formation, compteur à 0).

Références

Historique Git sur la branche mission1-tri-playlist : compteur, tri asc/desc, adaptation contrôleur, dépôt et templates front.

Obstacles

Conflits potentiels de routes ; cohérence du cumul filtres + tri ; durée supérieure au créneau de 2 h prévu.

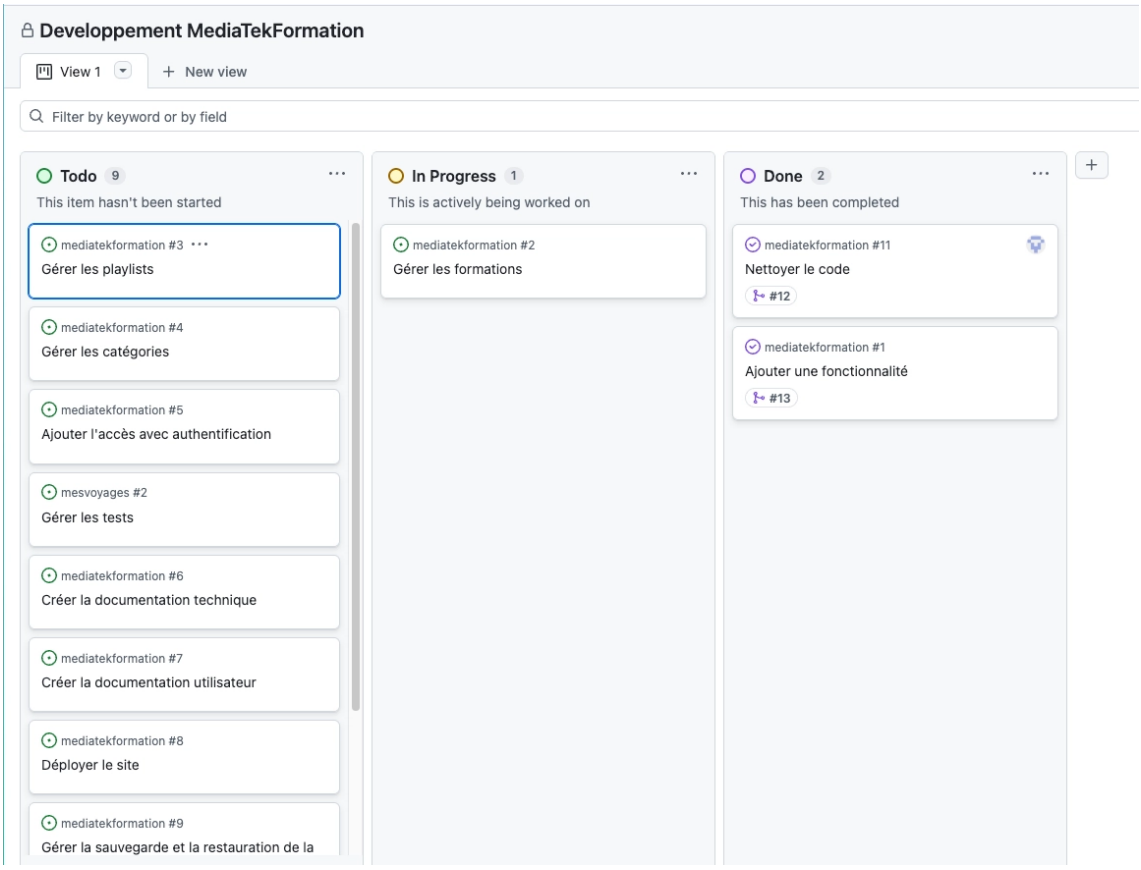
3. Mission 2 : Coder la partie back-office

3.1 Tâche 1 : Gérer les formations

Liste des formations avec suppression (après confirmation) et modification ; si suppression, retrait de la playlist associée. Mêmes tris et filtres que le front. Bouton vers formulaire d'ajout. Contrôles de saisie : description et catégories facultatives ; playlist et catégories en listes ; date sélectionnable, non postérieure au jour. Édition sur le même formulaire prérempli. Temps estimé 5 h, temps réel 4 h 30.

Actions réalisées

FormationAdminController sous /admin : index (liste, filtres, tris), new (formulaire vide, persist, flush), edit (prérempli, flush), delete (token CSRF, suppression de la relation playlist si besoin, remove, flush). FormationType : title NotBlank, description optionnel, playlist, categories multiples, publishedAt avec contrainte de date non future. Templates admin/formation/index.html.twig et form.html.twig ; baseadmin.html.twig avec menu Formations, Playlists, Catégories.



Kanban, tâche en cours.

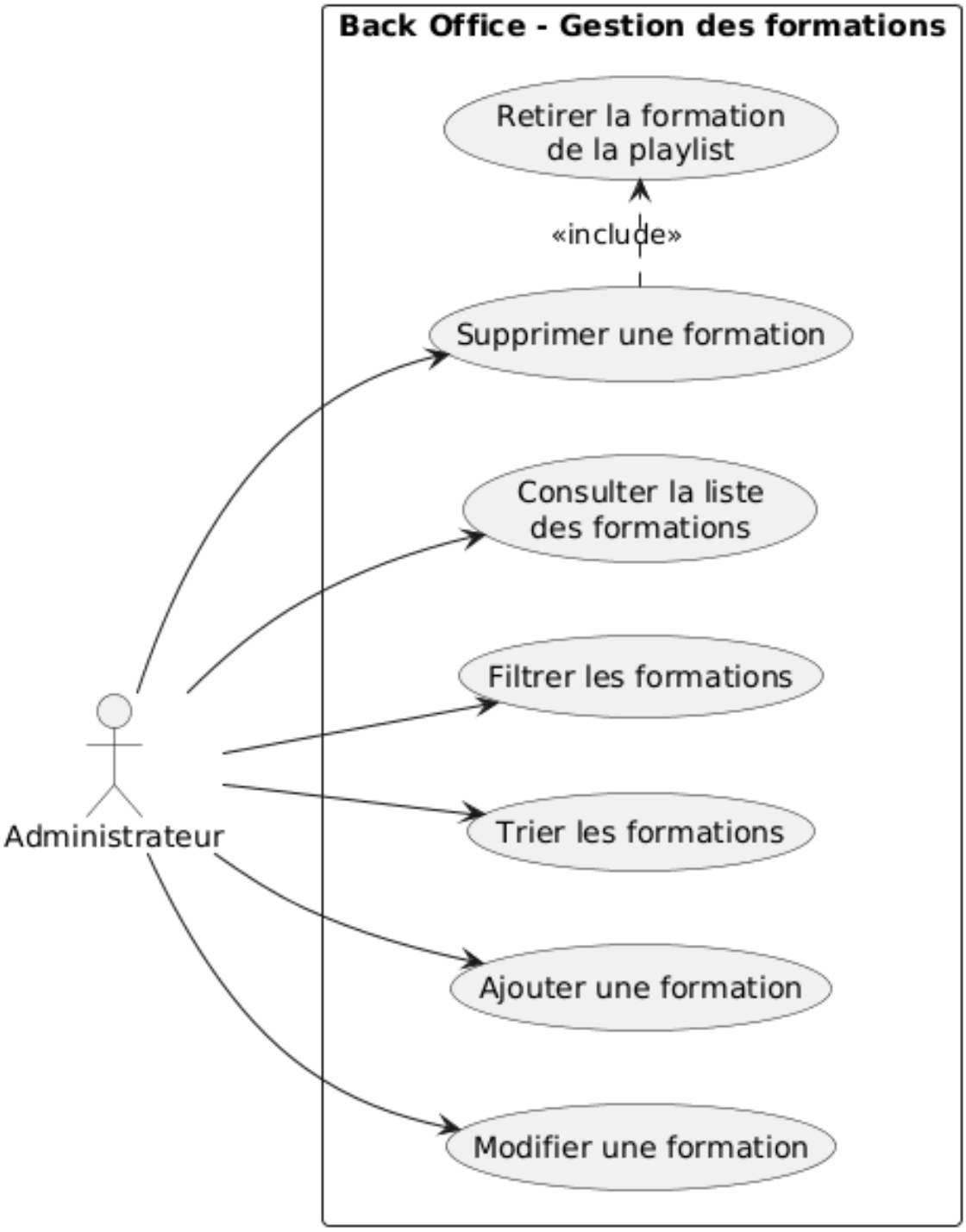
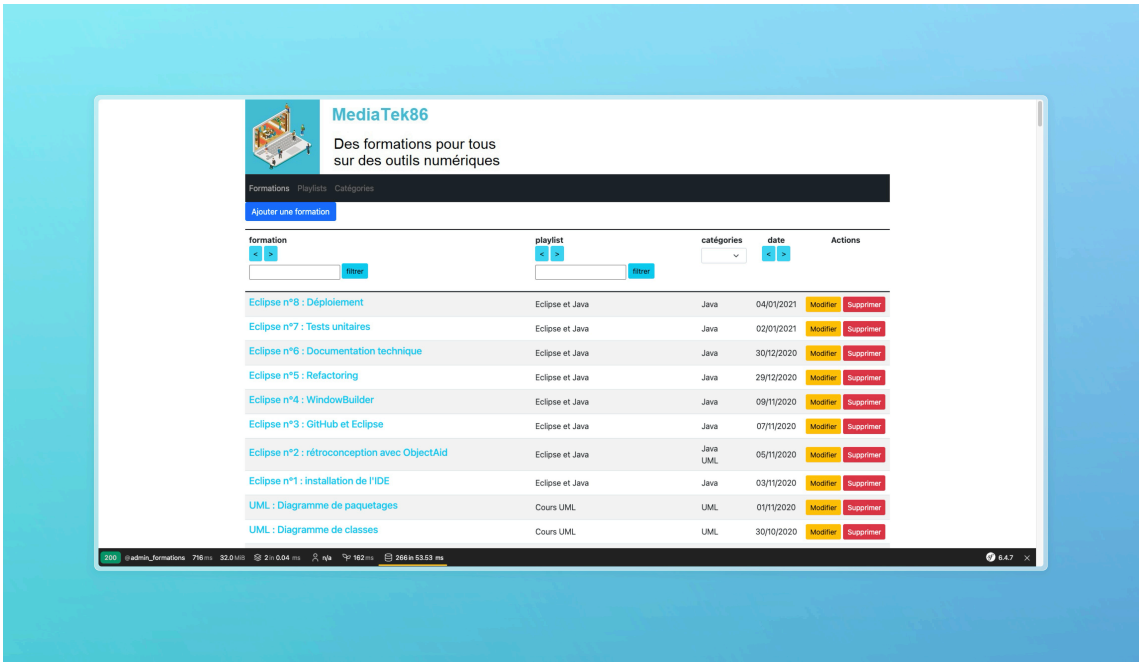
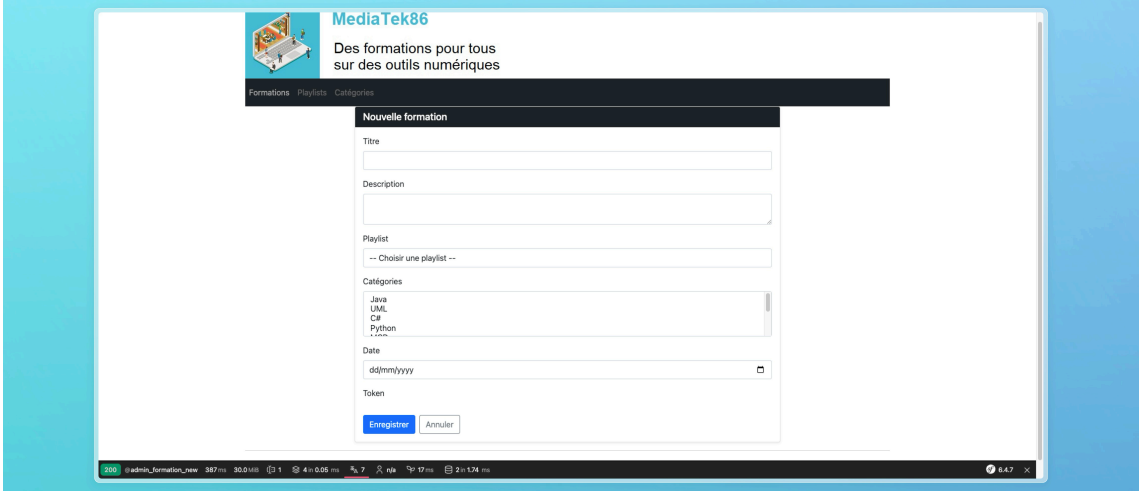


