

# AUDIT DE QUALITÉ DU CODE & PERFORMANCE DE L'APPLICATION ToDo&Co

## **SOMMAIRE:**

### **I) QUALITÉ DU CODE**

1) Migration	2
2) CDN jQuery	2
3) PopConfirm.js	2
4) Boutons, formulaires, responsive	3
5) Codacy et Scrutinizer	5
6) PHP_CodeSniffer	5
7) Tests unitaires et fonctionnels	6
8) Intégration continue: Travis	7

### **II) PERFORMANCES DE L'APPLICATION**

1) Utilisation de Blackfire	8
2) A améliorer	8
3) Solutions	9
4) Outil de comparaison de Blackfire	12

# Qualité du code

## 1) MIGRATION A EFFECTUER

Le framework Symfony (version 3.1.10) est utilisé pour ce projet. Il n'y a plus de support. Il est donc conseillé de migrer vers les versions 3.4 voire 4.1.

Une migration a donc été effectuée vers la version 3.4 (version stable supportée jusqu'en 2020/2021).

Symfony Configuration

3.4.16	app	dev
Symfony version	Application name	Environment

Symfony Status	Bugs are fixed until	Security issues are fixed until	
STABLE VERSION	November 2020	November 2021	<a href="#">View roadmap</a>

## 2) CDN jQuery ajouté

Le fichier jQuery appelé dans base.html.twig est absent. Un CDN minifié a été utilisé pour remplacer ce fichier manquant. Un CDN minifié bootstrap.js ainsi que bootstrap.css ont été utilisés. Les fichiers Bootstrap ont donc été supprimés des dossiers web/css et web/js.

## 3) Popconfirm.js ajouté

Avant de supprimer une tâche, il est conseillé d'ajouter un pop-up afin de confirmer la suppression de celle-ci. Un fichier popconfirm.js a été ajouté.



#### **4) Problèmes boutons, responsive, formulaires...**

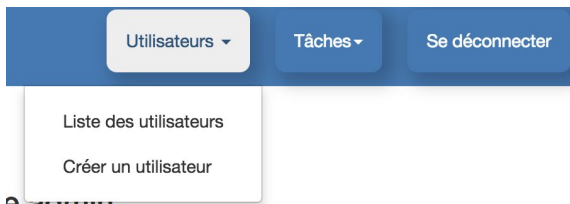
- a) Le bouton “consulter la liste des tâches terminées” ne renvoie vers aucune URL.
- b) Le bouton “consulter la liste des tâches à faire” nous dirige vers toutes les tâches.
- c) 2 boutons “Créer une tâche” apparaissent l’un à côté de l’autre dans le cas où aucune tâche n’est disponible.
- d) Sur grand écran, les 3 tâches sur une rangée ne sont pas bien alignées à cause du même bouton “créer une tâche”.
- e) Sur mobile, il y a bien une tâche par rangée mais les boutons “supprimer” et “marquer non terminée” sont rognés.
- f) Il n’y a pas de menu grâce auquel nous pourrions accéder à chaque partie de l’application (menu à vérifier en responsive)
- g) Sur les tous les écrans, il est préconisé de mettre une tâche sur une seule rangée et de pouvoir afficher le contenu de la tâche dans une fenêtre modale, ce pour une meilleure lecture de celui-ci.
- h) Le formulaire de connexion devrait être amélioré

#### **Solutions proposées:**

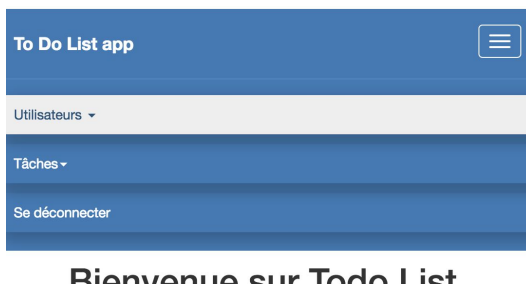
- a) Menu Utilisateur



## b) Menu Administrateur



## c) Menu responsive



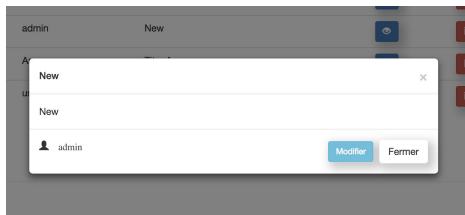
## d) Liste des tâches

<div>Créer une tâche</div> <div>Trier par</div>						
#	Date	Auteur	Titre	Voir	Suppr	Statut
1	26-09-2018	user	Tâche user modifiée			✗
2	27-09-2018	admin	New			✗
3	27-09-2018	Anonyme	Titre Anonyme			✓
4	28-09-2018	user	ef			✓

