Audit de qualité & performances

---

1. Contexte

1.1. Présentation

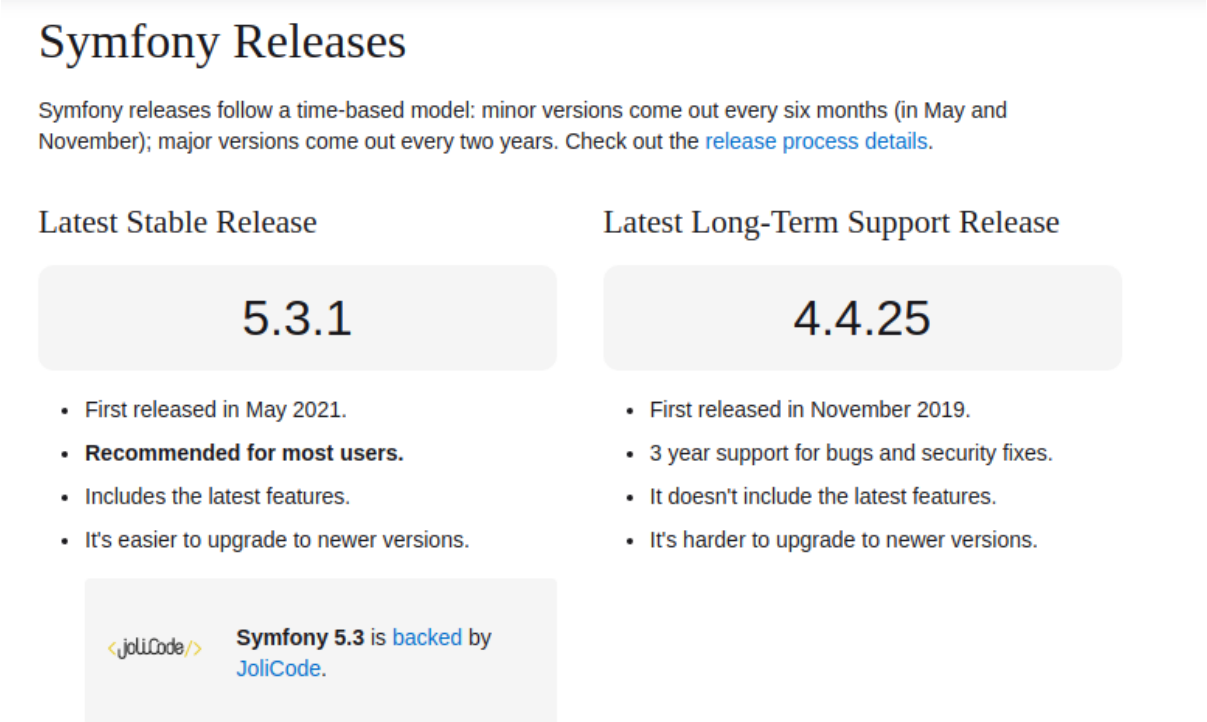
L'application ToDolist a dû être développée à toute vitesse pour permettre de montrer à de potentiels investisseurs que le concept est viable.
Le but de cet audit est d'améliorer la qualité du code et les performances de l'application.

1.2. Urgence de version

L'application a un besoin urgent d'être migrée vers une version plus récente de la solution par souci de maintenance et de sécurité.
En effet, la version 3.1 de Symfony n'est plus maintenue, et la reprise du projet est complexifiée car il y'a beaucoup de dépendances dépréciées.

J'ai fait le choix de proposer une migration de la solution vers la version 5.3 de Symfony.
Ce n'est pas une version LTS, mais c'est une version qui est maintenue jusqu'en 2022, et qui précède la prochaine version LTS (5.4). Il sera donc plus simple de migrer d'une version mineure, de la 5.3 vers la 5.4. LTS

https://symfony.com/releases



1.2.1. Application avant migration

Symfony 3.1.10
php 5.5.9
doctrine-bundle 1.6
doctrine-orm 2.5

1.2.2. Application après migration

Symfony 5.2.10
php "^7.2||^8.0"
doctrine-bundle 2.3.2
doctrine-orm 2.9.2

1.3. Évolutions techniques et corrections des fonctionnalités

J'ai réalisé un document exposant point par point les corrections effectuées après avoir repris le projet, mais aussi les évolutions techniques, mais aussi relatives à la sécurité, etc.

