

Audits de code et de performance

Projet 8 de la formation OpenClassrooms Développeur d'applications PHP/ SYMFONY



























I. Mise en place des tests

Les tests sont implémentés avec **PHPUnit** puis on valide le comportement de l'application

Pour consulter le résultat des tests unitaires, ouvrez dans votre navigateur le fichier :

```
http://{SERVER_NAME}/test-coverage/index.html
```

En voici un aperçu :

C:\Users\smr\MesProjets\P8\projet8-TodoList\src / (Dashboard)									
	Code Coverage								
		Lines			Functions and Methods			Classes and Traits	
Total		81.53%	128 / 157		83.64%	46 / 55		71.43%	10 / 14
Controller		100.00%	56 / 56		100.00%	10 / 10		100.00%	4 / 4
DataFixtures		50.00%	2 / 4		66.67%	2 / 3		50.00%	1 / 2
Entity		100.00%	42 / 42		100.00%	27 / 27		100.00%	2 / 2
Form		100.00%	12 / 12		100.00%	2 / 2		100.00%	2 / 2
Repository		0.00%	0 / 19		0.00%	0 / 7		0.00%	0 / 2
Security		100.00%	7 / 7		100.00%	2 / 2		100.00%	1 / 1
Kernel.php		52.94%	9 / 17		75.00%	3 / 4		0.00%	0 / 1

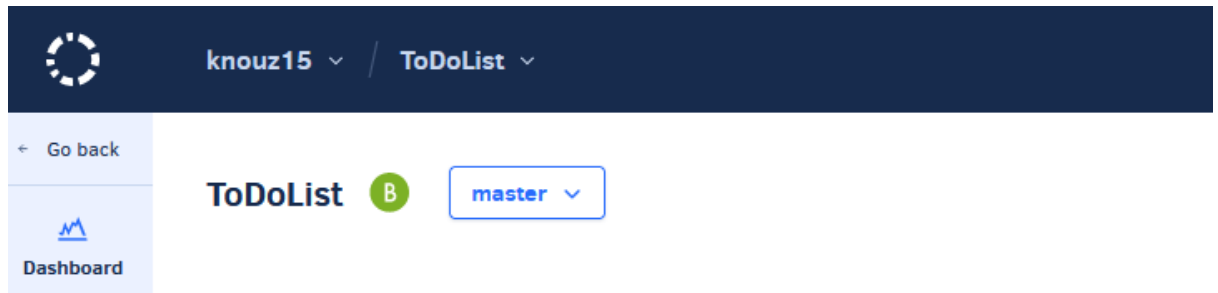
Conclusion : Le taux de couverture dépasse les 70 %.

II. Code quality

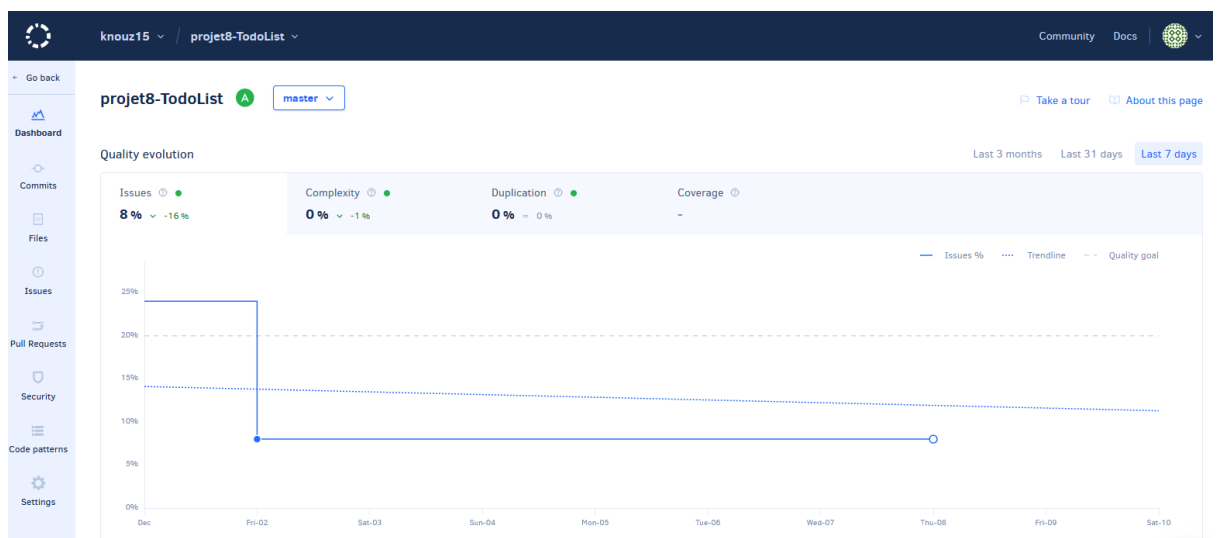
Deux choix optés pour analyser le code de l'application : Codacy et SymfonyInsight.

