MediaTekFormations

Rapport

Atelier professionnel 1 - BTS SIO 2e année

Paul Thorel

Introduction/Contexte

Dans le cadre de mon BTS Services Informatiques aux Organisations (SIO), option Solutions Logicielles et Applications Métier (SLAM), j'ai effectué dans le cadre de ma deuxième année, des ateliers professionnels visant à évaluer mes compétences acquises.

Ici, il fallait reprendre le travail existant d'un développeur de "l'équipe" sur un site de formations d'une médiathèque, en rajoutant certaines fonctionnalités, et en créant une partie back-office sécurisée pour gérer toutes les données (catégories/playlists/formations)

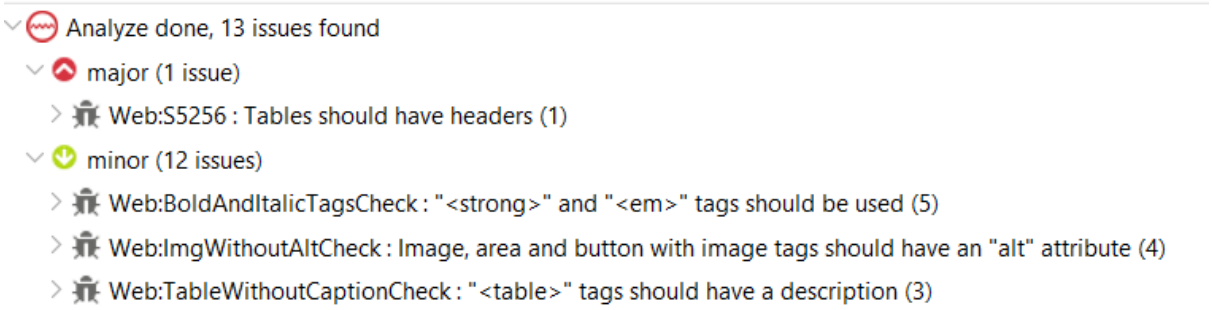
Ressources et outils mis en œuvre

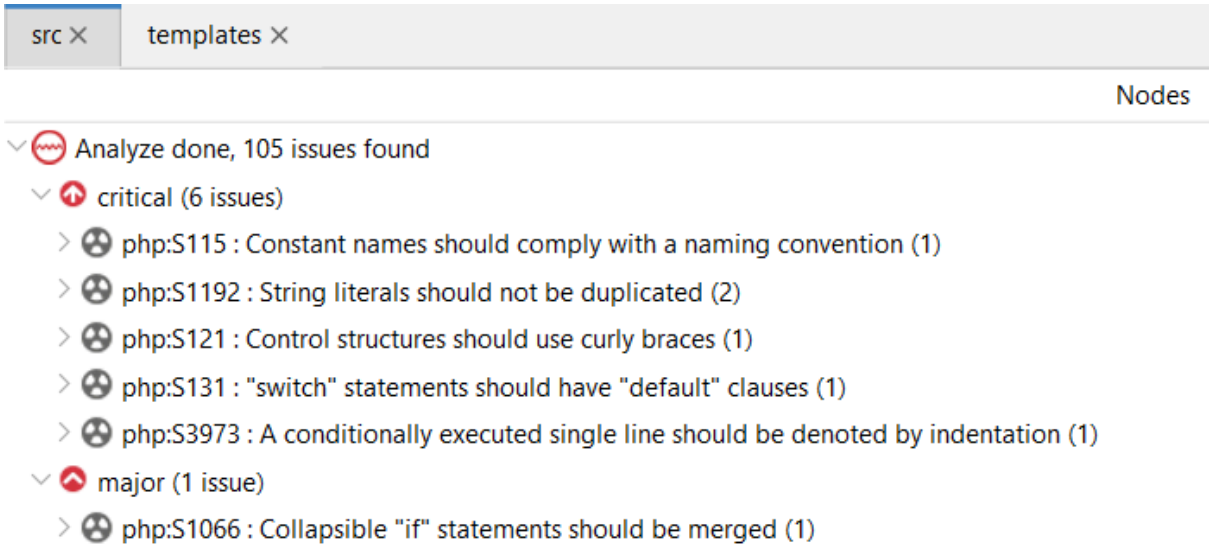
Durant cet atelier, j'ai utilisé:

- l'environnement de développement Apache netbeans et son plugin sonarlint (pour nettoyer dans un premier temps le code)
- PhpMyAdmin tout au long du développement, afin de vérifier la bonne modification et l'intégrité des données
- Git et GitHub pour gérer les différentes versions (faire des sauvegardes régulières ou restaurer la dernière version stable par exemple).
- Stackedit ainsi que google Docs pour rédiger/éditer les readmes et ce rapport
- Et enfin sublime text pour écrire rapidement du code à la volée

Mission 1 - Nettoyer le code

Afin de partir sur des bases propres, l'utilisation du plugin sonarlint a permis de renvoyer les éventuels points à changer sur le code.





