



TODO & CO
PLAN TO BE PRODUCTIVE

Audit de qualité du code et performance

Par Clément Thuet

Introduction

Ce document présente les différences d'un point de vu de la qualité du code et de la performance de l'application entre la version initiale de Todo&Co's to-do list (le MVP) et la version actuelle.

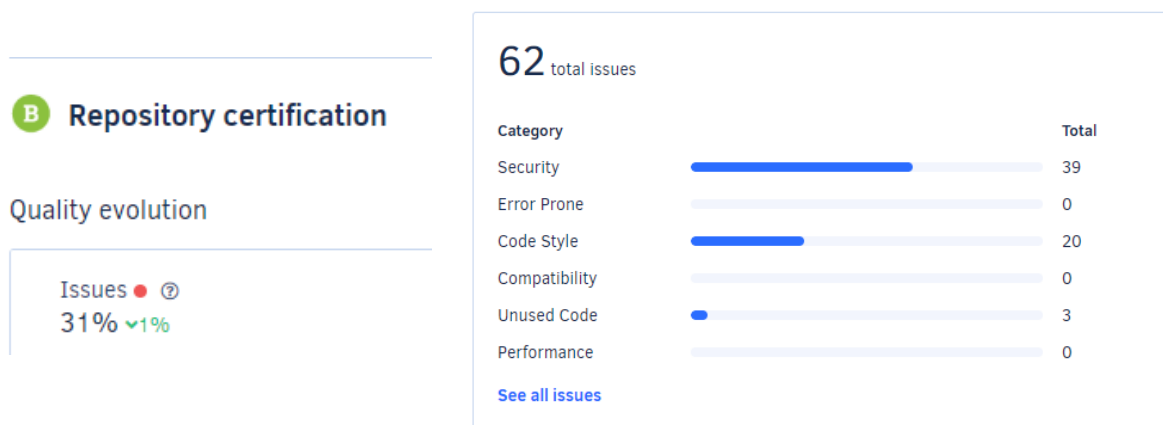
L'objectif lors de l'évolution du projet est de maintenir un niveau de qualité et de performance au moins égal à ce qui se fait actuellement.

Codacy est utilisé pour l'analyse des erreurs de style dans le code, les portions inutilisées, les incompatibilités et les problèmes de sécurité.

Pour étudier la performance l'outil Blackfire a été mis en place, il permet notamment d'analyser le temps d'exécution de chaque tâche et la mémoire utilisée.

I - Qualité du code

Version initiale :



L'application obtenait la note respectable de B avec 62 problèmes détectés, principalement des erreurs de sécurités liées à l'usage de fonction dont l'utilisation est découragée dans les fichiers de Symfony.

Current Issues ▼ master ▼

Filter ≡ All languages ▼ **Security** ▼ All levels ▼ All patterns ▼ All authors ▼ [Clear all](#)

app/AppKernel.php

The use of function `dirname()` is discouraged

```
38 return dirname(__DIR__).'./var/cache/'.$this->getEnvironment();
```

var/SymfonyRequirements.php

The use of function `call_user_func()` is discouraged

```
140 $fulfilled = call_user_func($evaluation, $cfgValue);
```

The use of function `is_dir()` is discouraged

```
420 is_dir(__DIR__.'../vendor/composer'),
```

The use of function `is_dir()` is discouraged

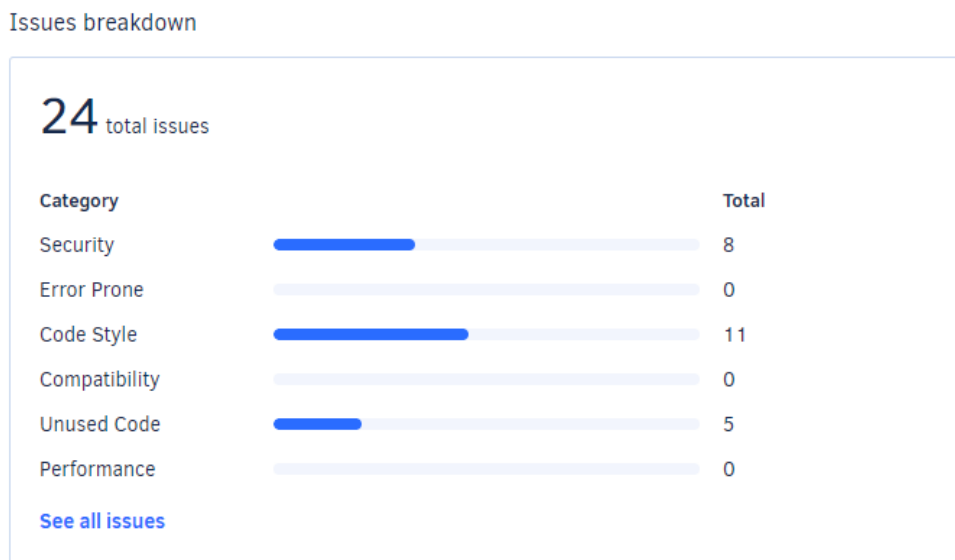
```
426 $cacheDir = is_dir(__DIR__.'../var/cache') ? __DIR__.'../var/cache' : __DIR__.'cache';
```

