

# AUDIT DE QUALITES ET PERFORMANCES

Application Web ToDoList

Réalisé par Gallot Benjamin  
Développeur d'Application Web PHP/Symfony

# SOMMAIRE

1. Introduction.....	1
2. Mise à jour version Symfony.....	2
3. Qualité du code.....	3
4. Méthodologie.....	5
5. Performance.....	7
6. Conclusion.....	8

# INTRODUCTION

L'objectif de ce document est de vous faire un état des lieux de la dette technique de la première version de l'application web ToDoList et en parallèle les améliorations qui ont été apportées afin de la réduire le plus possible.

Nous allons donc réaliser un audit de la qualité du code et de la performance de l'application. Afin de réaliser cette évaluation, je me suis basé sur mon expérience en tant que développeur pour percevoir la qualité du code et je me suis également appuyé sur des outils d'analyse de code comme Codacy, Travis-Ci et Blackfire.

# MISE A JOUR VERSION SYMFONY

Le premier constat est que l'application a été réalisé avec la version 3,1 de Symfony qui 'est plus stable. L'objectif est donc de mettre Symfony à jour vers la version 4.1. Nous avons procédé en plusieurs étapes.

Tout d'abord vérifier avant une mise à jour de version s'il y a des dépréciations de code. Nous pouvons vérifier avec le profiler de Symfony ou avec PHPUnit Bridge. Rien de majeur n'a été repéré. Nous avons donc commencé la mise à jour vers une version mineur stable 3.4.

Dans un deuxième temps, nous pouvons mettre à jour vers la version majeur 4.1. Mais avant cela vérifions les dépréciations de code. Plusieurs sont à corriger.

Log Messages		
Info. & Errors	1	Deprecations 10
Debug	33	PHP Notices 0
Container	573	
Log messages generated by using features marked as deprecated.		
Time	Channel	Message
15:10:09	php	User Deprecated: Symfony\Component\HttpKernel\Kernel::loadClassCache() is deprecated since Symfony 3.3, to be removed in 4.0. <a href="#">Show context</a> <a href="#">Show trace</a>
15:10:09	php	User Deprecated: Symfony\Component\HttpKernel\Kernel::doLoadClassCache() is deprecated since Symfony 3.3, to be removed in 4.0. <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-	Using the unquoted scalar value "levent" is deprecated since Symfony 3.3 and will be considered as a tagged value in 4.0. You must quote it in "C:\MAMP\htdocs\ToDoList\app/config/config_dev.yml" on line 20. <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-	Using the unquoted scalar value "levent" is deprecated since Symfony 3.3 and will be considered as a tagged value in 4.0. You must quote it in "C:\MAMP\htdocs\ToDoList\app/config/config_dev.yml" on line 23. <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-	Using the unquoted scalar value "ldoctrine" is deprecated since Symfony 3.3 and will be considered as a tagged value in 4.0. You must quote it in "C:\MAMP\htdocs\ToDoList\app/config/config_dev.yml" on line 23. <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-	The "framework.trusted_proxies" configuration key has been deprecated in Symfony 3.3. Use the Request::setTrustedProxies() method in your front controller instead. <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-	Not setting "logout_on_user_change" to true on firewall "main" is deprecated as of 3.4, it will always be true in 4.0. <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-(2 times)	Autowiring-types are deprecated since Symfony 3.3 and will be removed in 4.0. Use aliases instead for "Psr\Log\LoggerInterface". <a href="#">Show context</a> <a href="#">Show trace</a>
14:59:56	-	Symfony\Component\HttpKernel\DependencyInjection\Extension::addClassesToCompile() is deprecated since Symfony 3.3, to be removed in 4.0. <a href="#">Show context</a> <a href="#">Show trace</a>

## QUALITE DU CODE

Dans cette partie, vous pouvez observer la dette technique de la première version de l'application ToDoList et les corrections effectuées afin de la diminuer.

