
RAPPORT D'AUDIT DE QUALITE DE CODE ET DE PERFORMANCE DE L'APPLICATION TODOLIST

SOMMAIRE

<u>PRESENTATION DU PROJET</u>	<u>3</u>
1.CONTEXTE.....	3
2.DESCRPTION DU BESOIN	3
3.PLATEFORME TECHNIQUE	3
<u>AUDIT DE QUALITE DE CODE</u>	<u>4</u>
1. L'ANALYSE DU PROJET INITIAL PAR SENSIOLABSINSIGHT	4
2. L'ANALYSE DU PROJET ACTUEL PAR SENSIOLABSINSIGHT	5
3. L'ANALYSE DU PROJET ACTUEL PAR CODACY	7
4.SOLUTIONS FOURNIES.....	7
ERREURS CRITIQUE :	7
ERREURS MAJEURES :	8
ERREURS MINEURES :	9
ERREURS INFOS :	10
5.MODIFICATION DU CODE PHP	11
<u>AUDIT DE PERFORMANCE DE L'API.....</u>	<u>13</u>
1.PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL	13
2. PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL AVEC AMELIORATION	14

PRESENTATION DU PROJET

1.CONTEXTE

Todolist 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).

Le choix du développeur précédent a été d'utiliser le framework PHP Symfony.

Mon rôle ici est donc d'améliorer la qualité de l'application.

2.DESCRPTION DU BESOIN

Le projet est donc de faire les tâches suivantes :

- L'implémentation de nouvelles fonctionnalités ;
- La correction de quelques anomalies ;
- Et l'implémentation de tests automatisés.

Il m'est également demandé d'analyser le projet grâce à des outils vous permettant d'avoir une vision d'ensemble de la qualité du code et des différents axes de performance de l'application.

3.PLATEFORME TECHNIQUE

Framework Symfony :

Version du projet initial : 3.1.*

Version après correction : 3.4.*

Environnement de développement :

Version Wamp 3.0.6 - Windows 64bit

Version Apache 2.4.23

Version de PHP 7.0.10



Version de MySQL : 5.7.14


AUDIT DE QUALITE DE CODE

1. L'ANALYSE DU PROJET INITIAL PAR SENSIOLABSINSIGHT


SensioLabsInsight


DashboardDocumentationPricingMy accountBlog

 tuyetrinhvt / mytodolist 



 Edit project

Using the Free plan, you are limited to one analysis.

 Please upgrade to remove this limit.



Analyzed 4 minutes ago by tuyetrinhvt, duration: a few seconds

 **tuyetrinhvt / mytodolist #1** 

1 Critical3 Major5 Minor5 Info


3.5 days

to get the Platinum Medal

Critical security alerts

- Projects must not depend on dependencies with known security issues

To see this analysis:

 Upgrade to get the Full Report

Details


Source:
[git@github.com:tuyetrinhvt/myt...](https://github.com:tuyetrinhvt/mytodolist)

Project type: Symfony 3/4 web project

Default branch: master

Stats

Lines of code: 749

 **SensioLabsInsight**
No medal yet 2018-02-19

The screenshot displays the SensioLabsInsight interface. At the top, there's a navigation bar with links like 'Dashboard', 'Documentation', 'Pricing', 'My account', and 'Blog'. Below this, a header shows the project name 'tuyettrinhvt / mytodolist #1' and a status 'Analyzed 3 minutes ago by tuyettrinhvt, duration: a few seconds'. The main content area is divided into two columns. The left column contains a sidebar with a '3.5 days' timer, a search bar, and sections for 'Severity' (1 Critical), 'Category' (1 Security), 'Developer' (1 Collective), 'Stats' (Lines of code: 749, Nb of violations: 14), and 'Last commit' (Merge pull request #1 from tuyettrinhvt/create-project by TuyettrinhVO 3 days ago). The right column features a yellow banner prompting to 'Upgrade your plan to get full report access'. Below this, a section titled 'Projects must not depend on dependencies with known security issues' lists four CVEs: CVE-2017-16653, CVE-2017-16654, CVE-2017-16790, and CVE-2017-16652, each with a brief description of the vulnerability. At the bottom of this section, it states 'Time to fix: about 1 day' and provides buttons for 'Comment', 'Ignore', 'Open issue', and 'Permalink'.

La légende des types d'erreurs :


Critique : indique un point risquant de mettre en péril de bon fonctionnement de l'application.

Major : indique un point qui risque de créer des bugs, ou des points de sécurité moins graves que ceux marqués en critique.