1. CODACY

Codacy report état initial de l'application:



Codacy report après Amélioration:




BILAN : Obtention du Badge A, le meilleur grade de cet outil, suite aux améliorations apportées à l'application.

Dashboard Codacy de l'application:

<https://app.codacy.com/gh/knouz15/projet8-ToDoList/dashboard>


Analyse SymfonyInsight Etat initial de l'application:

 benknz /
projet8-ToDoList #1

Analyzed 2 months ago by benknz, duration: a minute

Edit configuration

Analyze

 33/100

1.5 days
to get the Platinum Medal


Severity
1 Critical
37 Major
4 Minor
1 Info

Risk
4 Productivity
36 Reliability
3 Reputation

Developer
35 Saro0h
3 Knouz15

Stats
Lines of code: 2,216
Nb of suggestions: 43

Last commit
[Merge pull request #4 from knouz15/UpdatingRecip](#)
ee by knouz15 2 months ago. master

 No medal

Changes: +43 suggestions 1 critical 37 major 4 minor 1 info

SymfonyInsight detected 9 upgrade issues in your project.
Upgrade your plan to see the details and how to upgrade

Suggestions (43) Fixed Ignored Stats

► Your project must use a custom favicon instead of the default one

Read doc Ignore all Reputation Critical

► Your project should not use invalid return types 29

Read doc Ignore all Reliability Major

► Your project should not use invalid parameter and return typehints 7

Read doc Ignore all Reliability Major

► Your project should not contain "FIXME" comments

Read doc Ignore all Productivity Major

► Your project should not contain commented code 2

Read doc Ignore all Productivity Minor

► Your project should not use an .htaccess file

Read doc Ignore all Reputation Minor


► Web applications should contain a site.webmanifest file

Read doc Ignore all Reputation Minor


► Text files should end with a valid new line character.

Read doc Ignore all Productivity Info

Analyse SymfonyInsight après Amélioration:

 SymfonyInsight


Dashboard Builds Docs What we cover Pricing Account

 benknz /
projet8-ToDoList #46

Analyzed 6 days ago, duration: a minute


Edit configuration

Analyze



Stats
Lines of code: 9,192
Nb of suggestions: 0


Last commit
[Effacer qqs fichiers](#) by knouz15 6 days ago.


 No medal


Changes: -1 suggestion -1 critical 0 major 0 minor 0 info

No suggestion Fixed (1) Ignored (3) Stats

Outstanding!
Despite a very thorough analysis, SymfonyInsight couldn't find a single suggestion in the whole codebase of this project.
This is very rare, and is worthy of the Platinum medal.
Congratulations to all the developers of this project for such a high quality!

 knouz15
25 commits

 knouz15
21 commits

 Saro0h
2 commits

BILAN : Obtention de la médaille Platinum, le meilleur grade de cet outil, suite à la mise à jour de Symfony et aux améliorations apportées à l'application.
Beaucoup de soucis ont été réglés.

Dashboard SymfonyInsight de l'application:

<https://insight.symfony.com/projects/7a29dd28-b529-4a40-aa4f-cb4d5e780faf>

III. Analyse et optimisation de performance

1. Les optimisations de performance

Les correctifs sont effectués en suivant la [Documentation Symfony](#) :

-Vider le conteneur de services dans un seul fichier :

```
# config/services.yaml

parameters:
    # ...
    container.dumper.inline_factories: true
```

-Configurer OPcache PHP:

```
;php.ini

[opcache]
opcache.preload=C:\Users\smr\MesProjets\P8\projet8-TodoList\config\preload.php
; required for opcache.preload:
opcache.preload_user=www-data

; maximum memory that OPcache can use to store compiled PHP files
opcache.memory_consumption=256

; maximum number of files that can be stored in the cache
opcache.max_accelerated_files=20000

opcache.validate_timestamps=0
```

-Configurer le realpath cache PHP :

```
;php.ini

[PHP]
; maximum memory allocated to store the results
realpath_cache_size=4096K

; save the results for 10 minutes (600 seconds)
realpath_cache_ttl=600
```

-Commande qui a optimisé l'autoloader (le chargeur automatique Composer):

`$ composer dump-autoload --no-dev --classmap-authoritative`

Elle sert à mettre en cache les classes utiles à l'application

2. Blackfire

Le service Blackfire collecte tous les appels d'une fonction dans un seul nœud et montre les fonctions, exécutées par le script PHP analysé, les moins performantes (les plus consommatrices de ressources matérielles, les plus coûteuses) en haut de la liste

En observant le bloc "Métriques" du service Blackfire.io, on distingue :

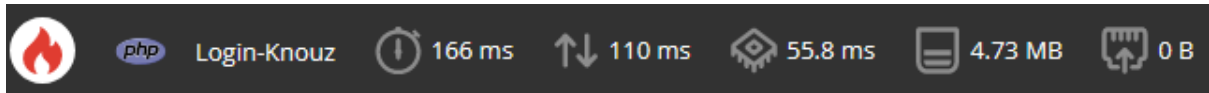
- le temps de génération de la page en PHP, découpé en temps d'accès au processeur et temps d'accès au système de fichiers ;
- la mémoire utilisée, en mégabytes ;
- l'utilisation du réseau.