Le fichier se nomme **evolutions_et_correctifs.pdf** et se situe dans le répertoire **/docs** à la racine du projet.

2. Analyse de performance

3. Configuration

3.1. Environnement de production

On passe en environnement de production pour être au plus près de l'environnement final et pour gagner en performances. En effet, la barre de debug Symfony présente en environnement de développement diminue les performances de l'application.

4. Optimisations recommandées (gain de performance considérable)

https://symfony.com/doc/current/performance.html#performance-checklists

4.1. Activation du cache PHP (OPCache)

https://symfony.com/doc/current/performance.html#performance-checklists

4.2. Optimisation de l'auto-loading de Composer.

https://getcomposer.org/doc/articles/autoloader-optimization.md

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

4.2.1. Level 1 Class map generation

Dans **composer.json,**

```
"config": {
      "optimize-autoloader": true,
```

4.2.2. Level 2

A. 4.2.2.1. Authoritative class maps

Dans **composer.json,**

```
"config": {
  "classmap-authoritative": true,
```

B. 4.2.2.2. APCu cache

Dans **composer.json,**

```
"config": {
  "apcu-autoloader": true
```

4.3. Recommandations de Blackfire

Dans le fichier **.blackfire.yaml** à la racine du projet, on insère le code suivant

```
recommendations:
    php.symfony.avoid_dot_env_parsing:
        enabled: false
    php.php_preloading:
        enabled: false
    php.dump_composer_autoloader_classmap:
        enabled: false
```

- On désactive le parsing du fichier. env en production
- On désactive php preloading
- On désactive classmap dump

5. Analyse de performance à l'affichage

J'ai analysé la performance de chaque page à l'aide de l'outil Blackfire sur les points suivants (onglets de gauche à droite)

- rapidité d'exécution du PHP pour interpréter le code et afficher la page
- correspond au temps d'exécution des opérations de la base de données etc.
- nous montre le temps d'exécution du CPU du serveur pour effectuer ses opérations
- mémoire consommée par PHP

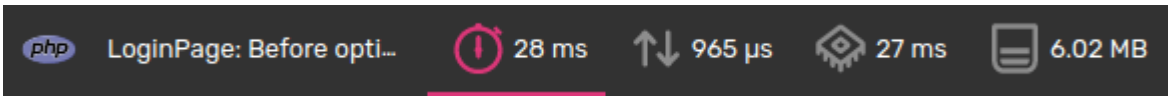
L'analyse a été effectuée 2 fois, tout d'abord dès la reprise du projet, puis après avoir travaillé sur la migration et la correction des fonctionnalités, et après avoir appliqué les optimisations citées au début du document.

J'ai ensuite effectué un comparatif des performances avant et après.

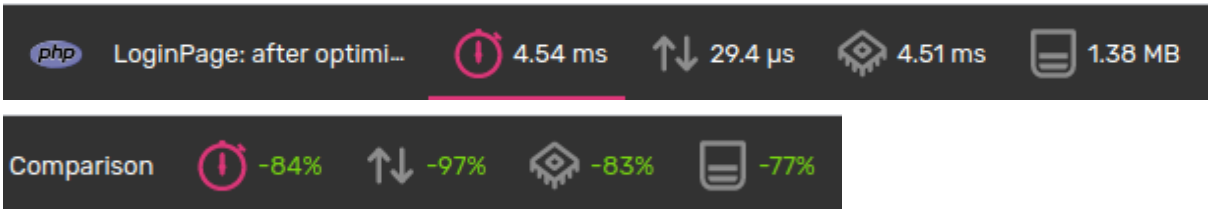
Nous pouvons voir des améliorations significatives sur pratiquement tous les points et toutes les pages et soumissions de formulaires.

