



To Do List App

Audit de qualité et performances

Réalisé par	Dubreucq Alexandra
À destination de	ToDo&Co OpenClassRooms
Date de réalisation	12/04/2018

Sommaire

Introduction	3
Mises à jour	3
Qualité de code	3
Tests et couverture de code	3
Outils de qualité de code	4
Codacy	4
CodeClimate	6
Travis CI	6
Performances	7
Audit	7
Pistes d'amélioration	7
Conclusion	7

Introduction

ToDo & Co est 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).

L'application a été développée avec la version 3.1 du framework Symfony.

En tant que développeuse, j'ai été en charge d'améliorer l'application en ajoutant de nouvelles fonctionnalités, en corrigeant des anomalies et en implémentant des tests.

Le but de ce document est de présenter la dette technique de l'application, ainsi qu'un rapport de l'audit de qualité de code et des performances après avoir effectué les modifications demandées. Ce document a également pour but de fournir des pistes pour continuer d'améliorer l'application.

Vous pouvez également trouver une revue complète de ce qui a été modifié ici :

<https://github.com/bhalex/todo-and-co/blob/master/documentation/WhatHaveBeenDone.md>

Mises à jour

Des mises à jour ont été appliquées à l'application, afin de lui assurer une meilleure maintenabilité.

En effet, la première version de l'application avait été réalisée sous Symfony 3.1 qui n'est à ce jour plus maintenue. Bien que la version 4 de Symfony soit sortie, c'est la version 3.4.8 qui a été retenue, afin d'éviter des changements majeurs, et parce qu'elle est encore maintenue par Sensio.

Côté front-end, le framework CSS Bootstrap a également été mis à jour, passant de la version 3.3.7 à la version 4.

La dernière version de Bootstrap n'incluant désormais plus d'icônes, c'est la dernière version de FontAwesome qui a été intégrée au projet.

Toutes les librairies front du projet sont chargées via des CDN afin d'améliorer les temps de chargement des pages.





Qualité de code

Il est important de réaliser des tests unitaires et fonctionnels pour assurer une meilleure maintenabilité du code, car ils permettent d'éviter toute régression.

Tests et couverture de code

Dans le cas de la version initiale de l'application, le taux de couverture du code est de 0.00% car aucun test n'a été effectué.

C:\wamp64\www\symfony\todo-and-co\src\AppBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total	<div></div>	0.00%	0 / 167	<div></div>	0.00%	0 / 34	<div></div>	0.00%	0 / 8
 Controller	<div></div>	0.00%	0 / 88	<div></div>	0.00%	0 / 12	<div></div>	0.00%	0 / 4
 Entity	<div></div>	0.00%	0 / 60	<div></div>	0.00%	0 / 20	<div></div>	0.00%	0 / 2
 Form	<div></div>	0.00%	0 / 19	<div></div>	0.00%	0 / 2	<div></div>	0.00%	0 / 2
 AppBundle.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%

Generated by [php-code-coverage 5.3.0](#) using [PHP 7.0.10](#) with [Xdebug 2.5.4](#) and [PHPUnit 6.5.7](#) at Wed Apr 4 12:08:39 UTC 2018.

Après avoir effectué les modifications demandées, des tests unitaires et fonctionnels ont été réalisés.

La couverture de code est désormais meilleure :

C:\wamp64\www\symfony\todo-and-co\src\AppBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total	<div></div>	97.56%	240 / 246	<div></div>	93.06%	67 / 72	<div></div>	76.92%	10 / 13
Command	<div></div>	98.81%	83 / 84	<div></div>	92.86%	13 / 14	<div></div>	50.00%	1 / 2
Controller	<div></div>	96.97%	64 / 66	<div></div>	84.62%	11 / 13	<div></div>	75.00%	3 / 4
DoctrineListener	<div></div>	100.00%	11 / 11	<div></div>	100.00%	5 / 5	<div></div>	100.00%	1 / 1
Entity	<div></div>	100.00%	46 / 46	<div></div>	100.00%	29 / 29	<div></div>	100.00%	2 / 2
Form	<div></div>	100.00%	15 / 15	<div></div>	100.00%	4 / 4	<div></div>	100.00%	2 / 2
Security	<div></div>	83.33%	15 / 18	<div></div>	60.00%	3 / 5	<div></div>	0.00%	0 / 1
Service	<div></div>	100.00%	6 / 6	<div></div>	100.00%	2 / 2	<div></div>	100.00%	1 / 1
AppBundle.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%