Minor : est des points moins importants tels que problèmes de style ou de lisibilité.


Info : est des points d'informations sur les bonnes procédures en vigueur ou conseils pouvant améliorer le contenu.

2. L'ANALYSE DU PROJET ACTUEL PAR SENSIOLABSINSIGHT

 **tuyettrinhvo / mytodolist #1**

Analyzed 2 minutes ago by tuyettrinhvo, duration: a few seconds


 **Edit project**




6.5 hours
to get the Platinum Medal


Stats
Lines of code: 450
Nb of violations: 10

Last commit
`edit config.yml` by tuyettrinhvo 2 minutes ago. **master**

 SensioLabsInsight Gold Medal 2018-03-03


Upgrade your plan to get full report access
Using the Free plan, you are limited to one analysis. To unleash the full power of SensioLabsInsight, you have to upgrade to a paid plan.
[Upgrade to get the Full Report](#)

 **tuyettrinhvo / mytodolist**


 **Edit project**

SensioLabsInsight really shines when each commit of your project is analyzed.
[Enable automatic analysis](#)

Using the Free plan, you are limited to one analysis.
[Please upgrade to remove this limit.](#)



Analyzed 2 minutes ago by tuyettrinhvo, duration: a few seconds

 **tuyettrinhvo / mytodolist #1**


10 Info

6.5 hours to get the Platinum Medal

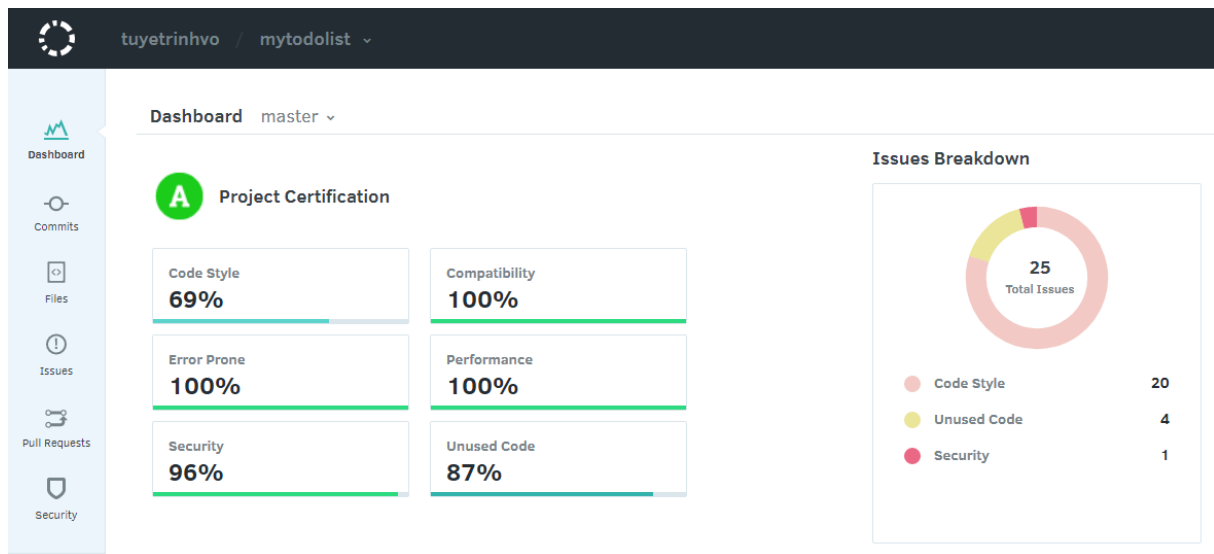
To see this analysis:
[Upgrade to get the Full Report](#)

Details
Source:
git@github.com:tuyettrinhvo/myt...
Project type: Symfony 3/4 web project
Default branch: master

Stats
Lines of code: 450

 SensioLabsInsight Gold Medal 2018-03-03

3. L'ANALYSE DU PROJET ACTUEL PAR CODACY



4.SOLUTIONS FOURNIES

Erreurs critiques :

La version **3.1.*** n'est plus maintenue par la société Sensiolabs, créatrice du Framework **Symfony** depuis 2017. J'ai donc mise à jour la version Symfony à **3.4.***

Dans le fichier `parameters.yml.dist`, j'ai corrigé la série de caractères du paramètre secret :

La norme Symfony Edition est livrée avec un secret par défaut : ThisTokenIsNotSoSecretChangelt.

Ce secret est utilisé pour renforcer les fonctions de sécurité dans Symfony, il est donc important que la valeur par défaut soit modifiée. C'est l'une des premières choses à faire lors de la configuration d'un nouveau projet Symfony.