5.1. Login page

5.1.1. Avant optimisations

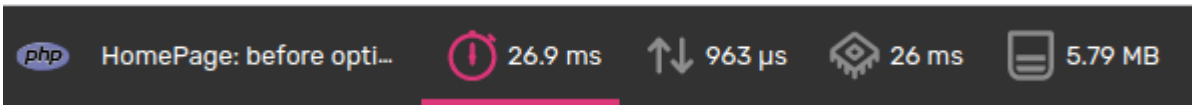


5.1.2. Après optimisations

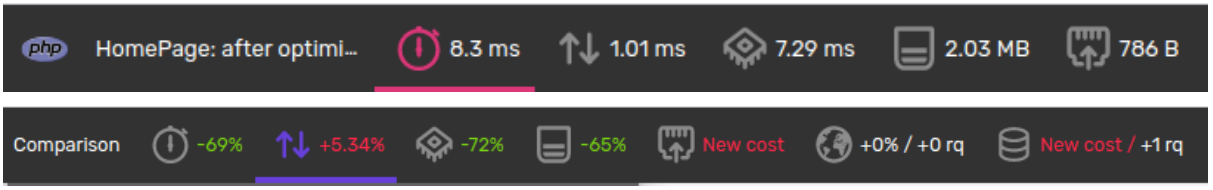

LoginPage: after optimi...
4.54 ms
29.4 µs
4.51 ms
1.38 MB
Comparison
-84%
-97%
-83%
-77%