Chaque tâche se trouve sur une seule ligne et une icône correspond à la lecture, la suppression et le statut de la tâche ( Il suffit de cliquer sur l'icône pour changer le statut de la tâche).

Aussi, un bouton "Trier par" a été ajouté afin de trier les tâches par "à faire", "terminé", "toutes", "par auteur", etc....

## e) Affichage du contenu dans une fenêtre modale




#### f) formulaire de connexion

g) Pour l'instant, seul l'administrateur peut créer un nouvel utilisateur. A terme, il est conseillé d'ajouter un lien d'inscription dans le menu pour que chacun puisse s'inscrire en tant qu'utilisateur simple. Puis, quand celui-ci affichera la liste des tâches, seules ses propres tâches seront affichées.

## **5) Codacy et Scrutinizer**

Codacy a été utilisé et permet d'avoir une idée de la qualité du code en retournant une note.

## **Todo-co\_Project8**

 code quality **A**

Scrutinizer **10.00**

Des erreurs peuvent être détectées ainsi que le non-respect des PSR. Scrutinizer propose des patches de correction.

## **6) PHP\_CodeSniffer**

Le respect des PSR peut être amélioré grâce à ce composant.

```
iMac-de-Lea:Todo-co_Project8 leazygomalas$ vendor/bin/phpcs src/AppBundle/Form
FILE: /Applications/MAMP/htdocs/Todo-co_Project8/src/AppBundle/Form/TaskType.php
FOUND 3 ERRORS AFFECTING 3 LINES
 2 | ERROR | Missing file doc comment
 9 | ERROR | Missing doc comment for class TaskType
11 | ERROR | Missing doc comment for function buildForm()
FILE: /Applications/MAMP/htdocs/Todo-co_Project8/src/AppBundle/Form/UserType.php
FOUND 3 ERRORS AFFECTING 3 LINES
 2 | ERROR | Missing file doc comment
13 | ERROR | Missing doc comment for class UserType
15 | ERROR | Missing doc comment for function buildForm()
Time: 160ms; Memory: 6Mb
```

## 7) Tests unitaires et fonctionnels

Le composant PHPUnit a été utilisé sur ce projet pour les tests unitaires et fonctionnels. Behat n'a pas été utilisé pour les tests fonctionnels mais a été installé en vue éventuellement de son utilisation par la suite.

Le code est donc exécuté afin d'être testé pour vérifier que tout s'est déroulé comme prévu. Un minimum de 70% pour le code-coverage est conseillé.

On vérifie également les fonctionnalités du point de vue utilisateur.

