Todolist – Documentation technique

Environnement du projet

Le language de programmation utilisé est PHP (version 8.1.3 aujourd'hui)

Le projet a été réalisé avec le framework Symfony (version 6.3 aujourd'hui).

Base de donnée Mysql.

L'ORM utilisé est Doctrine (inclus dans Symfony).

Vous pouvez regarder la documentation officielle de Symfony pour apprendre comment utiliserle framework pour créer des routes, utiliser les assertions, les formulaires, lesmigrations, créer des templates beaucoup d'autres fonctionnalitées.

Pour installer le projet je vous suggère de lire le README.MD.

Connection - deconnection

Les utilisateurs sont stockés en base de donnée.

L'entité relative aux utilisateurs est "\src\Entity\User.php".

Les routes de connection et deconnection "\src\Controller\LoginController.php" fichier.

Les deux viennent du framework Symfony.
Si vous voulez mettre à jour les url ou nom de route , vous devrez remplacer "\config\packages\security.yaml" dans "firewalls" -> "main" et "form_login" ou "logout" et également dans le controller.

Si vous avez besoin de plus d'informations, vous pouvez lire la documentation de Symfony: https://symfony.com/doc/6.3/security.html

Test

Le projet utilise le framework PhpUnit.
La documentation Symfony relative a phpunit :
https://symfony.com/doc/current/testing.html

Les fichiers de tests sont localisés dans le dossier "\tests\" dans le project.
L'architecture de tous les dossiers et fichiers doit être identique au dossier « \src\ ».

Les fichiers de test doivent avoir un suffix "Test" juste avant l'extension.

Par exemple :

Pour tester le fichier “\src\Controller\TaskController.php",

Vous devrez créer un fichier “\tests\src\Controller\TasksControllerTest.php”.

Aujourd’hui, le code coverage est a plus de 98%.

La configuration de test se fait dans le fichier “phpunit.xml” (à la racine du project).

Les noms des méthodes de test doivent commencer pa rle mot clé “test” et elle doit etre public (exemple : public function testMethodSuccess()).

Bundles et librairies additionnelles

Utiliser la commande de terminal “composer install” pour installer tous les bundles et librairies.

Ils sont situés dans le dossier “\vendor\”.

Les bundles sont listés dans le fichier “\config\bundles.php”.

Les librairies et les bundles sont visibles dans le fichier “\composer.json”.

Qualité du projet

Sur le repository Github, SymfonyInsight lance une analyse a chaque push.

Aujourd’hui, le rang atteint est Platinum, soyez sur de garder un bon niveau de rang en résolvant les suggestions lorsqu’il y en a.

Le projet utilise les librairies phpstan & cs fix pour empêcher d’introduire des erreurs et des format de code non conforme.

Un hook pre-push est configure et il déclenche une analyse phpstan, une analyse cs fix, et lance tous les tests unitaires pour empêcher toute regression.

Vous pouvez trouver toutes ces commandes pour lancer ces analyse dans le fichier README.MD.

Essayez, du mieux que vous le pouvez, de respecter les bonnes pratiques:
-Don’t repeat yourself (eviter la duplication de code)

-Le code doit etre lisible (nommage, complexité …)

-Suivez les principes SOLID (utiliser les interfaces, services …)

-Ajouter des tests quand vous créer du nouveau code (considerez l'approche test driven development)

-Gardez la logique coté back (fichier php), et pas dans les templates

Version 1.1 – Date: 13-12-2023

Commandes utiles

Installer un certificat (https)	*symfony server:ca:install*
Lancer le server local	*symfoy serve*
Installer les dépendences back-end	*composer install*
Installer les dépendences front-end	*npm install*
Créer la base de données	*symfony console doctrine:database:create* ou *symphony console d:d:c*
Créer une migration	*symfony console make:migration*
Jouer les migrations	*symfony console doctrine:migrations:migrate* ou *symfony console d:m:m*
Consulter le statut des migrations	*symfony console doctrine:migrations:status* ou *symfony console d:m:st*
Supprimer la base de données	*symfony console doctrine:database:drop –force* ou *symfony console d:d:d --force*
Vérifier la validité du schema	*symfony console doctrine:schema:validate*
Créer la base de données de test	*APP_ENV=test symfony console doctrine:database:create*
Lancer les migrations sur la base de données de test	*APP_ENV=test symfony console doctrine:migrations:migrate*
Supprimer la base de données de test	*APP_ENV=test symfony console doctrine:database:drop --force*
Créer la base de données de test ET jouer les migrations	*composer database:test:create*
Lancer les fixtures	*symfony console doctrine:fixtures:load* ou *symfony console d:f:l*
Lancer les tests unitaires	*vendor/bin/phpunit* ou *npm run tests*

Lancer les tests unitaires avec code coverage	*vendor/bin/phpunit --coverage-html public/test-coverage*
Lancer les tests unitaires d'un seul fichier	*vendor/bin/phpunit <chemin_vers_le_fichier>*
Lancer un test unitaire d'un fichier	*vendor/bin/phpunit –filter <nom_du_test> <chemin_vers_le_fichier>*
Lancer le cs fixer	*tools/php-cs-fixer/vendor/bin/php-cs-fixer fix src* ou *composer cs-fix* ou *npm run csfixer*
Lancer phpstan	*vendor/bin/phpstan analyse src tests* ou *composer run-phpstan* ou *npm run phpstan*
Consulter le statut git	*git status*
Ajouter un fichier (ou plusieurs) git	*git add <fichier>* ou *git add . (pour tout ajouter)*
Ajouter un commit	*git commit -m « Message du commit »*
Envoyer les changements	*git push origin <nom_de_la_branche>* ou *git push*
Récupérer les changements	*git pull origin <nom_de_la_branche>* ou *git pull*
Changer de branche	*git checkout <nom_de_la_branche>*
Créer une branche	*git checkout -b <nom_de_la_nouvelle_branche>*
Récupérer les branches	*git fetch*
Envoyer les changements en ne jouant pas le hook pre-push (husky) (pas recommandé)	*git push origin <nom_de_la_branche> --no-verify*