La plupart de ces modifications semblent peu importante, mais elles permettent de rendre le code plus lisible et plus flexible (notamment en remplaçant les chaines par des variables/constantes)

Mission 2 - Ajouter les fonctionnalités demandées

La première fonctionnalité demandée concerne l'affichage pour chaque playlist du nombre de formations qu'elle contient (que ce soit sur la page de consultation de chaque playlist ou sur la liste complète).

Les entités playlits contiennent déjà un champ pour récupérer les formations, il suffit de récupérer avec *playlist.formations|length* leur nombre.

```
            <br/><br/>
            <strong>Formations disponibles: {{ playlist.formations|lengt
 h}}</strong>
```

```
<td>
    <span class="align-middle">{{ playlists[k].formations|length}} formations</span>
</td>
```

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

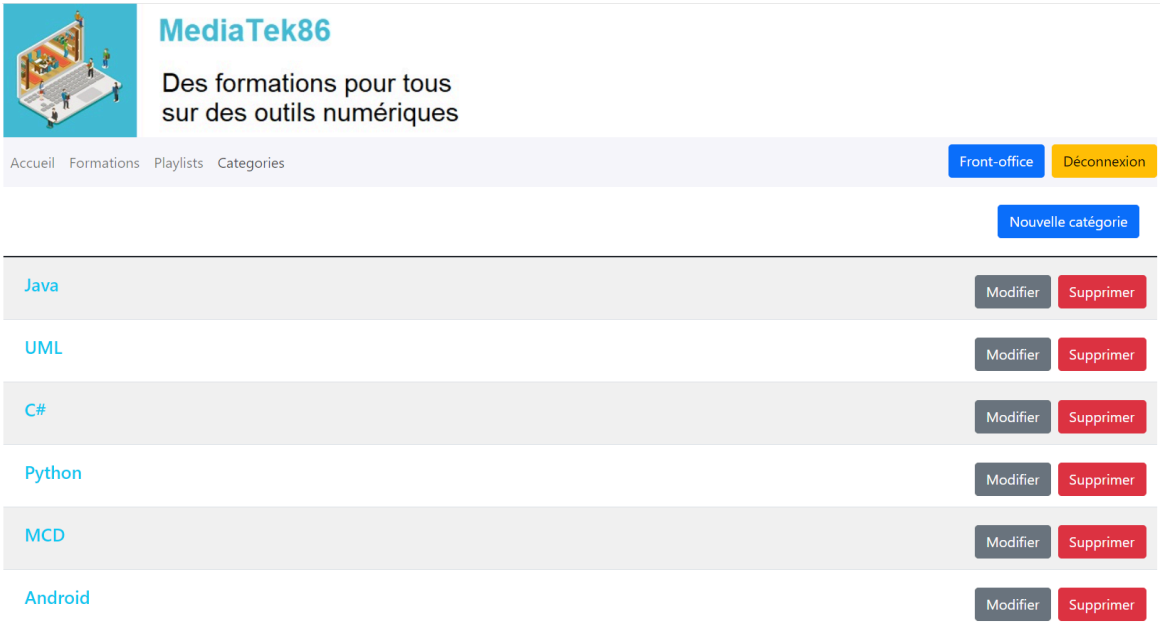
Et de gérer l'appel aux éventuelles interactions sur ces valeurs dans le controlleur:

```
public function findAllOrderByFormation($ordre): array{
    return $this->createQueryBuilder('p')
            ->leftjoin('p.formations', 'f')
            ->groupBy('p.id')
            ->orderBy('COUNT(f)', $ordre)
            ->getQuery()
            ->getResult();
}
```