Il est préférable de le faire avant la mise en ligne d'un projet, mais il peut également être modifié après la mise en ligne d'un projet, bien que certains utilisateurs puissent subir une perturbation mineure.

Erreurs majeures :

Les contrôleurs ne doivent contenir que des méthodes nommées avec le suffixe Action. Cette convention permet de distinguer les méthodes qui vont être appelées par le noyau des autres méthodes qui pourrait être crée au sein d'un contrôleur : Je n'ai pas besoin pour l'instant des méthodes loginCheck() et loginOut() du Security Controller. C'est pour cette raison que je les ai supprimés et mis les routes dans app/config/routing.yml

J'ai ajouté le champ « description » dans le fichier composer.json : **Car ce fichier devrait le contenir**

Dans le fichier config.yml : J'ai déclaré la version serveur de mysql utilisé par Doctrine, la 5.7.14 :

SensioLabsInsight utilise un moteur d'analyse dynamique pour démarrer des applications afin de trouver des bogues et des erreurs impossibles à trouver par une simple analyse statique. C'est pourquoi, lorsque notre application n'est pas amorçable, SensioLabsInsight déclenche une violation de la qualité du code et désactive certaines règles d'analyse.

Dans des circonstances normales, cette erreur est rarement visible dans les analyses effectuées par nos utilisateurs, mais un changement récent introduit par la version 2.5 du projet DBAL est de casser beaucoup de projets sur SensioLabsInsight. Fondamentalement, lors de l'appel de la méthode getDatabasePlatform() dans DBAL 2.5, cet appel doit établir une connexion afin d'évaluer la classe de plate-forme de base de données appropriée Doctrine\DBAL\Connection si elle n'est pas déjà connectée.

La solution consiste à configurer la version du serveur de base de données utilisée par Doctrine : doctrine/dbal/server_version.

Dans le dossier Web, j'ai supprimé le fichier config.php :

Ce fichier config.php situé dans le dossier /web sert à initialiser le projet. Une fois le projet initialisé, il est préférable de le supprimer. Sa présence crée une faille de sécurité pouvant donner des informations confidentielles à une personne malveillante. Par défaut ce fichier n'est accessible qu'en localhost, mais il est fortement conseillé de le supprimer une fois en mode production.

Erreurs mineures :

J'ai créé le dossier AppBundle/Form/Type et déplacé les fichiers TaskType.php et UserType.php dedans :

Lors du premier développement d'un projet, il est tentant de sauvegarder les types de formulaires directement dans le Form/répertoire du bundle, mais nous ne devrions pas le faire. Cela semble agréable et simple, mais au fur et à mesure que le développement progresse, vous devrez probablement étendre le composant de forme d'autres manières. Outre les types de formulaire, nous pouvons étendre le composant de formulaire avec :

Ajout de transformateurs de données personnalisés

Ajout de listes de choix personnalisés

Ajout d'écouteurs d'événement personnalisés

Toutes ces personnalisations, ainsi que les types de formulaires personnalisés, doivent être bien organisées dans la structure de répertoires de votre bundle. La meilleure pratique recommandée consiste à organiser les extensions de formulaire dans les sous-répertoires à l'intérieur du Form/répertoire, en fonction de leur point d'extension.

Dans le dossier Web/app.php, j'ai dé-commentée les lignes HttpCache pour mettre en place le cache Symfony (qui ont été mis en commentaire) :

Le code commenté réduit la lisibilité et réduit la confiance du code pour les autres développeurs. S'il s'agit d'un usage courant pour le débogage, il ne doit pas être validé.

En utilisant un système de contrôle de version, un tel code peut être retiré en toute sécurité. Dans le cas présent, par défaut, le fichier /web/app.php contient deux lignes destinées à offrir le choix d'activer ou non le cache de Symfony.

Dans le fichier config.yml : J'ai déclaré le nom du cookie de session :

Le nom par défaut utilisé pour un cookie de session en PHP est "PHPSESSID". Il est important que ce nom soit changé en quelque chose de spécifique à chaque application pour éviter que les sessions ne saignent d'une application à l'autre. Il est conseillé donc de les personnaliser afin d'éviter qu'une personne malveillante puisse y accéder.

Dans le fichier `base.html.twig`, j'ai inclus liens CDN pour les fichiers `bootstrap.min.js`, `bootstrap.min.css`, `jquery.min.js`.

Erreurs infos :

J'ai personnalisé les pages d'erreurs en créant les fichiers par exemple `error.html.twig` dans dossier `app/Resources/TwigBundle/views/Exceptions`.

