Audit de qualité du code & performance de l'application

I. Contexte du projet

Vous venez d'intégrer une startup dont le cœur de métier est une application permettant de gérer ses tâches quotidiennes. L'entreprise vient tout juste d'être montée, et l'application a dû être développée à toute vitesse pour permettre de montrer à de potentiels investisseurs que le concept est viable (on parle de Minimum Viable Product ou MVP).
Le choix du développeur précédent a été d'utiliser le framework PHP Symfony, un framework que vous commencez à bien connaître !
Bonne nouvelle ! ToDo & Co a enfin réussi à lever des fonds pour permettre le développement de l'entreprise et surtout de l'application.
Votre rôle ici est donc d'améliorer la qualité de l'application. La qualité est un concept qui englobe bon nombre de sujets : on parle souvent de qualité de code, mais il y a également la qualité perçue par l'utilisateur de l'application ou encore la qualité perçue par les collaborateurs de l'entreprise, et enfin la qualité que vous percevez lorsqu'il vous faut travailler sur le projet.

II. Missions

- Correction d'anomalies
- Implémentation de nouvelles fonctionnalités
- Implémentation de tests automatisés

III. Audit de qualité du code

1. Analyse de l'existant

Le code de départ qui nous a été fourni était sous Symfony 3.1 qui est une version qui n'est plus maintenue depuis 2018.
Après récupération et installation du projet, il y'a eu des bugs empêchant l'application de se lancer. Et après correction et lancement de l'application, la dépréciation de plusieurs composants a été constatée.

En plus des anomalies signalées dans les missions, d'autres ont été repérées :

- Absence d'un lien vers la page d'accueil
- Position des boutons du formulaire de création de tâches contre intuitive
- Boutons visibles pour tous les utilisateurs même ceux qui n'ont pas l'autorisation d'exécuter les actions spécifiées
- Pages d'erreurs non personnalisées

L'analyse de la qualité du code sur Codacy de l'application avait renvoyée un grade de **C**.

2. Améliorations apportées

L'ensemble des missions qui été spécifiées dans le cahier des charges ont été accomplies.

- **Migration vers Symfony 4.4**

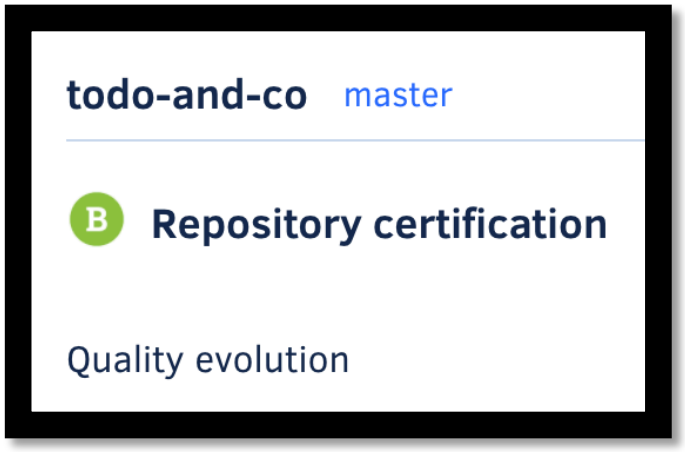
Le projet a été migré vers la version 4.4 de Symfony, qui est une version LTS (stable et maintenue sur le long terme).

Les changements relatifs à la migration et pour pallier aux dépréciations dans certaines parties du code ont été apportés :

- Changement de **AppBundle** par **App** pour les namespaces
- Remplacer les classes qui étendaient **Controller** par **AbstractController**
- Utiliser l'injection de dépendance plutôt que l'**AuthentificationUtils** dans le container
- Etc.

- **Amélioration sur Codacy**

Suite à l'analyse de qualité sur Codacy, des problèmes sur les aspects du code ont été corrigé, faisant passer le grade du projet de la note de **C** à une note de **B**.

