Application web Mediatek86 de gestion des formations (PHP)

SOMMAIRE

1. Contexte

La société InfoTech Services 86 (ou *ITS 86*) est une ESN constituée d'une équipe de 32 collaborateurs, administratifs, ingénieurs et techniciens. Elle organise ses activités autour de deux pôles : développement et Systèmes et réseau. Ses clients sont des TPE et des PME.

Le pôle développement propose des solutions d'hébergements sur des serveurs dédiés, intégration de services et développement logiciel avec gestion et création de bases de données.

ITS 86 a récemment obtenu la gestion du parc informatique du réseau des médiathèques de la Vienne, *MediaTek86*. Les médiathèques mettent à disposition de leurs adhérents des documents sur divers supports (papier, cédérom, livre numérique, DVD, etc...).

Dans chaque médiathèque, il existe plusieurs types d'employés :

- Ceux du service Administratif, qui gèrent notamment les achats de livres
- Ceux du service Prêt, qui s'occupent des prêts et du rangement des documents dans la médiathèque
- Ceux du service Médiation culturelle, qui communiquent sur les activités de la médiathèque

MediaTek86 gère le portail web unique, qui permet aux adhérents de réserver et d'emprunter plus de 200 000 documents (livres, journeaux, CD, DVD, jeux), revues et ressources numériques. *Mediatek86* est un réseau qui regroupe les responsables de chaque médiathèque, un chef de projet numérique, une chargée de communication numérique et un infographiste. Les projets numériques sont pilotés par le chef de projet numérique qui est chargé de la maîtrise d'ouvrage auprès des ESN auxquelles *MediaTek86* fait appel.

Pour améliorer l'attractivité des médiathèques, *MediaTek86* veut proposer, en complément de l'activité principale de la médiathèque, des formations aux outils numériques et des autoformations en ligne. Pour cela, elle a fait appel à ITS 86 pour développer une application web permettant d'accéder aux vidéos d'auto-formation proposées par la médiathèque.

Le site mediatekformation, utilisant le framework Symfony, permet d'accéder aux vidéos d'auto-formation proposées par la médiathèque et qui sont aussi accessibles sur YouTube. Elle était constituée de 3 pages :

- Une page d'accueil présentant le site web et avec les deux dernières formations ajoutées
- Une page présentant la liste de toutes les formations, avec la possibilité de filtrer par titre et de trier par titre et par date de parution.
- Une page détaillant la formation sélectionnée, qui affiche la vidéo de formation, son titre, sa description et sa date de parution.

Les évolutions demandées sont les suivantes :

1. Ajouter des niveaux aux formations, pour indiquer à qui elles s'adressent.
2. Coder la partie back office de l'application, pour pouvoir gérer les formations et les niveaux directement depuis l'application.
3. Réaliser une vidéo de démonstration d'utilisation de l'application

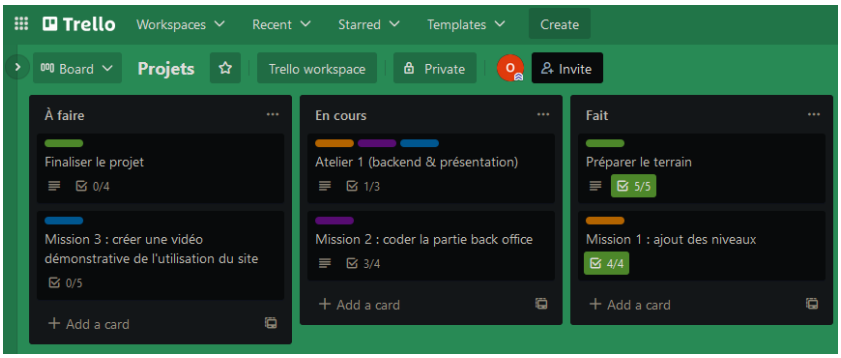
2. Outils utilisés

Les outils utilisés pour cette situation professionnelle sont les suivants :

- IDE : **Apache NetBeans 12.5**
- Logiciel de gestion de versions : **GitHub**
- Outil de suivi de projet : **Trello**

3. Réalisation des missions

Tout au long de cette situation professionnelle, j'ai utilisé GitHub et Trello pour gérer correctement l'évolution du projet, en effectuant un commit sur GitHub pour chaque tâche à effectuer sur Trello.