Les tests peuvent être exécutés en ligne de commande

```
iMac-de-Lea:Todo-co_Project8 leazygomalas$ vendor/bin/phpunit
PHPUnit 7.3.5 by Sebastian Bergmann and contributors.

..... 22 / 22 (100%)

Time: 3.33 seconds, Memory: 18.00MB

OK (22 tests, 34 assertions)

Code Coverage Report:
 2018-09-29 22:58:51

Summary:
Classes: 44.44% (4/9)
Methods: 60.38% (32/53)
Lines: 26.27% (62/236)

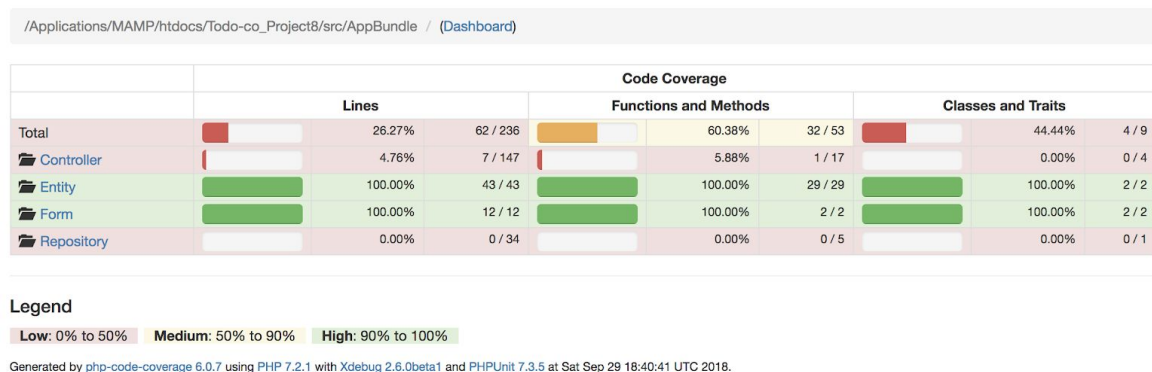
\AppBundle\Controller::AppBundle\Controller\SecurityController
Methods: 33.33% ( 1/ 3) Lines: 77.78% ( 7/ 9)
\AppBundle\Entity::AppBundle\Entity\Task
Methods: 100.00% (14/14) Lines: 100.00% ( 22/ 22)
\AppBundle\Entity::AppBundle\Entity\User
Methods: 100.00% (15/15) Lines: 100.00% ( 21/ 21)
\AppBundle\Form::AppBundle\Form\TaskType
Methods: 100.00% ( 1/ 1) Lines: 100.00% ( 3/ 3)
\AppBundle\Form::AppBundle\Form\UserType
Methods: 100.00% ( 1/ 1) Lines: 100.00% ( 9/ 9)
```

A la reprise du projet, aucun test n'était implémenté.

Nous pouvons voir de façon plus graphique le résultat de ces tests (en cours au moment de l'imprim écran) en faisant :

```
./vendor/bin/phpunit --coverage-html web/test-coverage
```

Il suffit alors de se rendre dans le dossier "web/test-coverage" et d'afficher le fichier "index.html" dans votre navigateur.




Plus de précisions sont apportées en cliquant sur le lien "Dashboard" en haut des résultats.

## 8) Intégration continue: Travis


Un fichier **.travis.yml** a été ajouté à la racine du projet (et une inscription sur Travis a été effectuée) de façon à ce qu'à chaque commit il nous soit rappelé si des erreurs de code sont rencontrées ou non. Aussi, à chaque pull request ou avant déploiement en (pré)production, Travis nous indique si le code est stable ou non et apporte donc des informations indispensables à la pérennité de l'application.

All checks have passed  
2 successful checks

✓

 Travis CI - Branch — Build Passed [Details](#)

✓

 continuous-integration/travis-ci/push — The Travi... [Details](#)

# Performance de l'application

## 1) Utilisation de Blackfire

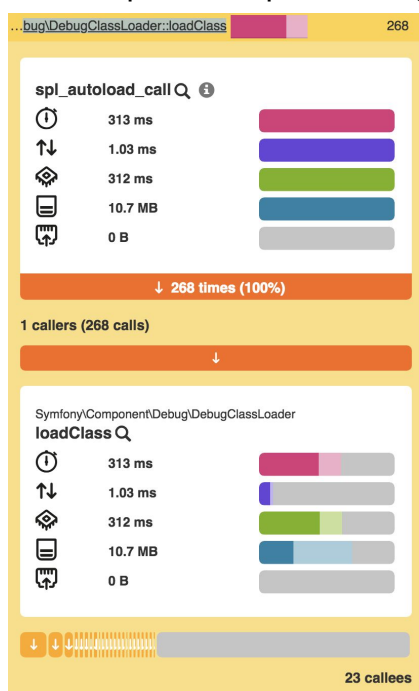
Au-delà du Web Profiler, nous avons utilisé l'outil Blackfire pour effectuer des mesures de performance.

## 2) A améliorer

Pour toutes les pages, le chargement de classes est le plus coûteux en terme de temps et de mémoire comme on peut le voir dans la colonne de gauche.