5.2. Home page

5.2.1. Avant optimisations

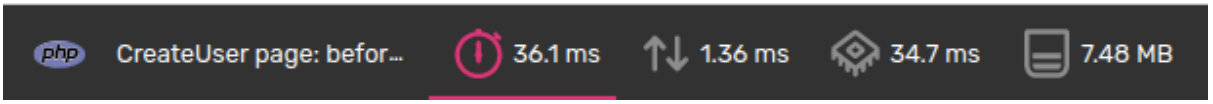


5.2.2. Après optimisations

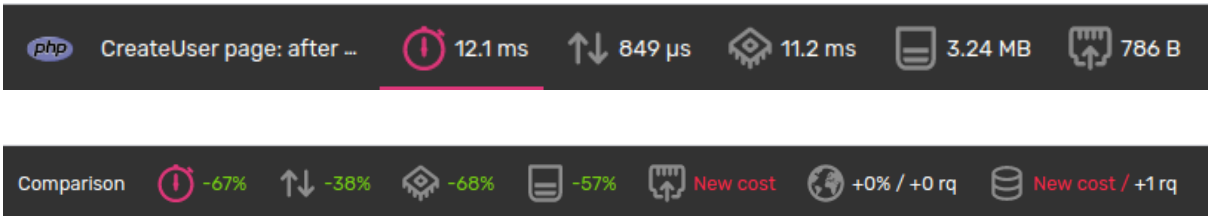


5.3. Create User page

5.3.1. Avant optimisations



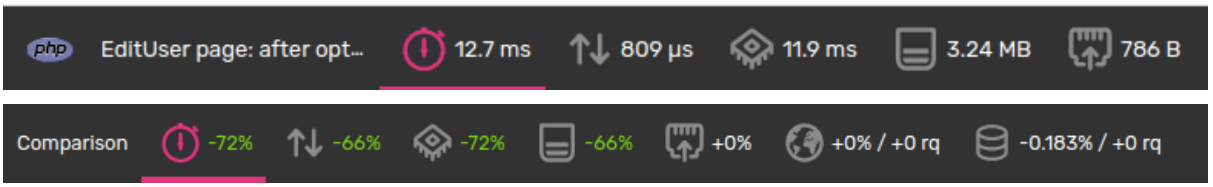
5.3.2. Après optimisations



5.4. Edit User page

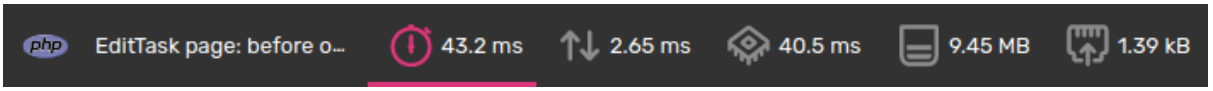
5.4.1. Avant optimisations

5.4.2. Après optimisations

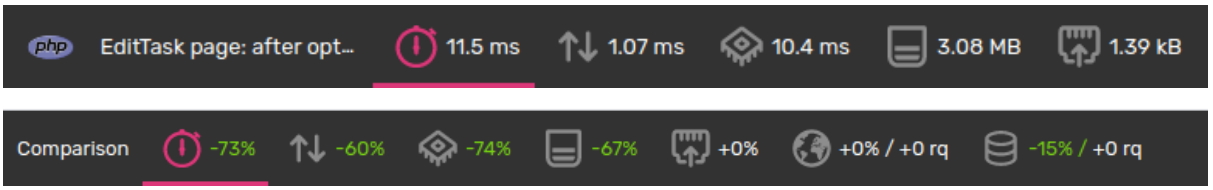


5.5. Edit Task Page

5.5.1. Avant optimisations

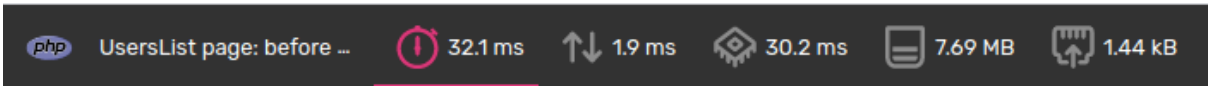


5.5.2. Après optimisations

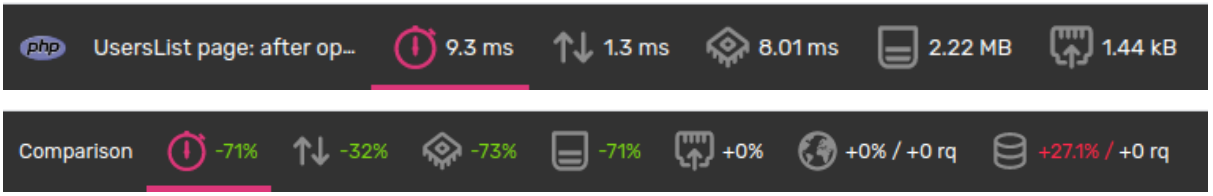


5.6. User list page

5.6.1. Avant optimisations

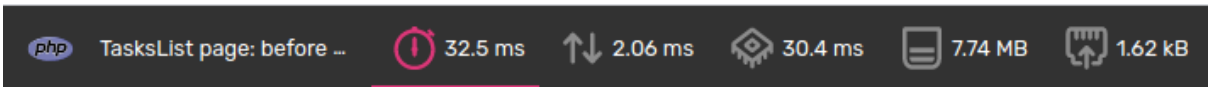


5.6.2. Après optimisations

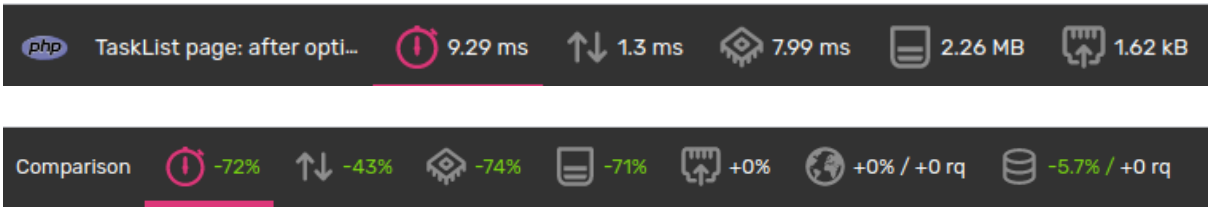


5.7. Task list page

5.7.1. Avant optimisations



5.7.2. Après optimisations



6. Analyse de performace des soumissions de formulaires avec cURL

L'analyse de performance des soumissions de formulaires a été effectuée à l'aide de requêtes cURL.

Le procédé a été le même pour tous les formulaires, mais je vous fais part ici uniquement de l'exemple d'une requête et d'une réponse pour la page login check.

J'ai ensuite pu comparer les performances avant et après telles que pour les analyses des pages à l'affichage.
Pareillement, on y voit un gain de performances assez important.