J'ai changé les favicons dans le dossier Web : **Cela révèle le moteur backend de l'application et le rend plus vulnérable. Donc il faut utiliser un favicon personnalisé à la place.**

J'ai ajouté les méthodes http GET, POST correspondantes aux routes dans les `_Controller.php` :

Le routeur Symfony prend en charge la requête RESTful en acheminant les demandes de la même ressource vers différentes actions, en fonction de la méthode de requête HTTP. Par exemple, une requête GET /tasks rendre une liste de tâches, tandis qu'une requête POST /tasks peut insérer une nouvelle tâche dans la base de données. Il est important que chacune de vos routes soit spécifique quant aux méthodes autorisées.

Dans le fichier `composer.json` : J'ai ajouté le namespace « `AppBundle\\` » du champ « `autoload` » :

Si l'autoload est défini avec un namespace vide, lors d'un appel à une classe, l'autoload va rechercher dans l'ensemble des fichiers en partant de la racine. Pour éviter cela et donc gagner en performances, il est conseillé de définir un namespace dans lequel l'autoload ira chercher les classes appelées.

J'ai mis à jour le fichier `composer.json` avant de versionner sur Git.

5.MODIFICATION DU CODE PHP

Sur les méthodes de création ou de modification par formulaire : j'ai corrigé `Form::isValid()` en `Form::isSubmitted() && Form::isValid()`. **Car l'utilisation de la méthode `isValid()` avec une form non soumise est obsolète depuis la version 3.2 et lèvera une exception dans la version 4.0.**

J'ai changé `$em` en `$entityManager`.

6.ERREURS SUR CODACY NON RESOLUE

Unused code – Code Inutilisé

Dans `src/AppBundle/Command/EditOldTasksCommand.php` : Codacy nous conseille d'éviter les paramètres inutilisés tels que `'$ input'`.

Dans `src/AppBundle/Controller/SecurityController.php` : Codacy nous conseille d'éviter les paramètres inutilisés tels que `'$ request'`.

Dans `src/AppBundle/Form/Type/TaskType.php` et `UserType` : Codacy nous conseille d'éviter les paramètres inutilisés tels que `'$option'`.

Security

Sur `Web/app_dev.php` : On s'attend à ce que la prochaine chose soit une fonction d'échappement (voir Codex pour 'Data Validation'), pas `'basename'`

Code style – Style de code

La classe AppKernel a un couplage entre la valeur des objets de 14. Codacy nous conseille de réduire le nombre de dépendances de moins de 13.

Dans `src/AppBundle/Commande/EditOldTasksCommand.php` et `tests/AppBundle/Controller/TaskControllerTest.php` : La méthode `execute` utilise une expression `else`. `Else` n'est jamais nécessaire et nous pouvons simplifier le code pour travailler sans `else`.

Codacy nous conseille d'éviter les variables avec des noms courts comme `$id`. La longueur minimale configurée est 3.

La classe `TaskControllerTest` a 11 méthodes non `getter` et `setter`. Codacy nous conseille de refactoriser `TaskControllerTest` pour conserver le nombre de méthodes inférieur à 10.

La classe `SymfonyRequirements` a une complexité globale de 51 qui est très élevée. Le seuil de complexité configuré est 50.

La méthode `__construct()` a une complexité `NPath` de 2580480. Le seuil de complexité `NPath` configuré est 200.

La méthode `__construct()` a 383 lignes de code. Le seuil actuel est fixé à 100. Codacy nous conseille d'éviter les méthodes très longues.

La méthode `__construct()` a une complexité cyclomatique de 40. Le seuil de complexité cyclomatique configuré est de 10.

Codacy nous conseille d'éviter d'utiliser l'accès statique à la classe `'DateTimeZone'` dans la méthode `'__construct'`.

Codacy nous conseille d'éviter d'utiliser l'accès statique à la classe 'PDO' dans la méthode '__construct'.

Codacy nous conseille d'éviter d'utiliser l'accès statique à la classe '\ Symfony \ Component \ Intl \ Intl' dans la méthode '__construct'.

AUDIT DE PERFORMANCE DE L'API

1.PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL

Le test de performance est le processus permettant de déterminer la vitesse ou l'efficacité de l'application.

Blackfire Profiler, développé par la société Sensiolabs, est un outil qui instrumentalise les applications PHP pour collecter des données sur les ressources serveur consommées telles que la mémoire, le temps CPU et les opérations d'I/O.