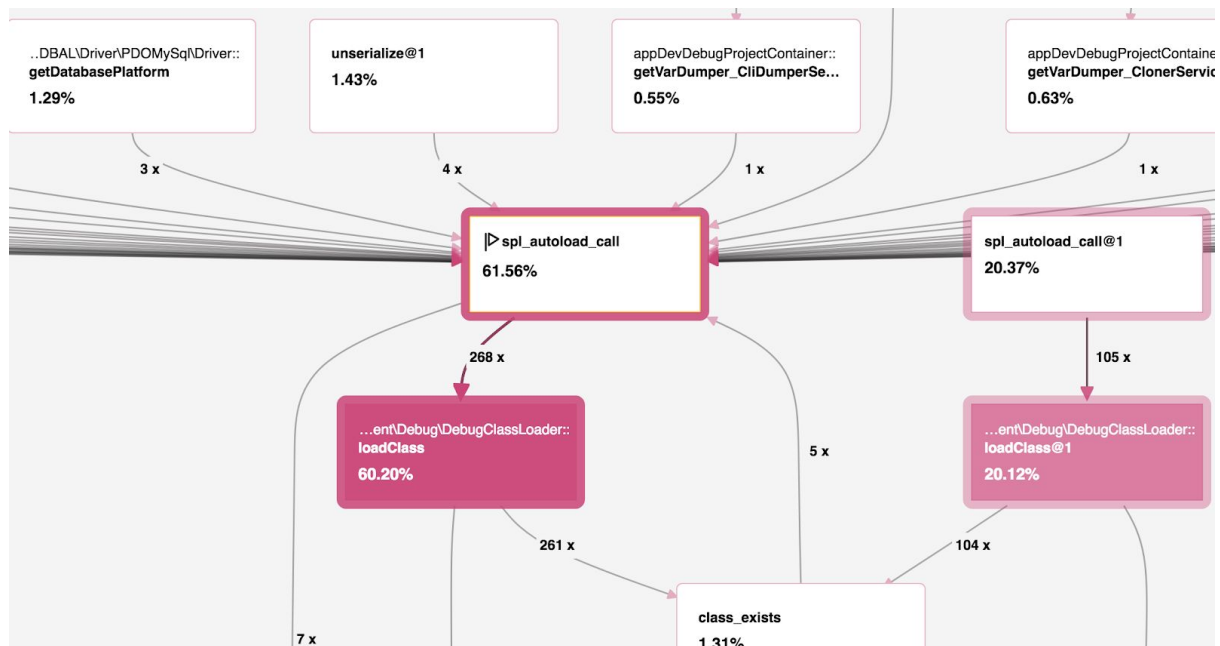
Function / Metric	% Excl. ▾	% Incl.	Calls
...bug\DebugClassLoader::loadClass	<div><div></div></div>		268
...\DebugClassLoader::loadClass@1	<div><div></div></div>		105
...DevDebugProjectContainer::get@2	<div><div></div></div>		66
...ClassLoader::findFileWithExtension	<div><div></div></div>		427

Si on clique sur la première ligne :





On retrouve le temps d'exécution, le I/O, le temps CPU et la mémoire.  
Network a une valeur à 0 en raison du fait que l'application est en local au moment du test.



