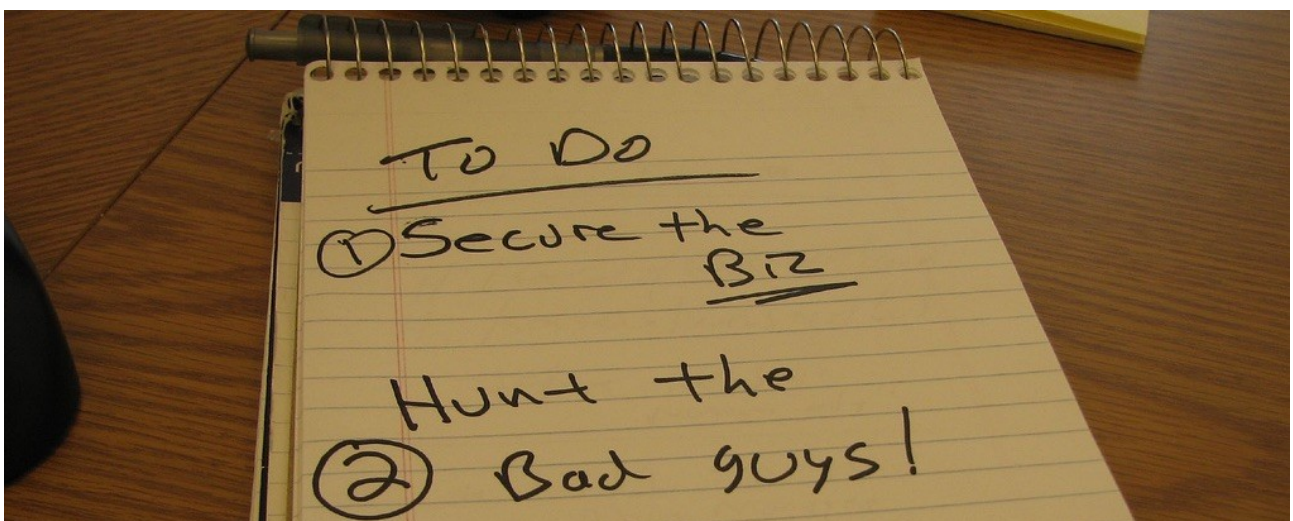


ToDo & Co

Audit de qualité et de performance de l'application



Contexte

ToDo list est une application développée par la startup ToDo & Co. Elle permet aux utilisateurs de gérer leurs 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. Mon travail ici a été d'ajouter de nouvelles fonctionnalités, de mettre à jour l'application, d'écrire tous les tests nécessaires, de surveiller les performances de celle-ci ainsi que de m'assurer de la bonne qualité du code. Dans cet audit je détaillerais le travail qui a été effectués en termes de mise à jour de version, performances de l'application et qualité du code.

Version

Upgrade de version

Avant toute chose, une mise à jour de Symfony à été faite sur l'application afin de l'améliorer en termes de performances et de sécurité. Nous sommes donc passés de la version 3.1 à la version 6.4 de Symfony. Cette version à été choisie car c'est la « Long-Term Support Release », signifiant qu'elle bénéficiera d'une maintenance concernant les bugs et la sécurité pendant une période de trois ans.

