
# Todo & Co - TodoList

## Documentation Technique

—

Héligon Sébastien
OpenClassrooms - Développeur d'application - PHP / Symfony
Projet 8 - Améliorez une application existante de Todo & Co

# Sommaire

# 1. Le Projet

## 1.1. Présentation

L'objectif fondamental de cette application TodoList est de fournir à l'entreprise ToDo & Co un outil efficace pour la gestion de ses tâches quotidiennes.

## 1.2. Technologies

L'application a été récemment mise à jour vers la dernière version de Symfony, la 6.3, afin de bénéficier des toutes dernières fonctionnalités, optimisations et sécurités offertes par ce framework de développement. Cette mise à jour a également nécessité le passage à PHP 8, la dernière version du langage de programmation, pour garantir une performance optimale et une sécurité renforcée.

Cette mise à jour vers Symfony 6.3 et PHP 8 représente un saut significatif en termes de performances, de sécurité et de fonctionnalités pour l'application. Veillez donc à suivre ces recommandations pour garantir une expérience utilisateur optimale.

# 2. Prérequis

## 2.1. Librairies

Afin de garantir le bon fonctionnement de l'application, il est essentiel d'utiliser "***Composer***", l'outil de gestion des dépendances PHP. Composer permet de gérer les bibliothèques et les packages requis par l'application, assurant ainsi une intégration fluide des composants Symfony 6.3.

## 2.2. Installation

Pour mettre en marche l'application, veuillez suivre les étapes ci-dessous

### 2.2.1. Clonage du projet

Tout d'abord, récupérez le projet à partir de GitHub en utilisant la commande suivante:

```
git clone git@github.com:heligonSeb/TodoList.git
```

### 2.2.2. Installation des dépendances

Une fois le projet récupéré, assurez-vous d'installer toutes les dépendances requises en utilisant Composer:

```
composer install
```

### 2.2.3. Création de la base de données

Pour finaliser l'installation, vous aurez besoin d'une base de données. Pour ce faire, je vous invite à créer un fichier **.env.local** basé sur le fichier **.env** déjà présent dans le projet. Après avoir configuré ce fichier en fonction de vos paramètres de base de données, exécutez les commandes suivantes pour créer la base de données, la configurer et insérer des données de test:

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

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

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

# 3. L'Authentification

## 3.1. Frontend

Pour accéder à l'application, les utilisateurs doivent passer par un formulaire de connexion situé dans le fichier "***templates/security/login.html.twig***". Avant de pouvoir se connecter, l'utilisateur doit préalablement être ajouté à la base de données par un administrateur. Cette mesure de sécurité garantit que seules les personnes autorisées peuvent utiliser entièrement l'application.

La soumission du formulaire est envoyée au "controller" ***SecurityController.php*** et à la méthode "Login" pour faire l'authentification.

L'administration des utilisateurs est réservée aux administrateurs. Ces derniers disposent de privilèges spéciaux qui leur permettent d'accéder à trois pages supplémentaires cruciales pour la gestion des utilisateurs:

➔ **Liste des utilisateurs:**

La première page, accessible via le fichier "***templates/user/list.html.twig***", affiche la liste complète des utilisateurs enregistrés dans l'application. Cette vue permet aux administrateurs d'avoir une vue d'ensemble des utilisateurs et de leurs informations.

➔ **Création des utilisateurs:**

La deuxième page, correspondant au fichier "***templates/user/create.html.twig***", est dédiée à la création de nouveaux utilisateurs. Les administrateurs peuvent utiliser ce formulaire pour ajouter de nouveaux membres à l'application en spécifiant leurs informations de connexion et leurs rôles.

➔ **Modification d'un utilisateur:**

Enfin, la troisième page, basée sur le fichier "***templates/user/edit.html.twig***", permet aux administrateurs de mettre à jour les informations d'un utilisateur existant. Cela inclut la modification de données telles que le "username", le mot de passe, ou les rôles.

