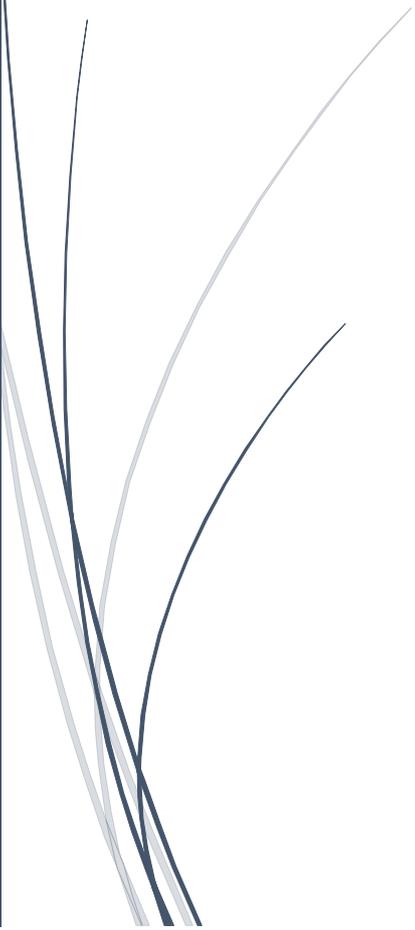


Atelier professionnel n°1 : Mediatekformation

Compte rendu de l'activité



FRANCART Jérémy
BTS SIO DEUXIEME ANNEE

Sommaire :

CONTEXTE DE L'ATELIER	3
MISSION GLOBALE	3
REALISATION DE LA MISSION	4
ÉTAPE 1 : PREPARATION DE L'ENVIRONNEMENT DE TRAVAIL.....	4
ÉTAPE 2 : NETTOYER ET OPTIMISER LE CODE EXISTANT	7
ÉTAPE 3 : CODER LA PARTIE BACK-OFFICE	17
ÉTAPE 4 : TESTER ET DOCUMENTER	25
ÉTAPE 5 : DEPLOYER LE SITE ET GERER LE DEPLOIEMENT CONTINU	31
BILAN	34

Contexte de l'atelier

Cet atelier prend place au sein de l'entreprise InfoTech Services 86 dans laquelle nous travaillons en tant que développeur junior. Cette entreprise a remporté des appels d'offres pour effectuer de multiples interventions pour le réseau de MediaTek86. MediaTek86 est un réseau qui gère les médiathèques de la Vienne, et qui a entre autres pour rôle développer la médiathèque numérique pour l'ensemble des médiathèques du département. C'est dans ce dernier but que MediaTek86 a fait recours aux services de InfoTech Services 86. Parmi ces interventions, nous avons été confiés la création d'un site internet pour exposer les formations en ligne de MediaTek86 aux utilisateurs

Mission globale

Notre mission consiste à développer et déployer un site internet utilisant le framework PHP Symfony et le SGBDR MySQL qui répond aux attentes de MediaTek86 : lister les formations, leurs playlists et leurs catégories pour tous les utilisateurs et gérer ces formations, playlists et catégories depuis un accès demandant une authentification. Il faudra aussi construire une documentation technique et utilisateur.

Un premier développeur a déjà commencé le travail, il a réalisé la majeure partie liée à l'affichage pour les utilisateurs (front-office), nous allons donc continuer ce qu'il a commencé.

Nous gérerons de multiples aspects qui vont être divisés en plusieurs étapes :

Étape 1 : Préparation de l'environnement de travail, de l'IDE (environnement de développement intégré), du site internet et de sa base de données

Étape 2 : Nettoyage et optimisation du code source existant

Étape 3 : Codage de la partie d'administration du site (back-office)

Étape 4 : Gestion des tests et de la documentation du site internet

Étape 5 : Déploiement du site internet en ligne et gestion du déploiement continu

Étape 6 : Construction du compte rendu et intégration du projet au portfolio

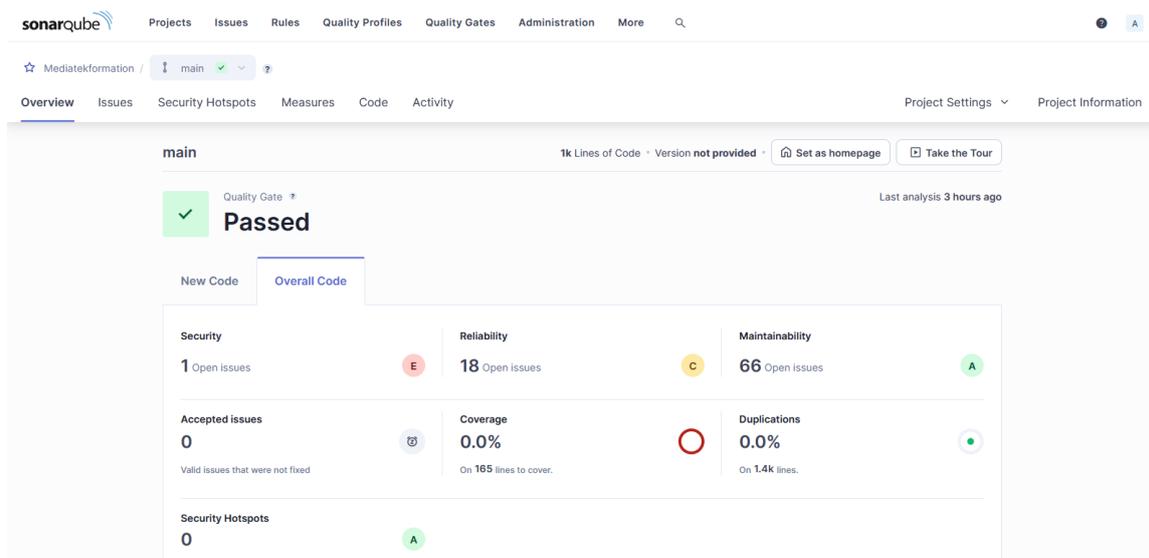
Réalisation de la mission

Étape 1 : Préparation de l'environnement de travail

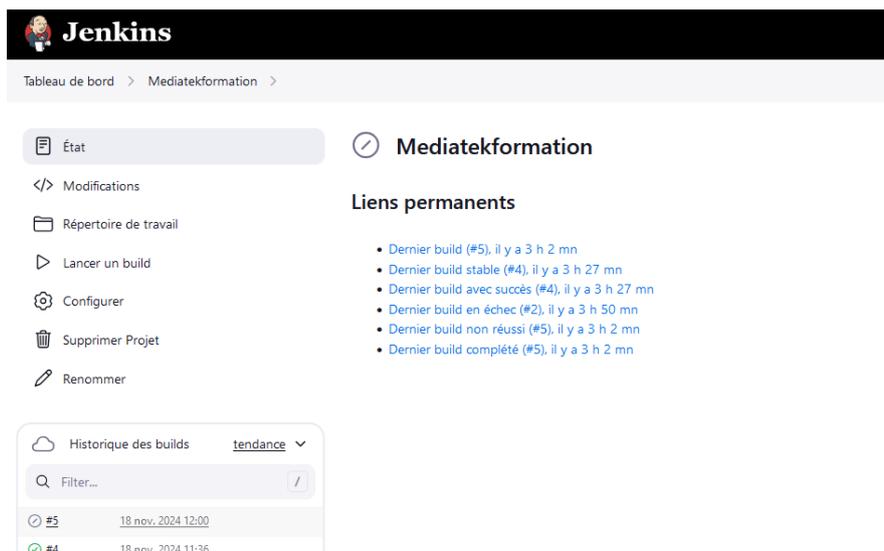
J'ai commencé par créer le dépôt GitHub de l'application d'où j'y ai ajouté les commits du premier développeur.

J'ai donc cloné le code source original que j'ai placé sous WampServer (une solution qui regroupe Windows, Apache, MySQL et PHP pour pouvoir tester le site en local).

Afin de gérer la qualité de code, j'ai créé un projet pour le site internet sur SonarQube, une plateforme de qualité de code.



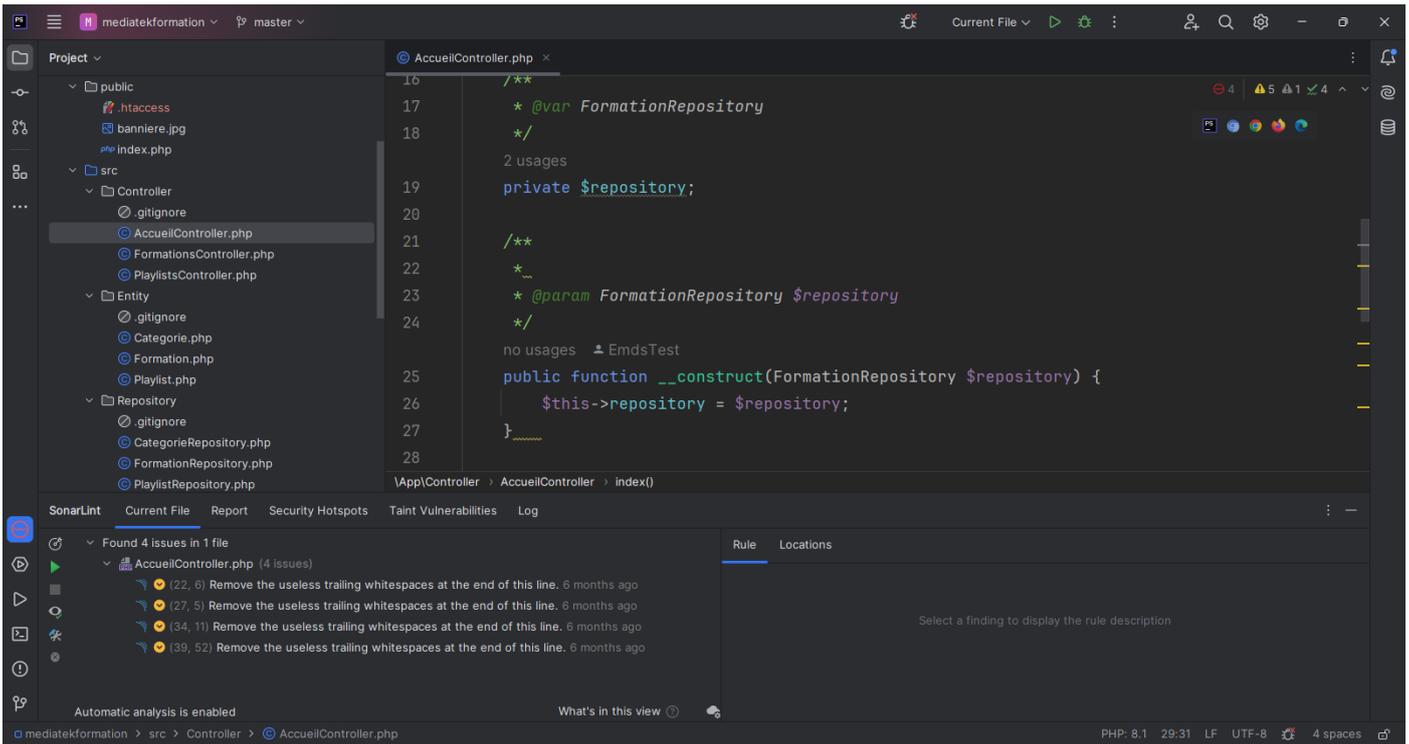
Puis, j'ai utilisé Jenkins afin de pouvoir semi-automatiser la mise à jour de SonarQube lorsque je réalisais un commit.