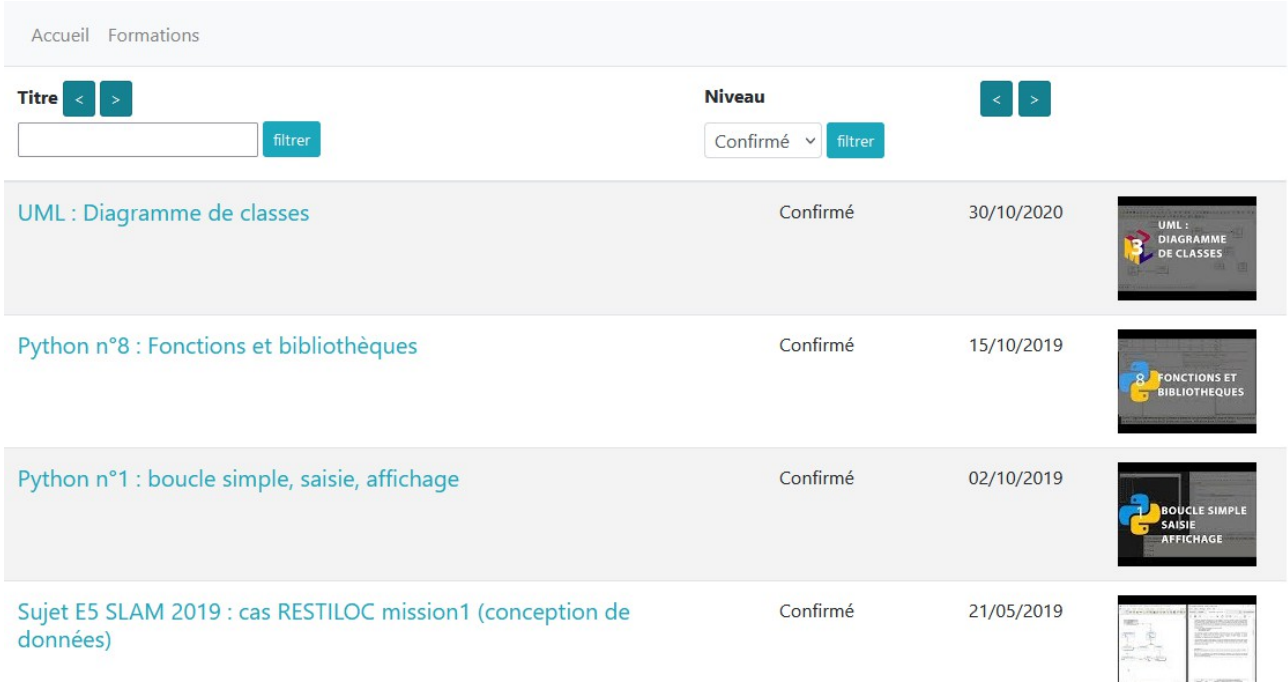
3.1. Ajout des niveaux

Cette mission consiste en la modification de l'application de façon à permettre l'enregistrement d'un niveau pour chaque formation.

Pour cela, il a fallu ajouter une table "Niveau" à la base de données, en passant par des lignes de commandes de composer. En effet, l'application utilisant le framework Symfony, les modifications au niveau de la base de données doivent se faire via composer, qui se charge ensuite de créer les objets correspondant dans le code de l'application. Il faut également utiliser composer pour modifier l'entité Formation et la structure de la table formation, pour ajouter en clé étrangère l'id du niveau.