→ <https://symfony.com/releases>

Performances

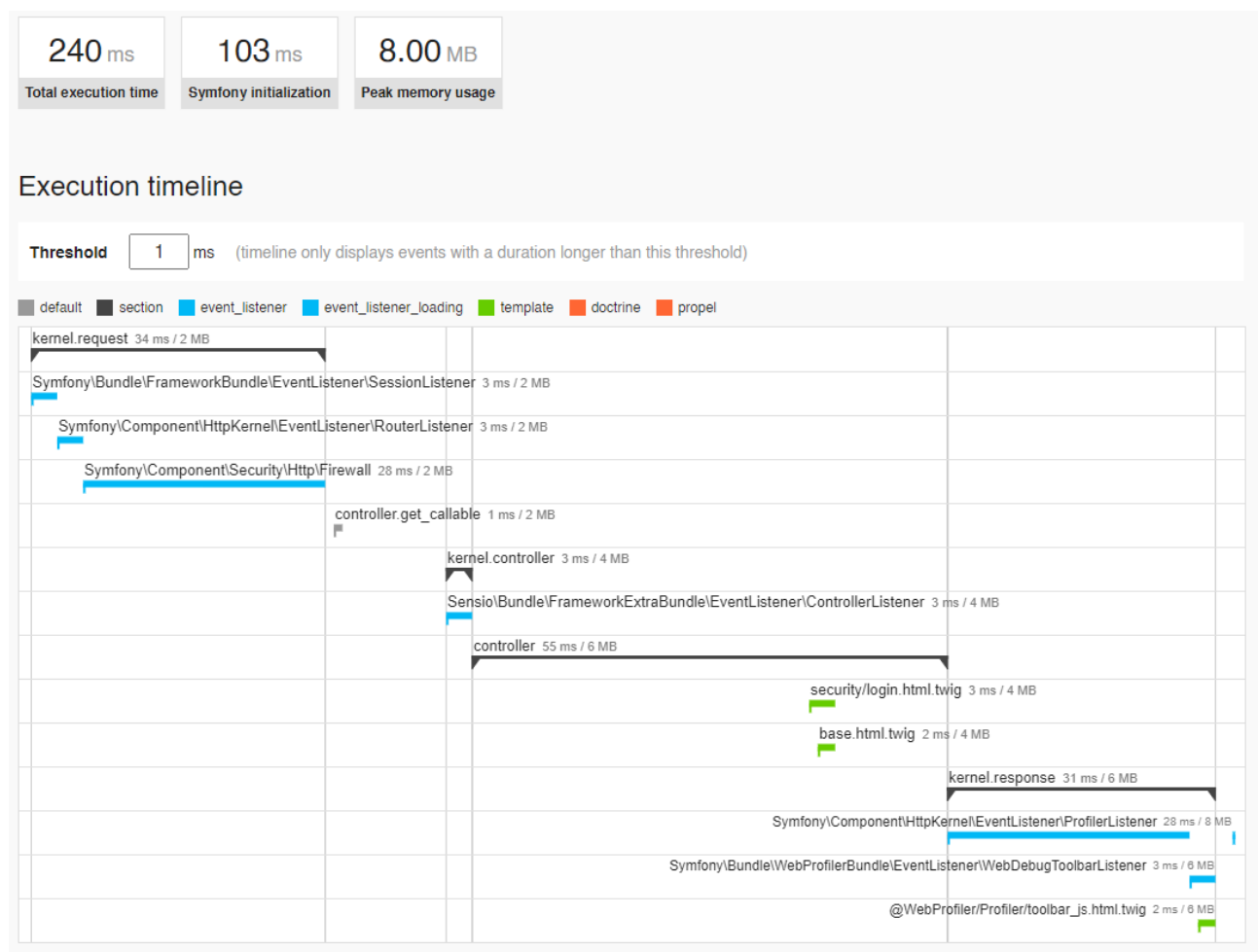
L'upgrade de version de Symfony (passage de la 3.1 à la 6.4) à été l'élément essentiel dans l'amélioration de l'application. Cette upgrade nous à d'un côté permis d'avoir une version à jour de Symfony en terme de sécurité, mais également maintenue et ce pour les trois prochaines années. En effet comme cité plus haut cette version est la LTS Release et donc signifie qu'elle sera maintenue pendant trois ans.

L'upgrade s'est également faite sur la version de PHP, puisque désormais c'est la version 8.1 de PHP qui est utilisée, permettant une amélioration des performances du code.