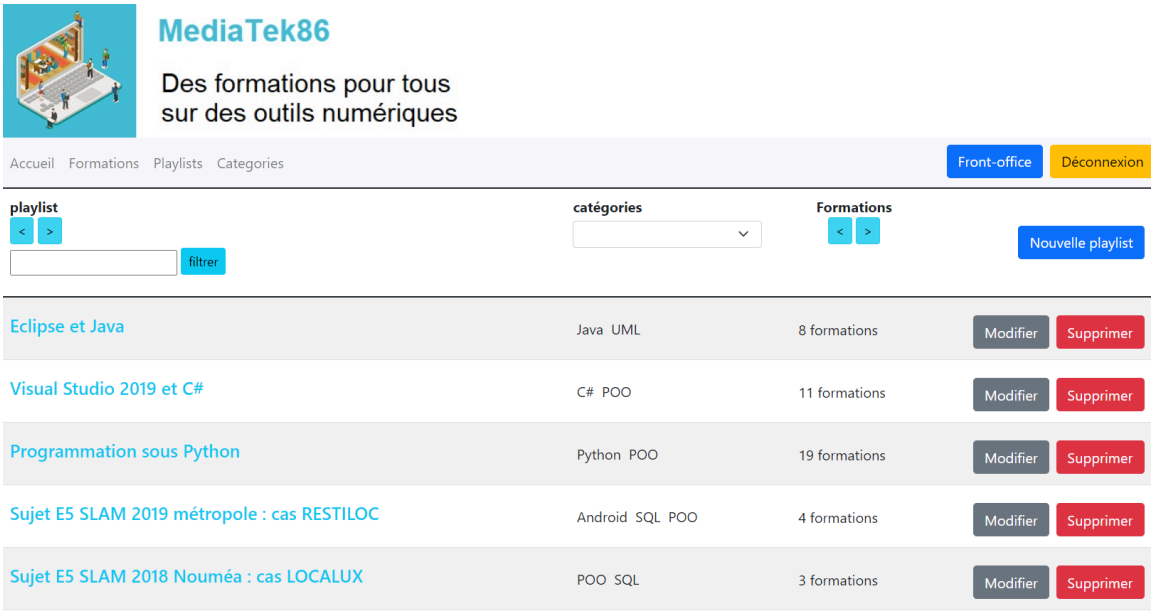
```
case "formations":
    $playlists = $this->playlistRepository->findAllOrderByFormation($ordre);
```

Ensuite, il va falloir rajouter toutes les pages accessibles uniquement à l'administrateur, pour le back-office
Pour l'heure, ce verrou de sécurité n'est pas encore activé, car il faut d'abord vérifier que tout fonctionne (étant testé en local de manière fermée, et des sauvegardes régulières effectuées avec phpmyadmin, il n'y avait pas lieu de trop sécuriser)

J'ai donc créé les différentes vues (certaines étant bien entendu modifiées le long de cette phase voire un peu après) -> notamment pour ce qui est des en-têtes avec les boutons de déconnexion, par exemple.



Page de gestion des catégories



Page de gestion des playlists

Il faut ensuite créer un controller (ici appelé adminBackController), qui va gérer tous les appels (ce qui permettra ensuite de vérifier avec une méthode simple si l'utilisateur est connecté en tant qu'admin , le tout sans sortir de ce controller).

Exemple avec certaines méthodes:

- suppression d'une formation

```php
    #[Route('/admin/formations/suppr/{id}', name: 'admin.formations.suppr', methods: ['GET','POST'])]
    public function supprimerFormation(int $id, Request $request): Response
    {

        if(!$this->is_admin()) return $this->redirectToRoute('accueil');


        $formation = $this->formationRepository->find($id);

        if (!$formation) {
            $this->addFlash('danger', 'Formation non trouvée.');
            return $this->redirectToRoute('admin.formations');
        }

        // Vérification du token CSRF
        if (!$this->isCsrfTokenValid('delete_formation_' . $id, $request->request->get('_token'))) {
            $this->addFlash('danger', 'Token CSRF invalide.');
            return $this->redirectToRoute('admin.formations');
        }

        $this->formationRepository->remove($formation);

        $this->addFlash('success', 'Formation supprimée avec succès.');

        return $this->redirectToRoute('admin.formations');
    }
```

- ajout d'une playlist

```php
#[Route('/admin/playlists/nouvelle_playlist/', name: 'admin.playlists.nouvelle', methods: ['POST'])]
public function nouvellePlaylist(Request $request): Response
{


    if(!$this->is_admin()) return $this->redirectToRoute('accueil');


    $nom = $request->get("name");

    if ($nom) {
        if (!$this->isCsrfTokenValid('nouvelle_playlist', $request->request->get('_token'))) {
            $this->addFlash('danger', 'Token CSRF invalide.');
        return $this->redirectToRoute('admin.playlists');
        }

        $this->playlist = new Playlist();
        $this->playlist->setName($nom);
        $this->playlistRepository->add($this->playlist);

        $this->addFlash('success', 'Playlist ajoutée avec succès.');

    } else {
        $this->addFlash('error', 'La playlist n\'a pas pu être créée.');
    }

    return $this->redirectToRoute('admin.playlists');
}
```

