

TODO & Co

Audit de qualité du code et performance de l'application

SOMMAIRE

1. Cadre
 - 1.1. Contexte
 - 1.2. Missions
2. Qualité du code
 - 2.1. Migration
 - 2.2. Outil qualité
 - 2.3. Couverture des tests
3. Performance
 - 3.1. Analyse de la page d'accueil
4. Axes d'amélioration

1. Cadre

1.1. Contexte

La startup **ToDo & Co** a obtenu une levée de fonds afin de développer son application de gestion de tâches quotidiennes.
Un Minimum Viable Product a déjà été réalisé et présenté aux investisseurs.

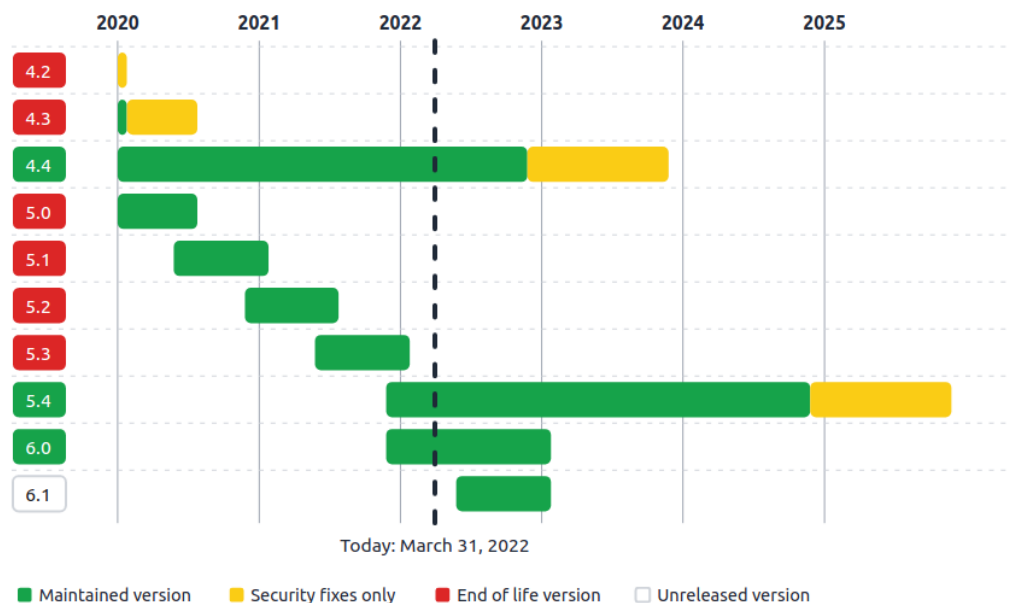
1.2. Missions

- Corrections d'anomalies
- Implémentations de nouvelles fonctionnalités
- Mise en place de tests automatisés

2. Qualité du code

2.1. Migration

Le MVP fourni était sous Symfony 3.1, une version actuellement non maintenue.



Nous pouvons voir sur ce roadmap que la version 4.4 arrive également bientôt en fin de maintenance et je ne souhaite pas avoir une nouvelle migration à faire très prochainement. J'ai donc opté pour la version Long Time

Support, qui est la 5.4, complètement maintenue jusque fin 2024 et jusqu'à 2026 pour la sécurité.

Une mise à jour des contenus dépréciés a été effectuée tout au long de la phase de développement.

Le reste de l'audit de qualité et de performance a été effectué après les corrections d'anomalies et l'ajout de nouvelles fonctionnalités.

2.2. Outil qualité

Utilisation de **Symfony Insight** pour assurer que l'application est sûre, fiable et maintenable.

Symfony Insight exécute automatiquement une analyse lors de pull request.

On peut y voir des erreurs et leur importance.


▼ Text files should end with a valid new line character. 2 Read doc Ignore all Productivity Info

in .gitignore, line 42

```
37. /public/test.html
38. ###< phpunit/phpunit ###
39.
40. ### test coverage ###
41. /coverage/test-coverage
42. ### test coverage ###
```

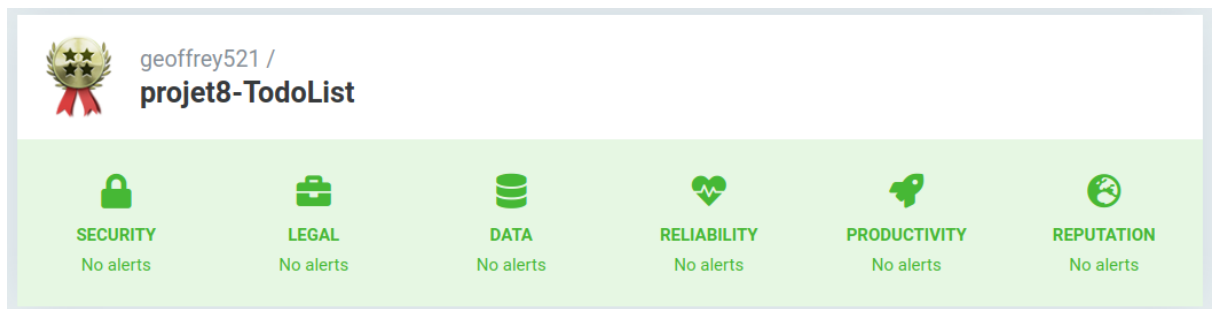
This file ends with no newline character, or with a different newline character than other files in your project. It won't render properly on a terminal, and it's considered a bad practice.