	Dette Technique	Correction Apportée
<u>Version</u>	Version 3.1 du Framework Symfony.	Mise à jour vers la version 4.1
<u>Authentication</u>	Les tâches n'étaient pas liées aux utilisateurs.	Les tâches créées sont liées à l'utilisateur en cours. Ajout propriété \$user dans entité Task. Les tâches plus anciennes sans auteur peuvent être liées à un utilisateur anonyme avec la commande php bin / console: load anonymous.
	Nous ne sommes pas en mesure de choisir un rôle pour l'utilisateur lors de sa création.	Deux rôles ont été implémentés et peuvent désormais être affectés à l'utilisateur lors de sa création:  •ROLE_USER  •ROLE_ADMIN  Modification du type de formulaire Form \ UserType en ajoutant un champ de rôles.
	Tous les utilisateurs ont été autorisés à accéder à la zone de gestion des utilisateurs.	La zone de gestion des utilisateurs est réservée aux utilisateurs admin (ROLE_ADMIN).
	Aucunes règles pour supprimer ou modifier des tâches.	Création du service TaskVoter.  L'utilisateur peut supprimer seulement les tâches qu'il a créées.  Les tâches liées à l'utilisateur anonyme peut seulement être effacées par des utilisateurs ayant le rôle admin.
<u>Tests</u>	Aucune Couverture	Implementer tests avec PHPUnit Bridge. Couverture à 97% Utilise command vendor/bin/simple-phpunit – coverage-html cov\ pour créer un rapport de couverture en html. Utilisation de Travis.ci avec Github pour la vérification de code de chaque Pull Request.
	Absence de Fixtures	Création de Fixtures avec Doctrine Data Fixtures Bundle.

<u>Refactoring</u>	Code encodepassword dupliqué in UserController.	Creation d'un EventSubscriber avec UserSubscriber.php et declaration du service dans services.yaml.
	Logique métier présent dans les controllers.	Création des FormsHandler pour chaque action des controllers.
	Pages erreurs présentes	Gestion des pages d'erreurs. Création du fichier error403.html.twig pour gérer la page d'erreur en production(par exemple pour les accès refusés).
	Erreur flash message ToggleIsDone or unDone Message inversé.	Corrigé dans TasksController.php
	Propriété \$roles manquante dans l'entité User.	Ajout propriété \$role and ajout nullable =true Creation setRoles().
	@UniqueEntity pour username manquant dans l'entité User et message manquant pour email.	* @UniqueEntity(fields={"username"}, message="Ce nom est déjà utilisé par un utilisateur.") * @UniqueEntity(fields={"email"}, message="Cet email est déjà utilisé par un utilisateur.")
	Dans le controller \$form ->isValid seul est déprécié.	Remplacer par \$form->isSubmitted() && \$form->isValid().
	Dans SecurityController: Request \$request inutile comme argument dans loginAction	Simplification du code and effacement dépendance Request.
	Dans UserController deleteAction() manquant.	Creation de deleteAction() dans UserController and ajout bouton supprimer dans view User list.html.twig. Création de tests fonctionnels pour cette action.
	Le code n'est pas documenté dans les entités.	Chaque getters et setters ont été documentés.
<u>Documentation</u>	Aucune documentation sur l'application TodoList.	Création d'un pdf pour expliquer la gestion de l'authentification. Création d'un fichier markdown pour expliquer comment contribuer au projet. Amélioration du Readme.md Création de Class Diagram – Use Case Diagram – Sequence Diagram - MPD












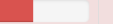











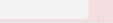
## METHODOLOGIE

Il est important quand on parle de qualité du code de mettre en place une méthodologie de travail afin d'assurer une meilleure maintenabilité du code. Il est important que chaque Pull Request soit analysé pour vérifier la qualité du code. Il y a deux étapes importantes.


### TESTS

Tout d'abord, il est important de tester son code avec des tests fonctionnels et unitaires. Le composant PHPUnit Bridge permet de réaliser de nombreux tests pour toutes les fonctionnalités de l'application. Avec la commande `vendor/bin/simple-phpunit --coverage-html cov/` vous créez un rapport html disponible à l'URI `/cov`. Cela vous donne le taux de couverture du code de votre application. Il est important que ce dernier soit supérieur à 90%.