Il ne me restait plus qu'à exécuter ce script que j'ai créé pour lancer une analyse du code et la mettre à jour sous SonarQube.

```
Invite de commandes
D:\jenkins-cli>.\jenkins-build-Mediatekformation.bat
```

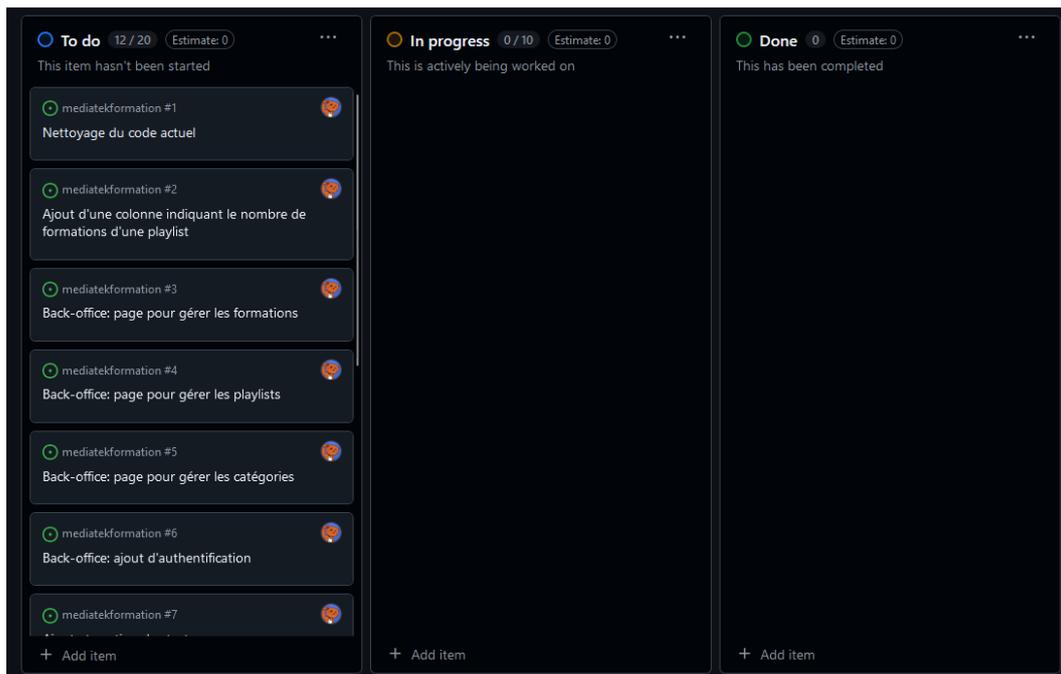
J'ai ensuite configuré PhpStorm, l'environnement de développement intégré choisi pour développer l'application :



Ensuite, j'ai ajouté et configuré la base de données existante du site sous phpMyAdmin.

Pour finaliser la configuration du site, j'ai exécuté `composer install` pour obtenir les bundles PHP nécessaires à l'application et j'y ai ajouté le bundle `symfony/apache-pack` pour avoir le fichier de redirection `.htaccess` et pour réparer la barre de débogage Symfony qui ne s'affichait pas.

Enfin, j'ai mis en place un Kanban GitHub automatisé que j'ai rempli avec les tâches qui nous sommes confiés permettant la gestion du projet de manière organisée.



Back-office: page pour gérer les formations #3

Open 0 comments

Refragg 32 minutes ago

- La page doit contenir la liste des formations avec les tris possibles comme sur la partie front-office
- Pour chaque formation, il doit y avoir un bouton supprimer et un bouton modifier
- Le clic sur le bouton supprimer entrainera une confirmation suivie de la suppression de la formation et sa suppression dans les playlists associées
- Un bouton ajouter doit être présent qui redirige vers un formulaire d'addition d'une formation avec saisies contrôlées
 - Le champ description n'est pas obligatoire
 - Une formation peut n'avoir aucune catégorie
 - La playlist et la / les catégories doivent être sélectionnées dans une liste
 - 1 seule playlist par formation
 - 0 ou plusieurs catégories par formation
 - La date ne doit pas être saisie mais sélectionnée (en utilisant un objet graphique aidant à la sélection d'une date)
 - La date ne doit pas être postérieure à la date du jour
- Le clic sur le bouton modifier entrainera une redirection vers le même formulaire que pour l'addition mais, il devra être prérempli

Refragg self-assigned this 32 minutes ago

github-project-automation (bot) added this to Mediatekformation 32 minutes ago

Assignees: Refragg

Labels

Projects: Mediatekformation
Status: To do +5 more

Milestone

Development – create a branch

Notifications: You're receiving notifications because you're watching this repository.
None All Status

1 participant

Et voici le site pour le moment :

MediaTek86

Des formations pour tous sur des outils numériques

Accueil Formations Playlists

Bienvenue sur le site de MediaTek86 consacré aux formations en ligne

Vous allez pouvoir vous former à différents outils numériques gratuitement et directement en ligne.

Dans la partie [Formations](#), vous trouverez la liste des formations proposées. Vous pourrez faire des recherches et des tris. En cliquant sur la capture, vous accèderez à la présentation plus détaillée de la formation ainsi que la vidéo correspondante.

Vous pouvez aussi retrouver les vidéos regroupées dans des playlists, dans la partie [Playlists](#).

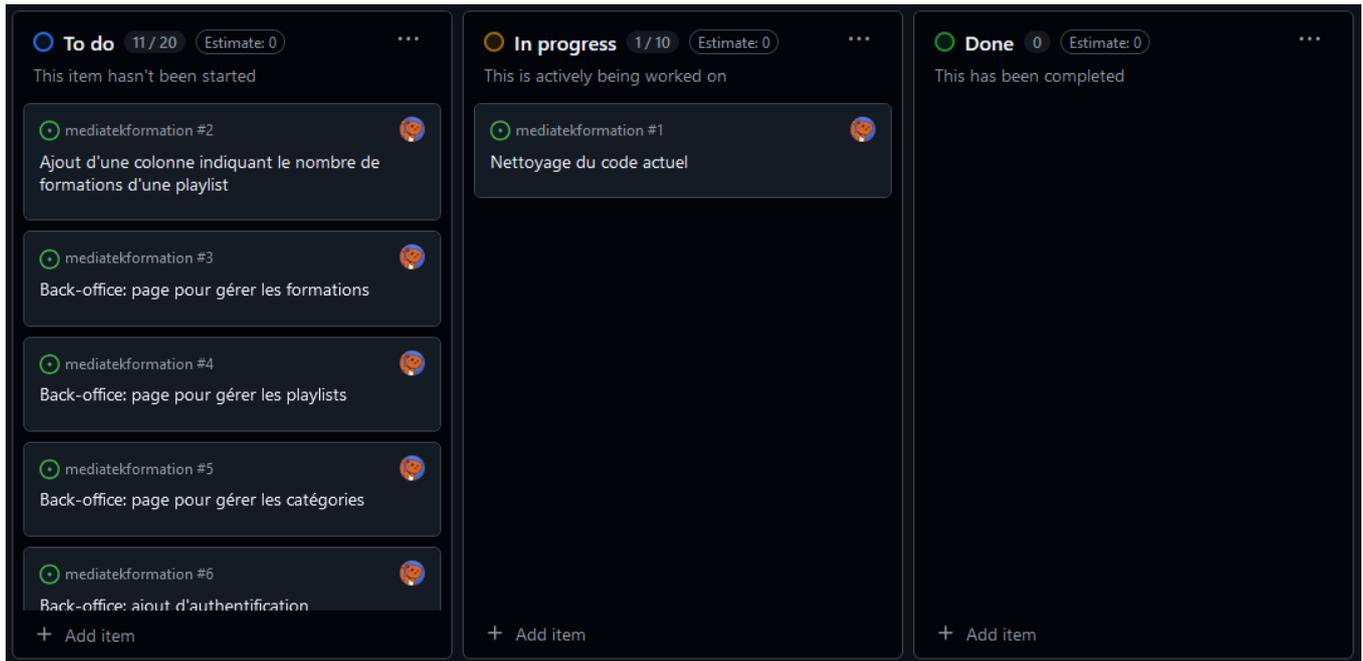
Voici les **deux dernières formations** ajoutées au catalogue :

	04/01/2021 Eclipse n°8 : Déploiement playlist : Eclipse et Java catégories : Java		02/01/2021 Eclipse n°70 : Tests unitaires playlist : Eclipse et Java catégories : Java
---	---	--	--

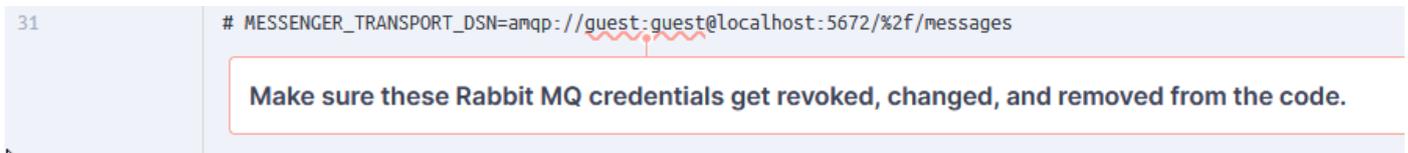
200 @accueil 8399 ms 24.0MiB 2 4 in 5.84 ms n/a 99 ms 4 in 147.89 ms 6.4.7