Generated by php-code-coverage 5.3.2 using PHP 7.0.10 with Xdebug 2.5.4 and PHPUnit 6.5.7 at Tue Apr 10 15:57:40 UTC 2018.

Outils de qualité de code

D’autres outils ont été utilisés, afin d’évaluer la qualité du code de l’application, permettant d’y appliquer des modifications pour améliorer la dette technique et respecter les bonnes pratiques en vigueur. Ces outils sont intégrés de façon à effectuer leurs vérifications à chaque nouvelle pull request, nous affichant donc leurs résultats avant de pouvoir effectuer un « merge ».

Codacy

Codacy nous permet de vérifier les bonnes pratiques au niveau du code de notre application, mettant l’accent sur la sécurité, la complexité et mettant en avant toute duplication.

To Do List App a désormais le badge de certification « A ».

bhalexx / todo-and-co

Dashboard master

A

Project Certification

Code Style

96%

Compatibility

100%

Error Prone

100%

Performance

100%

Security

100%

Unused Code

69%

Issues Breakdown

6

Total Issues

Unused Code

4

Code Style

2

Documentation

0

Code style – 96%

Codacy recommande d'éviter les noms de variables trop courts (acceptant pour minimum 3 caractères). Cependant, nous ne modifierons pas ces deux variables et nous garderons « \$id ».

src/AppBundle/Entity/Task.php

Avoid variables with short names like \$id. Configured minimum length is 3.

```
19 private $id;
```

src/AppBundle/Entity/User.php

Avoid variables with short names like \$id. Configured minimum length is 3.

```
24 private $id;
```

Unused code – 69%

Codacy recommande de supprimer tous les paramètres non utilisés dans les méthodes. Cependant, pour les méthodes suivantes, nous les garderons afin d'en respecter la signature.

src/AppBundle/Command/UpdateOldTasksCommand.php

Avoid unused parameters such as '\$input'.

```
32 protected function execute(InputInterface $input, OutputInterface $output)
```

src/AppBundle/Form/TaskType.php

Avoid unused parameters such as '\$options'.

```
12 public function buildForm(FormBuilderInterface $builder, array $options)
```

src/AppBundle/Form/UserType.php

Avoid unused parameters such as '\$options'.

```
27 public function buildForm(FormBuilderInterface $builder, array $options)
```