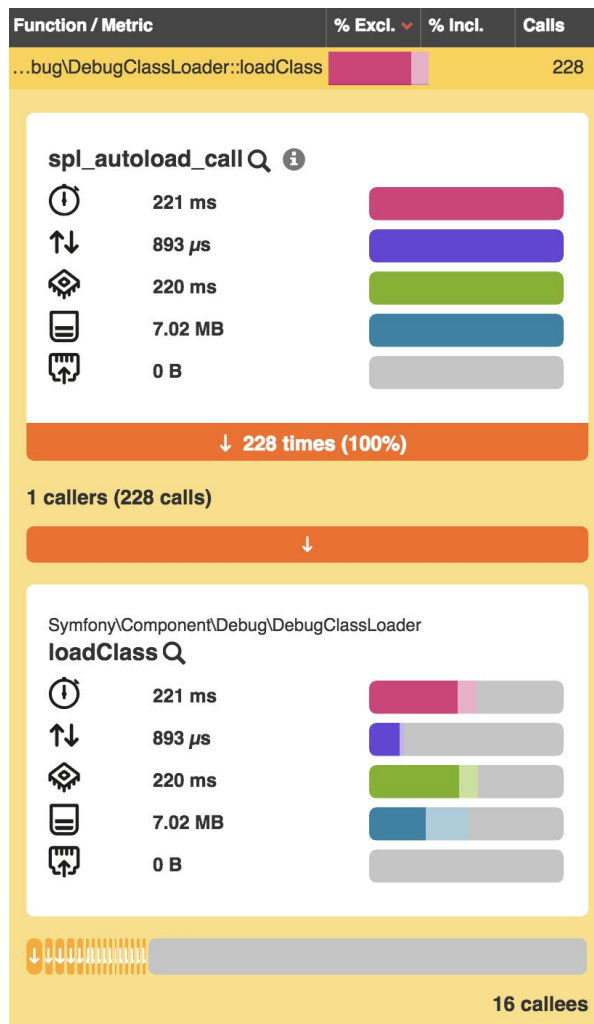
Nous avons un noeud et un chemin en surbrillance. Ils constituent ce qui a le plus d'impact en termes de performances au sein de l'application.  
Le pourcentage représente le coût inclusif de la métrique de la fonction.

### **3) Solutions**

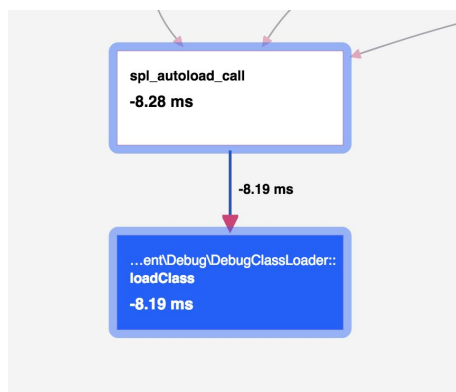
Une commande est capable d'optimiser le temps de chargement des classes (on l'exécute à la racine du projet).

composer dump-autoload -o

On peut déjà constater une baisse des indicateurs suite à cette commande :



Le graphe passe en bleu si une amélioration est constatée.



Même si un gain de temps est constaté, le poids reste sensiblement le même, même avec cette commande.

```
$ composer dump-autoload --optimize --no-dev --classmap-authoritative
```