Étape 2 : Nettoyer et optimiser le code existant

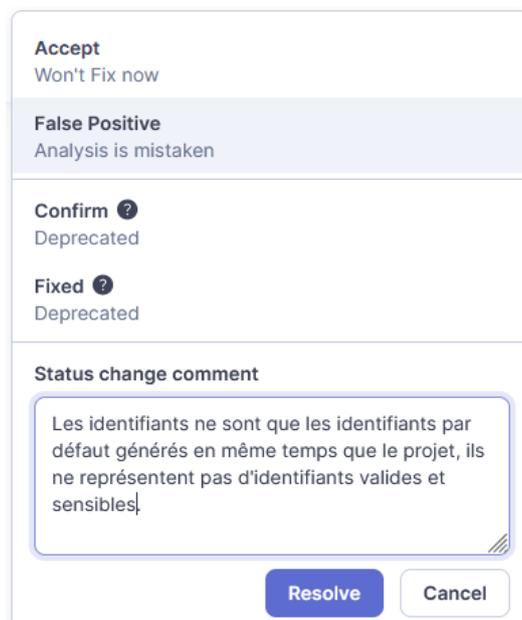
Tache 1 : Nettoyage du code actuel – Temps estimé 2 heures / temps réel 2 heures



Nous allons utiliser SonarQube (et son intégration sous PhpStorm) afin de détecter et régler les problèmes de qualité de code. Nous allons voir les différents problèmes reportés par SonarQube et la (ou les) façon(s) de les régler.



Ce premier problème apparaît dans le fichier .env du site où on y retrouve les secrets, identifiants et configurations du site. Ce problème est en fait un faux-positif pour les raisons évoquées dans le commentaire de résolution :



Bulk Change Select issues ▲ ▼ Navigate to issue ◀ ▶ 6 issues 30min effort

config/preload.php

[Replace "require" with namespace import mechanism through the "use" keyword.](#) Adaptability
Maintainability No tags +
 Open ▾ Not assigned ▾ L4 • 5min effort • 8 months ago • Code Smell • Major

[Replace "require" with "require_once".](#) Consistency
Reliability No tags +
 Open ▾ Not assigned ▾ L4 • 5min effort • 8 months ago • Bug • Minor

public/index.php

[Replace "require_once" with namespace import mechanism through the "use" keyword.](#) Adaptability
Maintainability No tags +
 Open ▾ Not assigned ▾ L5 • 5min effort • 8 months ago • Code Smell • Major

Ces problèmes concernent du code autogénéré par Symfony que l'on doit ignorer dans la recherche de problèmes.

Change 6 issues

Change Status

- Accept
6 issues
- Confirm
6 issues
- False Positive
6 issues
- Fixed
6 issues

Resolution comment

Code généré automatiquement

Formatting Help : *Bold* ``Code`` * Bulleted point

Send Notifications

Apply

Cancel

```

  ▾ AccueilController.php (4 issues)
    (22, 6) Remove the useless trailing whitespaces at the end of this line. 8 months ago
    (27, 5) Remove the useless trailing whitespaces at the end of this line. 8 months ago
    (34, 11) Remove the useless trailing whitespaces at the end of this line. 8 months ago
    (39, 52) Remove the useless trailing whitespaces at the end of this line. 8 months ago
  
```

Beaucoup de problèmes concernent juste des espaces en trop à la fin des lignes, on les retrouve dans plusieurs fichiers de l'application, on enlève juste ces espaces en trop pour régler ces problèmes.

```

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

1 usage
#[ORM\Id]
#[ORM\GeneratedValue]
#[ORM\Column]
private ?int $id =
  
```

SonarLint: Rename this constant "cheminImage" to match the regular expression `^[A-Z][A-Z0-9]*_[A-Z0-9]+*$`.

© Formation

cheminImage = "https://i.ytimg.com/vi/": string

Début de chemin vers les images

Source: .../src/Entity/Formation.php

Pour ce problème, il est indiqué que le nommage utilisé de cette constante ne respecte pas la convention de nommage pour les constantes, il faut renommer cette constante selon la convention de nommage « SCREAMING_SNAKE_CASE », on utilise les fonctionnalités de l'IDE afin de la renommer en plus de ces références.

```

2 usages
private const CHEMIN_IMAGE = "https://i.ytimg.com/vi/";
  
```

```

return $this->render( view: "pages/formations.html.twig", [
    'formations' => $formations,
    'categories' => $categories
]);
  
```

🔴 (22, 36) Define a constant instead of duplicating this literal "pages/formations.html.twig" 4 times. [+3 locations]

Ce problème indique que l'on répète une chaîne en dur. Pour régler ce problème, on peut simplement créer une constante de classe que l'on va réutiliser plusieurs fois (ce type de problème apparaît aussi dans d'autres fichiers, on fait la même chose pour le régler).

```

4 usages
private const PAGE_FORMATIONS = 'pages/formations.html.twig';
  
```

```

return $this->render( view: self::PAGE_FORMATIONS, [
    'formations' => $formations,
    'categories' => $categories
]);
  
```

```
no usages  EmdsTest
function __construct(FormationRepository $formationRepository, CategorieRepository
    $this->for Explicitly mention the visibility of this constructor "__construct".
    $this->cat SonarLint: Show rule description 'php:S1784' Alt+Maj+Entrée More actions... Alt+Entrée Ctrl+. Alt+Maj+F10
}
no usages  Emd
#[Route('/form ): FormationsController
```

Ce constructeur doit pouvoir être appelé par Symfony donc il est de toute évidence public, on le marque comme tel pour régler le problème (le même problème apparaît avec d'autres constructeurs).

```
no usages  EmdsTest *
public function __construct(FormationRepository $for
    $this->formationRepository = $formationRepository
    $this->categorieRepository = $categorieRepository
```

```
100     $categories = new ArrayCollection();
101     foreach($this->formations as $formation){
102         $categoriesFormation = $formation->getCategories();
103         foreach($categoriesFormation as $categorieFormation)
104             if(!$categories->contains($categorieFormation->getName())){
105                 $categories[] = $categorieFormation->getName();
106             }
107     }
108     return $categories;
109 }
110
```

\\App\Entity > Playlist > getCategoriesPlaylist()

SonarLint Current File Report Security Hotspots Taint Vulnerabilities Log

Found 3 issues in 1 file

- Playlist.php (3 issues)
 - (103, 12) Add curly braces around the nested statement(s). 8 months ago
 - (103, 12) Use curly braces or indentation to denote the code conditionally executed by this "foreach". [+1 location] 8 months ago
 - (87, 12) Merge this if statement with the enclosing one. 8 months ago

Il y a 2 problèmes ici : il manque des accolades pour dénoter la boucle « foreach » puis, le code au sein de la boucle « foreach » doit être indenté afin de rendre la lecture du code plus facile.

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

```
83 public function removeFormation(Formation $formation): static
84 {
85     if ($this->formations->removeElement($formation)) {
86         // set the owning side to null (unless already changed)
87         if ($formation->getPlaylist() === $this) {
88             $formation->setPlaylist(playlist: null);
89         }
90     }
91
92     return $this;
93 }
94
```

\App\Entity > Playlist > removeFormation()

SonarLint Current File Report Security Hotspots Taint Vulnerabilities Log

Found 1 issue in 1 file

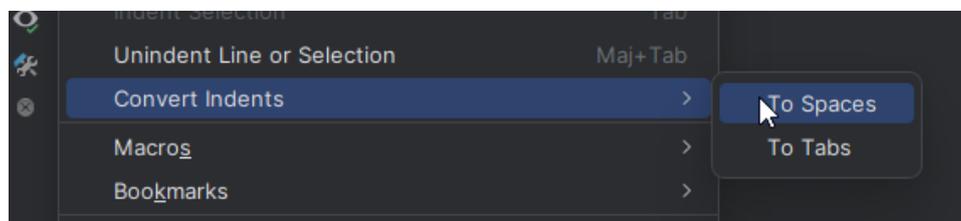
Playlist.php (1 issue)

(87, 12) Merge this if statement with the enclosing one. 8 months ago

Cette méthode a été autogénérée donc on l'accepte comme elle l'est.

```
PlaylistRepository.php (1 issue)
(0, 0) Replace all tab characters in this file by sequences of white-spaces. 8 months ago
```

Ce fichier utilise des caractères de tabulation au lieu de remplacer les tabulations par des espaces. Pour régler ce problème, on peut utiliser l'IDE pour convertir les indentations en espaces.



```

65     #[Route('/playlists/tri/{champ}/{ordre}', name: 'playlists.sort')]
66     public function sort($champ, $ordre): Response{
67         switch($champ){
68             case "name":
69                 $playlists = $this->playlistRepository->findAllOrderByName($ordre);
70                 break;
71         }
72         $categories = $this->categorieRepository->findAll();
73         return $this->render( view: self::PAGE_PLAYLISTS, [
74             'playlists' => $playlists,
75             'categories' => $categories
76         ] );
77     }

```

SonarLint Current File Report Security Hotspots Taint Vulnerabilities Log

Found 3 issues in 1 file

- PlaylistsController.php (3 Issues)
 - (67, 8) Add a "case default" clause to this "switch" statement. 8 months ago

Ce problème nous indique qu'il faut ajouter un cas par défaut dans le switch dans le cas où champ ne correspond pas à "name".

Comme un seul champ est filtré dans ce switch, on peut le changer par un bloc « if, else »

```

if ($champ === 'name') {
    $playlists = $this->playlistRepository->findAllOrderByName($ordre);
} else {
    $playlists = $this->playlistRepository->findAll();
}

```

```


</>

```

Add an "alt" attribute to this image.