Blackfire peut être utilisé à n'importe quelle étape du cycle de vie de l'application : pendant le développement, le test, la mise en scène et la production, pour profiler, tester, déboguer et optimiser ses performances.

Ci-dessous le tableau récapitulatif des requêtes testés sur l'environnement de production, donc le cache Symfony et httpCache sont activés :

URI	Time	I/O Wait	CPU Time	Memory	Network	SQL Queries
/login	266 ms	218 ms	48.1 ms	8.24 MB	0 B	0 µs / 0 rq
/	362 ms	299 ms	62.9 ms	11.9 MB	753 B	347 µs / 1 rq
/tasks/create	386 ms	318 ms	68.3 ms	14.2 MB	753 B	515 µs / 1 rq
/tasks/1/edit	388 ms	320 ms	67.7 ms	14.1 MB	1.35 kB	842 µs / 2 rq
/tasks	328 ms	276 ms	52 ms	11.3 MB	1.91 kB	1.52 ms / 2 rq
/users/create	386 ms	318 ms	68.3 ms	14.2 MB	753 B	515 µs / 1 rq
/users/1/edit	384 ms	316 ms	68 ms	14.2 MB	1.32 kB	779 µs / 2 rq
/users	308 ms	258 ms	49.7 ms	11.2 MB	2.46 kB	733 µs / 2 rq

2. PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL AVEC AMELIORATION

La performance de l'application après les mises à jour est raisonnable, en plus l'application est testée en local, et sous Windows.

Nous ne pouvons pas vraiment analyser la performance d'une application sans le server réel.

Cependant, j'ai suivi les recommandations du Blackfire, toujours testé sur l'environnement de production avec le cache Symfony et httpCache sont activés, j'ai obtenu ensuite les résultats ci-dessous :

Après avoir mis en place le cache Doctrine orm

L'utilisation d'annotations pour le mappage a un coût : Doctrine doit transformer cette configuration en code PHP standard exécuté par l'application. Dans les applications réelles avec beaucoup d'entités complexes, ce processus de conversion a un impact sévère sur les performances. C'est pourquoi nous devons mettre en cache l'analyse des annotations Doctrine en production.

URI	Time	I/O Wait	CPU Time	Memory	Network	SQL Queries
/login	128 ms	95.2 ms	32.6 ms	7.54 MB	0 B	0 μ s / 0 rq
/	173 ms	133 ms	39.4 ms	10.5 MB	753 B	341 μ s / 1 rq
/tasks/create	223 ms	168 ms	55.3 ms	14 MB	753 B	344 μ s / 1 rq
/tasks/4/edit	225 ms	169 ms	56.9 ms	14 MB	1.41 kB	620 μ s / 2 rq
/tasks	181 ms	136 ms	44.7 ms	11.2 MB	3.47 kB	1.19 ms / 3 rq
/users/create	226 ms	167 ms	59.2 ms	14.2 MB	753 B	310 μ s / 1 rq
/users/1/edit	229 ms	168 ms	61 ms	14.2 MB	1.32 kB	687 μ s / 2 rq
/users	175 ms	133 ms	41.7 ms	11 MB	2.08 kB	626 μ s / 2 rq

Après avoir activé la cache d'opcode PHP (Zend OPcache) en plus :

Chaque fois que PHP charge une classe, Composer doit rechercher le fichier correspondant sur le système de fichiers, ce qui est un processus lent.

Parce que le classmap peut être énorme, il est fortement recommandé d'avoir un cache d'opcode PHP installé (comme Zend OPcache).

URI	Time	I/O Wait	CPU Time	Memory	Network	SQL Queries
/login	33.7 ms	17.6 ms	16.1 ms	1.59 MB	0 B	0 μ s / 0 rq
/	50.4 ms	29.7 ms	20.6 ms	2.29 MB	753 B	317 μ s / 1 rq
/tasks/create	61.5 ms	33.6 ms	27.9 ms	3.14 MB	753 B	333 μ s / 1 rq
/tasks/4/edit	61.6 ms	32.5 ms	29 ms	3.17 MB	1.41 kB	646 μ s / 2 rq
/tasks	52.1 ms	30.3 ms	21.8 ms	2.39 MB	3.47 kB	898 μ s / 3 rq
/users/create	64.4 ms	31.8 ms	32.6 ms	3.33 MB	753 B	352 μ s / 1 rq
/users/1/edit	69.1 ms	35.9 ms	33.1 ms	3.35 MB	1.32 kB	637 μ s / 2 rq
/users	47.4 ms	27.7 ms	19.7 ms	2.31 MB	2.08 kB	619 μ s / 2 rq