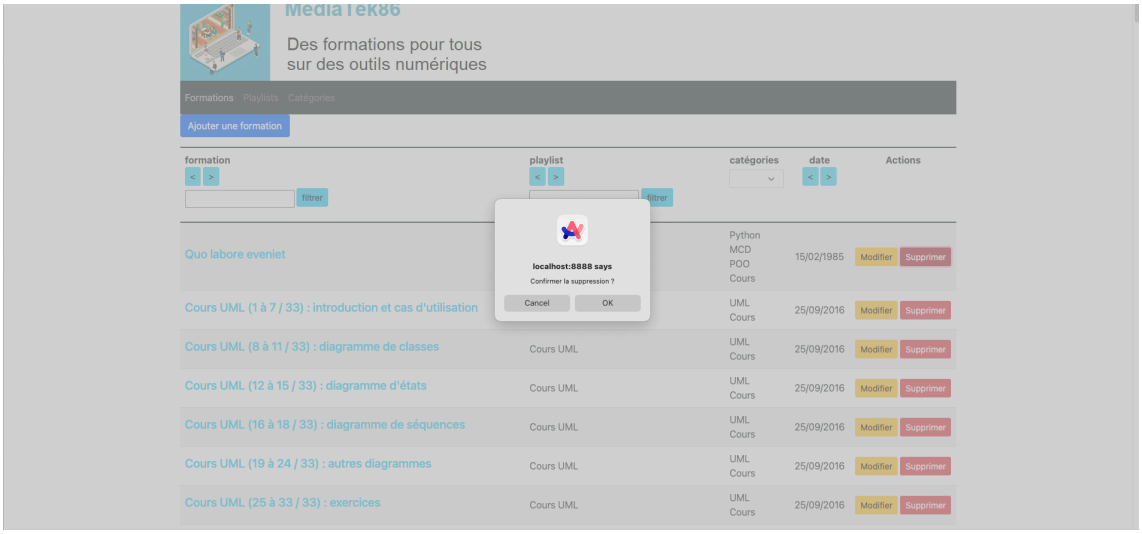
Diagramme de cas d'utilisation.



Maquette liste des formations.



Maquette formulaire.



Maquette confirmation de suppression.

Bilan

Back-office des formations opérationnel avec Doctrine (paramètres liés), CSRF sur les actions destructrices et validation côté formulaire.

Ce que j'ai appris

CRUD admin Symfony, réutilisation de la logique de tri et filtre du front, gestion des relations formation-playlist à la suppression, homogénéisation des formulaires avec le reste du back-office.

Références

- src/Controller/Admin/FormationAdminController.php
- src/Form/FormationType.php
- templates/admin/formation/index.html.twig
- templates/admin/formation/form.html.twig
- templates/baseadmin.html.twig

Branche Git mission2-gerer-formations.

Obstacles

Limiter la duplication avec le front tout en gardant tris et filtres ; suppression avec mise à jour de la relation playlist ; style des listes et formulaires.

3.2 Tâche 2 : Gérer les playlists

Liste, suppression (si aucune formation rattachée) et modification ; tris et filtres comme le front ; ajout avec nom et description (name obligatoire). Formations de la playlist affichées en modification sans ajout ni suppression depuis ce formulaire. Temps estimé 5 h, temps réel 4 h.

Actions réalisées

PlaylistAdminController : index avec filtres et tris (nom, nombre de formations), new, edit avec formations en lecture seule, delete conditionnel avec messages flash. PlaylistType : name obligatoire, description optionnelle. Templates admin/playlist/index.html.twig et form.html.twig.

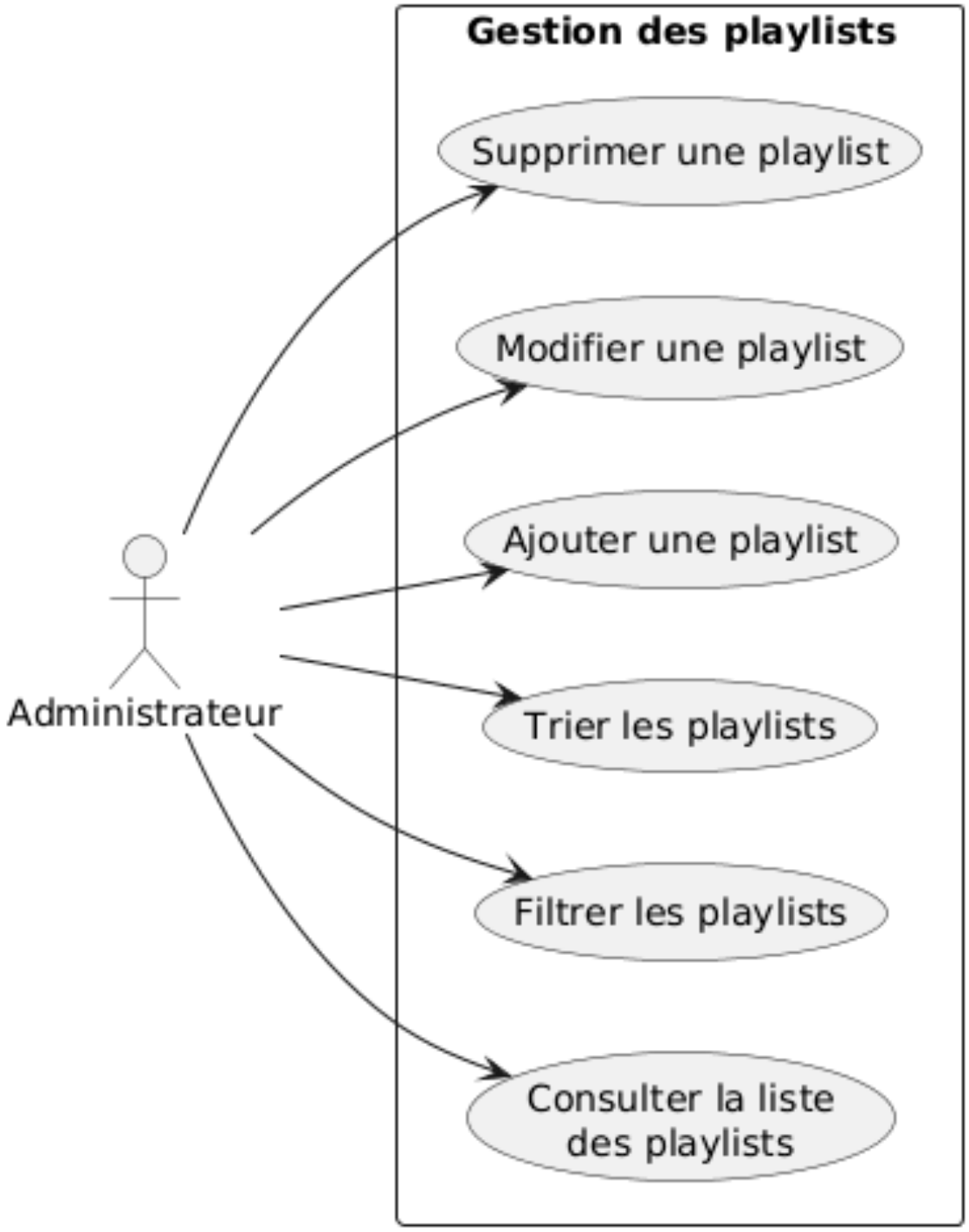
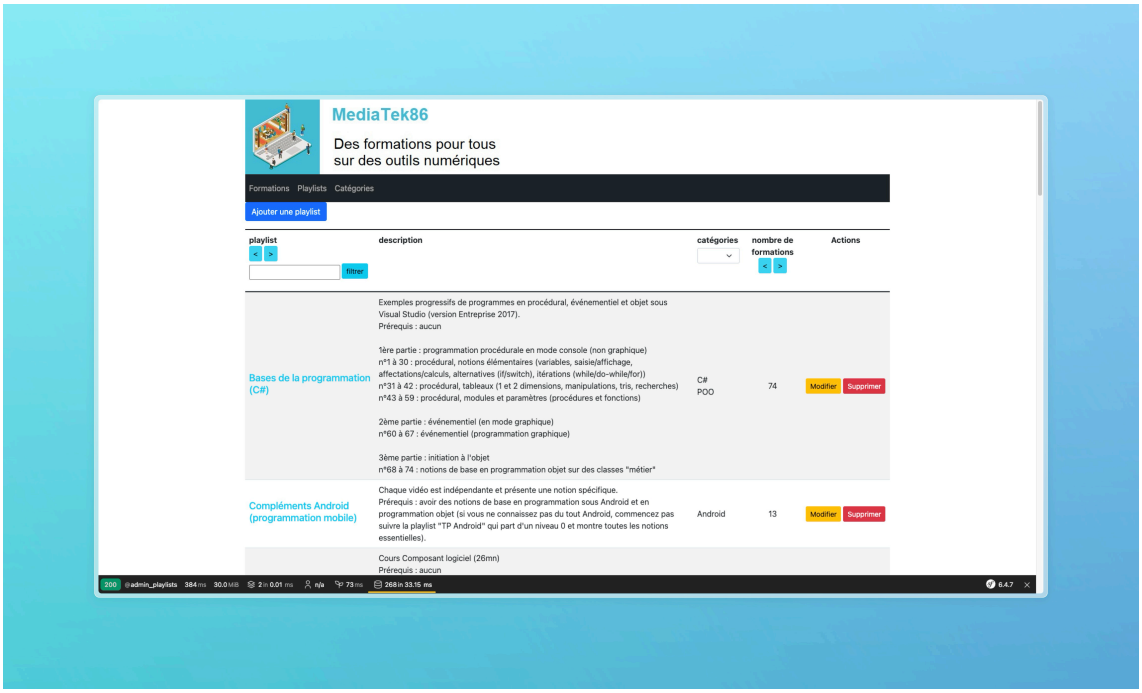
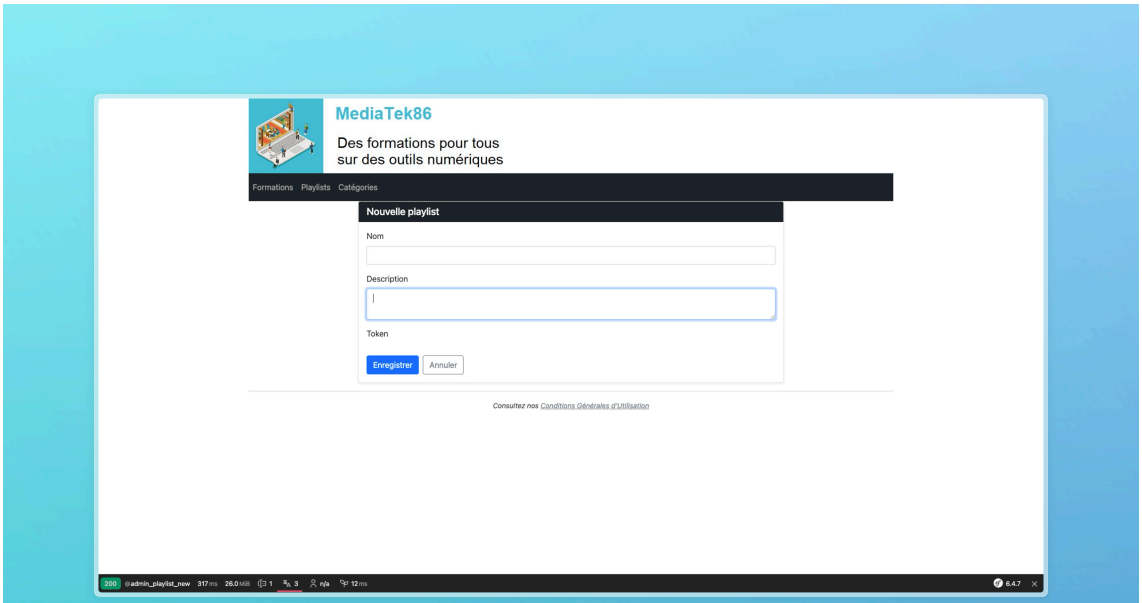