Ce problème nous indique que pour des raisons d'accessibilité au site, il est important d'ajouter un attribut « alt » aux balises HTML « img ».

```



```

```
'formations.sort', {champ:'title', ordre:'ASC'}) }}" class="btn btn-info btn-sm active" role="button" aria-pressed="true">
'/formations/tri/{champ}/{ordre}/{table}', {champ:'title', ordre:'DESC'}) }}" class="btn btn-info btn-sm active" role="but
-inline mt-1" method="POST" action="{{ path('/formations/recherche/{champ}/{table}', {champ:'title'}) }}">
orm-group mr-1 mb-2">
pe="text" class="sm" name="recherche"
lue="{{% if valeur|default and not table|default %}}{ valeur }}{% endif %}">
```

Int Current File Report Security Hotspots Taint Vulnerabilities Log

Found 13 issues in 1 file

- formations.html.twig (13 Issues)
 - (8, 20) Add a 'onKeyPress|onKeyDown|onKeyUp' attribute to this <a> tag. 8 months ago
 - (8, 20) Use <button> or <input> instead of the button role to ensure accessibility across all devices. 8 months ago

Ce problème provient du fait que l'on utilise les attributs « role » et « aria-pressed » sur des balises HTML « a ». Ces attributs sont censés être utilisés sur des boutons. On supprime donc ces attributs inutiles et on fait pareil pour tous les boutons.

```
<a href="{{ path('formations.sort', {champ:'title', ordre:'ASC'}) }}" class="btn btn-info btn-sm active"></a>
```

Maintenant que les différents problèmes ont été réglés, on peut effectuer un commit et utiliser le script Jenkins pour mettre à jour le projet sous SonarQube

Le seul problème restant concerne l'utilisation d'une table dans l'accueil pour disposer les 2 dernières formations dans l'accueil, notre responsable a indiqué que ce problème n'était pas à régler. Cependant, il pourrait facilement être réglé en remplaçant la table par un élément « div » disposant les éléments d'une manière particulière.

Bulk Change Select issues ▲ ▼ Navigate to issue ◀ ▶ 1 issues 2min effort

templates/pages/accueil.html.twig

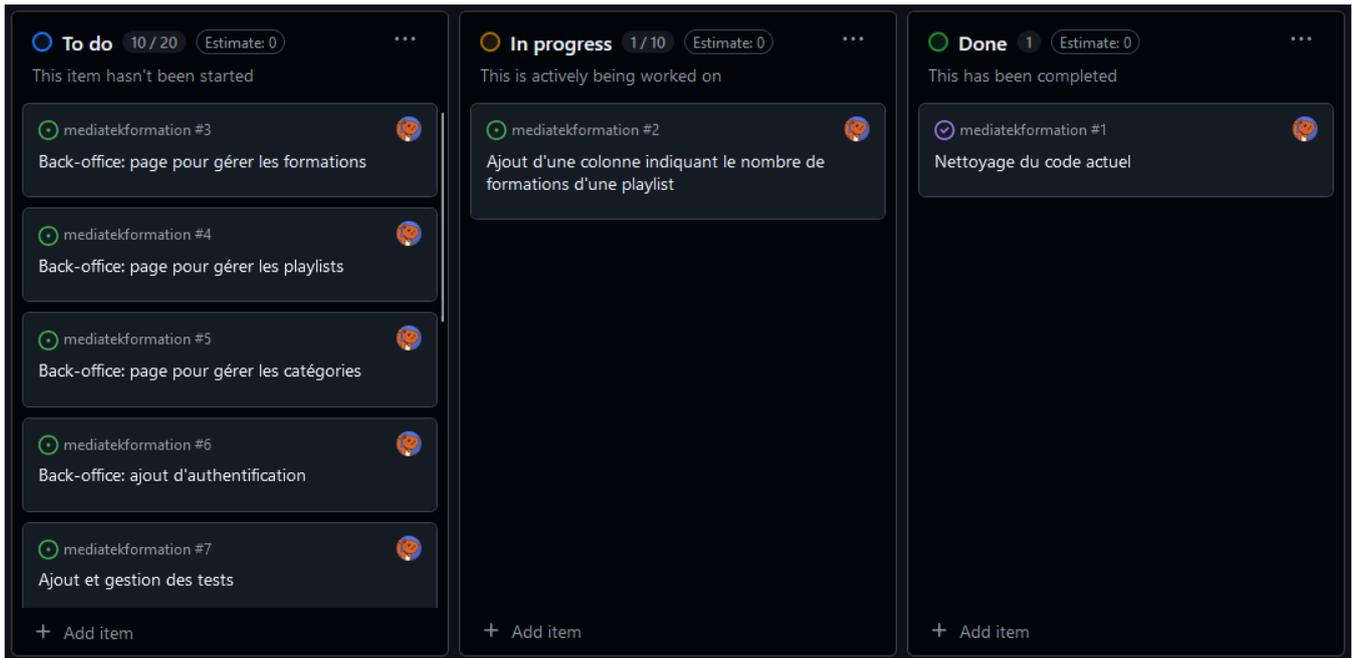
Add "<th>" headers to this "<table>". Intentionality

Reliability accessibility wcag2-a +

○ Open ▼ Not assigned ▼ L16 • 2min effort • 8 months ago • 🐛 Bug • 🗨 Major

1 of 1 shown

Tache 2 : Ajout d'une fonctionnalité – Temps estimé 2 heures / temps réel 2 heures



Dans cette tâche, il nous est demandé d'ajouter une colonne indiquant le nombre de formations par playlist dans la page listant les playlists. Cette colonne doit aussi permettre les tris comme les autres colonnes.

On commence par ajouter une fonction dans l'entité playlist pour obtenir le nombre de formations pour une playlist :

```
/**
 * Retourne le nombre de formations contenues dans la playlist
 * @return int
 */
no usages new *
public function getFormationsCount(): int {
    return count($this->formations);
}
```

Puis, on ajoute dans le contrôleur de tri le champ « count » pour qui va nous permettre de savoir que l'on doit trier sur le nombre de formations

```
no usages EmdsTest +1 *
#[Route('/playlists/tri/{champ}/{ordre}', name: 'playlists.sort')]
public function sort($champ, $ordre): Response{
    if ($champ === 'name') {
        $playlists = $this->playlistRepository->findAllOrderByName($ordre);
    } elseif ($champ === 'count') {
        $playlists = $this->playlistRepository->findAllOrderByCount($ordre);
    } else {
        $playlists = $this->playlistRepository->findAll();
    }
}
```

On crée la méthode `findAllOrderByCount($ordre)` utilisée dans le contrôleur afin de pouvoir récupérer les playlists triés par rapport à un ordre :

```
/**
 * Retourne toutes les playlists triées sur le nombre de formations dans la playlist
 * @param string $ordre
 * @return Playlist[]
 */
1 usage new *
public function findAllOrderByCount($ordre): array{
    return $this->createQueryBuilder( alias: 'p')
        ->leftJoin( join: 'p.formations', alias: 'f')
        ->groupBy( ...groupBy: 'p.id')
        ->orderBy( sort: 'COUNT(f.id)', $ordre)
        ->getQuery()
        ->getResult();
}
```

Ensuite, du côté de l'affichage, on ajoute des boutons pour trier par nombre de formations et on affiche le nombre de formations dans une playlist pour chaque playlist en utilisant la méthode `getFormationsCount` de l'entité.

```
<th class="text-center align-top" scope="col">
    nb formations<br/>
    <a href="{{ path('playlists.sort', {champ:'count', ordre:'ASC'}) }}" class="btn btn-info btn-sm active"></a>
    <a href="{{ path('playlists.sort', {champ:'count', ordre:'DESC'}) }}" class="btn btn-info btn-sm active"></a>
</th>
```

```
<td class="text-center">
    <h5>
        {{ playlists[k].formationsCount }}
    </h5>
</td>
<td class="text-center">
    <a href="{{ path('/playlists/playlist/{id
```

Enfin dans la page de détails d'une playlist, on affiche le nombre de formations de cette playlist.

```
<br /><br />
<strong>nombre de formations :</strong><br />
    {{ playlist.formationsCount }}
```

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

Bases de la programmation (C#)

catégories : C# POO

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"

nombre de formations :

74



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 (calculs)

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)

Étape 3 : Coder la partie back-office

Dans cette étape, on va coder la partie d'administration du site (back-office). Pour ce faire, on va créer des routes sous /admin/ qui redirigeront vers l'administration du site

Tache 1 : Gestion des formations – Temps estimé 5 heures / temps réel 5 heures

Tache 2 : Gestion des playlists – Temps estimé 5 heures / temps réel 5 heures

Tache 3 : Gérer les catégories – Temps estimé 3 heures / temps réel 1 heure 30 minutes

Ces 3 tâches sont regroupées car elles sont similaires. Il faut en fait que ces pages listent les formations / playlists / catégories de la même manière que sur la partie front-office avec les tris et filtres présents. La différence majeure est qu'une colonne actions doit être ajoutée pour pouvoir créer, modifier et supprimer les formations / playlists (pour les catégories on supportera seulement la création et suppression). Le clic sur les boutons ajouter et modifier doivent amener vers un formulaire d'addition / modification permettant d'ajouter / modifier l'entité. Le clic sur le bouton de suppression devra entraîner la suppression de l'entité après confirmation.

J'ai commencé par créer des maquettes pour les différentes pages ainsi que les diagrammes de cas d'utilisation.

Voici l'une des maquettes pour la gestion des formations :

Gestion des formations

Ajout d'une formation

Modification d'une formation

MediaTek86

Des formations pour tous sur des outils numériques

Se déconnecter

Administration du site

Retour à l'accueil Formations Playlists Catégories

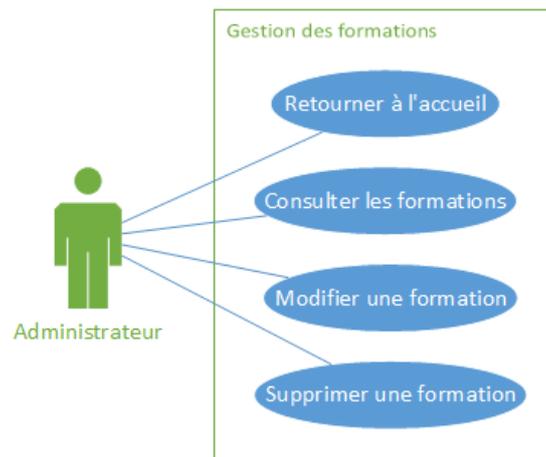
formation playlist catégories date actions

< > < > v < > créer une formation

filtrer filtrer

formation	playlist	catégories	date	actions
Android Studio (complément n°1)	Compléments Android	Android	09/07/2018	Editer Supprimer
Android Studio (complément n°2)	Compléments Android	Android	10/07/2018	Editer Supprimer
Android Studio (complément n°3)	Compléments Android	Android	11/07/2018	Editer Supprimer
Android Studio (complément n°4)	Compléments Android	Android	12/07/2018	Editer Supprimer

Voici un des diagrammes de cas d'utilisation pour la gestion des formations :



Le reste des maquettes et diagrammes est disponible en téléchargement sur la page dédiée au projet sur mon portfolio

Une fois les maquettes et diagrammes créés, j'ai créé les contrôleurs et les templates Twig pour les différents onglets d'administration du site. J'ai pu réutiliser du code déjà présent dans la partie front-office pour pouvoir afficher, trier et filtrer les listes, j'y ai rajouté une dernière colonnes actions contenant les boutons créer, modifier, supprimer. Un clic sur un bouton ajouter / modifier emmène vers un formulaire Symfony créé grâce à la commande `make:form` des outils en ligne de commande de Symfony. J'ai appliqué la template de formulaires Bootstrap 5 afin d'améliorer le visuel des formulaires.