The use of function `is_writable()` is discouraged

```
429 is_writable($cacheDir),
```

Rien de gênant n'était à déplorer, l'objectif était donc de ne pas introduire de nouvelles erreurs tout en supprimant celles déjà existantes si possible.

Après refonte :



L'application obtient toujours la note de B mais le nombre d'erreur a été pratiquement divisé par 3. Les problèmes restants proviennent majoritairement de point syntaxique discutable comme des noms de variables tels que « \$id » ou « \$em » jugés trop court alors qu'ils sont compréhensibles.

Les différentes analyses sont accessibles à cette adresse :

<https://app.codacy.com/manual/ClementThuets/ToDo-Co/dashboard?bid=17117717>

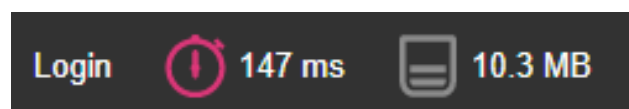
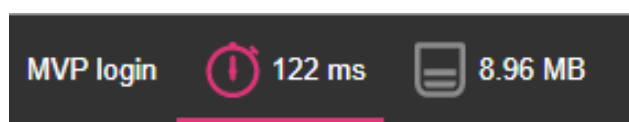
II - Performance

Afin de comparer l'impact des modifications sur le projet et particulièrement le changement de version de Symfony, j'ai utilisé l'outil Blackfire pour analyser différentes actions.

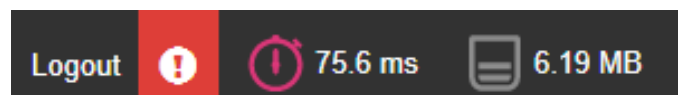
Le nombre de gauche correspond au temps d'exécution en milliseconde et le nombre de droite à la mémoire utilisée pour effectuer l'opération.

Voici le comparatif avec à gauche l'ancienne version (MVP) et à la droite la nouvelle :

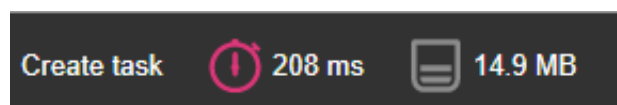
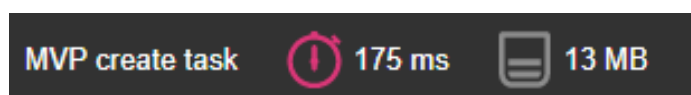
La connexion (-20,4%)



La déconnexion (-24,9 %)



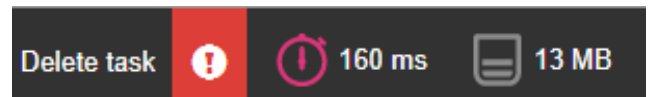
Création d'une tâche (-18,8%)



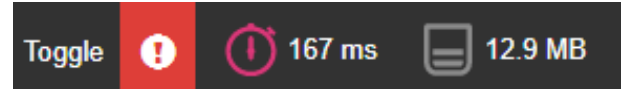
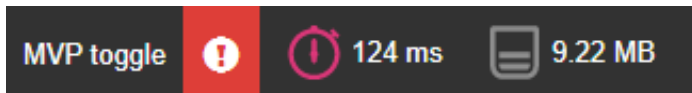
Edition d'une tâche (-22,2%)



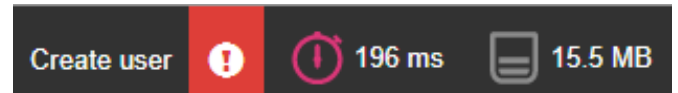
Suppression d'une tâche (-18,5%)



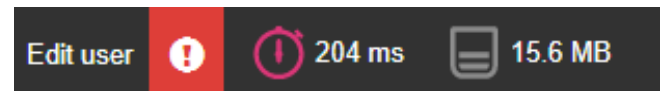
Marquer une tâche comme (non) terminée (-24,6%)



Création d'un utilisateur (-27,2%)



Edition d'un utilisateur (-29,1%)



On se rend compte d'une baisse de performance moyenne en terme de temps d'exécution de 23,21% et l'augmentation de la mémoire utilisée est également proportionnelle.

