3.4 B-11 : Develop an authentication method

Introduction :

L'authentification est un élément essentiel de nombreuses applications Web. Dans ce projet, j'ai mis en place un système d'authentification en utilisant une application React en frontend et une API Node.js en backend. J'ai créé les routes d'authentification, mis en place des formulaires de connexion et d'inscription, en utilisant des cookies sécurisés pour stocker les jetons JWT, et sécuriser les routes nécessitant une connexion à l'aide d'un middleware nommée authguard.

1. **Configuration du Backend :**

- Création des routes : J'ai configuré les routes signin, signup et logout dans l'API. Ces routes gèrent respectivement la connexion, l'inscription et la déconnexion des utilisateurs.

  Lien :

  https://github.com/Lucas-Moreno/code-epitech/blob/main/back/src/controllers/auth.controller.ts

- Middleware d'Authentification : Pour sécuriser les routes nécessitant une connexion, j'ai utilisé un middleware qui vérifie la validité du jeton JWT présent dans les en-têtes de la requête, ce middleware est utilisé sur les routes protégées par un jeton JWT.

  ```
  router.get('/user', (req: Request, res: Response) => {
    authMiddleware(req, res, () => {
      getInfoUser(req, res);
    });
  });
  ```

  Lien :

  https://github.com/Lucas-Moreno/code-epitech/blob/main/back/src/middleware/auth.middleware.ts

2. **Configuration de l'Application React :**

- Création des pages : J'ai créé des pages de connexion, inscription et un bouton déconnexion dans la page d'accueil.
- Gestion des routes : à l'aide de 'react-router-dom', j'ai configuré les routes pour afficher les pages en fonction de l'URL.

3. **Processus d'inscription :**

- Formulaire d'inscription : J'ai créé un formulaire d'inscription qui recueille le nom, l'e-mail et le mot de passe de l'utilisateur.
- Appel API : lors de la soumission du formulaire, une requête POST est envoyée à la route signup de l'API pour créer un nouvel utilisateur, cette logique dans le front est externaliser dans un fichier authService.ts

  Lien :
  https://github.com/Lucas-Moreno/code-epitech/blob/main/front/src/services/authService.ts

4. **Processus de connexion :**

- Formulaire de connexion : J'ai créé un formulaire de connexion qui recueille l'e-mail et le mot de passe de l'utilisateur.
- Stockage du Jeton JWT : Lors de la connexion réussie, l'API renvoie un jeton JWT qui est stocké de manière sécurisée dans un cookie.

  Lien :
  https://github.com/Lucas-Moreno/code-epitech/blob/main/front/src/services/authService.ts

5. **Protection des Routes :**

- AuthGuard : J'ai mis en place un authguard pour protéger les routes nécessitant une connexion. Si l'utilisateur n'est pas connecté, il est redirigé vers la page de connexion. Ce authguard fonctionne en analysant dans l'application si le cookie est bien présent, si oui alors le router redirige sur le bonne route sinon cela renvoie à la page de connexion.

  Lien :

https://github.com/Lucas-Moreno/code-epitech/blob/main/front/src/middlewares/Authguard.tsx

6. **Processus de déconnexion :**

- Déconnexion : Lorsque l'utilisateur choisit de se déconnecter; une requête POST est envoyée à la route logout de l'API pour supprimer le jeton stocké dans le cookie, cette route est appelée dans la page d'accueil de l'application via un bouton déconnexion.

7. **Conclusion :**

En combinant les fonctionnalités de l'application React avec les routes et les middlewares de l'API Node.js, j'ai créé un système d'authentification complet. Les utilisateurs peuvent s'inscrire, se connecter et se déconnecter en toute sécurité, et les routes sont protégées pour assurer la confidentialité des données. Cette mise en place d'authentification fournit une base solide pour le développement d'applications Web sécurisées et conviviales.