OPENCLASSROOMS

ToDo & Co - ToDoList

Documentation Technique

Sommaire

1. Librairies....3
1.1 Commande de mise a jour....3
2. Installation du projet....3
2.1 Clonez le repo....3
2.2 Modifier le .env avec vos informations & mettre en place la bdd....3
2.3 Installez les dépendances....3
3. Fichiers d'authentification....4
4. Système d'encryptage de mot de passe....4
5. Contrôle des accès....5
6. Stockage des utilisateurs....5
7. Base de données....6
8. Modification et création d'une table....7
8.1 Création & Mise à jour d'une entité....7
8.2 Création des fichiers de migration....7
8.3 Mise à jour de la structure de la base de données selon les migrations....7
9. Fonctionnement de l'authentification....7
9.1 Page authentification....7
9.2 Soumission du formulaire....8
9.3 Redirection de l'utilisateur....8

1. Librairies

Les librairies sont toutes installés par composer, aut pas hésiter à faire des mises à jours, car cela peut corriger une potentielle faille de sécurité.

1.1 Commande de mise a jour

```
composer install
```

2. Installation du projet

2.1 Clonez le repo

Pour installer le projet vous devrez faire les manipulations suivantes

```
git clone https://github.com/AzzeddDev/p8_todolist.git
```

2.2 Modifier le .env avec vos informations & mettre en place la bdd

Modifier le .env avec vos informations & mettre en place la bdd

```
DATABASE_URL=mysql://root:@127.0.0.1:3306/p8-todolist?charset=utf8mb4
```

2.3 Installez les dépendances

Commande :

```
composer install
```

3. Fichiers d’authentification

Type	Fichier	Description
Description	config/packages/security.yaml	Configuration du processus d’authentification
Entité	src/Entity/User.php	Entité utilisateur
Contrôleur	src/Controller/SecurityController.php	Contrôleur connexion / déconnexion
Authentification	src/Form/UserType.php	Méthodes du processus d’authentification de l’application
Vue	templates/security/login.html.twig	Template du formulaire de connexion

4. Système d’encryptage de mot de passe

Le mot de passe est encrypté dans la basse de donnée, pour plus de sécurité. L’encryptage est en mode automatique pour le moment, ce qui correspond au meilleur choix pour la version de symfony du site.

```
password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: ‘auto’
```

5. Contrôle des accès

Cette partie du fichier « security.yam », sert au contrôle des accès des différentes partie du site. Elles peuvent bien entendu être modifiées pour changer le rôle de l'utilisateur pour la route.

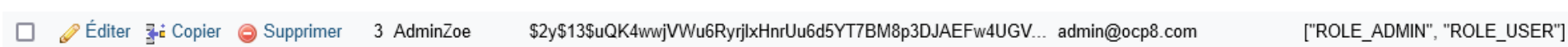
```
access_control:
     - { path: ^/login, roles: PUBLIC_ACCESS }
     - { path: ^/users/create, roles: PUBLIC_ACCESS }
     - { path: ^/users, roles: ROLE_ADMIN }
     - { path: ^/, roles: ROLE_USER }
```

- Path représente la route exemple : /users
- Roles représente le rôle de l'utilisateur exemple : ROLE_ADMIN

6. Stockage des utilisateurs

Les donnés utilisateurs sont stockés dans une basse de données MySql. Plus précisément dans la table user. Il faut savoir que le champ Username est unique et il est possible de rajouter un autre champs unique. Il faudra simplement aller dans l'entité user : src/entity/user, et de rajouter « unique = true » à l'annotation **@Column.**

```
/**
 * @ORM\Column(type=»string», length=25, unique=true)
 * @Assert\NotBlank(message=»Vous devez saisir un nom d'utilisateur.»)
 */
private string $username;
```



Pour faire des requêtes sur la bdd il faudra utiliser le composant Doctrine.
Les tables sont représentées en entité, et sont visitable dans le dossier : src/Entity/. Exemple de requête sur la table user.

```
        $users = $userRepository->findAll();
        return $this->render('user/list.html.twig', [
            'users' => $users
        ]);
```

On peut retranscrire ça avec la requête SQL :

```
SELECT * FROM User
```

7. Base de données

Si pour une raison ou une autre vous voudriez changer de base de donnée. Il faudra changer les informations de connexion dans le fichier .env à la racine du projet

```
DATABASE_URL=mysql://root:@127.0.0.1:3306/p8-todolist?charset=utf8mb4
```

8. Modification et création d'une table

Pour modifier ou créer une table, il faut utiliser le système d'entité de doctrine. Pour ce faire, il faut rentrer ces commandes dans un terminal à la racine du projet. Voici la marche à suivre.

8.1 Création & Mise à jour d'une entité

```
php bin/console make:entity
```

8.2 Création des fichiers de migration

```
php bin/console make:migrations
```

8.3 Mise à jour de la structure de la base de données selon les migrations

```
php bin/console doctrine:migrations:migrate
```

9. Fonctionnement de l'authentification

9.1 Page authentification

Dans un premier temps, la personne va devoir rentrer ces 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.

9.2 Soumission du formulaire

L'authentification s'opère de cette manière :
- Récupération des champs du formulaire grâce à la méthode getCredentials
- Récupération de l'utilisateur dans la BDD par son username, en passant par la vérification de validité d'un token Csrf à la méthode getUser
- Vérification du password associé à l'utilisateur par la méthode checkCredentials

9.3 Redirection de l'utilisateur

Suivant ci, la personne c'est bien authentifier ou pas. Une fois l'authentification a eu lieu, le User sera rediriger vers la page de défaut. A contrario, un message d'erreur sera affiché au-dessus du formulaire de connexion.