Activer l'OpCache semble être la bonne solution pour réduire le temps de chargement de classes.<sup>1</sup>

```
[opcache]
; Determines if Zend OPcache is enabled
opcache.enable=1

; Determines if Zend OPcache is enabled for the CLI version of PHP
opcache.enable_cli=1

; The OPcache shared memory storage size.
opcache.memory_consumption=256

; The amount of memory for interned strings in Mbytes.
opcache.interned_strings_buffer=16

; The maximum number of keys (scripts) in the OPcache hash table.
; Only numbers between 200 and 1000000 are allowed.
opcache.max_accelerated_files=10000

; The maximum percentage of "wasted" memory until a restart is scheduled.
opcache.max_wasted_percentage=5

; When this directive is enabled, the OPcache appends the current working
; directory to the script key, thus eliminating possible collisions between
; files with the same name (basename). Disabling the directive improves
; performance, but may break existing applications.
opcache.use_cwd=1

; When disabled, you must reset the OPcache manually or restart the
; webserver for changes to the filesystem to take effect.
opcache.validate_timestamps=0

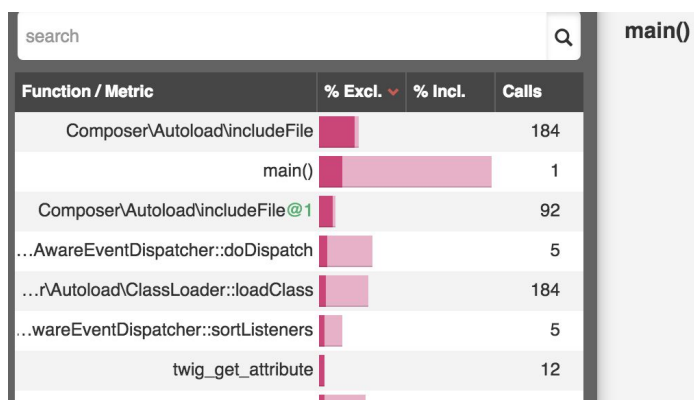
; How often (in seconds) to check file timestamps for changes to the shared
; memory storage allocation. ("1" means validate once per second, but only
; once per request. "0" means always validate)
opcache.revalidate_freq=0

; Enables or disables file search in include_path optimization
opcache.revalidate_path=0

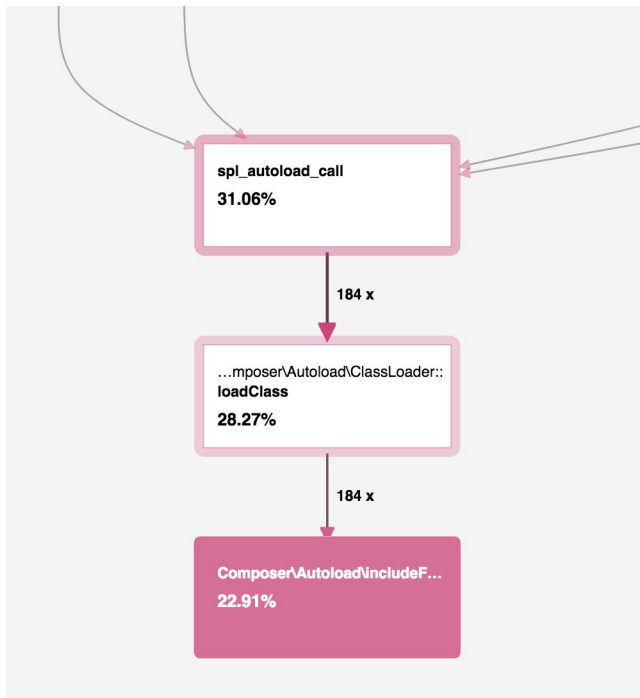
; If disabled, all PHPDoc comments are dropped from the code to reduce the
; size of the optimized code.
opcache.save_comments=1

; If enabled, a fast shutdown sequence is used for the accelerated code
; Depending on the used Memory Manager this may cause some incompatibilities.
opcache.fast_shutdown=1
```

En environnement de production et avec l'OpCache activé, nous obtenons un gain de performance conséquent.



<sup>1</sup> <https://symfony.com/doc/3.4/performance.html>



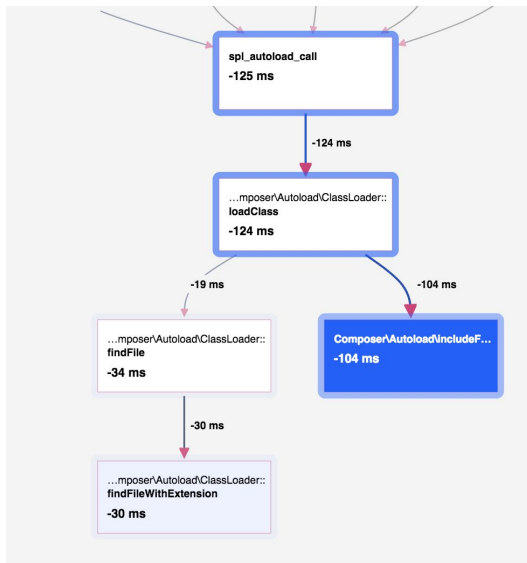
#### 4) Outil de comparaison de Blackfire

Nous avons comparé ces 2 profils avant et après l'activation de l'OpCache en environnement de production.

#### Profil avant activation OpCache

#### Profil après activation OpCache

Comparison <span>-77%</span> <span>↑↓ -83%</span> <span>-77%</span> <span>-78%</span> <span>Eliminated cost</span>			
Functions	search		main()
	Function / Metric	% Incl.	Calls
	main()	<div></div>	+0
	run_init::web/app.php	<div></div>	+0
	AppKernel::handle	<div></div>	+0
	spl_autoload_call	<div></div>	-1



- Gain de 77% sur le temps de génération de la page.  
(77% CPU et 83% file system).
- Gain de 78% sur la mémoire utilisée.