Utilisation de l’application

Qu'est-ce que ToDoList

Todolist est une application permettant de gérer ses tâches quotidiennes.

C'est une application web sous le framework symfony. Il y est possible de créer, modifier et supprimer des tâches. Aussi les valider ou non.

Des utilisateurs sont aussi présents sur l'application, un compte utilisateur comporte la liste des taches créer par celui-ci. Excepter pour un utilisateur administrateur qui lui peut accéder à toute les taches.

Il est donc possible de créer un utilisateur et d'en choisir le rôle.

Comment ça marche

Symfony

Symfony est un framework web permettant notamment de séparer le code et coder avec cohérence et propreté.

> *« Symfony est un ensemble de composants PHP, un framework d'applications Web, une philosophie et une communauté – tous travaillant en harmonie. »*

Symfony fonctionne sur le design patern MVC (Model Vue Controller), comme son nom l'indique cette manière de coder permet de séparer le model (les entités à persister), la vue (tout ce qui s'affiche à l'écran) et les controller (gère les actions de l'application)

Pour plus d'information sur symfony :

https://symfony.com/

Authentification

ToDoList comporte une gestion d'utilisateur, pour rappel chaque utilisateur doit pouvoir être créer, doit pouvoir être modifiable.

De plus un utilisateur n'a pas tous les droits sur l'application, il ne doit avoir accès seulement à ses taches.

Comment s'opère l'authentification

L'authentification de ToDoList est mise en place grâce à Guard.Guard est un composant qui a été intégré à Symfony en septembre 2015, il a été déjà développé par KnpUniversity.

Il s'agit du composant Guard intégré au module de sécurité de la version 2.8. Il vient résoudre le problème majeur du système d'authentification natif en centralisant et standardisant la gestion de tout le processus dans une seule classe PHP appelée « Authenticator ». Cette classe implémente l'interface GuardAuthenticatorInterface contenant uniquement 7 méthodes à implémenter :

- *getCredentials* : Permet d'extraire les identifiants de la requête.
- *getUser* : Permet de récupérer un utilisateur en fonction de ses identifiants.
- *checkCredentials* : Permet de vérifier les identifiants.
- *createAuthenticationToken* : Permet de créer le token d'authentification.
- *onAuthenticationFailure* : Retourne un objet Response dans le cas où l'authentification échoue.
- *onAuthenticationSuccess* : Retourne un objet Response dans le cas où l'authentification réussit.
- *supportsRememberMe* : Permet de spécifier si la stratégie d'authentification supporte la fonctionnalité « Se souvenir de moi ».

https://symfony.com/doc/current/security/guard_authentication.html

Où sont stockés les utilisateurs

Doctrine est un ORM (couche d'abstraction à la base de données) pour PHP.

Un ORM est une technique de programmation faisant le lien entre le monde de la base de données et le monde de la programmation objet. Elle permet de transformer une table en un objet facilement manipulable via ses attributs.

Doctrine est l'ORM par défaut du framework symfony, De plus, il peut être utilisé avec de nombreux autres frameworks tels que Zend Framework, CodeIgniter, FLOW3 ou encore Lithium.

Doctrine fait donc gagner énormément de temps et de stabilité (moins de bugs que les classes faites par soi-même) en générant automatiquement les classes à partir du schéma de la base de données, avec les bonnes méthodes.

Les utilisateurs sont stockés dans une base de données sous forme d'entités. Chaque utilisateur est unique via son id.

Rejoindre le projet

Procéder

En tant que nouveau développeur sur ce projet il vous faut vous adapter à la manière de développer l'application.

Afin de commencer à développer il vous faut avoir un compte gitHub. Il vous faut développer sur une branche afin de pouvoir développer en communauté. De plus pour versionner le code il est nécessaire de faire des commits à chaque ajout/modification/suppression de fonctionnalités.

Enfin pour rassembler des branches il faut faire une pull-request et attendre la validation ou non si des modifications sont nécessaires.

Afin de ne pas vérifier à la main le code et permettre toute modification vérifiable facilement. Veilliez à faire des tests de vos fonctionnalités, document pouvant aider :

https://symfony.com/doc/current/testing.htm