6.1. Login Check

6.1.1. Requête

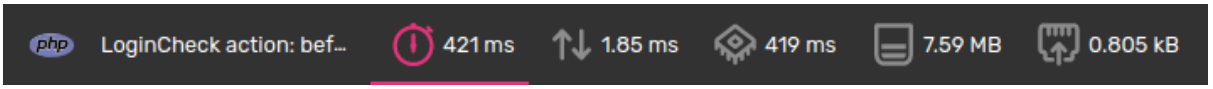
```
sudo blackfire curl 'https://127.0.0.1:8002/login_check' \
>   -H 'authority: 127.0.0.1:8002' \
>   -H 'pragma: no-cache' \
>   -H 'cache-control: no-cache' \
>   -H 'sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90",
"Google Chrome";v="90"' \
>   -H 'sec-ch-ua-mobile: ?0' \
>   -H 'upgrade-insecure-requests: 1' \
>   -H 'origin: https://127.0.0.1:8002' \
>   -H 'content-type: application/x-www-form-urlencoded' \
>   -H 'user-agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93
Safari/537.36' \
>   -H 'accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.9' \
>   -H 'sec-fetch-site: same-origin' \
>   -H 'sec-fetch-mode: navigate' \
>   -H 'sec-fetch-user: ?1' \
>   -H 'sec-fetch-dest: document' \
>   -H 'referer: https://127.0.0.1:8002/login' \
>   -H 'accept-language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7' \
>   -H 'cookie: cookieconsent_dismissed=yes;
PHPSESSID=ki4p3an1lb62ucf2jg7v7op4gi' \
>   --data-raw '_username=admin&_password=admin' \
>   --compressed
```

6.1.2. Réponse

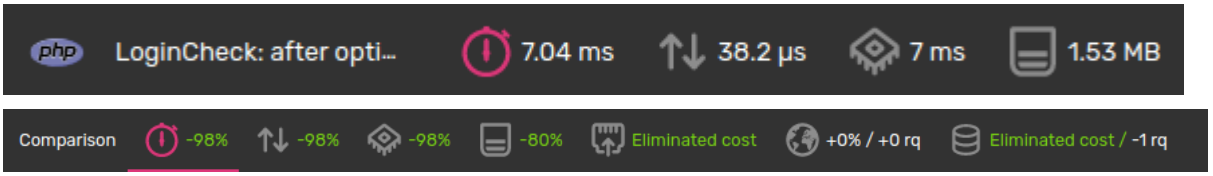
```
Profiling: [########################################] 100%
Blackfire cURL completed
Graph
https://blackfire.io/profiles/b46fc8bc-0eea-499e-876c-fac56d007f76
/graph
Timeline
https://blackfire.io/profiles/b46fc8bc-0eea-499e-876c-fac56d007f76
/graph?settings%5Bdimension%5D=timeline
No tests!              Create some now
https://blackfire.io/docs/testing-cookbooks/tests
4 recommendations
https://blackfire.io/profiles/b46fc8bc-0eea-499e-876c-fac56d007f76
/graph?settings%5BtabPane%5D=recommendations

Wall Time      421ms
I/O Wait      1.85ms
CPU Time       419ms
Memory        7.24MB
Network          n/a      n/a      n/a
SQL            289µs      1rq
```