Un exemple de la commande pour créer un formulaire :

```
C:\wamp64\www\mediatekformation>php bin/console make:form

The name of the form class (e.g. AgreeableChefType):
> ExempleType

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

created: src/Form/ExempleType.php

Success!

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

La route d'édition du contrôleur d'administration des formations où l'on affiche le formulaire ou on gère la modification :

```

128  * @param int $id L'identifiant de la formation à modifier
129  * @param Request $request La requête actuelle (injecté par Symfony)
130  * @return Response
131  */
no usages ▲ Refrags
132  #[Route('/admin/formations/edit/{id}', name: 'admin.formations.edit')]
133  public function edit(int $id, Request $request): Response {
134      $formation = $this->formationRepository->find($id);
135      $formFormations = $this->createForm( type: FormationsType::class, $formation);
136
137      $formFormations->handleRequest($request);
138      if ($formFormations->isSubmitted() && $formFormations->isValid()) {
139          $this->formationRepository->addOrEdit($formation);
140          return $this->redirectToRoute( route: 'admin.formations');
141      }
142
143      return $this->render( view: 'pages/admin/admin.formations.edit.html.twig', [
144          'formation' => $formation,
145          'formformations' => $formFormations->createView()
146      ]);
147  }

```

L'utilisation de la template Bootstrap 5 dans les formulaires (fichier de configuration de twig):

```

1 twig:
2 file_name_pattern: '*.twig'
3 + form_themes: ['bootstrap_5_layout.html.twig']

```

L'objet « formformations » envoyé depuis le contrôleur peut ensuite être utilisé dans la template twig afin de personnaliser comment le formulaire va s'afficher

```

1  {{ form_start(formformations) }}
2  <div>
3      <div class="row">
4          <div class="col">
5              {{ form_row(formformations.title) }}
6          </div>
7          <div class="col">
8              {{ form_row(formformations.description) }}
9          </div>
10         <div class="col">
11             {{ form_row(formformations.publishedAt) }}
12         </div>
13     </div>
14     <div class="row" style="...">
15         <div class="align-content-center justify-content-center" style="...">
16             {{ form_row(formformations.submit) }}
17         </div>
18     </div>
19     {{ form_end(formformations) }}

```

Enfin, on inclus cette template de formulaire dans la template d'édition / addition :

```

1  {% extends 'baseadmin.html.twig' %}
2
3  {% block body %}
4      <h2>Détails formation :</h2>
5      {% include 'pages/admin/admin.formations.form.html.twig' %}
6  {% endblock %}
7
8  {% block title %}
9      Édition d'une formation
10  {% endblock %}

```

Pour la validation du formulaire, il est possible d'ajouter des contraintes directement dans l'entité et les erreurs vont être gérées automatiquement par Symfony :

```

4 usages
#[ORM\Column(type: Types::DATETIME_MUTABLE, nullable: true)]
#[Assert\LessThanOrEqual(['value' => 'now'])]
private ?\DateTimeInterface $publishedAt = null;

```

(cette contrainte valide que la date et heure n'est pas postérieure à la date et heure actuelle)

Aussi, afin de pouvoir ajouter, supprimer et modifier les entités, certaines méthodes ont dû être rajoutées dans le repository des entités.

Au final, voici les fichiers modifiés lors de l'ajout de l'administration du site et voici les résultats :

Showing 20 changed files with 914 additions and 3 deletions.		Split	Unified
config/packages/twig.yaml	+1 -0	■	■
src/Controller/admin/AdminCategoriesController.php	+79 -0	■	■
src/Controller/admin/AdminFormationsController.php	+117 -0	■	■
src/Controller/admin/AdminPlaylistsController.php	+139 -0	■	■
src/Entity/Formation.php	+2 -0	■	■
src/Form/FormationsType.php	+64 -0	■	■
src/Form/PlaylistsType.php	+38 -0	■	■
src/Repository/CategorieRepository.php	+47 -1	■	■
src/Repository/FormationRepository.php	+6 -1	■	■
src/Repository/PlaylistRepository.php	+6 -1	■	■
templates/baseadmin.html.twig	+45 -0	■	■
templates/pages/admin/admin.categories.html.twig	+47 -0	■	■
templates/pages/admin/admin.formations.add.html.twig	+10 -0	■	■
templates/pages/admin/admin.formations.edit.html.twig	+10 -0	■	■
templates/pages/admin/admin.formations.form.html.twig	+36 -0	■	■
templates/pages/admin/admin.formations.html.twig	+98 -0	■	■
templates/pages/admin/admin.playlists.add.html.twig	+10 -0	■	■
templates/pages/admin/admin.playlists.edit.html.twig	+54 -0	■	■
templates/pages/admin/admin.playlists.form.html.twig	+22 -0	■	■
templates/pages/admin/admin.playlists.html.twig	+83 -0	■	■



MediaTek86

Des formations pour tous sur des outils numériques

Administration du site

[Retour à l'accueil](#) [Formations](#) [Playlists](#) [Catégories](#)

formation	playlist	catégories	date	actions
<input type="text"/> <input type="button" value="filtrer"/>	<input type="text"/> <input type="button" value="filtrer"/>	<input type="text"/>	<input type="text"/> <input type="button" value="filtrer"/>	<input type="button" value="créer une formation"/>
Eclipse n°8 : Déploiement	Eclipse et Java	Java	04/01/2021	<input type="button" value="Editer"/> <input type="button" value="Supprimer"/>
Eclipse n°70 : Tests unitaires	Eclipse et Java	Java	02/01/2021	<input type="button" value="Editer"/> <input type="button" value="Supprimer"/>
Eclipse n°6 : Documentation technique	Eclipse et Java	Java	30/12/2020	<input type="button" value="Editer"/> <input type="button" value="Supprimer"/>

200 @admin.formations 808 ms 10.0 MIB admin 419 ms 266 in 172.90 ms 6.4.7 X



MediaTek86

Des formations pour tous sur des outils numériques

Administration du site

[Retour à l'accueil](#) [Formations](#) [Playlists](#) [Catégories](#)

Détails formation :

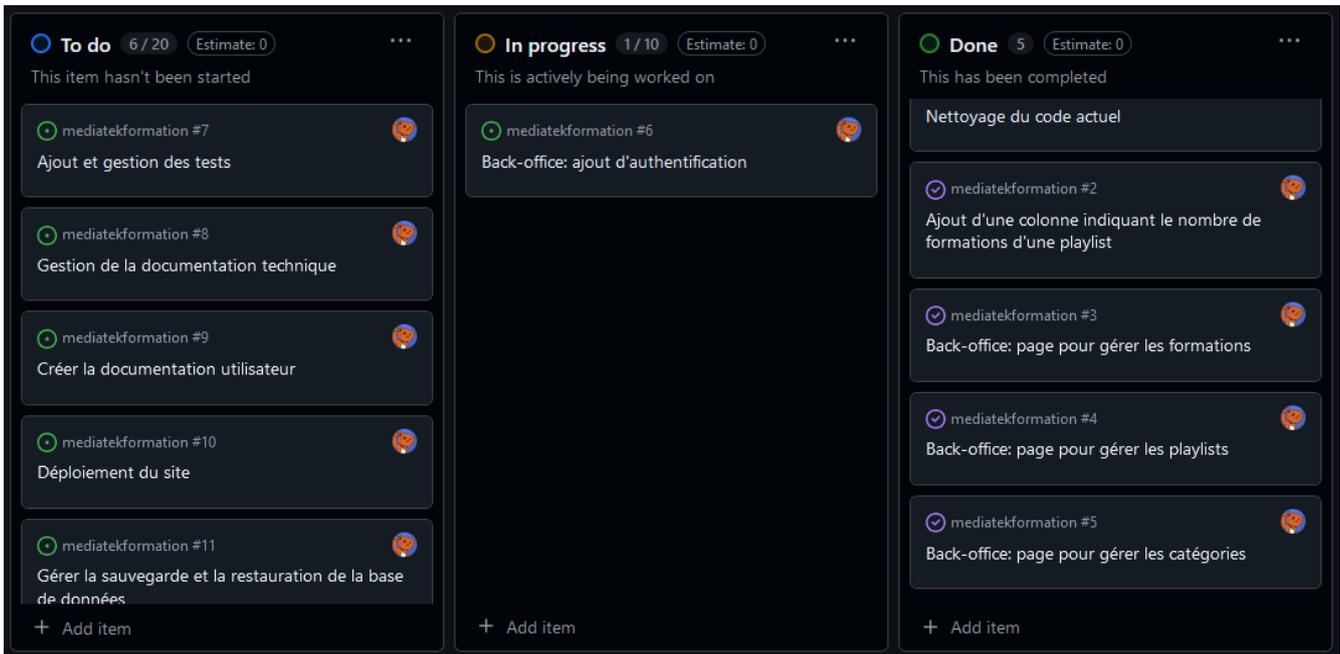
Titre <input type="text" value="Eclipse n°8 : Déploiement"/>	Description <input type="text" value="Exécution de l'application en dehors de l'IDE, en invite de commande. Création d'un fichier jar pour le déploiement de l'application."/>	Date de publication <input type="text" value="04 / 01 / 2021 17 : 00"/>
Identifiant de la vidéo <input type="text" value="Z4yTTXka958"/>	Catégories <ul style="list-style-type: none">JavaUMLC#Python	Playlist <input type="text" value="Eclipse et Java"/>

200 @admin.formations.edit 532 ms 8.0 MIB 1 4 in 0.09 ms 7 A 146 ms 6 in 4.63 ms 6.4.7 X



Tache 4 : Ajouter l'accès avec authentification – Temps estimé 4 heures / temps réel 1 heure

A travers cette tâche, il nous est demandé de gérer l'authentification pour la partie back-office afin que tout le monde ne puisse pas accéder au panel administratif.



Pour commencer, la maquette et le diagramme de cas d'utilisation ont été créés (disponible en téléchargement sur la page dédiée au projet dans le portfolio)

Maintenant nous allons créer une entité « User » avec la commande Symfony `make:user` :

```
C:\wamp64\www\mediatekformation>php bin/console make:user

The name of the security user class (e.g. User) [User]:
>

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
>

Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:
> username

Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed by
another system. (yes/no) [yes]:
>

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml

Success!