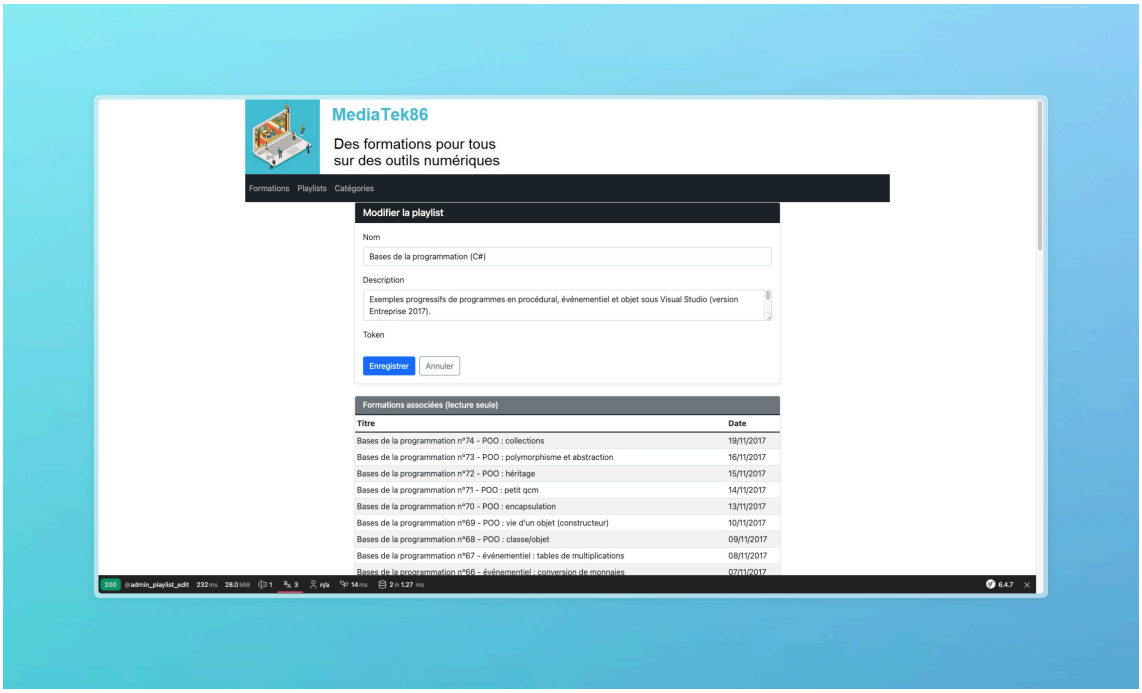
Diagramme de cas d'utilisation.



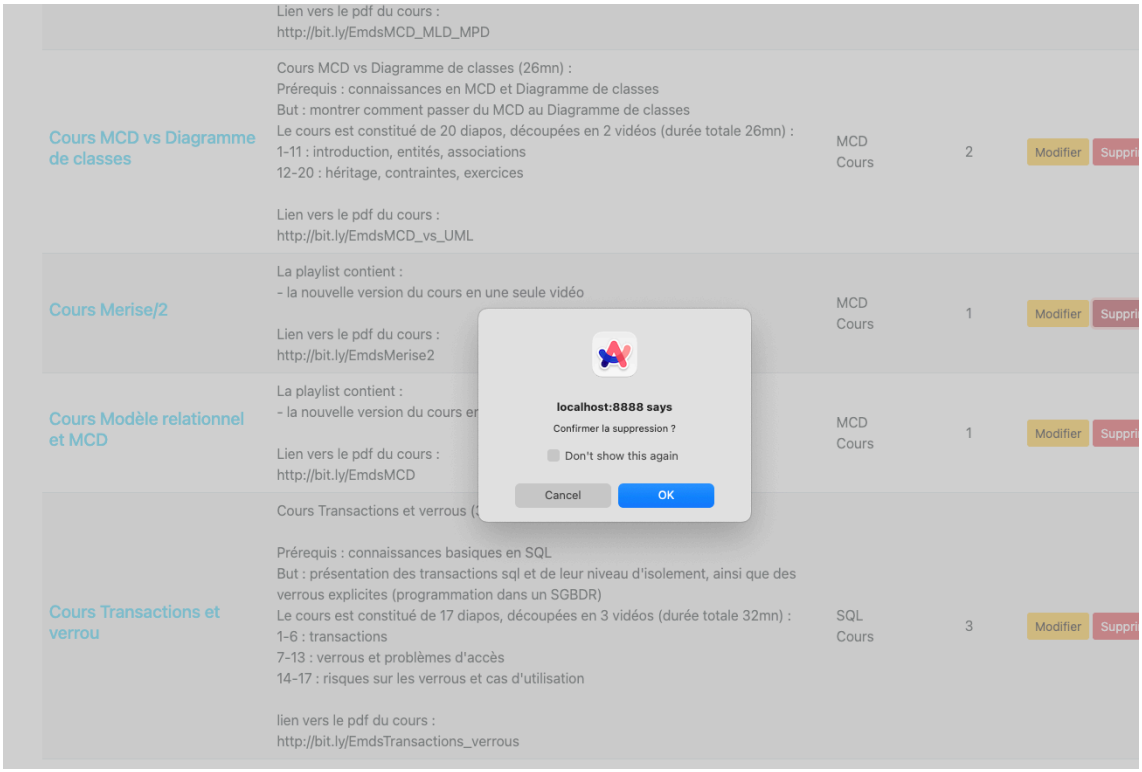
Maquette liste des playlists.



Maquette formulaire d'ajout.



Maquette formulaire de modification.



Maquette confirmation de suppression.

Bilan

Suppression bloquée lorsque des formations sont liées, avec message clair ; filtres et tris alignés sur le front ; respect du cahier des charges pour l'édition des formations depuis le seul formulaire formation.

Ce que j'ai appris

Règles métier sur suppression conditionnelle, messages flash, affichage en lecture seule des collections liées.

Références

- src/Controller/Admin/PlaylistAdminController.php
- src/Form/PlaylistType.php
- templates/admin/playlist/index.html.twig
- templates/admin/playlist/form.html.twig

Branche Git mission2-gerer-playlists.