Une fois la table et "l'entity" Niveau créés avec composer, j'ai rempli la nouvelle table avec des exemples, et j'ai mis un id de niveau aléatoire à toutes les formations existantes.

Il a fallu ensuite modifier la page des formations pour afficher les niveaux dans une nouvelle colonne, entre le titre et la date de parution. Un bouton pour filtrer les formations par niveau a aussi dû être ajouté.



Le niveau actuellement filtré est mémorisé et reste affiché après avoir cliqué sur le bouton filtrer.

Fonction gérant le tri sur les niveau :

```
public function findAllContain($champ, Request $request): Response{
    if($this->isCsrfTokenValid('filtre_'.$champ, $request->get('_token'))){
        $niveaux = $this->repositoryNiveau->findAll();
        $valeur = $request->get("recherche");
        if ($champ == "niveau") {
            $formations = $this->repository->findByLevel("libelle", $valeur);
        }
        else {
            $formations = $this->repository->findByContainValue($champ, $valeur);
        }
        return $this->render(self::PAGEFORMATIONS, [
            'formations' => $formations,
            'niveaux' => $niveaux,
            'niveauselection' => $valeur
        ]);
    }
    return $this->redirectToRoute("formations");
}
```

Enfin, il a également fallu gérer l'affichage du niveau de la formation dans la page détaillant la formation sélectionnée.

Après avoir terminé cette mission, les objectifs suivants étaient complétés :

- Évolution de la base de données pour gérer les niveaux
- Affichage de niveaux pour les formations
- Possibilité de trier les formations par niveau

3.2. Coder la partie back-office

Il était ici demandé de coder intégralement une partie de l'application permettant de gérer les formations et les niveaux. Il fallait également que cette partie soit sécurisée par une authentification. Enfin, il était demandé que l'application soit sécurisée avec des requêtes paramétrées, des tokens dans les formulaires et des contrôles de saisie.

Pour toutes les pages d'administration, j'ai créé un sous-dossier *admin* dans le dossier *templates* qui contient toutes les pages. Les pages pour l'administration étant très similaires à celles du front end, j'ai simplement eu à utiliser un template "baseadmin" pour afficher les nouvelles pages dans la barre de navigation.

Pour la page de gestion des formations, j'ai simplement eu à copier-coller le contenu de la page listant les formations dans le front end. Ensuite, j'ai ajouté les boutons pour supprimer et modifier les formations, et le bouton d'ajout. J'ai dû faire des recherches sur Bootstrap pour cette partie, pour réussir à correctement aligner les objets graphiques et leur donner le bon aspect.

MediaTek86

Des formations sur des outils numériques pour tous

Accueil Gestion formations Gestion niveaux Se déconnecter

Titre < > filtrer

Niveau < > Débutant filtrer

Actions Ajouter

Eclipse n°8 : Déploiement	Maître	28/12/2020		Modifier Supprimer
Eclipse n°7 : Tests unitaires	Expert	28/12/2020		Modifier Supprimer
Eclipse n°6 : Documentation technique	Débutant	28/12/2020		Modifier Supprimer

La suppression de formation fonctionne avec une simple confirmation dans la vue, avant l'appel de la méthode de suppression dans le controleur.