Next Steps:
- Review your new App\Entity\User class.
- Use make:entity to add more fields to your User entity and then run make:migration.
- Create a way to authenticate! See https://symfony.com/doc/current/security.html
```

Une fois l'entité l'utilisateur créé, on utilise la commande `make:migration` pour générer un fichier de migration de base de données que l'on va ensuite exécuter avec `doctrine:migrations:migrate`.

Ensuite, on va initialiser un utilisateur admin dans la base de données. Pour ce faire, on va utiliser le bundle orm-fixtures, ce bundle va nous permettre d'ajouter un utilisateur utilisant les interfaces de programmation de Symfony.

On peut maintenant définir la méthode load dans un fichier Fixtures spécifique pour initialiser un utilisateur :

```
no usages  ↳ Refragg
public function load(ObjectManager $manager): void
{
    $user = new User();
    $user->setUsername( username: 'admin');
    $password = 'admin';
    $hashedPassword = $this->passwordHasher->hashPassword($user, $password);
    $user->setPassword($hashedPassword);
    $user->setRoles(['ROLE_ADMIN']);
    $manager->persist($user);
    $manager->flush();
}
```

On exécute cette Fixture avec `doctrine:fixtures:load`. Notre base de données possède maintenant un compte administrateur.

Maintenant, on va paramétrer l'authentification dans le fichier de configuration security.yaml pour rediriger vers la route d'un contrôleur.

```
main:
  lazy: true
  provider: app_user_provider
  form_login:
    login_path: app_login
    check_path: app_login
    enable_csrf: true
  logout:
    path: logout
```

On crée ensuite le contrôleur Login pour gérer l'authentification.

```
no usages
10 class LoginController extends AbstractController
11 {
12     #[Route('/login', name: 'app_login')]
13     public function index(AuthenticationUtils $authenticationUtils): Response
14     {
15         // Récupération éventuelle de l'erreur
16         $error = $authenticationUtils->getLastAuthenticationError();
17         // Récupération éventuelle du dernier nom d'utilisateur utilisé
18         $lastUsername = $authenticationUtils->getLastUsername();
19
20         return $this->render( view: 'login/index.html.twig', [
21             'last_username' => $lastUsername,
22             'error' => $error,
23         ]);
24     }
25
26     no usages
27     #[Route('/logout', name: 'logout')]
28     public function logout() {
29         // Géré par le bundle security de Symfony
30     }
```

Il faut maintenant créer le formulaire de login puis, on définit dans security.yaml qu'il faut le rôle administrateur pour accéder aux routes sous /admin/. Cela va déclencher le processus d'authentification si aucun utilisateur n'est connecté

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

Enfin, on ajoute un bouton se déconnecter dans les templates baseadmin et basefront.

```
<div class="text-left" style="...">
  
  {% if app.user %}
    <a href="{{ path('logout') }}" class="btn btn-secondary" style="...">Se déconnecter</a>
  {% endif %}
</div>
```

Voilà le résultat :



MediaTek86
Des formations pour tous sur des outils numériques

Administration du site

[Retour à l'accueil](#) [Formations](#) [Playlists](#) [Catégories](#)

Authentifiez-vous

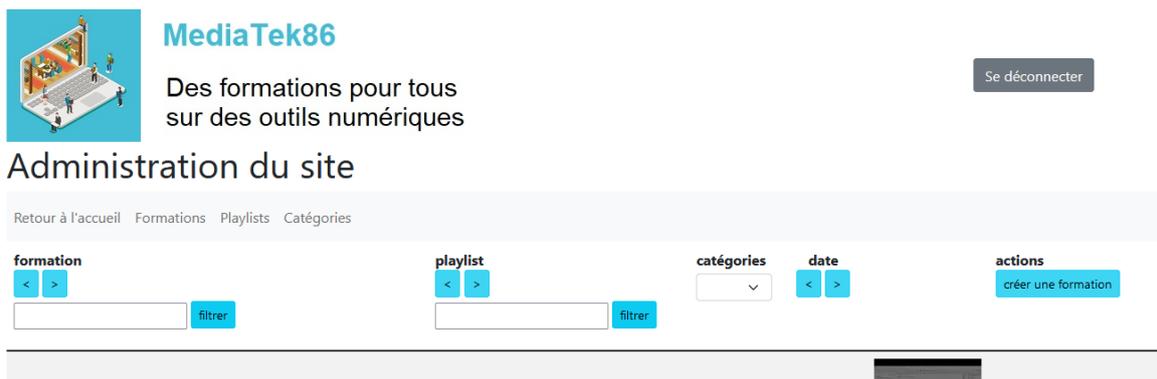
Nom d'utilisateur :

Mot de passe :

[Se connecter](#)

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

Puis, le bouton « Se déconnecter » apparaît quand on est connecté :



MediaTek86
Des formations pour tous sur des outils numériques

[Retour à l'accueil](#) [Formations](#) [Playlists](#) [Catégories](#)

[Se déconnecter](#)

Administration du site

[Retour à l'accueil](#) [Formations](#) [Playlists](#) [Catégories](#)

formation [<](#) [>](#) [filtrer](#)

playlist [<](#) [>](#) [filtrer](#)

catégories [v](#)

date [<](#) [>](#)

actions [créer une formation](#)

Étape 4 : Tester et documenter

Tâche 1 : Gestion des tests – Temps estimé 7 heures / temps réel 6 heures

Pour cette tâche, il nous est demandé de créer plusieurs tests au travers du site internet : des tests unitaires, d'intégrations, fonctionnels et de compatibilité.

Les tests doivent respecter le cahier des charges et le PV de recette (disponible depuis le portfolio).

Pour commencer, on réalise des tests unitaires sur la méthode `formation.getPublishedAtString()`

```

8 class FormationTest extends TestCase
9 {
10 public function testGetPublishedAtStringDateValide() {
11     $date = new \DateTime( datetime: '2025-01-04 17:00:12');
12
13     $formation = new Formation();
14     $formation->setPublishedAt($date);
15
16     $this->assertEquals( expected: '04/01/2025', $formation->getPublishedAtString());
17 }
18
19 public function testGetPublishedAtStringDateVide() {
20     $formation = new Formation();
21
22     $this->assertEquals( expected: '', $formation->getPublishedAtString());
23 }
24 }
25

```

Ensuite, on réalise des tests d'intégration sur les règles de validation d'une formation

```

10 class FormationValidationsTest extends KernelTestCase
11 {
12     3 usages
13     private function getFormation() {
14         $formation = new Formation();
15         $formation->setTitle( title: "Formation de test");
16         $formation->setDescription( description: "Description de test");
17         $formation->setVideoId( videoid: "rUnuYTjaBoU");
18         return $formation;
19     }
20     public function testPublishedAtPosterieurAMaintenant() {
21         $datePosterieure = (new \DateTime( datetime: 'now'))->add(new \DateInterval( duration: 'PT2H'));
22         $formation = $this->getFormation()->setPublishedAt($datePosterieure);
23
24         self::bootKernel();
25         $validateur = self::getContainer()->get(ValidatorInterface::class);
26         $erreurs = $validateur->validate($formation);
27         $this->assertCount( expectedCount: 1, $erreurs);
28     }
29     public function testPublishedAtAnterieurAMaintenant() {...}
30     public function testPublishedAtEgalAMaintenant() {...}
31 }
32

```

Ensuite, il faut des tests d'intégration sur les méthodes ajoutées dans les repositories, pour cela, j'ai créé une base de données de test afin de ne pas modifier la base de données de production.

J'ai utilisé GitHub pour voir l'historique et les méthodes que j'ai ajoutées dans les repositories puis, j'ai créé des classes de tests pour les différents repositories

```
9 class PlaylistRepositoryTest extends KernelTestCase
10 {
11     2 usages
12     private function getRepository() {
13         self::bootKernel();
14         return self::getContainer()->get(PlaylistRepository::class);
15     }
16
17     public function testFindAllOrderByCountAscendant() {
18         $repository = $this->getRepository();
19         $playlists = $repository->findAllOrderByCount('ASC');
20
21         $this->assertEquals( expected: 'playlist test', $playlists[0]->getName());
22         $this->assertEquals( expected: 'Cours Informatique embarquée', $playlists[1]->getName());
23         $this->assertEquals( expected: 'Cours Merise/2', $playlists[2]->getName());
24     }
25
26     public function testFindAllOrderByCountDescendant() {...}
27
28 }
29
30 }
```

Ensuite, des tests fonctionnels ont été construits, ici, on va tester les contrôleurs. Pour ce faire, on effectue des requêtes HTTP avec un client et on vérifie le résultat

Par exemple pour l'accueil :

```
10 class AccueilControllerTest extends WebTestCase
11 {
12     public function testIndex() {
13         $client = self::createClient();
14         $client->request( method: 'GET', uri: '/');
15         $this->assertResponseIsSuccessful();
16     }
17 }
```

On teste aussi sur les pages avec les listes les tris, les recherches puis on teste un des liens pour voir si le contenu est celui attendu :