src/AppBundle/Service/TwigDateRequestListener.php

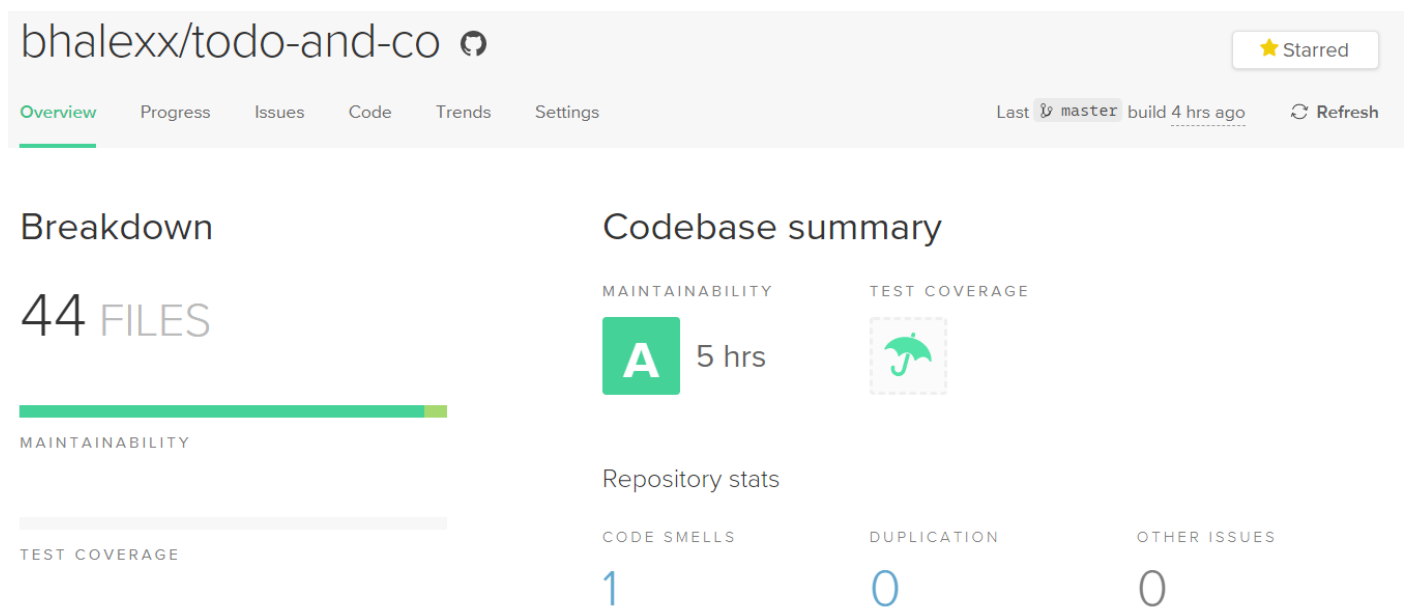
Avoid unused parameters such as '\$event'.

```
17 public function onKernelRequest(GetResponseEvent $event) {
```

CodeClimate

CodeClimate est un autre outil qui, comme Codacy, effectue des vérifications automatisées, offrant un contrôle de qualité du code et mettant l'accent sur la maintenabilité de celui-ci.

To Do List App a désormais le badge « A » en termes de maintenabilité de code.



