Todolist – Technical Documentation

Project environment

Programmation language is PHP (version 8.1.3 at this moment)

The project was made with the framework Symfony (version 6.3 at this moment).

MySql database.

The ORM is Doctrine (included in Symfony).

You can see the Symfony official documentation to learn how to use the framework to create route, assertions, form, migrations, templates and many other features.

To install the project I suggest you to read the README.MD.

Login – Logout

Users are stored in the database.

The Entity related to the users is “\src\Entity\User.php”.

The login and logout routes are in the “\src\Controller\LoginController.php” file.

Both of them comes from Symfony framework.
If you want to update the route url or name, you must replace it in the “\config\packages\security.yaml” in “firewalls” -> “main” then “form_login” or “logout” and also in the controller.

If you need more information, you can read the Symfony’s documentation here:
https://symfony.com/doc/6.3/security.html

Test

Project use the PhpUnit framework.
Symfony documentation for test with phpunit :
https://symfony.com/doc/current/testing.html

Tests files are located in the “\tests\” folder in the project.
All folder and files locations must be a copy of the “\src\” folder.

Tests files must have the suffix “Test” just before the extension.

For example :

To test the “\src\Controller\TaskController.php” file,

You must create a “\tests\src\Controller\TasksControllerTest.php” file.

At this moment, the code coverage is up to +98%.

Test config is in “phpunit.xml” file (root of the project).

Test methods names must began with the keyword “test” and be public (example : public function testMethodSuccess()).

Additionnal Bundles and Libraries

Use the terminal command “composer install” to install all bundles and libraries.

They are stored in the “\vendor\” folder.

Bundles are listed in the “\config\bundles.php” file.

Libraries and bundles appear in the “\composer.json”.

Project quality

On the Github repository, SymfonyInsight launch an analyze for every push.

At this moment, rank is at Platinum, be sure to keep a good rank by solving the suggestions when there is.

Project use the libraries phpstan & cs fix to prevent error and wrong format code.

A pre-push hook is configured and it launch a phpstan analyze, a cs fix analyze, and run all the tests to prevent regression.

You can find all the terminal command to run theses analyzes in the README.MD file.

Try, the best you can, to respect good practices:
-Don’t repeat yourself (avoid code duplication)

-Code must be lisible (naming, complexity …)

-Follow SOLID principles (use interfaces, services …)

-Add tests when you write new code (consider test driven development)

-Keep logic in back side (php files), not in templates

Usefull commands

Install certificat (https)	*symfony server:ca:install*
Launch the local server	*symfoy serve*
Install back-end dependencies	*composer install*
Install front-end dependencies	*npm install*
Create the database	*symfony console doctrine:database:create* or *symphony console d:d:c*
Create a migration	*symfony console make:migration*
Run the migrations	*symfony console doctrine:migrations:migrate* or *symfony console d:m:m*
See migrations status	*symfony console doctrine:migrations:status* or *symfony console d:m:st*
Delete the database	*symfony console doctrine:database:drop –force* or *symfony console d:d:d --force*
Check schema validity	*symfony console doctrine:schema:validate*
Create test database	*APP_ENV=test symfony console doctrine:database:create*
Run the migrations for test database	*APP_ENV=test symfony console doctrine:migrations:migrate*
Delete the test database	*APP_ENV=test symfony console doctrine:database:drop --force*
Create test database AND run migrations	*composer database:test:create*
Run the fixtures	*symfony console doctrine:fixtures:load* or *symfony console d:f:l*
Run unit tests	*vendor/bin/phpunit* or *npm run tests*
Run unit, tests with code coverage	*vendor/bin/phpunit --coverage-html public/test-coverage*
Run unit tests for a test file only	*vendor/bin/phpunit <chemin_vers_le_fichier>*
Run one unit test in a test file	*vendor/bin/phpunit –filter <method_namet> <path_file>*

Run the cs fixer	*tools/php-cs-fixer/vendor/bin/php-cs-fixer fix src* or *composer cs-fix* or *npm run csfixer*
Run phpstan	*vendor/bin/phpstan analyse src tests* or *composer run-phpstan* or *npm run phpstan*
Check git status	*git status*
Add one or more updated file(s)	*git add <file>* or *git add . (to add allr)*
Add commit	*git commit -m « Commit message »*
Send changes	*git push origin <branch_name>* or *git push*
Get changes	*git pull origin <branch_name>* or *git pull*
Switch branch	*git checkout <branch_name>*
Create a new branch	*git checkout -b <new_branch_name>*
Get all branches	*git fetch*
Send changes skipping the pre-push hook (husky) (not recommanded)	*git push origin <branch_name> --no-verify*
Compile at every update of css files	*npm run watch*
Compile, regroup and optimize css files	*npm run build*