```

public function testLienPlaylistFonctionne() {
    $client = self::createClient();
    $client->request( method: 'GET', uri: '/playlists');
    $client->clickLink( linkText: 'Voir détail');
    $response = $client->getResponse()->getContent();
    self::assertStringContainsString( needle: '<h4 class="text-info mt-5">Bases de la programmation (C#)</h4>', $response);
}

public function testSortPlaylistAscendant() {
    $client = self::createClient();
    $client->request( method: 'GET', uri: '/playlists/tri/name/ASC');
    $this->assertSelectorTextContains( selector: 'h5.text-info', text: 'Bases de la programmation (C#)');
}

public function testSortPlaylistDescendant() {...}

public function testRecherchePlaylist() {
    $client = self::createClient();
    $client->request( method: 'POST', uri: '/playlists/recherche/name', ['recherche' => 'test']);
    $this->assertSelectorCount( expectedCount: 1, selector: 'h5.text-info');
    $this->assertSelectorTextSame( selector: 'h5.text-info', text: 'playlist test');
}

```

Pour pouvoir tester les contrôleurs admin, on commence par s'identifier avec un utilisateur ayant le rôle admin puis on peut effectuer les mêmes types de vérifications :

```

12     private function loginClientAsAdmin($client) {
13         $userRepository = self::getContainer()->get(UserRepository::class);
14         $adminUser = $userRepository->findOneBy(['id' => 1]);
15         return $client->loginUser($adminUser);
16     }
17
18     public function testIndexAdminFormations() {
19         $client = self::createClient();
20         $this->loginClientAsAdmin($client);
21         $client->request( method: 'GET', uri: '/admin/formations');
22
23         $this->assertResponseIsSuccessful();
24     }

```

Voilà le résultat de tous les tests qui passent :

```

Test Results 18 sec 576 ms ✓ Tests passed: 55 of 55 tests - 18 sec 576 ms
C:\wamp64\bin\php\php8.2.13\php.exe C:
Testing started at 15:53 ...
PHPUnit 9.6.19 by Sebastian Bergmann a

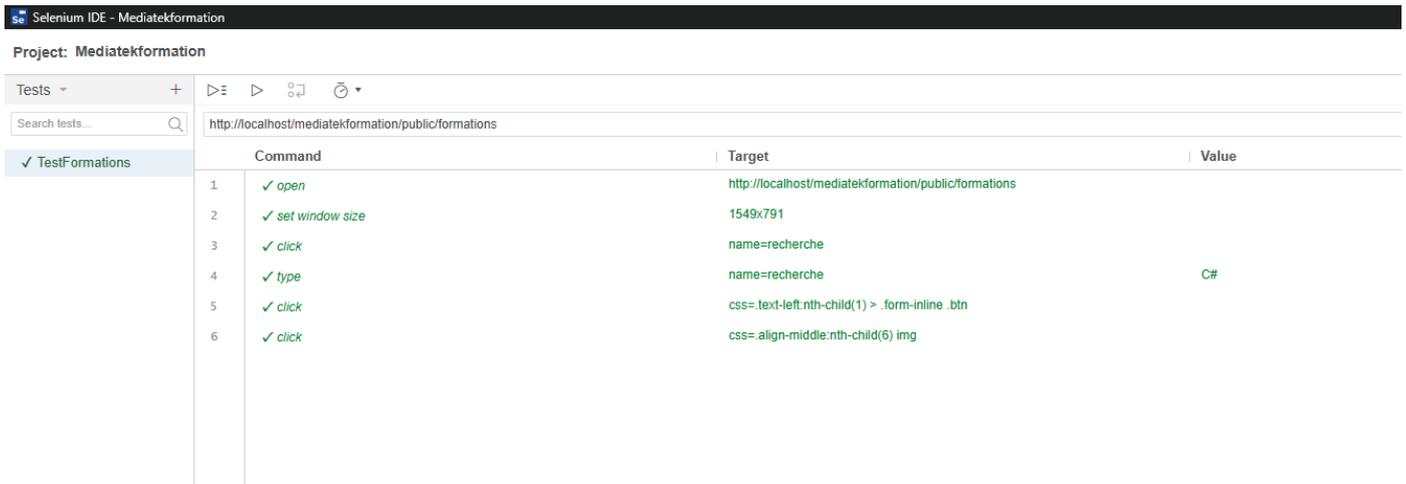
Testing

Time: 00:18.791, Memory: 54.00 MB
OK (55 tests, 107 assertions)
Process finished with exit code 0

```

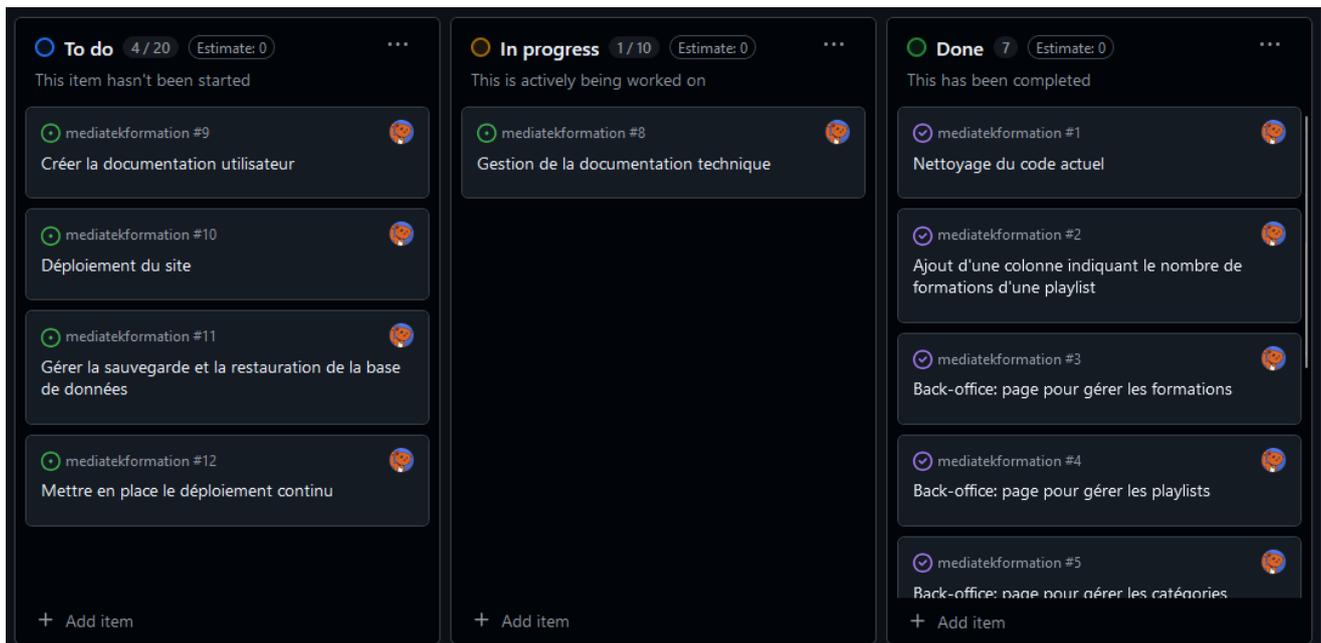
Enfin, pour les tests de compatibilité, on utilise Selenium pour simuler un navigateur et on teste un scénario pour voir s'il se rejoue avec succès.

On peut exporter ce test et l'exécuter sur différents navigateurs pour voir si cela fonctionne partout.



Tâche 2 : Créer la documentation technique – Temps estimé 1 heure / temps réel 1 heure

Pour cette tâche, il nous est demandé d'ajouter des commentaires normalisés à travers tout le code créé afin de pouvoir générer une documentation technique du site complète.



J'ai donc commencé par ajouter des commentaires normalisés dans tout le code qui a été créé par les développeurs. Ensuite j'ai testé la génération en local avec phpDocumentor. Puis, j'ai mis en place la génération et publication automatique vers GitHub Pages grâce à une GitHub Action. La documentation est maintenant accessible depuis <https://docs.mediatekformation.ifrancart.fr>

The screenshot shows a web browser displaying a documentation page for an application. The page has a dark theme. On the left, there is a sidebar with a search bar and a navigation menu. The main content area is titled 'Application' and contains a 'Table of Contents' section with a 'Classes' subsection. The 'Classes' list includes:

- [AccueilController](#) (Description of AccueilController)
- [AdminCategoriesController](#) (Contrôleur gérant les pages d'administration des catégories)
- [AdminFormationsController](#) (Contrôleur gérant les pages d'administration des formations)
- [AdminPlaylistsController](#) (Contrôleur gérant les pages d'administration des formations)
- [FormationsController](#) (Controleur des formations)
- [LoginController](#) (Contrôleur gérant la connexion / déconnexion des utilisateurs)
- [PlaylistController](#) (partially visible)

Tâche 3 : Créer la documentation utilisateur - Temps estimé 2 heures / temps réel 2 heures

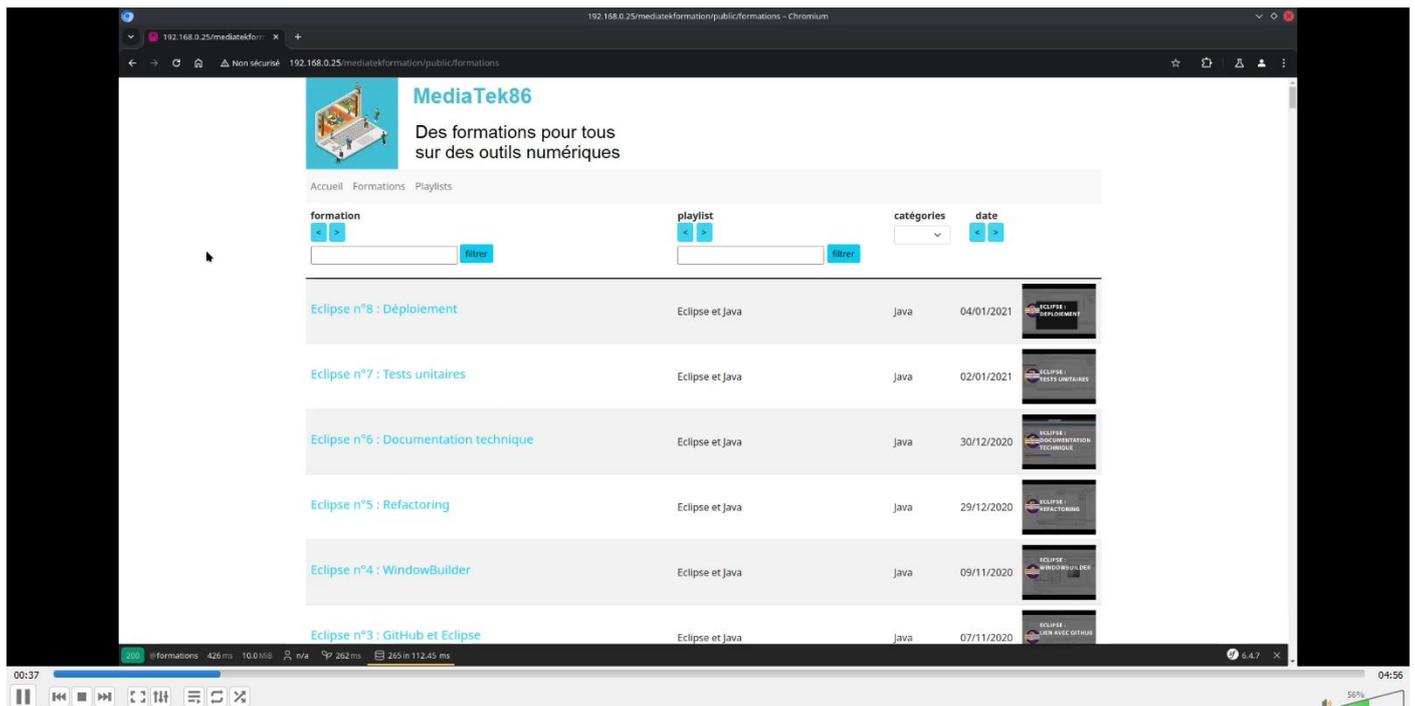
Pour cette tâche, il nous est demandé de créer une vidéo montrant toutes les fonctionnalités du site internet avec explication orales et faisant moins de 5 minutes.

The screenshot shows a Kanban board with three columns: 'To do', 'In progress', and 'Done'. Each column has a header with a count and an 'Estimate: 0' label. The 'To do' column has 3 items, 'In progress' has 1 item, and 'Done' has 8 items. The tasks are as follows:

- To do (3 / 20):**
 - mediatekformation #10: Déploiement du site
 - mediatekformation #11: Gérer la sauvegarde et la restauration de la base de données
 - mediatekformation #12: Mettre en place le déploiement continu
- In progress (1 / 10):**
 - mediatekformation #9: Créer la documentation utilisateur
- Done (8):**
 - mediatekformation #1: Nettoyage du code actuel
 - mediatekformation #2: Ajout d'une colonne indiquant le nombre de formations d'une playlist
 - mediatekformation #3: Back-office: page pour gérer les formations
 - mediatekformation #4: Back-office: page pour gérer les playlists
 - mediatekformation #5: Back-office: page pour gérer les catégories

J'ai donc enregistré une vidéo après avoir préparé les outils et le script. J'ai ensuite coupé quelques parties de la vidéo afin de la rendre plus fluide.

Capture d'écran de la vidéo :



Étape 5 : Déployer le site et gérer le déploiement continu

Tâche 1 : déployer le site – Temps estimé 2 heures / temps réel 2 heures

Dans cette tâche, il faut déployer le site chez un hébergeur en mettant à jour notamment les conditions générales d'utilisation et la configuration pour rendre le site fonctionnel sur l'infrastructure de l'hébergeur.

J'ai pour ma part déployé le site sur un VPS hébergé chez IONOS en utilisant Docker.

Après connexion sur le VPS, j'ai commencé par cloner et paramétrer un projet utilisant docker-compose-lamp. Ce dépôt contient une base pour déployer un stack LAMP facilement avec Docker compose.

Pour déployer le site correctement, plusieurs étapes ont été réalisées :

- Clonage du code source de Mediatekformation
- Configuration DNS du domaine mediatekformation.jfrancart.fr
- Paramétrage de la base docker-compose-lamp pour l'adapter au projet
- Ajout de la base de données depuis le panel phpMyAdmin
- Configuration du proxy inversé de test utilisé (permet la gestion centralisée de plusieurs sites sur un serveur)
- Modification des paramètres d'accès à la base de données dans le fichier .env.local
- Installation des bundles utilisés dans l'application avec `composer install`
- Une fois que la configuration fonctionne sur le proxy inversé de test, déploiement sur celui de production
- Changement du mot de passe administrateur pour la partie back-office
- Création de services systemd pour automatiser le démarrage du site lors du redémarrage du serveur

Une fois ces étapes réalisées, le site est déployé et accessible sur <https://mediatekformation.jfrancart.fr>

Tâche 2 : Gérer la sauvegarde et la restauration de la base de données – Temps estimé 1 heure / temps réel 1 heure

Pour cette tâche, il nous est demandé d'automatiser la sauvegarde et de permettre la restauration de la base de données.

Pour la sauvegarde automatique, on peut créer un simple script qui va exécuter mysqldump et sauvegarder le résultat dans un dossier :