**Important : Aucune couverture de code sur la première version de ToDoList. Nous y avons remédiés. Veuillez trouver ci-joint le taux de couverture.**

	Code Coverage							
	Lines		Functions and Methods			Classes and Traits		
Total		97.16% 205 / 211		93.65% 59 / 63		88.24% 15 / 17		
Command		100.00% 33 / 33		100.00% 5 / 5		100.00% 1 / 1		
Controller		100.00% 15 / 15		100.00% 10 / 10		100.00% 4 / 4		
Entity		91.43% 32 / 35		91.67% 22 / 24		50.00% 1 / 2		
EventSubscriber		100.00% 11 / 11		100.00% 5 / 5		100.00% 1 / 1		
Form		100.00% 10 / 10		100.00% 2 / 2		100.00% 2 / 2		
Handler		100.00% 87 / 87		100.00% 12 / 12		100.00% 6 / 6		
Security		85.00% 17 / 20		60.00% 3 / 5		0.00% 0 / 1		
AppBundle.php		n/a 0 / 0		n/a 0 / 0		n/a 0 / 0		

Lorsque que le taux de couverture est réalisé. Il faut ensuite utiliser l'outil Travis CI. C'est un outil d'intégration continue qui exécute automatiquement l'intégralité de nos tests unitaires et fonctionnels à chaque pull request.