Obstacles

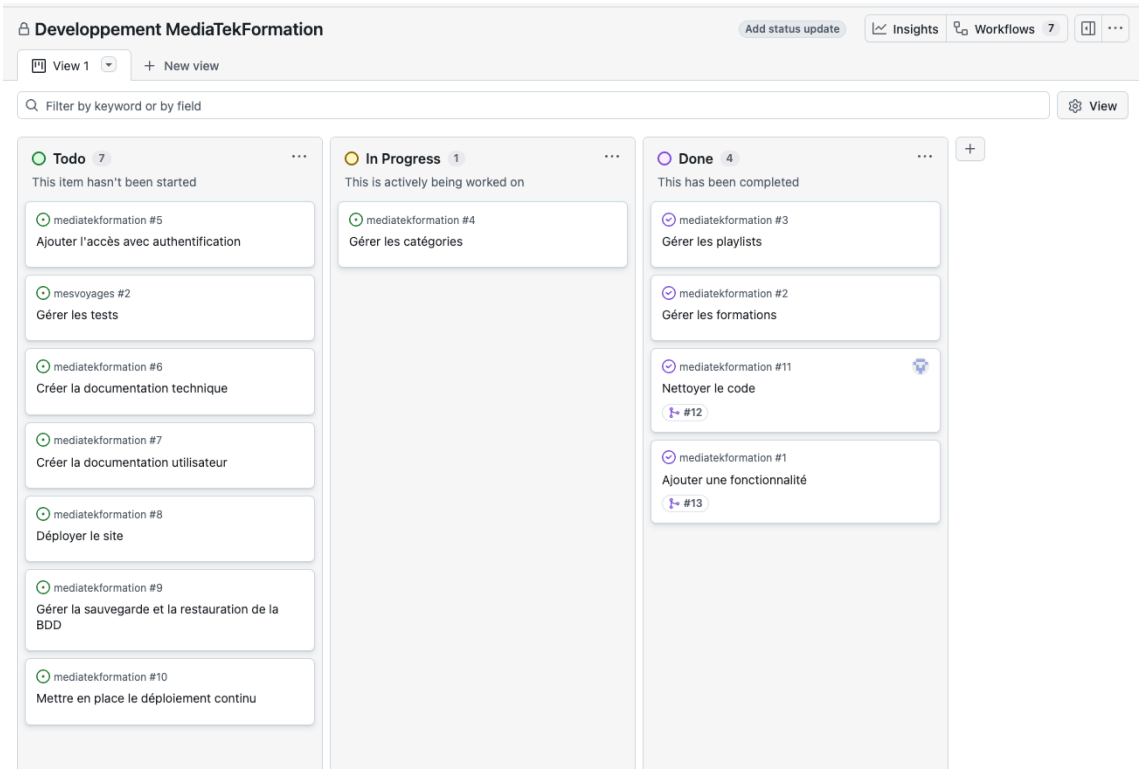
Message utilisateur clair si suppression impossible ; alignement filtres et tris ; séparation stricte entre gestion des formations et des playlists.

3.3 Tâche 3 : Gérer les catégories

Liste et suppression si aucune formation rattachée ; mini formulaire sur la même page pour ajouter une catégorie si le nom n'existe pas déjà. Temps estimé 3 h, temps réel 3 h.

Actions réalisées

CategorieAdminController : liste triée par nom, ajout avec contrôle de non-vide et unicité (sensible à la casse), suppression avec CSRF et contrôle des dépendances. Vue index.html.twig : messages flash, formulaire d'ajout, tableau avec nombre de formations et suppression avec confirmation.



Preuve d'avancement.

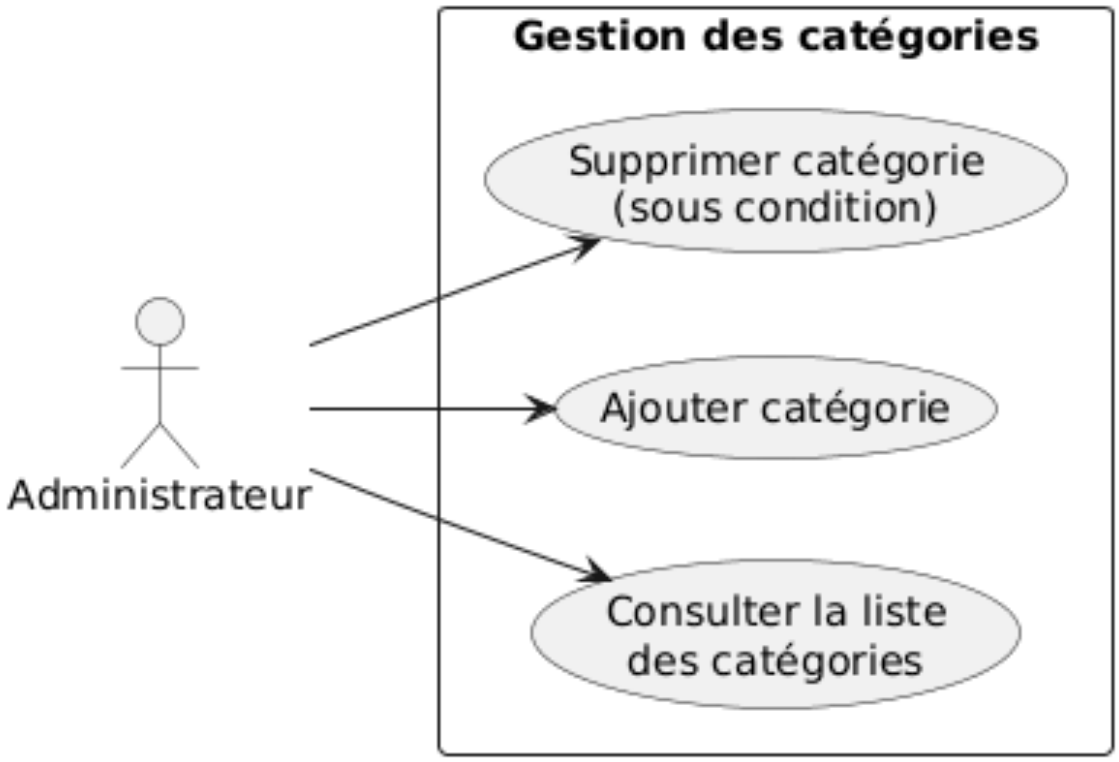
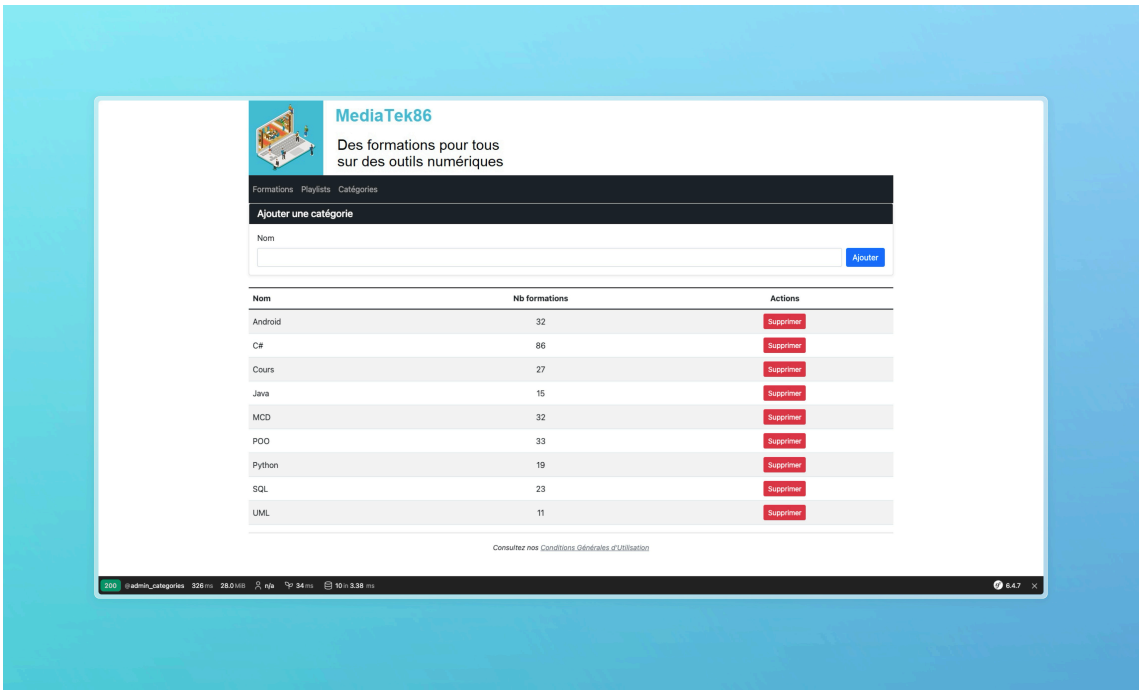
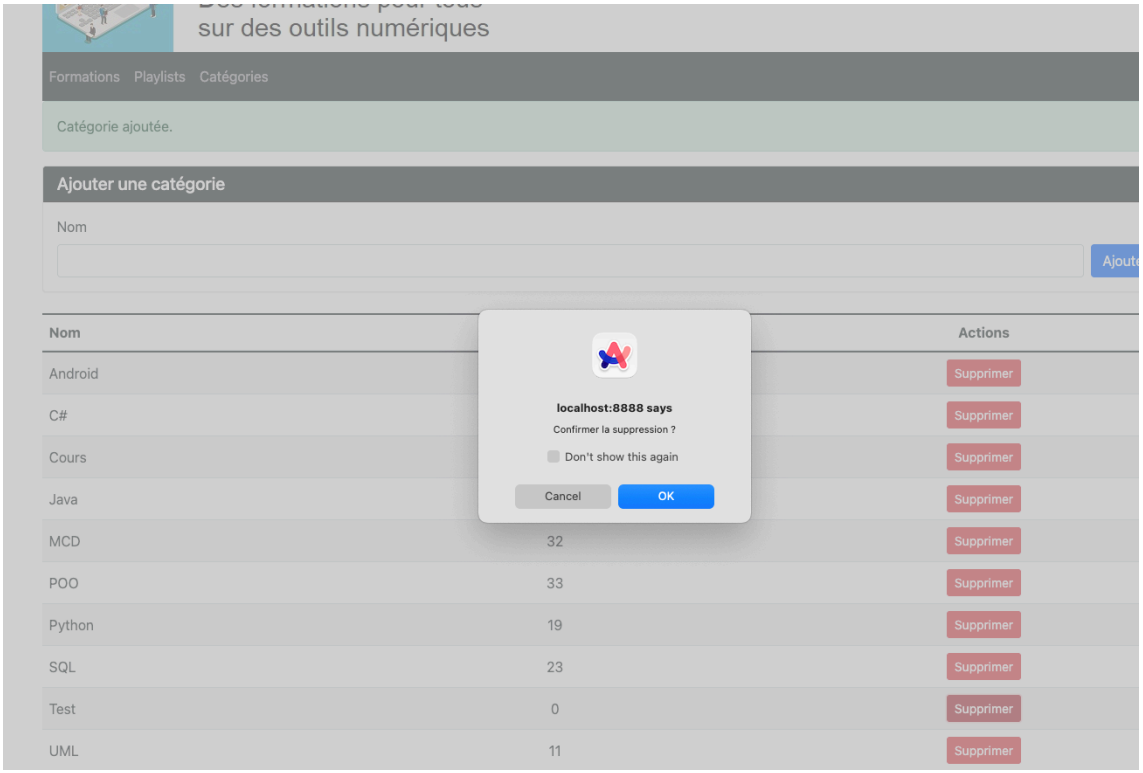


Diagramme de cas d'utilisation.