## 3.2. Backend

Au niveau de la gestion de l'authentification dans le backend de l'application, un élément crucial est le fichier "***security.yaml***" situé dans "***config/packages/security.yaml***". Ce fichier revêt une importance particulière pour deux raisons majeures. Tout d'abord, il est essentiel pour configurer le mécanisme d'authentification et spécifier quelle entité et quelle propriété seront à cette fin. Dans l'application, nous avons opté pour l'entité "***User***" et la propriété "***username***" pour l'authentification:

```
providers:
   # used to reload user from session & other features (e.g. switch_user)
   app_user_provider:
       entity:
           class: App\Entity\User
           property: username
```

La première partie de cette configuration permet de déterminer comment l'application va identifier les utilisateurs, en utilisant l'entité "***User***" et en considérant le champ "***username***" comme identifiant unique. Cette étape est cruciale pour assurer la sécurité et la personnalisation de l'authentification.

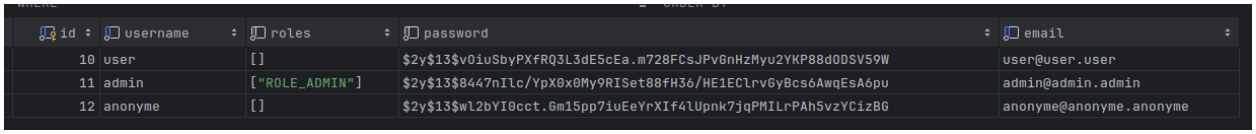
La deuxième composante essentielle de ce fichier concerne la gestion des accès aux différentes pages de l'application en fonction des rôles des utilisateurs. Cela signifie que nous pouvons spécifier quelles pages et quelles parties du site sont accessibles à certains types d'utilisateurs. Cette configuration de base est basée sur les lignes de code suivantes:

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

Dans notre application, nous avons mis en place des règles strictes pour que l'accès aux pages liées à la gestion des utilisateurs soit réservé aux administrateurs (ROLE_ADMIN), tandis que la page liée à la création d'une tâche ne soit accessible qu'aux utilisateurs enregistrés (ROLE_USER) car une tâche est forcément liée à un utilisateur .

Cette approche garantit une sécurité granulaire et un contrôle d'accès pour chaque utilisateur en fonction de son rôle. Ainsi, l'application peut garantir un niveau élevé de sécurité tout en permettant une expérience utilisateur personnalisée. Cette configuration associée à l'authentification basée sur l'entité "***User***" et la propriété "***username***", contribue à créer un environnement sécurisé et adapté aux besoins de l'application.

Comme nous l'avons expliqué précédemment, pour devenir un utilisateur de l'application, il est impératif d'être ajouté préalablement par un administrateur via un formulaire spécifique. La soumission de ce formulaire est gérée par le "controller" "***UserController***" et sa méthode "***create***", qui assure l'enregistrement de l'utilisateur dans la table "***user***" de la base de données.



| id | username | roles | password | email |
|---|---|---|---|---|
| 10 | user | [] | $2y$13$vOiuSbyPXfRQ3L3dE5cEa.m728FCsJPvGnHzMyu2YKP88dODSV59W | user@user.user |
| 11 | admin | ["ROLE_ADMIN"] | $2y$13$8447nIlc/YpX0x0My9RISet88fH36/HE1EClrvGyBcs6AwqEsA6pu | admin@admin.admin |
| 12 | anonyme | [] | $2y$13$wl2bYI0cct.Gm15pp7iuEeYrXIf4lUpnk7jqPMILrPAh5vzYCizBG | anonyme@anonyme.anonyme |

Une fois que l'utilisateur est enregistré, Symfony nous offre des outils puissants pour interagir avec la base de données. Par exemple, pour récupérer une liste de tous les utilisateurs, il suffit d'une seule ligne de code:

```
$entityManager->getRepository(User::class)->findAll();
```

Cette simple instruction équivaut à exécuter une requête SQL "***SELECT * FROM user***" en MYSQL. Symfony prend en charge la conversion des données de la base de données en objets PHP, ce qui simplifie considérablement la manipulation des données.