Ce qui nous permet d'analyser des pages de notre site

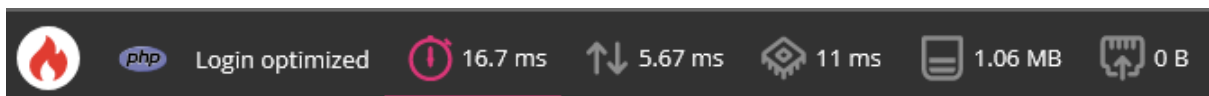
3. L'audit de performance Blackfire avec des données de tests

La page d'authentification :

- Avant Optimisation

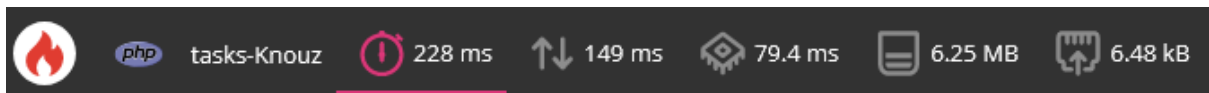


- Après Optimisation



La page tasks :

- Avant Optimisation

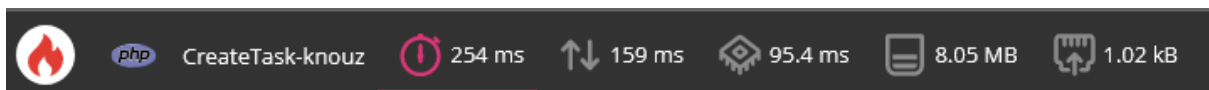


- Après Optimisation

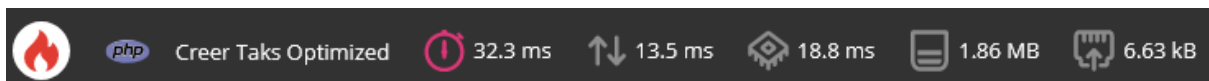


La page create task :

- Avant Optimisation

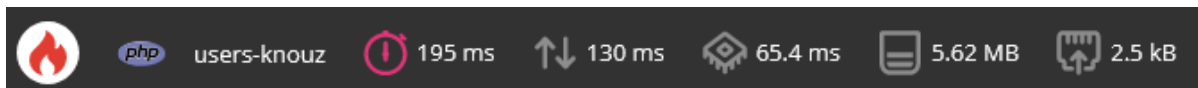


- Après Optimisation



La page users :

- Avant Optimisation



- Après Optimisation



La page create user :

- Avant Optimisation



- Après Optimisation



BILAN DE COMPARAISON:

On n'observe pas de régressions entre les deux analyses, mais plutôt une baisse considérable notamment dans les métriques suivantes :

- Le temps d'exécution
- La mémoire vive consommée

Remarque :

Ces analyses sont faites avec la licence gratuite de Blackfire, ce qui limite l'analyse du site. En mode Premium, des recommandations et indices sont disponibles pour nous aider à résoudre les problèmes de performance de l'application comme connaître exactement les requêtes SQL exécutées pour pouvoir les réduire. On pourra aussi écrire des tests qui seront exécutés pendant une analyse de la performance pour contrôler cette dernière dans le temps.

Conclusion

Cette application peut être optimisée avec des solutions telles :

- Refactorisation du code permettant de gagner des lignes de codes et donc du temps de chargement
- Réduction des requêtes SQL
- Allongement de la durée de vie du cache et mise en place de cookies permettant une optimisation du chargement lors de visites répétées.
- Mettre à jour régulièrement les librairies utilisées et supprimer celles qui ne sont plus utiles
- Maintenir un code de qualité, performant et propre et réduire les duplications
- Le choix du serveur et de l'hébergeur pour proposer un site réactif et optimal, autant sur la connexion que sur la configuration
- Evaluer les régressions entre les analyses de performance en écrivant des tests efficaces qui dépendent de l'évolution de l'application (non pas des métriques) et de se reposer sur les causes réelles qui entraînent l'évolution de la performance (non pas sur le temps ou la mémoire)