 **Travis CI**  
Succeeded — 6 hours ago

✓ **Travis CI - Branch**

### Travis CI - Branch

✓ **Success**

🕒 built 6 hours ago in 2 minutes

🔑 34f6050 by @Benj972

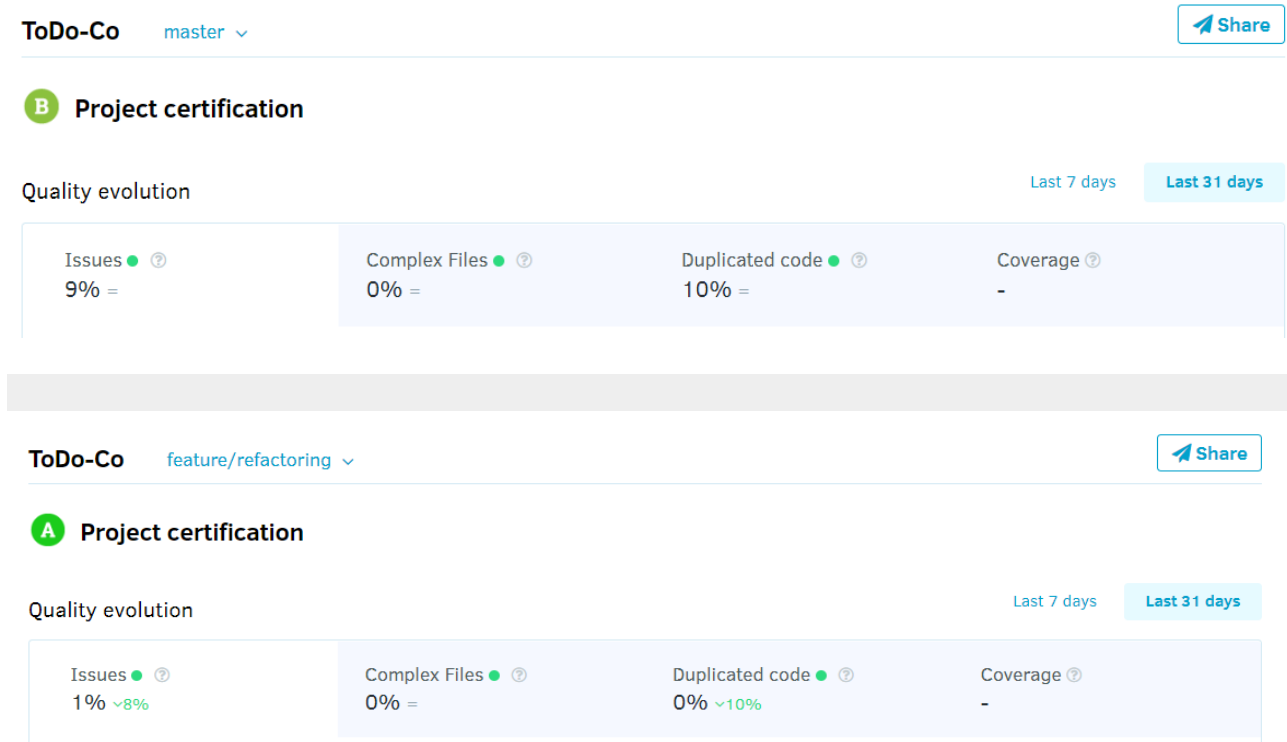
🔗 feature/refactoring

### Build Passed

✓ The build passed.

## EVALUER LE CODE


Dans un deuxième temps, il est important d'utiliser un outil comme Codacy pour vérifier la dette technique du code, les bonnes pratiques, les duplications de code, le code inutilisé entre autres. Cet outil doit également être intégré à chaque Pull Request afin qu'il y ait une vérification automatisée. Nous avons fait une analyse de la première version et à chaque création de nouvelles branches avec des améliorations. Voici un exemple :









*Nous pouvons voir qu'avec les différentes modifications apportées avec la feature/refactoring la qualité du code de l'application est passée de B à A. Le taux de problèmes à chuter de 9% et la duplication de code est passée à nul.*

## RESULTAT PULL REQUEST VALIDE

Add more commits by pushing to the **feature/refactoring** branch on Benj972/ToDo-Co.



-  **All checks have passed** [Hide all checks](#)  
3 successful checks
  -  **Codacy/PR Quality Review** — Up to standards. A positive pull request. [Details](#)
  -  **Travis CI - Branch** Successful in 1m — Build Passed [Details](#)
  -  **continuous-integration/travis-ci/push** — The Travis CI build passed [Details](#)
-  **This branch has no conflicts with the base branch**  
Merging can be performed automatically.

[Merge pull request](#)  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

## PERFORMANCE

Pour parvenir à établir un rapport de performances, l'outil Blackfire développé par SensioLabs a été utilisé. L'objectif est de vérifier le temps de réponse et la consommation de mémoire pour chaque fonctionnalité. Il est important qu'une réponse soit inférieure à trois secondes et qu'il n'y est pas de consommations de mémoire anormales.

*Cet audit de performances a été réalisé sur un serveur web local en conditions de développement et dans un environnement Windows.*

Version MVP(Minimum Viable Product) :

URL	TEMPS	MEMOIRE
/login	317ms	10MB
/	361ms	11,8MB
/tasks	387ms	12MB
/tasks/create	547ms	15MB
/tasks/{id}/edit	501ms	15.1MB
/users	396ms	11,9MB
/users/create	478ms	14,8MB
/users/{id}/edit	507ms	14.9MB

Version améliorée :

URL	TEMPS	MEMOIRE
/login	405ms	12,9MB
/	448ms	14,8MB
/tasks	522ms	14,9MB
/tasks/create	554ms	18,6MB
/tasks/{id}/edit	559ms	18,7MB
/users	458ms	15MB
/users/create	574ms	19MB
/users/{id}/edit	563ms	19,1MB

Nous constatons pour la version MVP que le temps de réponse est plus que correct pour ce type d'application et qu'il n'y a rien d'anormal à signaler sur la mémoire utilisée. Les résultats pour la version améliorée sont sensiblement équivalents. Cela montre que toutes les modifications qui ont été apportées pour réduire la dette technique n'ont pas d'impact négatif sur les performances de l'application ToDoList.



## CONCLUSION

Nous avons donc réaliser un audit sur ToDoList. Nous avons fait un bilan de la dette technique et apportées les améliorations nécessaires. Nous avons été tout particulièrement attentif à la stabilité du Framework, la qualité du code et la documentation nécessaire pour comprendre cette application et avoir la méthodologie adaptée. Cela fût une première approche, des points peuvent encore être amélioré comme l'implémentation du cache pour optimiser au mieux les performances ou encore un travail sur l'ergonomie du site avec le Framework d'interface Bootstrap.