

# TODO & CO

## TODOLIST

### RAPPORT D'AUDIT :

- QUALITE
- PERFORMANCE

Parcours OPENCLASSROOMS

Développeur d'Application PHP / Symfony

Projet d'étude 8

Développeur : Mathieu Bonhommeau

Ce document présente l'audit de qualité et de performance qui a été réalisé sur l'application TODOLIST de la startup TODO & CO.

## SOMMAIRE :

1. CONTEXTE DU PROJET

2. MISE A NIVEAU DE L'APPLICATION

3. OUTILS UTILISES

4. AUDIT QUALITE

5. AUDIT PERFORMANCES

6. AMELIORATIONS POSSIBLES

# 1 – CONTEXTE DU PROJET

TODO & CO est une startup qui a conçu une application dont le but est de pouvoir gérer ses tâches quotidiennes : [TODOLIST](#)

La version existante de l'application est une version *MVP (Minimum Viable Product)* développé en PHP avec le framework Symfony.

Le but du travail de développement qui a été effectué sur l'application était d'améliorer la qualité :

- Qualité du code
- Qualité perçue par les utilisateurs
- Qualité perçue par les collaborateurs de TODO & CO
- Qualité perçue lorsqu'il faut travailler sur le projet

## 2 – MISE A NIVEAU DE L'APPLICATION

- Migration vers une version *LTS* (*Long-Term Support version*)  
Le projet a été initialement développé à l'aide de Symfony 3.1.  
Il a donc été migré vers une version stable et maintenue sur le long terme :  
**Symfony 4.4.19.**
- Correction d'anomalies  
Différentes anomalies ont été détectées dans le projet.  
Celles-ci ont été corrigées.
  - Une tâche doit être attachée à un utilisateur
  - Un utilisateur doit avoir un rôle (admin ou user)
  - Correction de liens morts
  - Correction de mauvais messages flash
  - Redirection
- Implémentation de nouvelles fonctionnalités  
Plusieurs fonctionnalités ont été ajoutées :
  - Restriction des pages de gestion des utilisateurs aux administrateurs
  - Une tâche ne peut être supprimée que par l'utilisateur qui l'a créée
  - Les tâches anonymes ne peuvent être supprimées que par un administrateur
- Implémentation de tests automatisés  
Des test unitaires et fonctionnels ont été implémentés à l'aide de **PHPUNIT** pour s'assurer que le fonctionnement de l'application soit en adéquation avec les demandes.

Un rapport de couverture de code est disponible dans le projet au chemin suivant : </web/test-coverage/index.html>

## 3 – OUTILS UTILISES POUR L'AUDIT

- Qualité du code :  
[CodeClimate](#) et [Codacy](#)
- Performance de l'application  
[Blackfire](#)

## 4 – AUDIT DE QUALITE

Les analyses effectuées sur [CodeClimate](#) et [Codacy](#) ont décernées respectivement les notes suivantes :

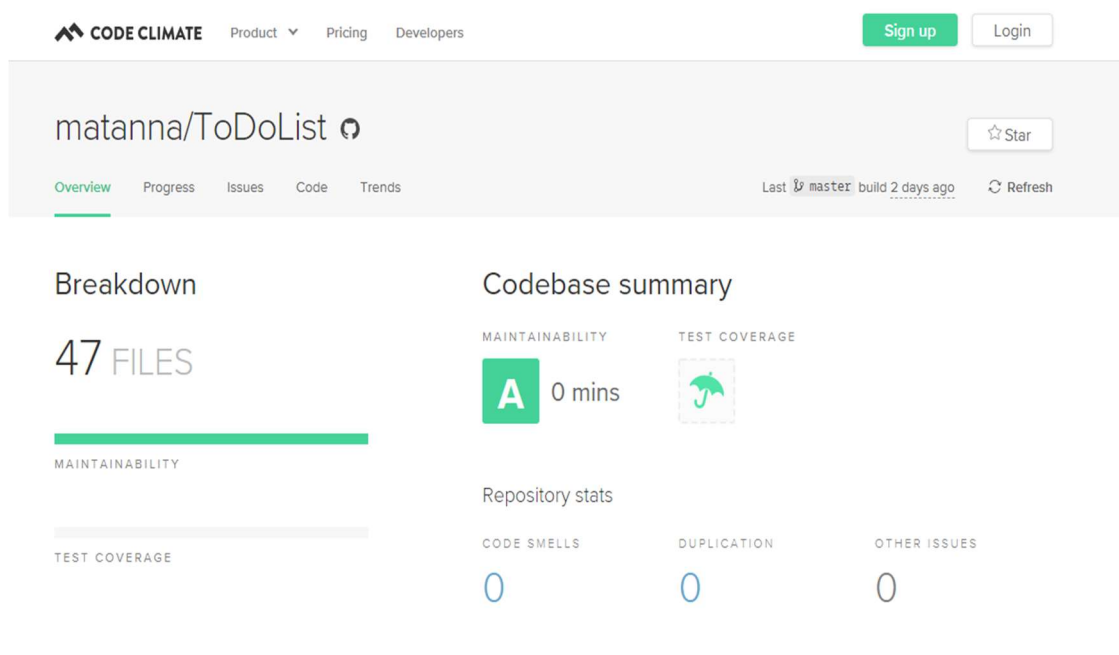
- Note A pour CodeClimate.