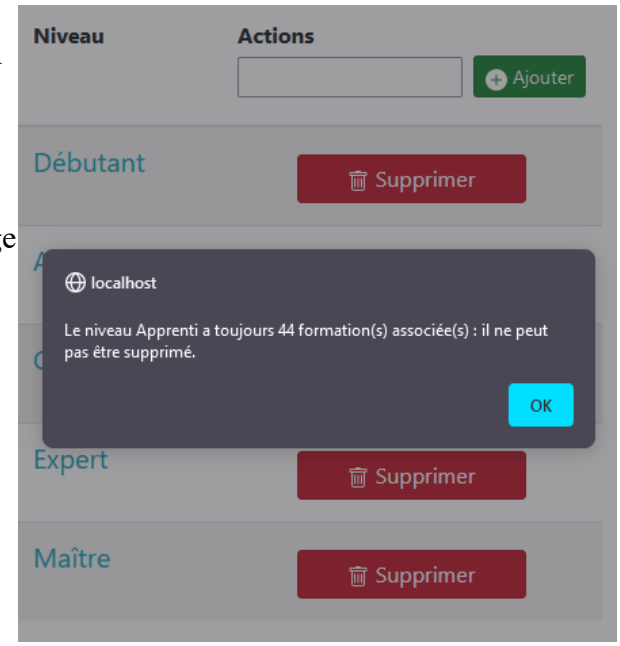
Pour l'ajout et la suppression de formation, un seul formulaire propre aux formations est utilisé. La plupart des saisies sont contrôlées avec des tags directement dans l'entity Formation, au dessus du paramètre contrôlé :

```
/**
     * @ORM\Column(type="string", length=91, nullable=true)
     * @Assert\NotBlank
     * @Assert\Length(
     *      max = 91
     * )
     */
    private $title;
```

Les champs 'miniature' et 'picture' ne sont cependant pas contrôlés de la même façon. Ceux-ci étant des URL, donc de type "string", il n'est pas possible de tester la taille de leur image directement. J'ai pour cela codé une fonction valideImages, avec le tag *@Assert\Callback*, pour tester que les URL contiennent bien une image, et que cette image fait la bonne taille :

```
/**
     * Vérifie que les images sont conformes
     * @Assert\Callback
     */
    public function validateImages(ExecutionContextInterface $context)
    {
        // Miniature
        if (strlen($this->getMiniature()) > 0) {
            // vérifie l'existence de la miniature à l'url indiquée
            $miniatureSize = @getimagesize($this->getMiniature());
            if (!is_array($miniatureSize) && strlen($this->getMiniature()) > 0) {
                $context->buildViolation("L'URL de la miniature est invalide")
                        ->atPath('miniature')
                        ->addViolation();
            }
            // vérifie que la miniature a la bonne taille
            elseif ($miniatureSize[0] !== 120 || $miniatureSize[1] !== 90) {
                $context->buildViolation("La miniature doit être de taille 120x90")
                        ->atPath('miniature')
                        ->addViolation();
            }
        }
        [...]
    }
```

J'ai ensuite créé la page de gestion des niveaux, qui contient la liste des niveaux avec un bouton permettant de les supprimer. Si le niveau à supprimer est utilisé par des formations, un message informatif s'affichera pour indiquer qu'il n'est pas possible de supprimer cette formation. Une saisie de texte en haut de la page permet d'ajouter un nouveau niveau.



Les ajouts, modifications et suppression sont protégés des CSRF par des tokens, mais aussi les tris et filtres. L'ensemble des requêtes SQL sont paramétrées pour empêcher les SQLi.

Pour sécuriser l'accès au back office, j'ai créé un formulaire avec Symfony, LoginFormAuthenticator, qui se charge d'authentifier les utilisateurs à partir de la table *user* de la base de données.

Le test unitaire FormationTest permet de tester le bon fonctionnement de la fonction setPublisedAt.

Enfin, j'ai généré la documentation technique et créé un scénario avec Selenium que j'ai testé sur Firefox et Chrome.

4. Bilan

Le site web de MediaTek86, mediatekformation, permet maintenant de visionner l'ensemble des formations proposées par la médiathèque, avec un niveau associé à chaque formation.

Après authentification, il est possible pour les administrateurs du site d'ajouter, de modifier et de supprimer des formations. Il leur est aussi possible de supprimer des niveaux si ceux-ci ne sont pas utilisés par des formations.

L'application est sécurisée avec des tokens contre les CSRF et tous les formulaires ont des contrôles de saisie stricts pour éviter les erreurs. Les requêtes SQL sont paramétrées pour empêcher les SQLi.

Enfin, la base de données a évolué automatiquement via composer, avec la création des tables *niveau* et *user.*