D'après les analyses Blackfire cette diminution provient principalement de l'Event Dispatcher, le passage à Symfony 4.4 semble donc en être responsable mais de nombreuses fonctionnalités comme les voters ont été rajoutées qui rajoutent aussi de la charge de travail au chargement de chaque page.

L'ajout de calcul afin de déterminer si l'utilisateur connecté a le droit de faire tel ou telle action afin d'afficher, ou non, les boutons de navigation semble peser également au moment du rendu du template twig.

Ces performances seront donc à surveiller attentivement lors des prochaines mises à jour.

Toutes les analyses présentées ci-dessus sont disponibles à l'adresse :
<https://blackfire.io/my/profiles>

III – Couverture de code

Dans l'objectif d'avoir une intégration le plus continue possible et d'éviter l'introduction de bug ce qui conduirait à une récession technique de l'application, il est important d'avoir de nombreux tests unitaires et fonctionnelles qui seront exécutés avant et après chaque mise à jour de l'application pour vérifier que tout fonctionne comme il est attendu.

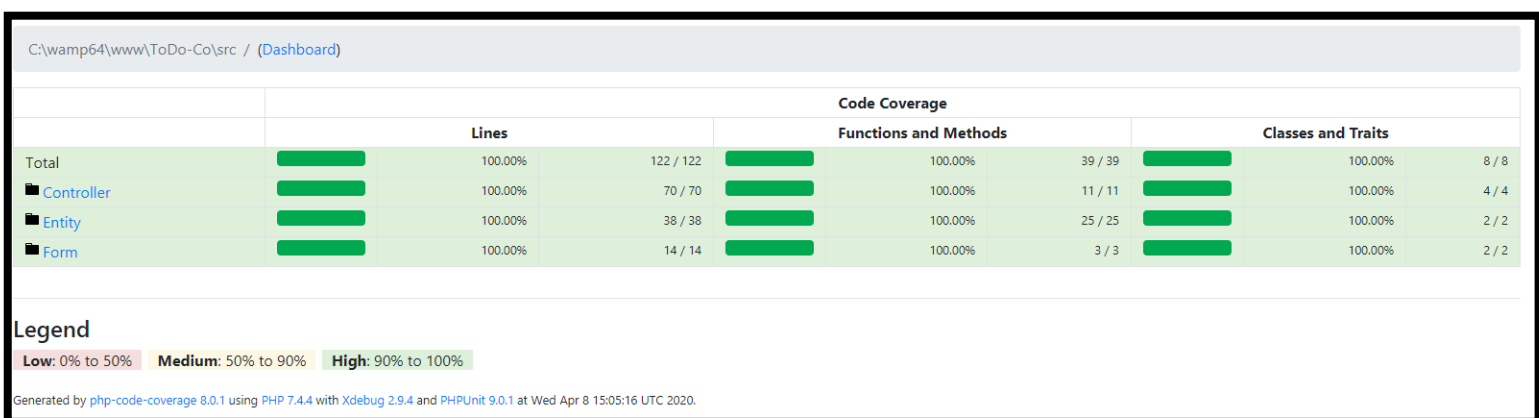
Le fonctionnement de ces tests est décrit dans le document recensant les bonnes pratiques pour contribuer au projet (disponible à cette adresse : <https://github.com/ClementThuet/ToDo-Co/blob/master/docs/contribute.md>)

Afin de s'assurer que le maximum de fonctionnalité fonctionnent correctement j'ai mis en place une analyse du code testé afin de déceler quelles parties du code n'étaient pas testé et corriger ces lacunes.

On utilise pour cela toujours PHPUnit et plus précisément la commande : **phpunit --coverage-html cover**

Celle-ci génère un rapport de couverture de code dans le dossier « cover » à la racine du projet. (Disponible ici : <https://github.com/ClementThuet/ToDo-Co/blob/master/cover/index.html>)

Voici un aperçu du tableau de bord une fois les corrections effectuées :



