Hacène Sahraoui

Conception P8 Openclassroom

ToDo & Co – TodoList
Documentation technique

Table des matières

1. Présentation Projet

1.1 Technologies

Le projet a été mis à jour vers une version plus récente de symfony: v5.4.21 Pour le bon fonctionnement du projet, la version 7.4.26 de PHP est requise.

À titre informatif la version 6 de symfony est déjà sortie, mais une utilisation de cette version n'est pas encore envisageable en l'état, car dans un premier temps elle demande une version minimal 8 de php ce qui n'est pour le moment pas possible, de plus tout les bundle ne sont pas encor totalement compatible, à l'avenir on pourrais éventuellement l'envisager pour une meilleurs sécurité mais surtout une maintenabilité meilleurs.

1.2 Librairies

Les librairies sont toutes installés par Composer, et sont donc visible sur le site https://packagist.org. Toutes les librairies sont listées dans le fichier composer.json à la racine du projet, il ne faut pas hésiter à faire des mises à jours, car cela peut corriger une potentielle faille de sécurité. Cependant, une consultation de la documentation est obligatoire pour éviter toutes complications ! Par exemple un conflit de dépendance .

Commande de mise à jour :

```
PHP composer update
```

1.3 Prérequis et conseils

Il est important de savoir, que pour l'utilisation de phpunit, xDebug est requis pour faire un rendu visuel en HTML. Bien entendu, pour l'utilisation de composer il faudra installer le logiciel. Et enfin, ci vous avez envie d'optimiser plus votre site je vous conseille blackfire .

Documentation xdebug : https://xdebug.org/docs/install
Documentation composer : https://getcomposer.org/doc/00-intro.md
Documentation blackfire :https://blackfire.io/docs/up-and-running/installation

1.3 Installation du projet

Pour installer le projet vous devrez faire les manipulations suivantes :

1. Clonez le repo

```
git clone https://github.com/haceneg2230/p8.git
```

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

Voir page 5

4. Installez les dépendances :

```
composer install
```

2.Paramétrage du site

2.1Fichiers d'authentification

Type	Fichier	Description
Configuration	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/Security/LoginFormAuthenticator.php	Méthodes du processus d'authentification de l'application
Vue	templates/security/login.html.twig	Template du formulaire de connexion

2.2Système d'encryptage de mot de passe

Le mot de passe est encrypté dans la base 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. Il est bien entendu possible de changer le mode dans le ficher de configuration « secrity.yam » dans le bloc « encoders ».

```
security:
        encoders:
                App\Entity\User:
                        algorithm: auto
```

Plus de détails sur les types d'encryptage : https://symfony.com/blog/new-in-symfony-4-3-native-password-encoder

2.3Contrôle 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: ^/user/register, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/task/new, roles: ROLE_USER }
```

```
    - { path: ^/user, roles: ROLE_ADMIN }

Path  représente la route exemple : /users
```

- Roles représente le rôle de l'utilisateur exemple : ROLE_ADMIN

Pour plus de détails sur le composant security : https://symfony.com/doc/current/security.html

2.4Stockage 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**.

Exemple :

```
/**
 *@ORM\Column(type="string", length=180, unique=true)
 */
private $username;
```

←T→		id	password	username	roles (DC2Type:json)	email
☐	Éditer Copier Supprimer	1	$argon2id$v=19$m=65536,t=4,p=1$OElZLmZ0cFNxVjdiTi9...	admin	["ROLE_ADMIN"]	admin@admin.fr

Ci-contre un exemple d'une entrée dans la table user. On peut voir que le password présent est bien encrypté.

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.

```
$user = $this->getDoctrine()->getRepository('App:User')->findAll()];
```

On peut retranscrire ça avec la requête SQL :

SELECT * FROM User

Plus de détails des requêtes avec doctrine : https://symfony.com/doc/current/doctrine.html#querying-for-objects-the-repository

3.Base de données

3.1 Modifications de la 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://db_user:db_password@127.0.0.1:3306/db_name
```

Détail configuration BDD : https://symfony.com/doc/current/doctrine.html#configuring-the-database

3.2Modification 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.

Création & Mise à jour d'une entité

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

Création des fichiers de migration

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

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

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

4.Fonctionnement de l'authentification

4.1 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.

4.2. Soumission du formulaire

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

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

4.3. Redirection de l'utilisateur

Suivant ci, la personne c'est bien authentifier ou pas. La méthode ***onAuthenticationSuccess***, la redirigera vers la dernière page qu'elle 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.

Détail création login form : https://symfony.com/doc/current/security/form_login_setup.html

5. Collaboration

Le développement de l'application doit respecter les normes de qualité PSR-4 et dois bien entendu respecter les deux contrôleurs de qualité mis en place. Les liens vers les Dashbords sont disponibles dans l'Audit.

Pour la collaboration en elle-même, le système est déjà mis en place grâce sur github. Mais je pense que la création d'une organisation sur le site serrais un plus pour le management du projet et des contributeurs.

Actuellement deux possibilités peuvent être mis en place pour la modification du code. Soit de fork le main repository ou bien d'invité le contributeur par le créateur du répository. Et ensuite, de faire les manipulations décrites sur la page 4.

Pour les bonnes pratiques de git, il faudra créer une branche par fonctionnalité a développé. Ou sinon une branche par développeur au minimum. Cela évitera les conflits de fichiers. Car ci un développeur modifie le même fichier qu'un autre il y aura un conflit, et cela pourrais