L'analyse a cependant remontée quelques erreurs à corriger :

- Méthodes de classes trop longues
- Variables déclarées mais non utilisées

Ceci a été corrigé et l'analyse relancée.

Le résultat est une Note A sans erreurs :



- Note B pour Codacy

L'analyse a remontée plus d'erreurs que sur CodeClimate mais ce système est beaucoup moins permissif.

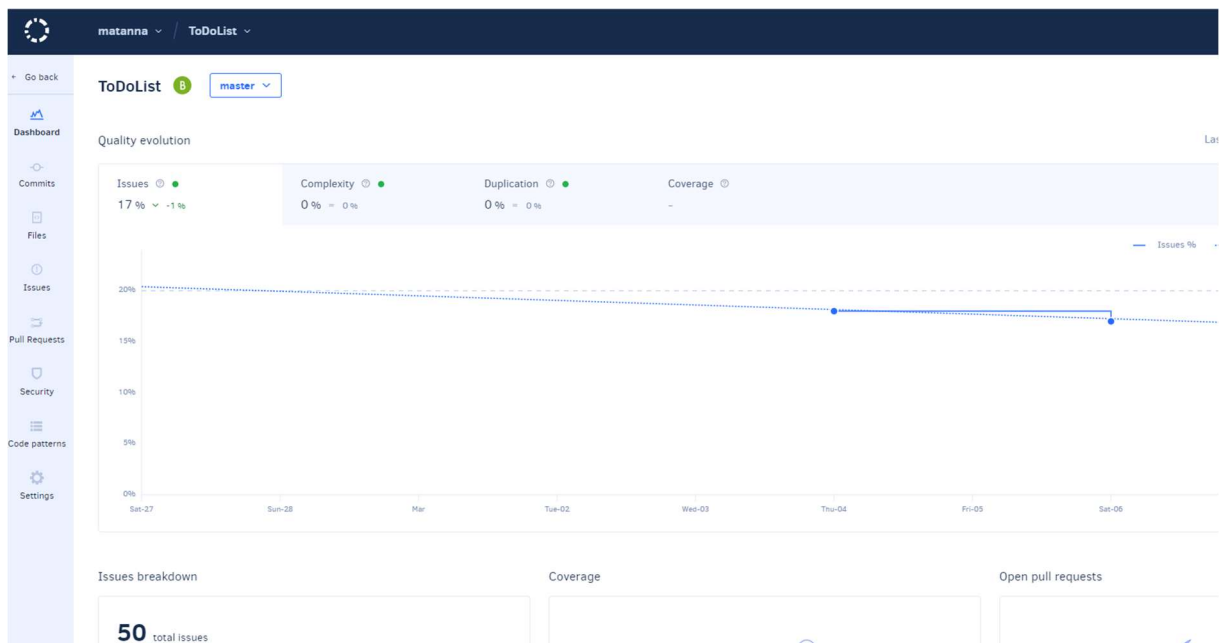
Ce qui a été modifié :

- Nom de variables trop court
- Variables non utilisées
- Suppression du else dans certaines conditions if.
- Suppression de paramètres non utilisés au niveau des méthodes

Ceci a été corrigé et l'analyse relancée.

Le résultat est toujours une note B.

Les autres erreurs remontées par l'analyse Codacy sont inhérentes au framework Symfony. Elles n'ont donc pas été corrigées.



Les analyses de qualité de code n'ont donc pas remonté de gros défauts.

Le code est donc conforme aux normes en vigueur et respecte bien les standards PHP.

## 5- AUDIT DE PERFORMANCES

Au préalable, l'application a été placée en **mode production**.