6.1.2.1. Avant optimisation



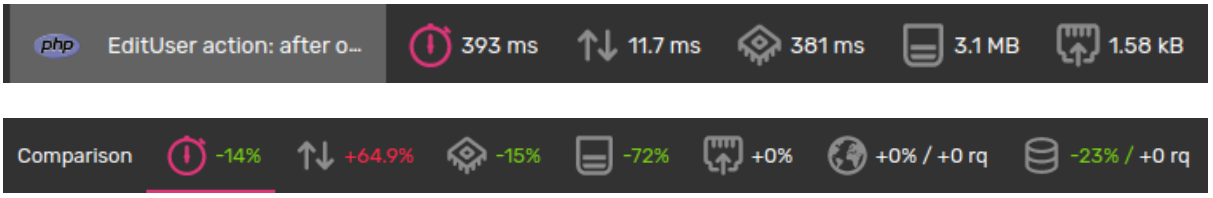
6.1.2.2. Après optimisation



6.2. EditUser

6.2.0.1. Avant optimisation

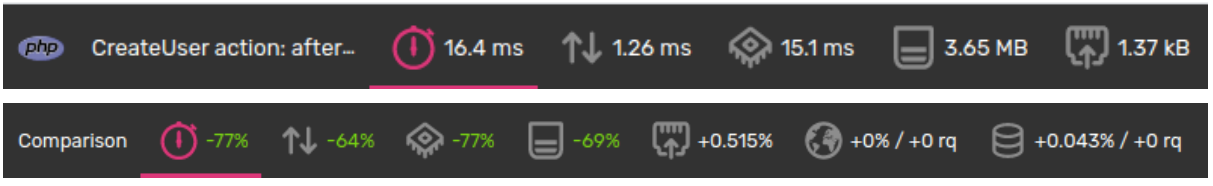
6.2.0.2. Après optimisation



6.3. CreateUser

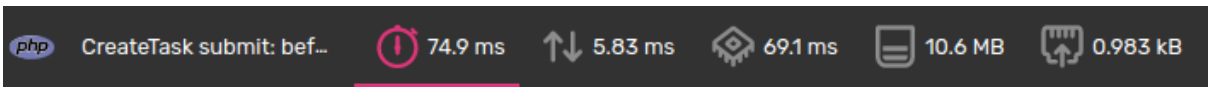
6.3.0.1. Avant optimisation

6.3.0.2. Après optimisation

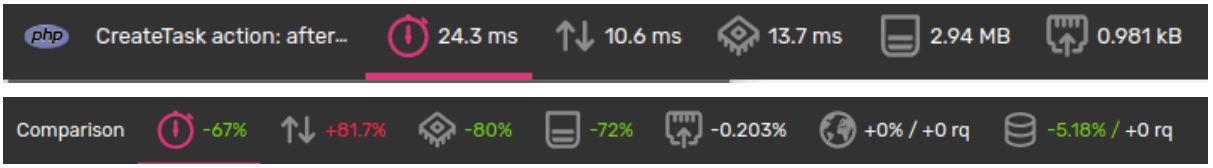


6.4. CreateTask

6.4.0.1. Avant optimisation

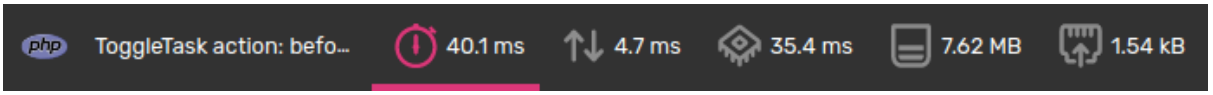


Après optimisation

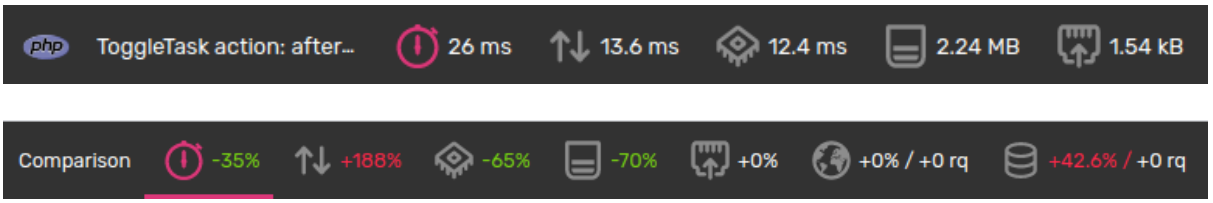


6.5. ToggleTask

6.5.0.1. Avant optimisation