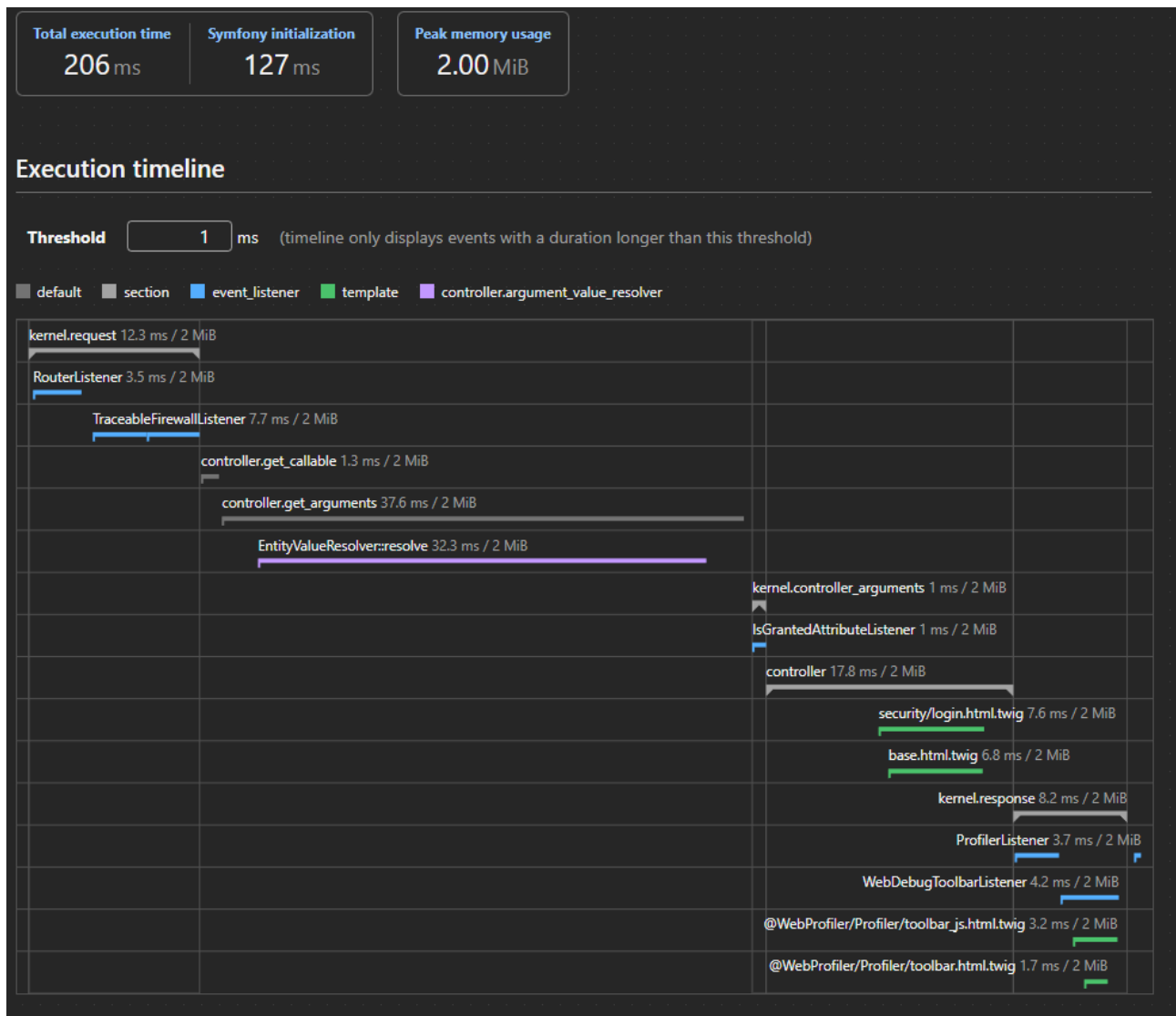
Grâce à ces améliorations, les lacunes techniques ont été corrigés et le projet est désormais à jour et nous permet donc de travailler sur de bonnes bases pour le futur.