```
1 #!/bin/sh
2 DATE=`date -I`
3
4 # Suppression de l'ancien fichier de sauvegarde de BDD
5 #find /root/mediatekformation-bdd-bkps/bdd* -mtime -1 -exec rm {} \;
6
7 # Sauvegarde de la BDD dans /root/mediatekformation-bdd-bkps/
8 docker exec mediatekformation-mariadb106 /usr/bin/mysqldump -u root --password=tiger --databases mediatekfor
  mation --single-transaction | gzip > /root/mediatekformation-bdd-bkps/bddbbackup_`${DATE}`.sql.gz
```

On teste le script et on observe le fichier résultant :

```
root@saesr-box:~# ./mediatekformation-bdd-bkp
root@saesr-box:~# ls -lah mediatekformation-bdd-bkps/
total 40K
drwxr-xr-x  2 root root  4.0K Feb  5 10:12 .
drwx----- 15 root root  4.0K Feb  5 10:40 ..
-rw-r--r--  1 root root  31K Feb  5 10:41 bddbbackup_2025-02-05.sql.gz
```

Ensuite, on peut modifier le « crontab » pour lui dire d'exécuter le script tous les jours à 12:00 :

```
23 # m h dom mon dow  command
24 00 12 * * * /root/mediatekformation-bdd-bkp
```

Tâche 3 : mettre en place le déploiement continu (estimé 1h réel 1h)

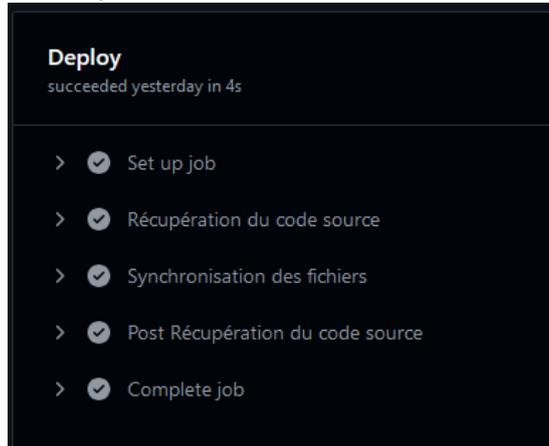
Pour cette dernière tâche, il nous est demandé de mettre en place le déploiement en continu, c'est-à-dire le déploiement automatique lors de changements dans le code source de l'application.

On va utiliser une action GitHub qui synchronise les fichiers avec le serveur à chaque fois qu'un commit est « push » dans le dépôt :

```
mediatekformation / .github / workflows / deploy.yml
Refragg Mise en place du déploiement continu ✓
Code Blame 18 lines (17 loc) · 549 Bytes
1 on: push
2 name: Déploiement du site après push
3 jobs:
4   web-deploy:
5     name: Deploy
6     runs-on: ubuntu-latest
7     steps:
8     - name: Récupération du code source
9       uses: actions/checkout@v2
10
11     - name: Synchronisation des fichiers
12       uses: SamKirkland/FTP-Deploy-Action@4.3.0
13       with:
14         server: ${{ secrets.ftp_server }}
15         server-dir: ${{ secrets.ftp_server_dir }}
16         port: ${{ secrets.ftp_port }}
17         username: ${{ secrets.ftp_username }}
18         password: ${{ secrets.ftp_password }}
```

Après configuration d'un serveur FTP sur le VPS et remplissage des secrets nécessaires dans le dépôt GitHub, on peut tester le déploiement continu en faisant une modification sur le site pour voir si elle est prise en compte automatiquement :

```
templates/pages/accueil.html.twig
@@ -2,7 +2,7 @@
1 {% block body %}
2   <p class="mt-3">
3     <h3>Bienvenue sur le site de MediaTek86 consacré aux formations en
4     ligne</h3>
5 +   <h3>Bienvenue sur le site de MediaTek86 consacré aux formations en
6     ligne !</h3>
7   </p>
8   <p class="mt-3">
9     Vous allez pouvoir vous former à différents outils numériques
10    gratuitement et directement en ligne.<br />
11  </p>
12 </block body %}
```



Nous pouvons observer que le site s'est mis à jour automatiquement



MediaTek86

Des formations pour tous
sur des outils numériques

Accueil Formations Playlists

Bienvenue sur le site de MediaTek86 consacré aux formations en ligne !

Vous allez pouvoir vous former à différents outils numériques gratuitement et directement en ligne.

Dans la partie [Formations](#), vous trouverez la liste des formations proposées. Vous pourrez faire des recherches et des tris. En cliquant sur la capture, vous accéderez à la présentation plus détaillée de la formation ainsi que la vidéo correspondante.

Vous pouvez aussi retrouver les vidéos regroupées dans des playlists, dans la partie [Playlists](#).

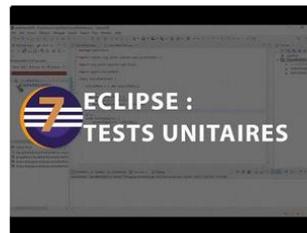
Voici les **deux dernières formations** ajoutées au catalogue :



04/01/2021

Eclipse n°8 : Déploiement

playlist : Eclipse et Java
catégories : Java



02/01/2021

Eclipse n°7 : Tests unitaires

playlist : Eclipse et Java
catégories : Java

ms 30.0MiB 2 in 0.02 ms n/a 17 ms 4 in 2.40 ms

Bilan

Le développement et le déploiement de ce site internet m'a permis d'élargir mes compétences notamment sur les langages et frameworks utilisés, la gestion des données d'un site, l'authentification, la gestion de projet pour réaliser les tâches à temps et autres.

Les objectifs de développement, documentation et déploiement ont pu être menés selon moi à bien malgré le fait que certaines libertés ont été prises ainsi ce qui a mené à certaines adaptations nécessaires.