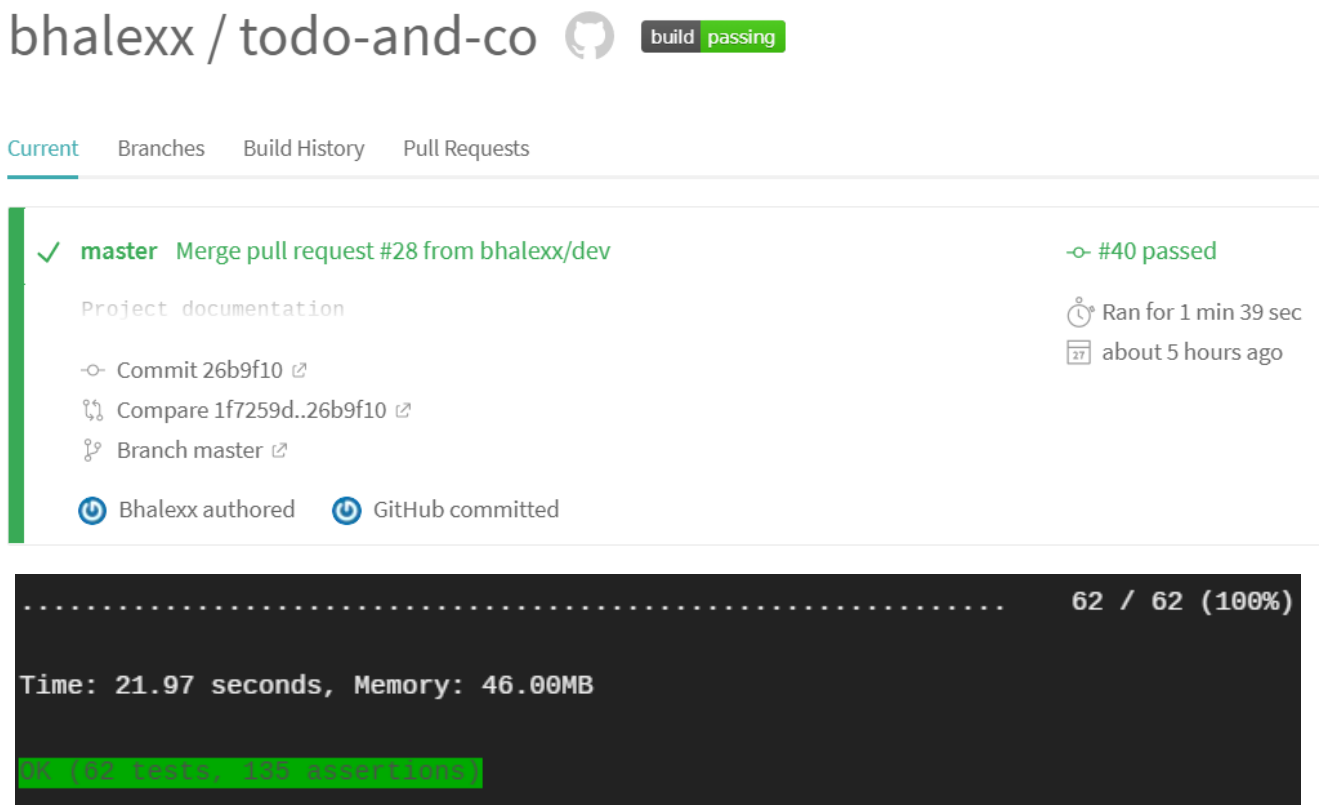
Code smells – 1

Cette alerte concernant le fichier `web/config.php`, propre à Symfony, nous n'y toucherons pas.

Travis CI

Travis CI est un outil d'intégration continue qui exécute automatiquement l'intégralité de nos tests unitaires et fonctionnels à chaque pull request.

La dernière modification effectuée au code est passée au travers de tous les tests :



Performances

Il a été demandé de réaliser un audit de performances une fois les modifications apportées à l'application.

Selon une étude de Google, 40% des visiteurs quittent un site web si le temps de chargement excède 3 secondes. Les performances de chargement d'une application sont donc à surveiller et améliorer dès que possible. Cependant, il ne s'agit pas du seul critère de fidélisation des visiteurs, la performance d'une application pouvant également dépendre de l'infrastructure technique du système, de sa robustesse et de sa capacité à monter en charge.

Audit

Pour parvenir à établir un rapport de performances, l'outil Blackfire, développé par Sensio, a été utilisé.

Attention : cet audit de performances a été réalisé en local et en conditions de développement. Ces résultats ne sont donc à interpréter qu'en termes d'indication, et sont à comparer dans un environnement de production.

Voici le tableau des résultats obtenus sur les requêtes d'URL :

URL	Temps	Mémoire
/login	781 ms	15.6 MB
/	580 ms	17.2 MB
/tasks	833 ms	18.5 MB
/tasks/create	889 ms	22.1 MB
/tasks/{id}/edit	925 ms	22.1 MB
/users	753 ms	18.5 MB
/users/create	950 ms	23.2 MB
/users/{id}/edit	953 ms	23.2 MB

Pour rappel, ces résultats ne sont pas à prendre à la lettre, étant donné que l'audit a été réalisé sur le serveur de développement, incluant du debug.

Pistes d'amélioration

Côté back, bien que les temps de chargements et taux de mémoire soient corrects, nous pouvons très aisément les améliorer en implémentant, comme le suggère Blackfire :

- Le système de cache de Symfony
- Le système de cache de Doctrine

Côté front, nous pouvons également améliorer les temps de chargements et taux de mémoire en apportant les modifications suivantes :

- Minification des fichiers JS et CSS, voire une concaténation de ceux-ci
- Alléger le poids des images ou implémenter un système de lazy loading

INFO

Le lazy loading est une technique qui consiste à retarder le chargement des éléments jusqu'au moment où ils s'apprêtent à être visibles dans la vue.

Conclusion

Les modifications apportées à l'application l'ont rendue davantage maintenable.

Veillez à maintenir les bonnes pratiques mises en place pour la pérenniser et continuer de l'améliorer.