- Optimisation de l'auto-chargement des classes de Composer  
L'analyse des performances à l'aide de Blackfire a permis de remonter une recommandation importante :
  - L'auto-chargement des classes de Composer doit être optimisé en production.

Voici un test de performances de la page d'accueil sans l'optimisation du chargement des classes :

The screenshot shows a web browser with a dark header displaying performance metrics: 212 ms, 149 ms, 62.7 ms, 14.6 MB, 768 B, 0 µs / 0 rq, and 328 µs / 1 rq. Below the header, there are buttons for 'Créer un utilisateur', 'Voir les utilisateurs', and 'Se déconnecter'. The main content area says 'Bienvenue sur Todo List, l'application vous permettant de gérer l'ensemble de vos tâches sans effort !'. Below this is a video of a spiral-bound notebook with the following handwritten text: 'TO DO', '① Secure the Biz', 'Hunt the', and '② Bad guys!'. At the bottom of the page, there are buttons for 'Créer une nouvelle tâche', 'Consulter la liste des tâches à faire', and 'Consulter la liste des tâches terminées'. The footer text reads 'Copyright © OpenClassrooms'.

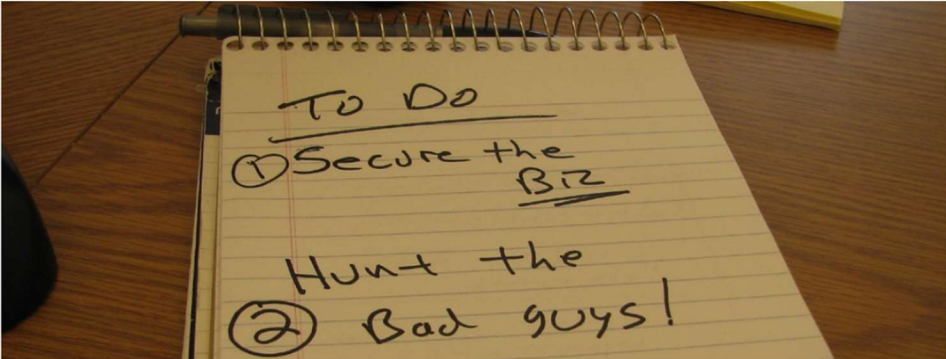
Voici un test de performances de la page d'accueil avec l'optimisation du chargement des classes :

- Ajout de **"optimize-autoloader"** : true dans **composer.json**



Créer un utilisateur Voir les utilisateurs Se déconnecter

Bienvenue sur Todo List, l'application vous permettant de gérer l'ensemble de vos tâches sans effort !



Créer une nouvelle tâche Consulter la liste des tâches à faire Consulter la liste des tâches terminées

Copyright © OpenClassrooms

Voici le comparatif des deux analyses :

Le temps de chargement de la page a donc été abaissé de 13%. L'utilisation de la mémoire a été en revanche augmenté de 10%. Le choix se porte tout de même sur la vitesse de chargement : la modification est donc concluante

Comparison -13% -19% +0.936% +10.1% +0% +0% / +0 rq -5.22% / +0 rq

Functions

From: To Do List app  
200 GET https://127.0.0.1:8000/  
5 minutes ago

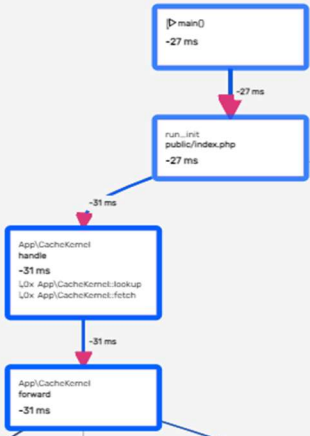
To: To Do List app  
200 GET https://127.0.0.1:8000/  
1 minute ago

Probe version: 1.51.0  
Agent version: 1.47.0  
Runtime: PHP 7.3.12 (cgi-fcgi)  
Probe OS: WINNT  
Agent OS: Windows 10 Home 10.0  
Samples: 10  
Timeline threshold: 10 ms  
Response size: 3.457 kB  
Transaction Name: App\Controller\DefaultController::indexAction