On voit ensuite au survol de chaque fichier à quel endroit chaque ligne de code est testé.

```
24     */
25     public function listDoneAction()
26     {
27         $this->denyAccessUnlessGranted('seeTask');
28         $tasksDone = $this->getDoctrine()->getRepository('App:Task')->findBy(['isDone'=>true]);
29         return $this->render('task/list.html.twig', ['tasks' => $tasksDone]);
30     }
31
32     /**
33      * @Route("/tasks/create", name="task_create")
34      */
35
36     public function createAction(Request $request, Form $form, EntityManagerInterface $em, Task $task);
37
38     {
39         $form->handleRequest($request);
40
41         if ($form->isSubmitted() && $form->isValid()) {
42             $em = $this->getDoctrine()->getManager();
43             $user = $this->getUser();
44             $task->setUser($user);
45             $em->persist($task);
46             $em->flush();
47             $this->addFlash('success', 'La tâche a été bien été ajoutée.');
```

2 tests cover line 40

- App\Tests\Controller\NavigationControllerUnitTest::testCreateTaskPageIsUp
- App\Tests\Controller\TaskControllerUnitTest::testCreateTask

```
48
49
50         return $this->redirectToRoute('task_list');
51     }
52
53     return $this->render('task/create.html.twig', ['form' => $form->createView()]);
```

Le code coverage est passé d'aucun test écrit à 100% de couverture, il est primordiale de conserver un haut taux de couverture (>70%) afin de s'assurer de la pérennité de l'application.

IV – Amélioration effectuées

Afin de réduire la dette technique de l'application le principal changement a été le passage de la version 3.1.9 à la version 4.4.5 de Symfony. Pour cela il a fallu revoir totalement le chargement des différents modules ainsi que l'architecture des fichiers et la configuration.

Par exemple les fichiers des vues *.html.twig ont été déplacés de */app/Resources/views/* à */templates/* et les Controller de *src/AppBundle/Controller* à *src/Controller* afin de respecter l'architecture de Symfony 4.

Cela facilitera l'évolution future de l'application notamment au moment du passage à la version 5 lorsque cela sera opportun.

De nouvelles fonctionnalités ont été implantés. Il est désormais possible de choisir à la création d'un utilisateur les rôles qui lui sont attribués.

Quand un utilisateur crée une tâche celle-ci lui est associé ce qui n'était pas le cas auparavant.

Une gestion plus fine des droits d'accès a été mise en place grâce aux voters, dorénavant seuls les utilisateurs avec le rôle admin peuvent traiter les tâches créées par l'utilisateur anonyme (concrètement, les tâches déjà existantes qui n'étaient liées à aucun utilisateur).

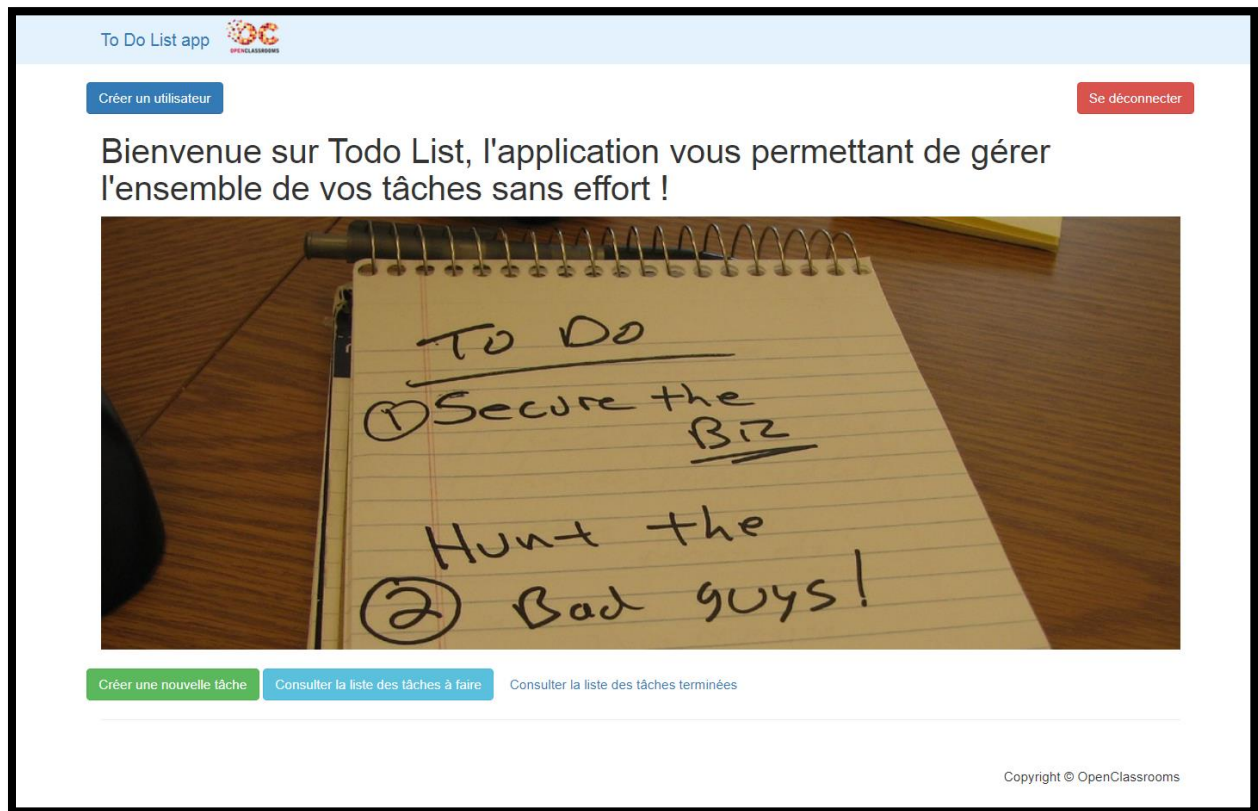
En termes d'interface les menus ont été aérés afin de rendre la navigation plus esthétique, l'image du menu principale a été changée au profit d'une proposition de logo.