Lorsque vous souhaitez modifier les informations d'un utilisateur, Symfony facilite également cette tâche. Il vous suffit de spécifier l'ID de l'utilisateur dans l'URL, et dans les paramètres de la méthode du contrôleur, déclarez une variable de type "**User**" (par exemple "**$user**"). Symfony récupérera automatiquement l'utilisateur correspondant à l'ID fourni et le placera dans la variable "**$user**", qui peut ensuite être utilisée pour effectuer les modifications nécessaires. Voici un exemple tiré de l'application:

```
#[Route('/users/{id}/edit', name: 'user_edit')]
public function edit(User $user, Request $request,
UserPasswordHasherInterface $passwordHasher, EntityManagerInterface
$entityManager): Response
```

Dans cet exemple, on définit une route permettant l'édition d'un utilisateur avec un ID spécifique. Symfony prend en charge l'extraction de l'utilisateur associé à cet ID et le passe automatiquement à la méthode "***edit***". Cela simplifie considérablement la gestion des opérateurs CRUD (Create, Read, Update, Delete) dans l'application, garantissant une expérience de développement fluide et efficace.

En résumé, Symfony offre une gestion efficace et élégante des opérations liées aux utilisateurs, allant de la création à la modification, grâce à son système de contrôle d'accès et à son intégration transparente avec la base de données. Cela permet de gagner du temps et de garantir une manipulation sécurisée des données des utilisateurs. Je vous invite à consulter la documentation sur le site de Symfony pour plus d'informations.

https://symfony.com/doc/current/index.html

# 4. Collaboration

Pour entamer une collaboration, je vous invite à visiter le dépôt ***GItHub*** du projet et à consulter le fichier dédié à la collaboration, accessible depuis la racine du dépôt. Vous pouvez accéder directement à ce fichier en suivant ce lien :
https://github.com/heligonSeb/TodoList/blob/main/contribution.md

Il existe deux principales méthodes pour contribuer au projet, demander une invitation pour collaborer directement ou réaliser un fork du dépôt. Voici Comment procéder pour chacune des ces options:

**Option 1: Demander une invitation pour collaborer**

Si vous souhaitez collaborer étroitement avec notre équipe de développement, vous pouvez demander une invitation pour devenir un collaborateur officiel du projet. Cela signifie que vous aurez des droits de modification directs sur le dépôt.

- Consultez d'abord le fichier de collaboration pour comprendre les directives et les normes à suivre pour contribuer au projet de manière efficace et cohérente.
- Ensuite, sur la page GitHub du projet, cliquez sur l'onglet "**Issues**" pour consulter les problèmes ouverts , les demandes de fonctionnalités, etc... Choisissez une tâche sur laquelle vous souhaitez travailler
- Commentez cette tâche pour indiquer votre intention de contribuer. On examinera votre demande et si elle est approuvée nous vous enverrons une invitation pour collaborer.

**Option 2: Effectuer un fork du projet**

Si vous préférez contribuer de manière indépendante ou si vous ne souhaitez pas demander une invitation, vous pouvez créer un fork du dépôt. Un fork est essentiellement une copie du dépôt principal que vous pouvez modifier à votre guise. Voici les étapes:

- Consultez toujours le fichier de collaboration pour comprendre les directives et les normes de contribution
- Sur la page GitHub du projet, cliquez sur le bouton "**Fork**" en haut à droite. Cela créera une copie du dépôt sur votre propre compte GitHub.
- Clonez votre fork sur votre machine locale en utilisant la commande "***git clone***"

```
git clone https://github.com/VOTRE_NOM_UTILISATEUR/TodoList.git
```

- Effectuez vos modifications, puis réalisez un "**commit**" et poussez-les vers votre Fork.
- Ensuite, sur GitHub, ouvrez une "**Pull Request**" depuis votre fork vers le dépôt principal. On pourra ainsi examiner vos modifications et les fusionner dans le dépôt principal si elles sont approuvées.

Quelle que soit l'option que vous choisissez, il est crucial de respecter les normes et les processus de contribution du projet. Cela garantira une collaboration efficace et harmonieuse avec notre équipe de développement et contribuera à la qualité globale du projet.