Maquette liste des catégories.



Maquette confirmation de suppression.

Bilan

Création et suppression depuis une page unique ; unicité des noms vérifiée dans le contrôleur ; suppression impossible tant que la catégorie est utilisée par des formations.

Ce que j'ai appris

Validation dans le contrôleur pour un formulaire minimal sans CategorieType dédié ; explicitation du comportement casse pour les noms.

Références

- src/Controller/Admin/CategorieAdminController.php
- templates/admin/categorie/index.html.twig

Branche Git mission2-gerer-categories.

Obstacles

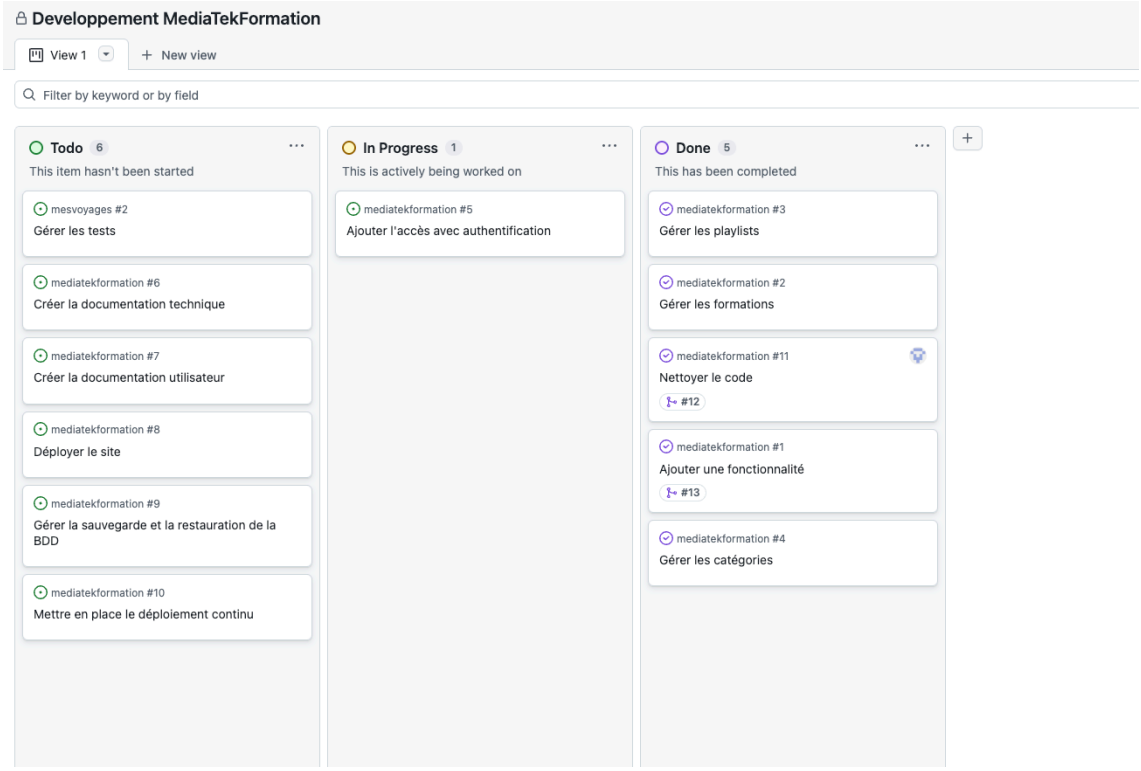
Choix de la validation dans le contrôleur ; unicité sensible à la casse (PHP et php distincts) ; branchement du lien Catégories dans la navigation admin.

3.4 Tâche 4 : Ajouter l'accès avec authentification

Back-office accessible après authentification ; un profil administrateur ; accès par segment admin sur les routes (formations/admin, playlists/admin, categories/admin dans cette réalisation) ; déconnexion sur toutes les pages admin. Temps estimé 4 h, temps réel 4 h 45.

Actions réalisées

Symfony Security : ROLE_ADMIN sur les routes admin. Fichiers ajoutés : User (UserInterface), UserRepository, CreateAdminCommand, LoginAuthenticator, SecurityController, login.html.twig. Modifications : security.yaml (utilisateurs en base, firewall, logout, remember_me, access_control), baseadmin.html.twig (lien Déconnexion), routes admin restructurées en /{ressource}/admin avec noms de routes inchangés.



Preuve d'avancement.

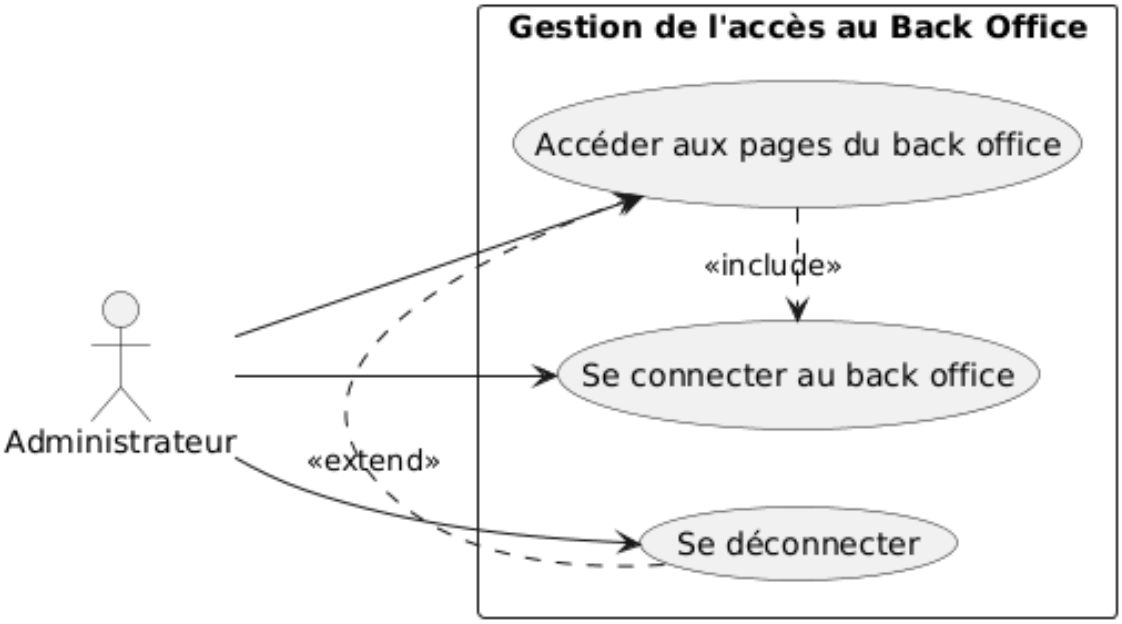
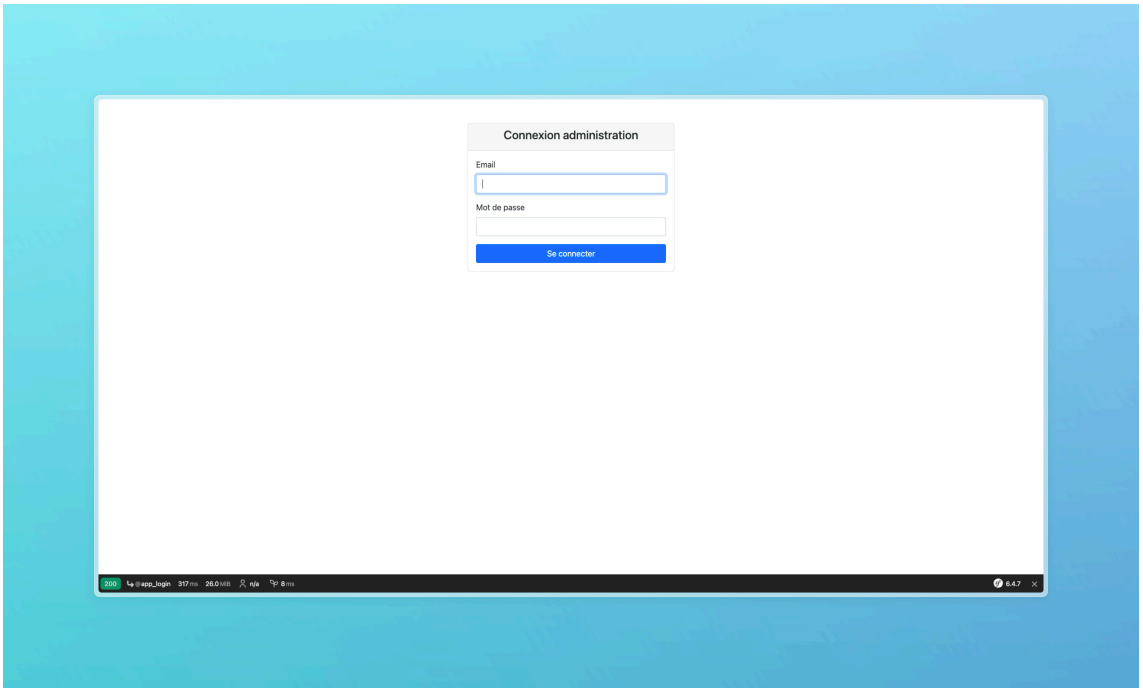


Diagramme de cas d'utilisation.



Maquette page de connexion.

Maquette back-office avec déconnexion.

Bilan

Flux : utilisateur non authentifié redirigé vers /login, succès vers la page demandée ou admin_formations ; sous-routes admin protégées.

Ce que j'ai appris

Bundle Security, entité User, hachage des mots de passe, configuration firewall et access_control, authenticateur personnalisé, commande console pour créer l'administrateur.