Enfin, après tous les tests effectués, il faut passer à la partie authentification
Je créé un fixture et une entity user (la modification n'étant effectuée qu'une fois, il suffit simplement d'exporter la bdd avec la table user qui a été créée au cas ou)

La documentation de symfony m'a aidé à généré (avec php bin/console) un controlleur pour l'authentification que j'ai ensuite adapté

```php
class LoginAppAuthenticator extends AbstractAuthenticator
{
    private UserPasswordHasherInterface $passwordHasher;
    private UrlGeneratorInterface $urlGenerator;

    // Injection des dépendances dans le constructeur
    public function __construct(UserPasswordHasherInterface $passwordHasher, UrlGeneratorInterface $urlGenerator)
    {
        $this->passwordHasher = $passwordHasher;
        $this->urlGenerator = $urlGenerator;
    }

    public function supports(Request $request): ?bool
    {
        return ($request->getPathInfo() === '/login' && $request->isMethod('POST'));
    }

    public function authenticate(Request $request): Passport
    {
        $name = $request->get("name");
        $pwd = $request->get("pwd");
        $token = $request->get("token_");

        return new Passport(
            new UserBadge($name),
            new CustomCredentials(fn($credentials, \App\Entity\User $user) => $this->passwordHasher->isPasswordValid($user, $credentials), $pwd)
        );
    }

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response
    {
        return new RedirectResponse($this->urlGenerator->generate('admin.accueil'));  // Redirection après succès
    }

    public function onAuthenticationFailure(Request $request, AuthenticationException $exception): ?Response
    {
        return new RedirectResponse($this->urlGenerator->generate('accueil'));  // Redirection en cas d'échec
    }
}
```

J'ai bien sur créé une page d'authentification, et j'ai rajouté dans les deux templates de bases (admin et front) les boutons de connexion/déconnexion,tout en modifiant certains fichiers utilisés par le framework pour gérer la déconnexion

```yaml
logout:
    path: /logout
    invalidate_session: true
    clear_site_data:
        - cookies
        - storage
```

```yaml
    # config/routes.yaml
app_logout:
    path: /logout
    methods: GET
```

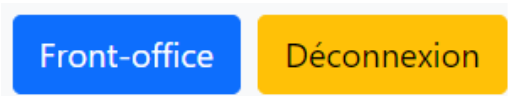
(fichiers routes.yaml et security.yaml)

Enfin, il faut donc modifier toutes les fonctions du controlleurs, afin de rajouter une méthode de test (ici un simple $this->is_admin()), sur l'authentification existante ou non de l'utilisateur)

```php
public function is_admin(): bool
{

    $is_admin = $this->security->isGranted('ROLE_ADMIN');
    if( !$is_admin ) $this->addFlash("error","Merci de s'authenfier pour accéder au back-office");
    return $is_admin;
}
```

Afin de rendre les interactions plus faciles, (sans doute visible dans les vues plus haut), j'ai rajouté un système (avec quelques routes à modifier) pour pouvoir passer de la partie admin à la partie front et inversement.



Mission 3 - Documentation textuelle et video

J'ai enregistré et monté une vidéo de démonstration de l'application, et j'ai également modifié et écrit le readme de l'application, disponible à cette adresse:

https://github.com/Manerr/mediatekformation/

Mission 4 - déploiement continu

Afin de déployer l'application, il suffit de copier la source de l'application sur un serveur disposant de wampserver (ou de php et mysql)
Ensuite, il suffit juste d'importer le script fourni (au format sql) dans phymyadmin ou adminer, et l'application est prête à fonctionner

Je n'ai pas pu faire fonctionner l'hébergement avec azure, les outils proposés par github action compiant bien les fichiers sur le "serveur", mais certains d'entre eux étant supprimés au passage, et le déploiement prenant parfois jusqu'à 30minutes

pour ce résultat.

Compétences acquises:

Lors de cet atelier, j'ai eu l'occasion d'approfondir mes connaissances, que ce soit en PHP ou en HTML , en terme de framework (avec symfony, que j'ai particulièrement appécié pour sa très grande flexibilité) ou en programmation objet.
J'ai également eu l'occasion d'améliorer mes compétences en terme de cybersécurité, en utilisant concrètement des tokens pour contrer des attaques de type csrf.

Conclusion:

Cet atelier m’a apporté une vision plus professionnelle dans le cadre du développement web, et plus largement m'a permis de m'enrichir dans plusieurs domaines, comme précisé plus haut.