Des titres <h1> décrivant le contenu de chaque page ont été mis en place.

Des boutons ont été rajoutés pour revenir en arrière ou au menu principale.

L'affichage des liens pour réaliser les différentes actions comme modifier ou supprimer une tâche se fait désormais en respectant les droits de chaque type d'utilisateur.

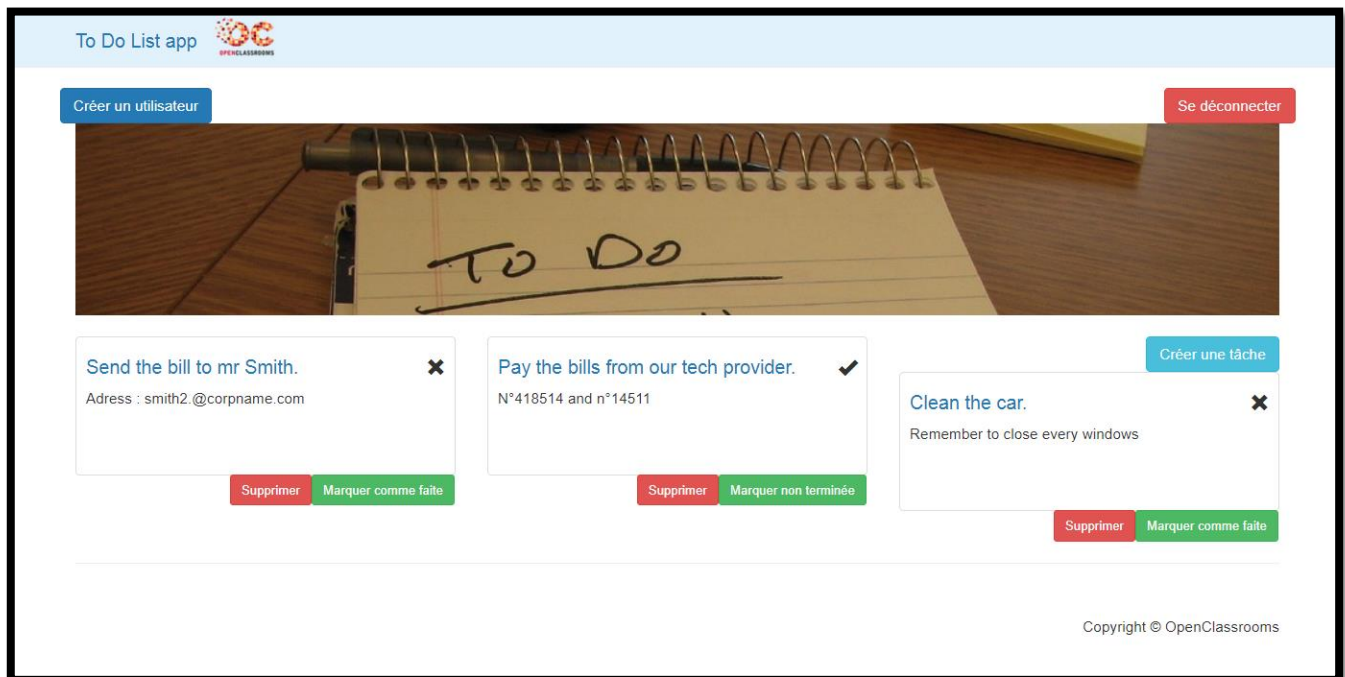
Avant :



Après :



Avant :



Après :