Références

Branche Git mission2-auth.

Obstacles

Mise en place de Security, entité User et hachage ; configuration security.yaml ; restructuration des routes admin avec segment cohérent et noms de routes stables ; création du compte via CreateAdminCommand.

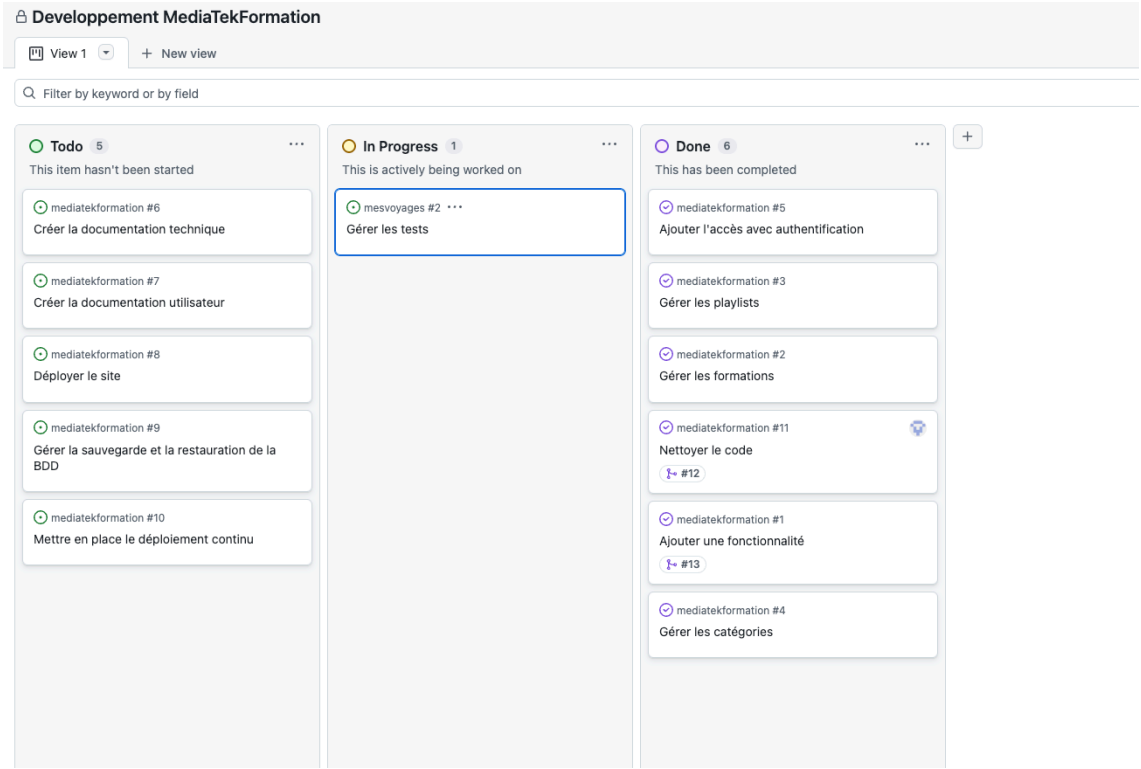
4. Mission 3 : Tester et documenter

4.1 Tâche 1 : Gérer les tests

Mise en œuvre des tests prévus : unitaires sur getPublishedAtString ; intégration sur les règles de validation de date (non postérieure à aujourd'hui) ; intégration sur les méthodes ajoutées aux repositories avec base de test dédiée ; fonctionnels sur l'accueil, listes, tris, filtres et pages de détail ; compatibilité sur Chrome, Firefox et Safari. Plan de tests par catégorie et PV de recette (colonne Faite) à exporter en PDF pour le portfolio. Durée indicative Missions 7 h ; temps noté sur Kanban 3 h hors plan détaillé ; temps réel global (plan, tests, exécutions) 4 h 15.

Actions réalisées

Rédaction et exécution des scénarios décrits dans le bilan (tableaux tests unitaires, intégration, fonctionnels, compatibilité) avec résultats OK attendus. Fichiers principaux : tests/Entity/FormationTest.php, tests/Validator/FormationValidationTest.php, tests/Repository/*.php, tests/Functional/AccueilTest.php, FormationsTest.php, PlaylistsTest.php, phpunit.xml.dist, tests/bootstrap.php.



Kanban, tâche tests en cours.

Bilan

Les cas listés dans le plan de tests (bilan) couvrent format de date, validation LessThanOrEqual, méthodes de dépôts, parcours front et navigateurs ; statuts OK indiqués dans le tableau du bilan.

Ce que j'ai appris

PHPUnit avec base dédiée, tests fonctionnels HTTP sur Symfony, jeux de données pour tris et filtres, recette multi-navigateurs documentée.

Références

Branche Git mission3-test.

Obstacles

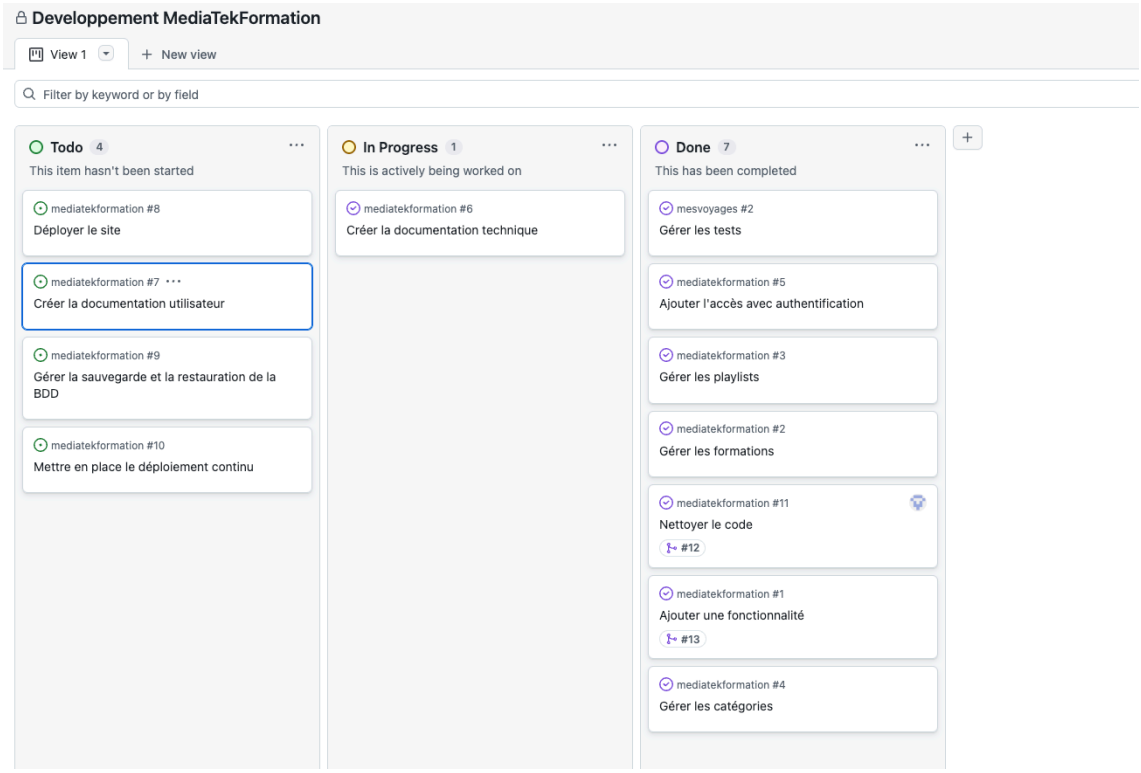
Temps global supérieur à la seule durée notée sur Kanban lorsque la rédaction détaillée du plan est incluse.

4.2 Tâche 2 : Créer la documentation technique

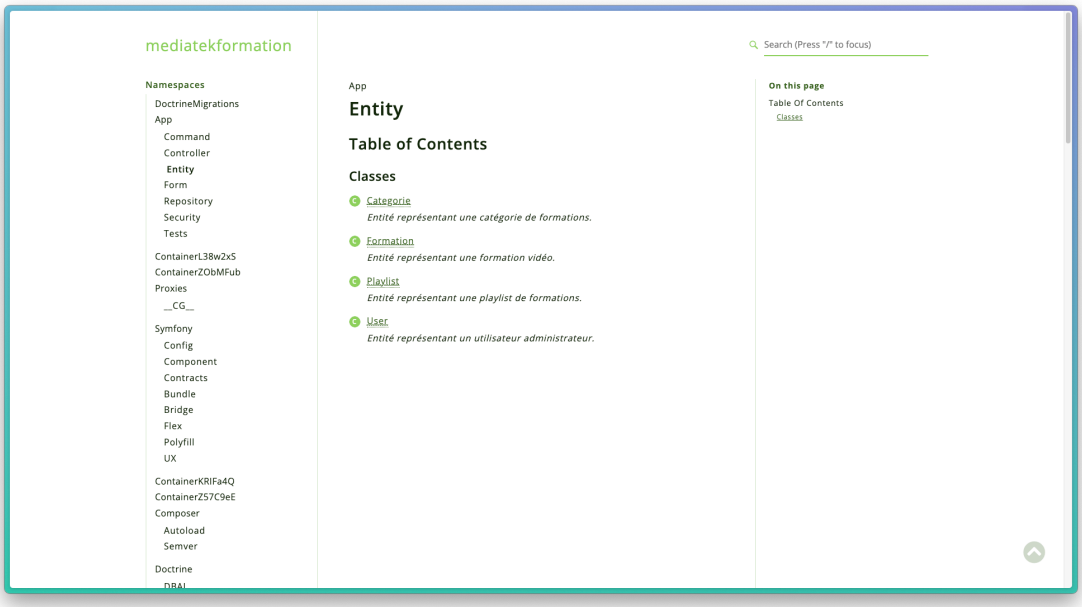
Contrôle des blocs PhpDoc nécessaires à phpDocumentor ; génération de la documentation du site (front et back) en excluant le code généré par Symfony et le dossier vendor, selon la procédure du wiki du dépôt d'origine et NetBeans. Temps estimé 1 h, temps réel 1 h 30.

Actions réalisées

Complétion des commentaires normalisés, configuration des chemins et exclusions, génération et contrôle des pages HTML.



Kanban, documentation en cours.



Première page de la documentation générée.