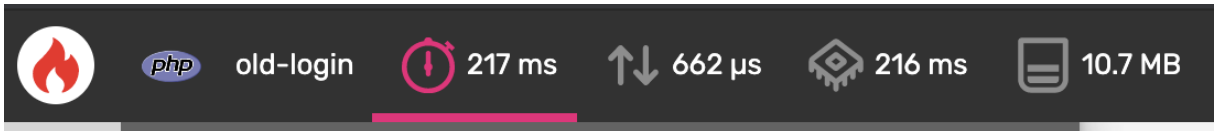


IV. Audit de performance avec Blackfire

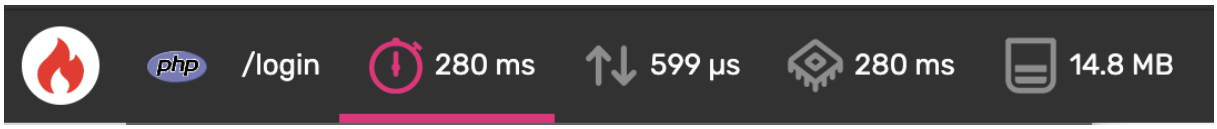
L'analyse de performance a été fait avec Blackfire et en environnement de production.

- La route **/login**

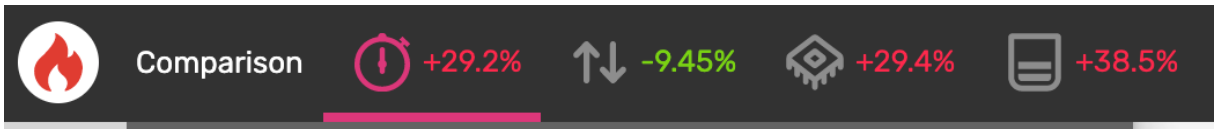
Avant



Après

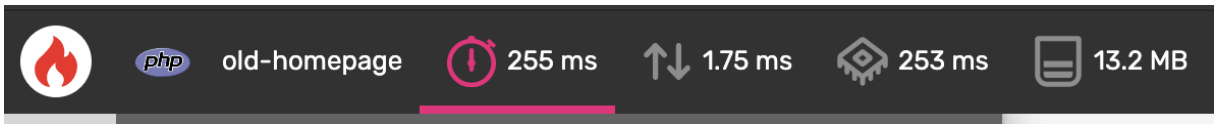


Comparaison

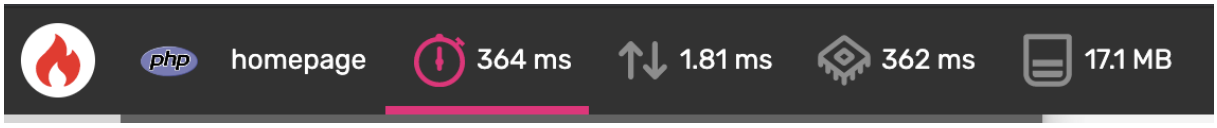


- La route **/**

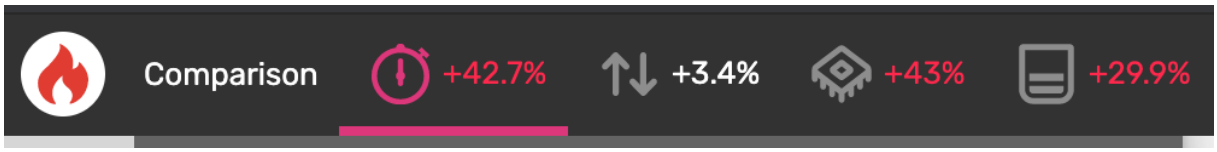
Avant



Après

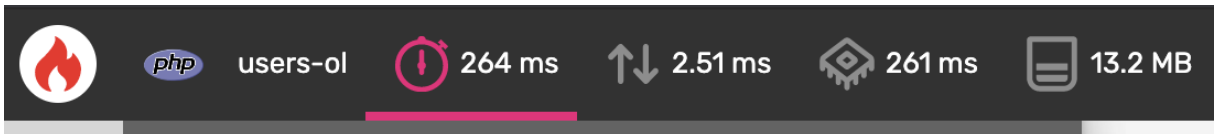


Comparaison

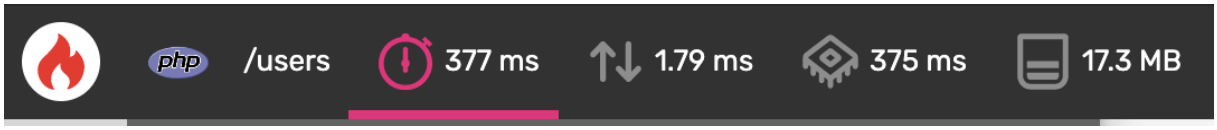


- La route **/users**

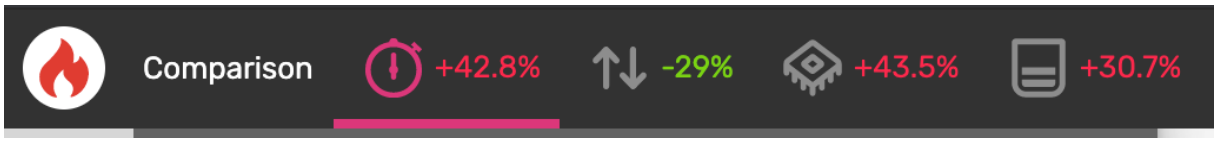
Avant



Après

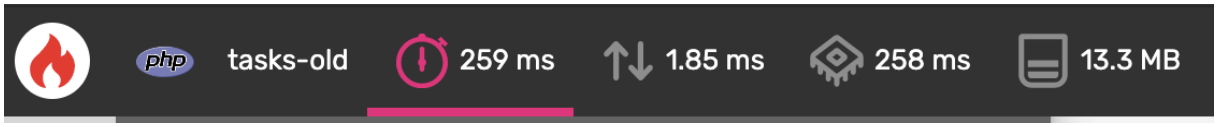


Comparaison

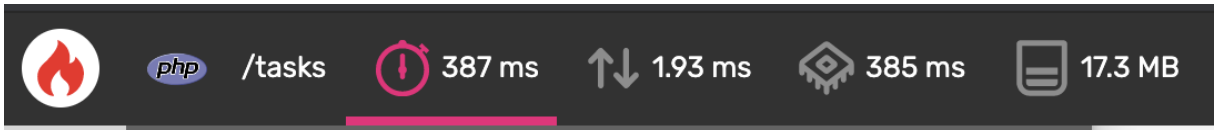


- La route **/tasks**

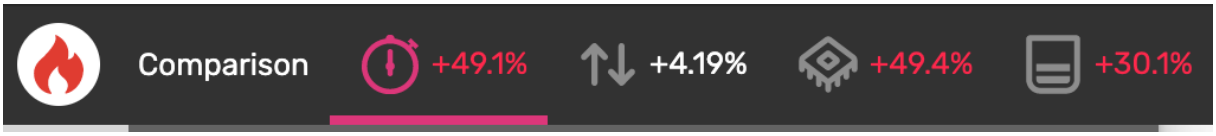
Avant



Après



Comparaison



Les metrics ci-dessus ont été réalisées sur 4 différentes routes de notre application : la page d'authentification (**/login**), la page d'accueil (**/**), la page de la liste des utilisateurs (**/users**) et la page de la liste des tâches (**/tasks**).
On constate une légère perte de performance après les corrections et changements apportés au projet.
Par exemple pour la route **/login** :

- le temps global d'exécution passe de 217ms à 280ms soit une perte de plus de 29,2%
- Le temps de réponse du processeur (CPU) passe de 216ms à 280ms
- La mémoire utilisée pour la génération de la page passe de 10.7MB à 14.8MB

Ces performances peuvent être améliorées en optimisant l'appel de fonctions tels que l'autoloader de Composer et en activant OPcache de PHP.