DOCUMENTATION TECHNIQUE

ToDo & Co

Créé par
ALKHALLOUFI FATIMA

TABLE DES MATIERES

I. Présentation Projet

1.1 Technologies

Le projet a été mis à jour vers une version de Symfony plus récente et stable : v6.4. Pour le bon fonctionnement du projet, la version 8.1 de PHP est requise.

1.2 Librairies

Les libraires sont toutes installés par Composer, et sont listées dans le fichier composer.json à la racine du projet.

1.3 Installation du projet

Pour installer le projet, il est important de suivre les étapes suivantes:

- Cloner le projet: https://github.com/fatialk/todolist.git
- Modifier le fichier .env avec vos informations et mettre en place la base de données
- Installez les dépendances avec la commande : `composer install`

II. Paramétrage du site

2.1 Fichiers d'authentification

Type	Fichier	Description
Configuration	config/packages/security.yaml	Configuration du processus d'authentification
Entity	src/Entity/User.php	User entity
Controller	src/Controller/SecurityController.php	Regroupe les méthodes loginAction et logout
Authentification	src/Security/UserAuthenticator.php	Méthodes du processus d'authentification de l'application
Template	templates/security/login.html.twig	Template du formulaire de connexion

2.2 Système d'encryptage de mot de passe

Les mots de passe enregistrés dans la base de donnée sont encryptés à l'aide du service PasswordHasher de Symfony :

```
security:
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:

Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
    providers:
        user_provider:
            entity:
                class: 'App\Entity\User'
                property: username
```

2.3 Contrôle des accès

Les pages de gestion des utilisateurs doivent être accessibles uniquement aux utilisateurs avec le rôle ADMIN. Pour contrôler l'accès à ces routes, ces dernières sont précédées par l'annotation :

```
#[IsGranted('ROLE_ADMIN')]
    #[Route('/register', name: 'app_register')]
```

```
#[IsGranted('ROLE_ADMIN')]
    #[Route('/users', name: 'user_list')]
```

```
#[IsGranted('ROLE_ADMIN')]
    #[Route('/users/{id}/edit', name: 'user_edit')]
```

Il également possible de contrôler l'accès aux routes depuis le fichier security.yaml :

```
access_control:
        # - { path: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
```

2.4 Stockage des utilisateurs

Les donnés utilisateurs sont stockés dans la base de données MySql. Plus précisément dans la table user.

Les champs Username et Email sont uniques. pour ajouter la contrainte d'unicité du champ, il suffit d'ajouter l'attribut (unique : true) à l'annotation ORM\Column :

```
#[ORM\Column(type: Types::STRING, length: 60, unique: true)]
    private string $email;
```

III. Base de données

3.1 Renommer le fichier .env.local en .env et modifier la connexion à la base de données :

```
DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name"
```

Détail configuration BDD : [lien de configuration de la BDD dans le site de Symfony]

3.2 Créer la base de données:

```
php bin/console doctrine:database:create
```

3.3 Créer le schéma de la base de données:

```
php bin/console doctrine:schema:update --force
```

3.4 Générer les fixtures:

```
php bin/console doctrine:fixtures:load
```

IV. Fonctionnement de l’authentification

4.1 Page authentification:

Dans un premier temps, l'utilisateur rentre ses identifiants sur la page login.html.twig à la route /login. Pour accéder à cette page, la méthode login est exécutée, sur le Controller : SecurityController. Cette méthode sert à générer la page, et à envoyer des données à la vue avec un render.

4.2 Soumission du formulaire:

Quand le formulaire est validé par l’utilisateur, les données sont récupérées par la classe UserAuthenticator, qui se charge de l’authentification. Cette classe est générique à la création du système d’authentification de la librairie Security-Bundle.

4.3 Redirection de l’utilisateur:

Suivant si, l'utilisateur s'est bien authentifié ou pas. La méthode onAuthenticationSuccess, le redirigera vers la dernière page qu’il a consulté, ou la page de défaut, en cas de succès. A contrario, un message d’erreur sera affiché au-dessus du formulaire de connexion.

V. Collaboration

Tout d'abord, veiller à bien prendre connaissance des normes de codage Symfony qui sont basées sur les normes PSR-1 , PSR-2 , PSR-4 et PSR-12

voir le lien : Normes de codage

Pour la collaboration via Github:

Vous suivre les manipulations décrites sur la page 1.3 (INSTALLATION DU PROJET) et la partie 3 (BASE DE DONNEES) de ce document.

Pour les bonnes pratiques de git, il faudra créer une branche par fonctionnalité a développer Ou sinon une branche par développeur au minimum. Cela évitera les conflits de fichiers.