Bilan

Documentation couvrant contrôleurs, entités, formulaires, sécurité, dépôts, sans bibliothèques tierces. Dossier .phpdoc/ exclu du déploiement FTP.

La documentation est disponible sur https://rabarijoy.github.io/mediatekformation-docs/

Ce que j'ai appris

Chaîne phpDocumentor, exclusions vendor et code généré, intégration au portfolio via pages HTML hébergées.

Références

Branche Git mission3-doc ; article wiki du dépôt d'origine.

Obstacles

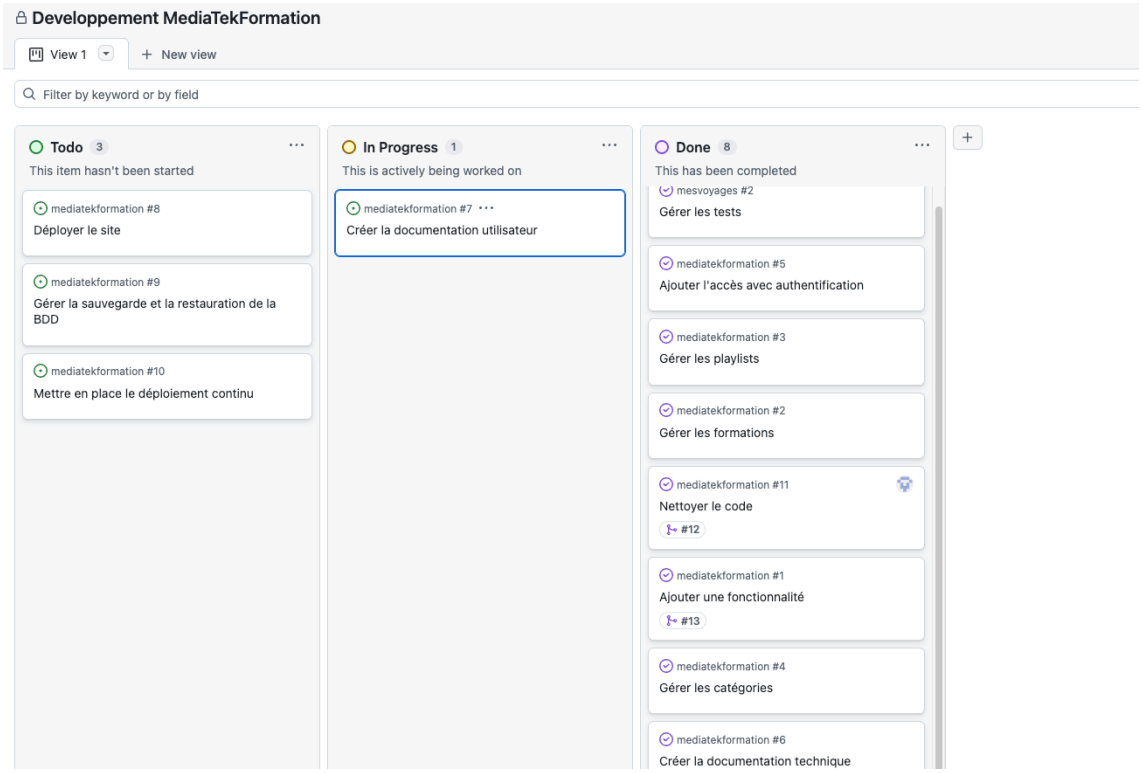
Homogénéité des commentaires ; configuration des chemins sources ; vérification des pages avant livraison.

4.3 Tâche 3 : Créer la documentation utilisateur

Vidéo présentant toutes les fonctionnalités (front et back), durée maximale 5 minutes, avec explications orales. Temps estimé 2 h, temps réel 1 h 50.

Actions réalisées

Enregistrement et montage d'une démonstration couvrant consultation, tris, filtres, administration et authentification dans la limite de temps imposée.



Preuve d'avancement (Kanban ou plateforme de publication).

Bilan

Vidéo conforme à la durée max et aux parcours principaux attendus.

Ce que j'ai appris

Scénarisation courte, enchaînement front puis back, choix d'outil et qualité audio-vidéo.

Obstacles

Condenser l'ensemble des fonctionnalités en moins de 5 minutes tout en restant compréhensible.

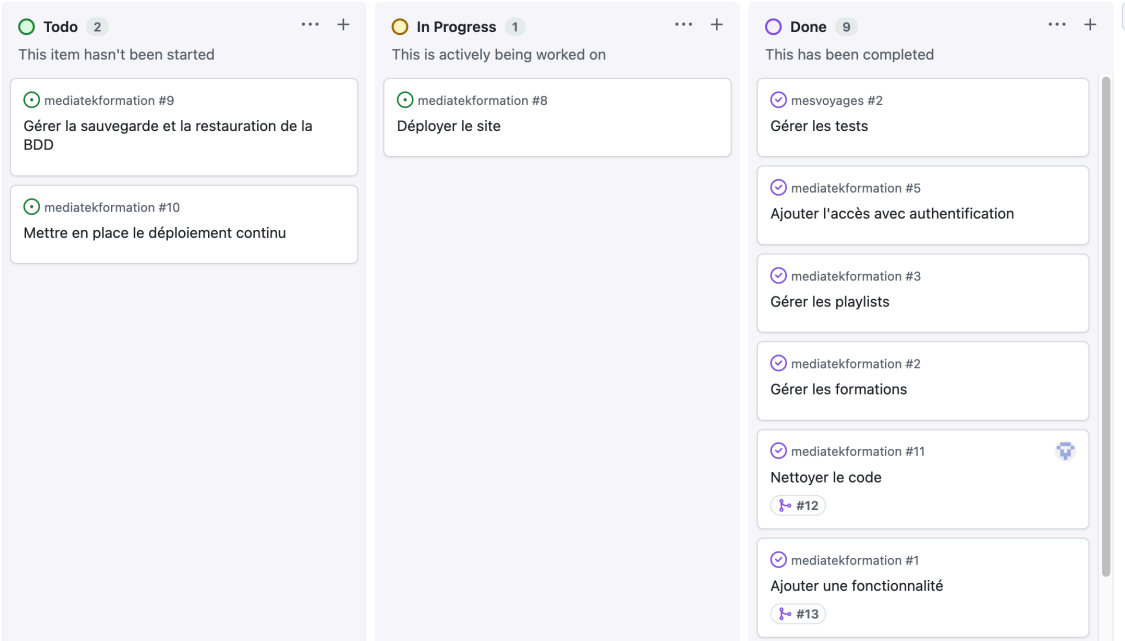
5. Mission 4 : Déployer le site et gérer le déploiement continu

5.1 Tâche 1 : Déployer le site

Déploiement du site, de la base et de la documentation technique chez un hébergeur ; mise à jour des CGU avec l'adresse publique. Temps estimé 2 h (indicatif Missions) ; temps réel 4h (déploiement manuel puis automatisé, itérations sur le workflow CI).

Actions réalisées

- Hébergement mutualisé Infomaniak avec FTP/FTPS ; base MySQL ; paramètres dans .env sur le serveur (non versionnés).
- Import du script SQL (mediatekformation.sql ou export équivalent) ; contrôle encodage et comptes applicatifs.
- Symfony en production : public/index.php selon hébergeur ; redirections HTTP 301 si besoin ; .user.ini ou réglages PHP (erreurs non affichées publiquement).
- Transfert du code et de vendor/ ; exclusion var/cache regénéré, tests/, etc.
- Documentation technique phpDocumentor en ligne (emplacement documenté dans le portfolio).
- CGU : templates/pages/cgu.html.twig avec URL https://mediatek.apiqa.mg et mentions Infomaniak (Genève, Suisse) ; commits feat sur URL et hébergeur.
- Vérification du back-office sécurisé en ligne (authentification, HTTPS).



Preuve d'avancement déploiement.

Bilan

Site et données en ligne sur l'hébergeur choisi, CGU alignées sur l'URL publique, accès admin vérifié.

L'application web est disponible sur https://mediatek.apiqa.mg/

Ce que j'ai appris

Chaîne de mise en production sur mutualisé, variables d'environnement hors dépôt, adaptation documentaire légale (CGU).

Obstacles

Secrets non commités ; compatibilité FTPS (passage à lftp en TLS explicite) ; warmup de cache en CI impossible sans BDD ; risque d'erreur 500 si miroir trop agressif (ajustement des options, exclusion de var/).

5.2 Tâche 2 : Gérer la sauvegarde et la restauration de la BDD

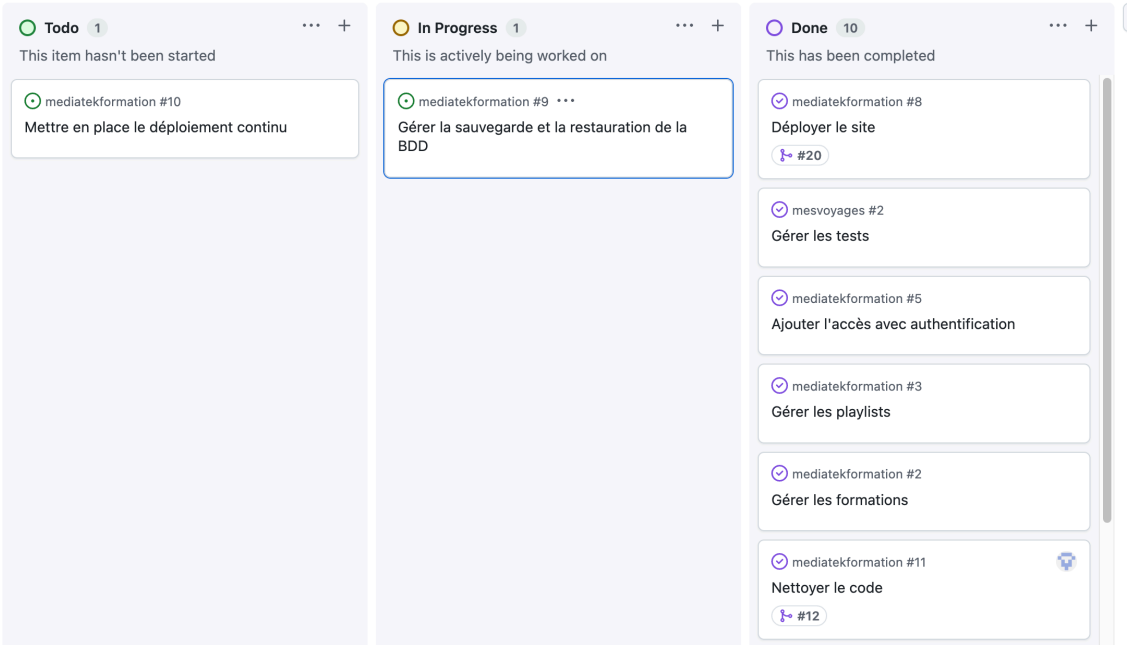
Sauvegarde journalière automatisée (wiki Automatiser la sauvegarde d'une BDD) ; restauration manuelle à partir d'un fichier produit par le script. Temps estimé 1 h ; temps réel 1h10.

