DOCUMENTATION TECHNIQUE

Implémentation de l'authentification à l'application ToDo

1 - Environnement de l'application

PHP 7.3.1

Symfony 5.2.4

Bundle gérant la sécurité : symfony/security-bundle

2 – Configuration du service d'authentification

Vous trouverez dans le fichier **config/packages/security.yaml** tous les paramètres nécessaires à la bonne configuration du module Security.

Encoder :

```
encoders:
    App\Entity\User:
        algorithm: bcrypt
```

Il permet de gérer l'algorithme de hash du mot de passe, ici c'est bcrypt

Providers :

```
providers:
    users_in_memory: { memory: null }
    app_user_provider:
        entity:
            class: App\Entity\User
            property: username
```

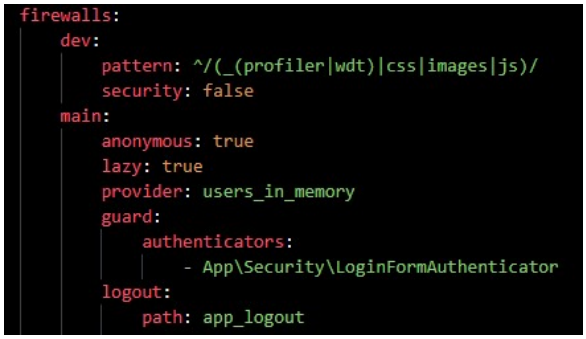
Il permet de gérer l'entité qui servira à s'authentifier lors de la connexion et sur quel attribut. Ici l'entité User avec son username.

Role_hierarchy :

```
role_hierarchy:
    ROLE_ADMIN:       ROLE_USER
```

Permet de gérer la hiérarchie des rôles définis aux utilisateurs. Ici le role_admin hérite du role_user il en aura tous les droits

Firewalls :



Cette section permet de définir comment l'utilisateur va s'authentifier. Ici les pages sont accessibles sauf cas contraire (voir après) à des personnes non connectées (anonymous), le système d'authentification ici LoginFormAuthenticator

Access_control : permet de gérer l'accès aux pages en fonction de l'utilisateur s'il est authentifié ou non ou s'il a le bon rôle. Ici les accès sont gérés autrement (voir après).

3 – Entité User

App\Entity\User

C'est ici que vous allez définir les relations de votre entité User avec la base de données.

Voici les différents paramètres de configuration à connaitre en lien avec le module Security.

@UniqueEntity permet d'établir des contraintes d'unicités dans la base.

@Assert permet d'établir des contraintes avant insertion dans la base pour les attributs.

getRoles() et **setRoles()** permettent respectivement de récupérer et de modifier le rôle d'un user

Les utilisateurs sont stockés dans une base de données MySQL dans une table user.

4 – Configuration de la connexion

App\Security\LoginFormAuthenticator

Permet de gérer tout le processus de connexion.

- supports() vérifie que la connexion se fait bien depuis la page de connexion et que les données sont envoyés en POST

- getCredentials() récupère les identifiants de l'utilisateur ainsi que le jeton de connexion

- getUser() vérifie la validité du jeton et récupère le User sur son username si il existe dans la base.

- checkCredentials() vérifie que le mot de passe saisie correspond au mot de passe crypté du User dans la base.

- onAuthenticationSuccess() défini ce que le projet doit faire après la réussite de l'authentification. Ici redirige vers la page d'accueil ou la page ciblée avant la connexion.

- getLoginUrl() génére l'url vers la page de connexion.

5 – Formulaire de connexion et d'ajout/modification d'un User

Le formulaire de connexion se trouve au chemin suivant **templates\security\login.html.twig**

FORMULAIRE D'AJOUT D'UN USER

App\Form\UserType

C'est ici que se construit et configure le formulaire d'ajout d'un User.

À noter que l'on retrouve sur ce formulaire 5 champs à savoir :

- login
- mot de passe
- confirmation du mot de passe
- email
- choix du rôle

Une variable a été intégrée au tableau $options, **$options['create']**, qui permet de savoir si le formulaire est implémenté via la route de création ou de modification d'un User et ainsi dans le deuxième cas de ne pas demander à retaper le mot de passe.

6 – Autorisation

L'autorisation d'accès aux pages est définie sur les **controllers** via l'annotation **@IsGranted** quand elle ne nécessite pas de configuration particulière.

Sinon si l'autorisation doit être plus précise il a été mis en place un système de **Voter**.

App\Security\Voter\TaskVoter

Mise en place de 2 fonctions sur la route task_delete à savoir, une qui autorise l'accès à la route uniquement pour l'utilisateur qui a créé la tâche et la deuxième qui autorise pour les utilisateurs ayant le rôle admin la suppression des tâches reliées à l'utilisateur anonyme.

App\Security\Voter\UserVoter

Mise en place de l'autorisation d'accès aux pages de gestion des utilisateurs uniquement aux utilisateurs ayant le rôle admin.

Notice pour contribuer au projet ToDo List

1 – Tout d'abord il vous faut créer une copie du projet en clonant le repository ou en faisant un fork du projet depuis cette adresse :

https://github.com/Gregory2911/OC_Projet8.git

2 – Créer ensuite une nouvelle branche du projet.

3 – Commencer par créer des tests unitaires et fonctionnels pertinents sur les modifications ou ajout que vous allez apporter.

4 – Apporter vos modifications au projet tout en respectant les normes et bonnes pratiques du langage PHP et du Framework Symfony. Nous respectons les règles PSR-1 et PSR-2 (PSR-12), PSR-4.

5 – Commenter votre code de façon claire et pertinentes sans surcharge d'informations inutiles.

6 – Tester vos modifications ou apports en exécutant vos tests.

7 – Créer une pull request, celle-ci sera vérifiée et mergée.