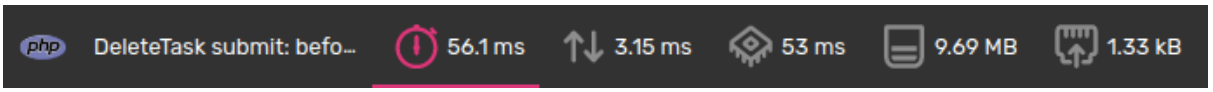


6.5.0.2. Après optimisation

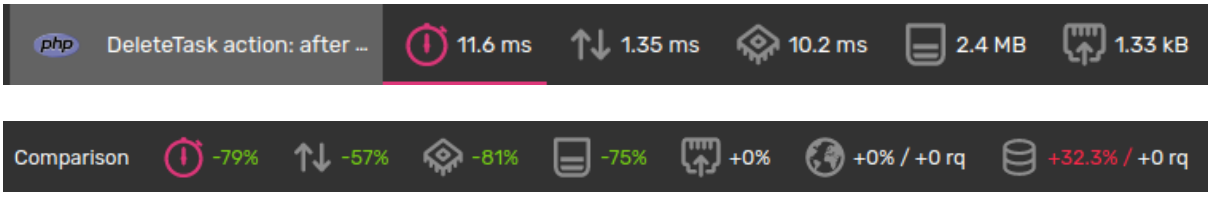


6.6. DeleteTask

6.6.0.1. Avant optimisation

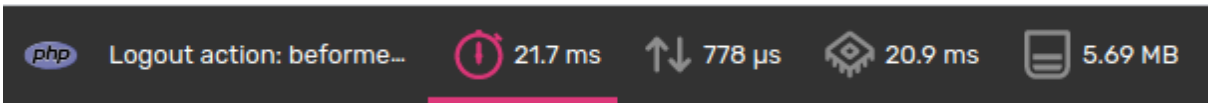


6.6.0.2. Après optimisations

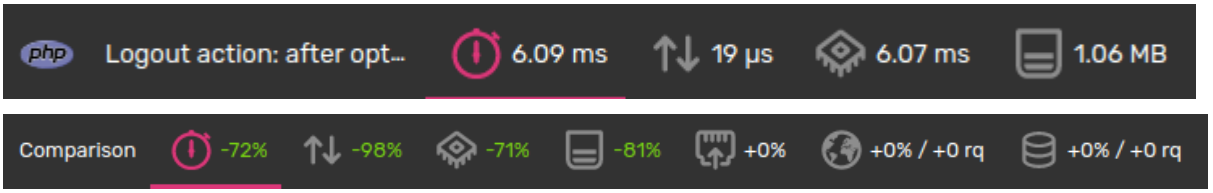


6.7. Logout

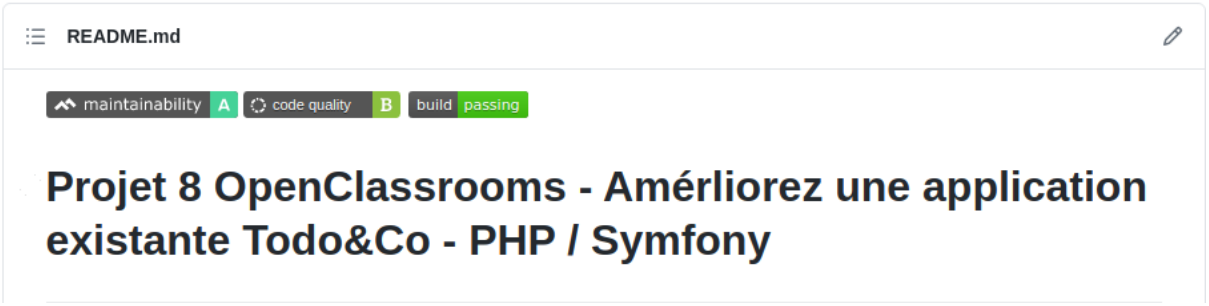
6.7.0.1. Avant optimisation



6.7.0.2. Après optimisation

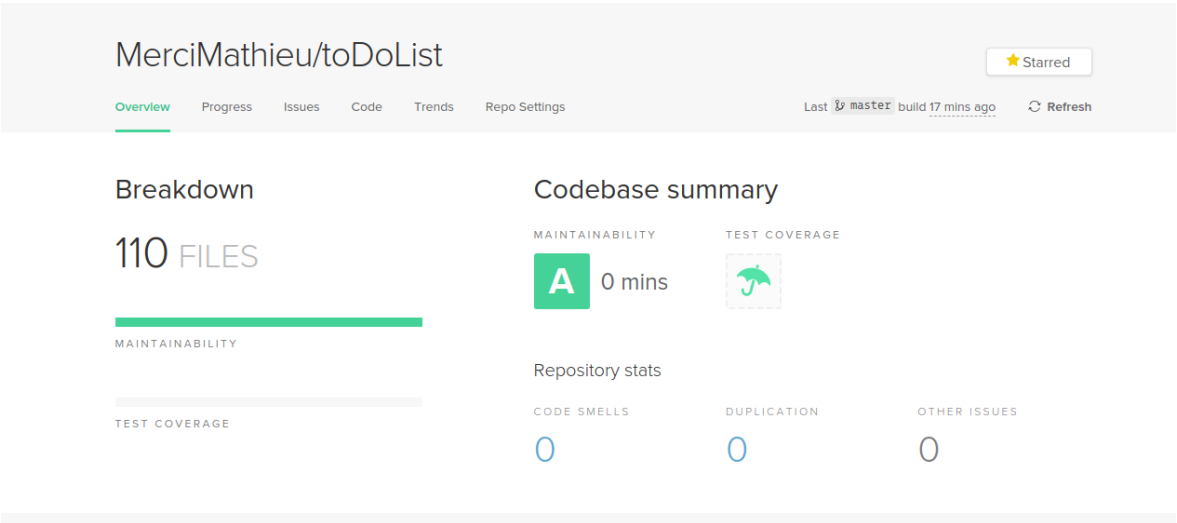


7. Analyse de qualité et maintenabilité du code



L'analyse s'appuie sur les mesures apportées par l' outil Code Climate pour la maintenabilité et qualité du code.