Time to fix: about 15 minutes Read doc Comment Ignore Open Issue Permalink

Last edited 13 hours ago by geoffrey521 

Nous pouvons voir ici une erreur de mauvaise pratique ainsi, sa description et une estimation du temps nécessaire pour la résoudre.

Obtention de la note la plus haut avec la médaille Platinum



2.3. Couverture des tests

Dans la version initiale, aucun test n'était effectué, sa couverture était de 0%.

Après la mise en place de tests, j'ai pu obtenir une couverture de **85.62%**.

	Lines		Code Coverage		Classes and Traits	
	Percentage	Count	Percentage	Count	Percentage	Count
Total	85.62%	125 / 146	89.09%	49 / 55	66.67%	8 / 12
Command	63.64%	14 / 22	66.67%	2 / 3	0.00%	0 / 1
Controller	100.00%	52 / 52	100.00%	11 / 11	100.00%	4 / 4
Entity	100.00%	39 / 39	100.00%	29 / 29	100.00%	2 / 2
Form	100.00%	14 / 14	100.00%	4 / 4	100.00%	2 / 2
Repository	14.29%	2 / 14	33.33%	2 / 6	0.00%	0 / 2
Security	80.00%	4 / 5	50.00%	1 / 2	0.00%	0 / 1
Kernel.php	n/a	0 / 0	n/a	0 / 0	n/a	0 / 0

Legend
 Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

Il aurait été possible de monter bien plus haut en faisant des tests sur les repositories, mais ceux ci étant générés par symfony et non modifiés, l'action n'aurait pas été pertinente.

Vous pouvez trouver tous les détails de cette analyse dans le dossier **html_coverage** situé à côté de ce fichier.

3. Performances

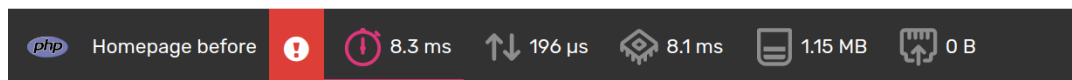
Attention: tous les tests sont réalisés en environnement de production mais en local (ce qui explique une haute vitesse d'exécution)

3.1. Analyse globale des performances

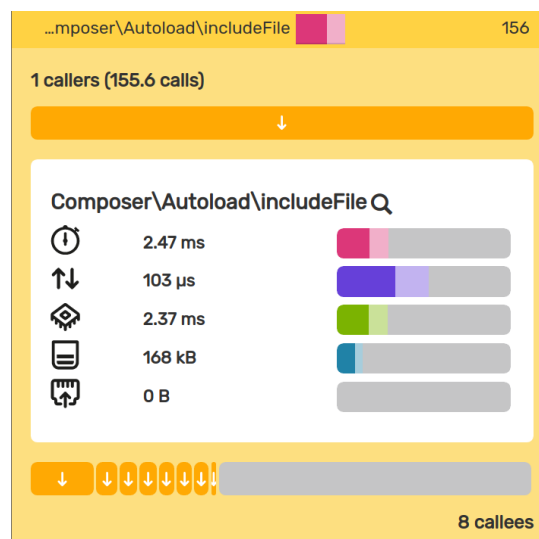
URL	Temps d'affichage / ms	Mémoire utilisée / mb
/login	13.2	1.68
/	8.3	1.15
/tasks	22	2.07

/tasks/create	23.1	2.08
/tasks/{id}/edit	22	2.08
/users	15.1	1.79
/users/create	19.6	2.5
/users/{id}/edit	20.1	2.52

3.2. Analyse de la page d'accueil



Nous pouvons voir ici, une vitesse d'exécution de 8.3ms, c'est rapide mais comme dit précédemment, c'est en local et nous pouvons quand même l'améliorer.



La fonction liée à composer, "IncludeFile" est la fonction la moins performante, elle a été appelée 156 fois et en appelle 8 autres, pour une vitesse de 2.47ms(18.6%) et une consommation de 168kb(10.5%).

Suggestion de Correctif de performance:

Optimisation de l'autoloader au niveau 2/B

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

Cette optimisation à faire une fois le projet déployé, permettra la mise en cache des recherches de classes, permettant d'obtenir une vitesse de chargement accrue lors des prochaines requêtes.

4. Axes d'amélioration

Améliorations de performances:

- Utiliser le système de cache de Doctrine et Symfony
- Utilisation de l'AJAX pour éviter de recharger toute une page à chaque interaction (exemple: lors du marquage d'une tâche comme faite)
- Implémenter un système de lazy loading pour les images

Améliorations Expérience Utilisateur:

- Amélioration nécessaire
 - Ajout de la fonctionnalité de suppression d'un utilisateur pour un Administrateur
 - Ajout de redirections personnalisées.
 - Ajout de la page de liste des tâches terminées
 - Ajout d'un lien d'accès à la liste des utilisateurs
- Amélioration recommandées
 - Ajout de catégories pour les tâches
 - Ajout de l'affichage des détails d'une tâche
 - Ajout de photo de profil utilisateur, que nous pourrions afficher aux miniatures des tâches