
ToDo & Co

Documentation

Kévin Mulot



1. Projet

1.1 Technologies

Todolist est une application utilisant le Framework Symfony.
Afin d'assurer la pérennité de celle-ci, la version à été mis-à-jour vers la version lts 4.4.
Une version de PHP 7.1 ou plus élevée est nécessaire pour le bon fonctionnement de l'application.

Documentation : <https://symfony.com/doc/4.4/setup.html>

1.2 Librairies

L'ensemble des librairies installées via Composer sont consultable dans le fichier `composer.json` à la racine du projet. Ces librairies sont également visibles sur le site packagist.org

Il est recommander de mettre à jour régulièrement ces librairies et de consulter leurs documentations afin d'assurer la sécurité et le bon fonctionnement de l'application.

1.3 Initialisation et contribution

Afin d'installer ToDoList, il vous sera nécessaire de consulter repository de celle-ci et de suivre les étapes indiquer dans le fichier `README.md` à la racine de celle-ci.

Pour contribuer à l'amélioration de l'application il vous faudra consulter le fichier `CONTRIBUTING.md` situé également à la racine.

Projet : <https://github.com/kevinmulot/OC-P8-Todolist/tree/main>

Readme : <https://github.com/kevinmulot/OC-P8-Todolist/blob/main/README.md>

Contributing : <https://github.com/kevinmulot/OC-P8-Todolist/blob/main/CONTRIBUTING.md>

2. Authentification

2.1 Utilisateurs

Une identification à été mis en place afin de définir l'identité des utilisateurs et de restreindre l'accès à celle-ci ou et uniquement certaines parties. Afin de s'authentifier un nom d'utilisateur (username) et un password devra être renseigner dans le formulaire d'authentification.

On distinguera trois type d'utilisateurs qui sont les suivant :

- **Utilisateur non authentifier** : Lorsqu'un utilisateur n'est pas authentifié il sera considéré comme anonyme et se verra automatiquement redirigé vers la page d'authentification s'il souhaite parcourir l'application.
- **Utilisateur normal** : Celui-ci dispose du rôle utilisateur simple (`ROLE_USER`) et ne pourra que créer ou modifier ses propres tâches. Il pourra également consulter les différentes listes de tâches.
- **Administrateur** : Cette utilisateur se voit attribué les droits les plus importants de par son rôle (`ROLE_ADMIN`). Il pourra créer et modifier un utilisateur, mais aussi modifier ses droits d'accès à l'application. Et créer, modifier et supprimer ses propres tâches ou celles ayant un auteur anonyme.

2.2 Fichiers

Voici la listes des fichiers important concernant l'authentification :

Type	Fichier	Description
Configuration	<code>config/packages/security.yaml</code>	Configuration de l'authentification et des accès
Entité	<code>src/Entity/User.php</code>	Entité utilisateur
Contrôleur	<code>src/Controller/SecurityController.php</code>	Contrôleur de connexion
Vue	<code>templates/security/login.html.twig</code>	Template du formulaire de connexion

2.3 Sécurité

La gestion de la sécurité et donc de nos utilisateur s'effectue via le fichier **security.yaml** qui contient plusieurs paramètres importants pour le bon fonctionnement de l'application.

Le mode d'encryptage du mot de passe d'un utilisateur stocké en base de données peut être sélectionnée ici :

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

Celui-ci est définie sur « auto » car il est recommandé de laisser le framework Symfony choisir la méthode d'encryptage la plus sécurisée pour sa version.

Le provider indique où aller chercher les information nécessaires à l'authentification :

```
providers:
  doctrine:
    entity:
      class: App\Entity\User
      property: username
```

Ici c'est la classe User que l'on utilise avec comme propriété le username (nom d'utilisateur) pour la connexion avec le mot de passe.

Le firewall (pare-feu) prend l'authentification en compte et restreint l'accès à certaines routes de l'application.

```
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false

  main:
    anonymous: true
    pattern: ^/
    form_login:
      login_path: login
      check_path: login_check
      always_use_default_target_path: true
      default_target_path: /
    logout: ~
```

Ces paramètres indiquent que si l'utilisateur n'est pas authentifié il se verra automatiquement rediriger vers la page de connexion . (login)

Le contrôle des accès se fait de via ces paramètres :

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

« path » définit la route accessible par l'utilisateur.

« roles » représente le niveau d'authentification nécessaire pour accéder à cette route.

On notera qu'un utilisateur non identifié n'aura accès que à la page de connexion (login) avant de se connecter et qu'on lui octroie l'accès à d'autres routes.

Documentation : <https://symfony.com/doc/4.4/security.html>

2.4 Stockage des données

Les utilisateurs sont stockés en base de données au sein de la table user représentée dans l'application par la classe `src/Entity/User.php`. Symfony permettant de créer notre base de données et nos tables uniquement grâce aux paramètres de nos entités.

```
/**
 * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
 */
class User implements UserInterface
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

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

L'implémentation de `UserInterface` est absolument nécessaire au bon fonctionnement de notre authentification.

Les paramètres de chaque donnée de notre table peuvent être modifiés à l'aide des annotations.

Voici à quoi ressemble notre table User en base de données :

	id	username	roles	password	email
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	admin	["ROLE_ADMIN"]	\$argon2id\$v=19\$m=65536,t=4,p=1\$01COeyGfVf4MCib73aX...	admin@todolist.fr
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	updated	["ROLE_USER"]	\$argon2id\$v=19\$m=65536,t=4,p=1\$vSocw6ekQs0Vb8jC6jI...	updated@todolist.fr
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	user2	["ROLE_USER"]	\$argon2id\$v=19\$m=65536,t=4,p=1\$vtCQrEUwwzmFQQtjlx...	user2@todolist.fr

Documentation : <https://symfony.com/doc/4.4/doctrine.html>

2.5 Formulaire

L'utilisateur non authentifié, comme vu précédemment, se verra redirigé en accédant à l'application, vers la page de connexion. (`templates/security/login.twig.html`)

```
/**
 * @Route("/login", name="login")
 */
public function loginAction(AuthenticationUtils $authenticationUtils)
{
    $error = $authenticationUtils->getLastAuthenticationError();
    $lastUsername = $authenticationUtils->getLastUsername();

    return $this->render( view: 'security/login.html.twig', array(
        'last_username' => $lastUsername,
        'error'         => $error,
    ));
}
```

Afin d'accéder à cette page la méthode **loginAction()** du **SecurityController** est exécutée. Celle-ci va alors générer notre vue et renvoyer des données si nécessaire.

Un formulaire est alors soumis à l'utilisateur pour poursuivre sa navigation.

Nom d'utilisateur :

Mot de passe :

Se connecter

Les identifiants, une fois renseignés, sont alors validés à l'aide de la librairie `security-bundle` après consultation de la base de données.

S'ils correspondent à un utilisateur, celui-ci se verra redirigé vers la page d'accueil. Dans le cas contraire une erreur s'affichera et il devra réitérer sa tentative.