En ce qui concerne les performances à proprement parler, les mesures ont été réalisée avec le web profiler de Symfony. Voici deux exemples mesures réalisées sur la page login :

Ancienne version de l'application :



Version actuelle de l'application :



Nous pouvons ici remarquer qu'il y a eu une nette amélioration en terme de mémoire utilisée. Par contre il n'y a pas une grande différence de vitesse entre l'ancienne version de l'application et l'actuelle. Cela s'explique par deux raisons :

Tout d'abord, nous avons ici affaire à une application très basique, qui n'implique pas une grande complexité de code, ce qui veut dire que malgré les upgrades réalisées les différences de performances sont à peine notables. Améliorer et optimiser le code afin d'avoir une nette amélioration des performances demanderait des outils d'analyse poussés, j'en viens à mon deuxième point.

L'utilisation de l'outil Blackfire permettrait, même dans une codebase simple comme celle que l'on a ici, d'analyser assez finement le code afin de pouvoir l'améliorer et par conséquent améliorer les performances de l'application. Malheureusement il n'existe plus de formules gratuites d'essai pour l'outil Blackfire. L'analyse des performances a donc dû se faire avec le profiler de Symfony, outil qui ne nous donne pas assez de détails nous permettant d'améliorer le code en vue d'optimiser les performances de l'application.

Malgré tout, grâce à la correction de la dette technique avec une version récente de Symfony et PHP ainsi qu'une codebase simple et de qualité (voir chapitre qualité du code), l'application ne souffre en aucun cas de problèmes de performances.

Il reste tout de même des points sur lesquels le site peut être amélioré, comme la mise en place d'un cache sur certaines pages ou encore l'utilisation de la version 8.2 de PHP qui permettrait d'optimiser encore un peu plus les performances de l'application.

Qualité du code

CodeClimate

Afin d'analyser la qualité du code j'ai utilisé CodeClimate. C'est un outil très intéressant qui permet de vérifier la qualité générale du code notamment en termes de maintenabilité. Il permet aussi de détecter certaines redondances dans le code ainsi que certaines mauvaises pratiques à corriger. Voici le rapport que nous donne CodeClimate concernant ce projet :

Breakdown

92 FILES



MAINTAINABILITY



TEST COVERAGE

Codebase summary

MAINTAINABILITY



7 hrs

TEST COVERAGE



Repository stats

CODE SMELLS

4

DUPLICATION

3

OTHER ISSUES

0

On peut voir que le code a un score de maintenabilité A, ce qui veut dire qu'il est aux normes et qu'il n'y a pas de problèmes majeures dans celui-ci.

Correctifs et ajouts



















Fonctionnalités ajoutées

- Seuls les admins peuvent avoir accès aux pages de gestion des utilisateurs.
- Lors de la création ou modification d'un utilisateur, un rôle peut être choisi (admin ou user).
- Lors de la création d'une tâche, celle-ci est automatiquement liée à l'utilisateur authentifié.
- Toutes les tâches déjà créées sont liées à un utilisateur anonyme.
- Une tâche peut être supprimée uniquement par l'utilisateur qui l'a créée.
- Une tâche anonyme peut être supprimée uniquement par un admin.
- Création de la page « Liste des tâches terminées ».
- Personnalisation des pages d'erreur.

Implémentation de tests unitaires et fonctionnels

De nombreux tests avec PHPUnit ont été implémentés pour valider et assurer le bon fonctionnement de l'application à long terme. Les entités ont été testés à l'aide de tests unitaires et des tests fonctionnels sont utilisés pour le comportement des différentes routes de l'application.

Voici le rapport de couverture du code :

		Code Coverage							
		Lines		Functions and Methods		Classes and Traits			
Total		100.00%	177 / 177		100.00%	49 / 49		100.00%	12 / 12
■ Controller		100.00%	76 / 76		100.00%	11 / 11		100.00%	4 / 4
■ DataFixtures		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
■ Entity		100.00%	43 / 43		100.00%	26 / 26		100.00%	2 / 2
■ Form		100.00%	36 / 36		100.00%	4 / 4		100.00%	2 / 2
■ Repository		100.00%	2 / 2		100.00%	2 / 2		100.00%	2 / 2
■ Security		100.00%	20 / 20		100.00%	6 / 6		100.00%	2 / 2
Kernel.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0