Actions réalisées

Script backup_bdd.sh à la racine : mysqldump avec transaction unique, routines, triggers ; fichier daté backups/backup_YYYY-MM-DD.sql ; compression gzip ; rétention 7 jours ; journalisation possible vers backups/backup.log. Paramètres HOST, DB_NAME, DB_USER, DB_PASSWORD par variables d'environnement. Dossier backups/ dans .gitignore.

```
0 2 * * * /chemin/absolu/vers/backup_bdd.sh >>
/chemin/absolu/vers/backups/backup.log 2>&1
```

Restauration : choisir le fichier, gunzip -k, puis mysql -h ... < fichier.sql, avec dump de sécurité préalable et vérifications applicatives.



Preuve crontab ou interface de planification.

Bilan

Procédure de sauvegarde quotidienne et de restauration manuelle documentée dans le bilan.

Ce que j'ai appris

Automatisation shell, mysqldump, rétention et planification cron, bonnes pratiques sur les secrets.

Obstacles

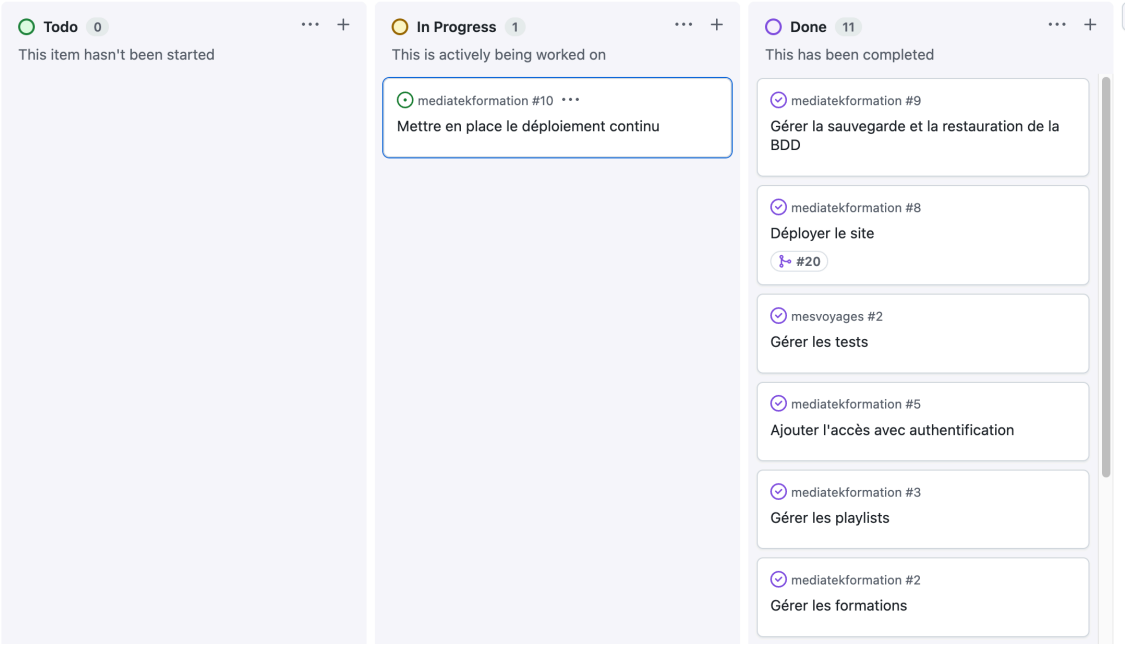
Droits d'exécution (chmod +x) ; stockage du mot de passe via variables d'environnement sur l'hébergeur ; fuseau horaire du cron.

5.3 Tâche 3 : Mettre en place le déploiement continu

Configuration GitHub pour mise à jour du site à chaque push sur master. Temps estimé 1 h ; temps réel : plusieurs itérations (commits fix(ci)).

Actions réalisées

- Workflow .github/workflows/deploy.yml sur push vers master.
- Checkout, PHP 8.1 et extensions, cache Composer, composer install --no-dev --no-scripts --optimize-autoloader.
- Si composer.lock modifié, déploiement complet de vendor/ ; sinon saut du transfert vendor/.
- Déploiement via lftp en FTPS, mirror --reverse, exclusions : .git, .github, vendor (sauf étape dédiée), var, tests, backups, Suivi, .phpdoc, nbproject, .env*, etc.
- Remplacement FTP-Deploy-Action par lftp pour Infomaniak ; suppression du cache warmup en CI ; ajustements --ignore-time et suppression de --delete sur le miroir principal ; correctifs PHP 8.4 / patches Symfony en build.



GitHub Actions après déploiement réussi.

Bilan

Pipeline décrit dans le dépôt pilote les transferts et accélère les déploiements lorsque vendor/ est inchangé.

Ce que j'ai appris

GitHub Actions, lftp et FTPS, stratégie conditionnelle sur composer.lock, exclusions pour préserver .env distant.

Obstacles

Authentification FTPS et certificats ; conflits sur .gitignore (backups/) ; durée des transferts avec vendor/ ; non-écrasement de la configuration secrète.

6. Mission Bilan : Compte rendu, README et portfolio

Durée indicative 6 h (hors détail ligne à ligne dans le fichier source). Cette mission regroupe la production du compte rendu, du README et de la page portfolio.

6.1 Tâche 1 : Compte rendu

Fusionner l'ensemble des bilans (dont ce document) dans un compte rendu unique au format PDF, avec sommaire (missions, tâches, numéros de page et liens internes), page de contexte (cadre, existant, langages et technologies), bilan final (objectifs atteints, difficultés). Le présent fichier est un document Word (.docx) qui reprend ces éléments pour le rendu.

Ce que j'ai appris

Structuration d'un livrable unique à partir des traces de missions et des preuves (captures, Kanban, schémas).

6.2 Tâche 2 : README du dépôt

Compléter le README : lien vers le dépôt d'origine et rappel de la présentation de l'application initiale ; fonctionnalités ajoutées ; installation locale ; test en ligne sans divulguer les identifiants administrateur (fiche officielle).

Ce que j'ai appris

Rédaction technique orientée reproduction du projet et distinction entre informations publiques et données sensibles.

6.3 Tâche 3 : Page portfolio

Page dédiée avec technologies, contexte, liens PDF (compte rendu, contrat, cahier des charges, missions, PV de recette), dépôt GitHub, site en ligne, documentation technique hébergée, intégration de la vidéo, liste des compétences B1/B2 couvertes (sans grilles complètes).

Obstacles (mission bilan)

Coordination des livrables (PDF, vidéo, liens publics vs données sensibles) ; relecture pour cohérence avec les missions officielles.

Bilan final

Résumé global du projet

Le projet MediaTekFormations est une application Symfony couvrant un front office de consultation (formations, playlists, tris, filtres, fiches détail) et un back-office sécurisé pour gérer formations, playlists et catégories. Le travail a démarré par la mise en place de l'environnement (IDE, SonarLint, Git, base locale, Kanban). La mission 1 a nettoyé le code selon SonarLint et ajouté le tri et l'affichage du nombre de formations par playlist avec requêtes DQL adaptées. La mission 2 a livré les écrans admin (CRUD, validations, CSRF, règles de suppression), puis l'authentification Symfony avec rôle administrateur et routes restructurées. La mission 3 a ajouté les tests (unitaires, intégration repository et validation, fonctionnels, compatibilité navigateurs), la documentation phpDocumentor et une vidéo utilisateur. La mission 4 a déployé l'application et la base chez Infomaniak, mis à jour les CGU pour https://mediatek.apiqa.mg, automatisé les sauvegardes MySQL et configuré un déploiement continu GitHub Actions vers le mutualisé (lftp, exclusions, stratégie vendor). La mission bilan prévoit la consolidation documentaire (ce compte rendu, README, page portfolio).

Objectifs atteints (selon les bilans)

- Environnement de développement opérationnel et suivi Kanban des tâches.
- Code front nettoyé (SonarLint) et fonctionnalité de tri et comptage sur les playlists.
- Back-office : formations, playlists, catégories, authentification administrateur.
- Tests et plan de recette documentés ; documentation technique générée ; vidéo utilisateur sous la limite de 5 minutes.
- Mise en ligne, CGU à jour, sauvegardes planifiées, workflow de déploiement sur push master.

Problèmes rencontrés et solutions ou traitements

- Versions PHP et extensions : alignement avec Symfony 6 et configuration locale.
- DATABASE_URL et première configuration : paramétrage progressif de l'environnement.

- Volume de règles SonarLint : traitement fichier par fichier, temps réel supérieur à l'estimation.
- Routes Symfony et DQL agrégées : ordre des routes et gestion des playlists sans formation.
- Duplication logique front/back : factorisation raisonnée des tris et filtres.
- Suppressions conditionnelles playlists/catégories : messages flash et règles métier explicites.
- Security et routes admin : security.yaml, authenticateur, commande de création d'admin, restructuration des chemins.
- CI/CD : passage à lftp/TLS, suppression du warmup sans BDD, ajustement mirror pour éviter les erreurs 500, déploiement vendor conditionnel à composer.lock.
- Sauvegarde : script shell, cron, rétention 7 jours, secrets via variables d'environnement.
- Mission bilan : coordination PDF, vidéo, liens publics et données sensibles.

Pistes d'amélioration réalistes

- Expliciter pour les utilisateurs le comportement d'unicité des noms de catégories sensible à la casse.
- Poursuivre une discipline de commentaires PhpDoc homogènes lors des évolutions du code (point déjà identifié lors de la documentation technique).
- En cas d'évolution du workflow de déploiement, conserver la vigilance sur l'authentification FTPS, les options de miroir lftp et l'exclusion des fichiers de configuration sensibles sur le serveur (obstacles décrits dans les bilans de déploiement).