Switch comparison direction

App\CacheKernel::forward +0  
...el\HttpCache\SubRequestHandler::handle +0

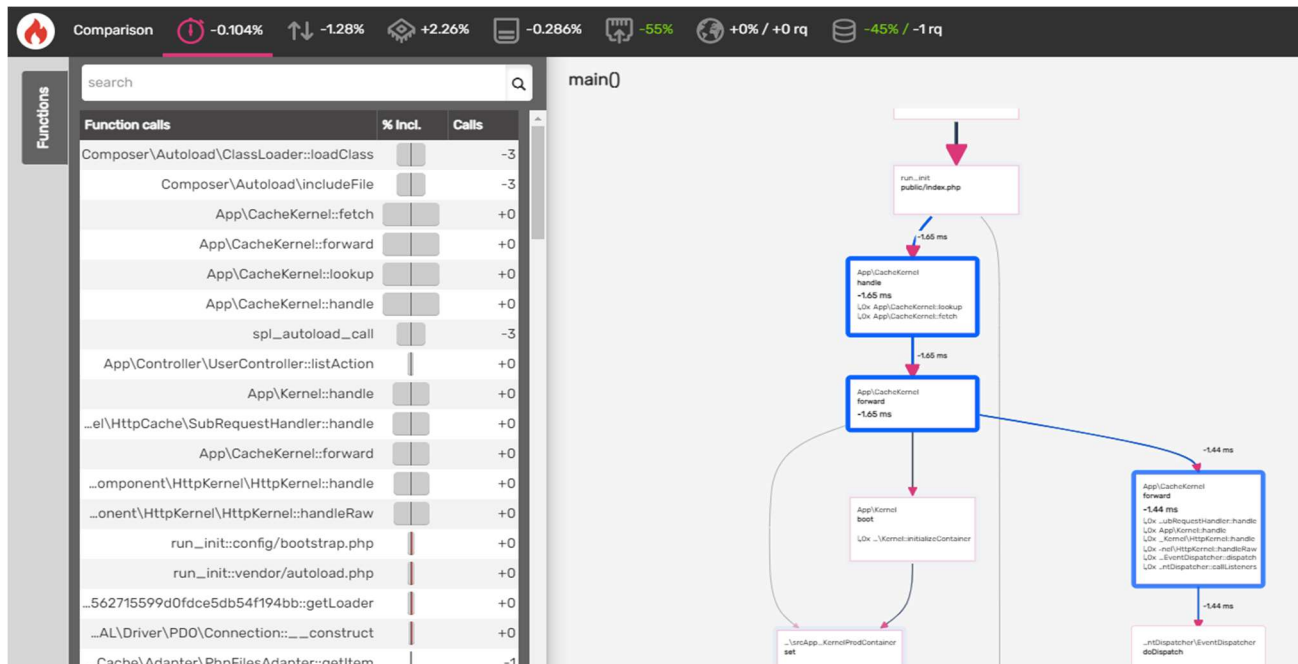
main()



- Mise en cache des données

Pour améliorer encore les performances de l'application, le choix a été fait de mettre certaines données en cache.

Pour illustrer cela, voici une comparaison entre l'affichage de la page listant les utilisateurs sans la mise en cache et avec la mise en cache.



La différence en temps de chargement n'est pas visible dans ce cas car pour l'instant, il n'y a pas beaucoup d'utilisateurs. Plus leur nombre augmentera, plus la différence sera visible.

En revanche, on remarque immédiatement une consommation réseau abaissée de 55% et un nombre de requêtes exécutées abaissées de 45%.

L'utilisation du cache http est également conseillé pour l'affichage des pages ce qui permettrait encore un gain de performances.

## 6 – AMELIORATIONS POSSIBLES

Plusieurs améliorations peuvent être apportées à l'application :

- **Système de catégories**  
Les tâches sont classées dans différentes catégories pour faciliter leur organisation et leur tri
- **Système d'accessibilité des tâches**  
Les tâches peuvent être privées (visibles que pour un utilisateur), publiques (visibles par tous) ou appartenant à un groupe de travail (visibles que par les utilisateurs appartenant à ce groupe)
- **Système de Deadline**  
A l'enregistrement d'une tâche, une date butoir est indiquée. Ceci permettra de prioriser les tâches à faire et d'implémenter un caractère d'Urgence.
- **Système de commentaires**  
Sur une tâche, chaque utilisateur pourra ajouter un ou des commentaires pour indiquer l'avancement de la tâche en question ou tout autre information qu'il juge nécessaire.
- **Page de profil**  
Créer une page de profil pour chaque utilisateur de l'application. L'utilisateur pourra ajouter une photo et d'autres informations.
- **Système de mail**  
Chaque utilisateur pourra envoyer des mails/messages via l'application à un autre utilisateur de son choix.