7.1. Code Climate



Le rapport de Code Climate nous indique une excellente maintenabilité du code, il n'y a pas de mauvaise pratique détectée ou duplication du code.

Lien vers le dashboard Code Climate:
https://codeclimate.com/github/MerciMathieu/toDoList

8. Maintenir la qualité du code

Nous utilisons certains outils et procédures pour permettre d'assurer une qualité optimale du code.

8.1. php-cs-fixer

Permet de respecter les standards php (psr-12)

8.2. Pull requests

Les développements sont validés par le chef de projet avant d'être livrés en production.

8.3. Code climate / Symfony insight

Permettent d'analyser le code et de fournir un rapport détaillé sur les erreurs, la qualité, la maintenabilité, etc.

8.4. Fichier de contribution

Mise en place du fichier CONTRIBUTING.md à la racine du projet.
Il donne des consignes aux développeurs afin de respecter les bonnes pratiques de code dans le projet.

8.5. Tests unitaires et fonctionnels avec taux de couverture

Les tests permettent de nous assurer que l'application fonctionne comme nous l'attendons d'elle, au niveau des fonctionnalités comme au niveau du comportement de l'application, ou de l'affichage.

Le code est testé à l'aide de PHPunit.

8.5.1. Résultat des tests

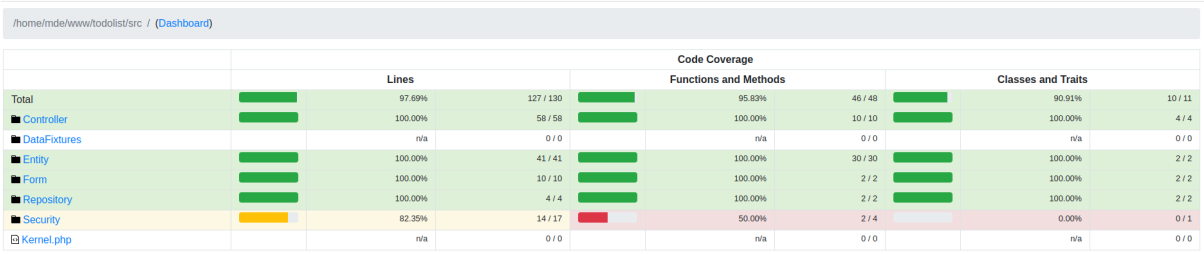
/usr/bin/php /tmp/ide-phpunit.php --configuration /home/mde/www/todolist/phpunit.xml.dist /home/mde/www/todolist/tests/AppBundle
Testing started at 01:13 ...
PHPUnit 9.4.4 by Sebastian Bergmann and contributors.

Testing /home/mde/www/todolist/tests/AppBundle
..... 42 / 42 (100%)

Time: 00:49.126, Memory: 54.50 MB

OK (42 tests, 79 assertions)
Process finished with exit code 0

8.5.2. Taux de couverture



/home/mde/www/todolist/src / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		97.69%	127 / 130		95.83%	46 / 48		90.91%	10 / 11
Controller		100.00%	58 / 58		100.00%	10 / 10		100.00%	4 / 4
DataFixtures		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Entity		100.00%	41 / 41		100.00%	30 / 30		100.00%	2 / 2
Form		100.00%	10 / 10		100.00%	2 / 2		100.00%	2 / 2
Repository		100.00%	4 / 4		100.00%	2 / 2		100.00%	2 / 2
Security		82.35%	14 / 17		50.00%	2 / 4		0.00%	0 / 1
Kernel.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0

https://127.0.0.1:8000/web/test-coverage/index.html

8.6. Travis CI

Travis CI est un outil d'intégration continue. Il fournit un service en ligne utilisé pour compiler, tester et déployer le code source des logiciels développés, notamment en lien avec le service d'hébergement du code source GitHub.

Il va effectuer tous les tests fonctionnels et unitaires développés dans l'application lors d'un merge ou d'un push sur la branche **master**

https://travis-ci.com/github/MerciMathieu/toDoList