

---

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

---

# SOMMAIRE

<b>PRESENTATION DU PROJET .....</b>	<b>3</b>
<b>1.CONTEXTE .....</b>	<b>3</b>
<b>2.DESCRPTION DU BESOIN .....</b>	<b>3</b>
<b>3.PLATEFORME TECHNIQUE .....</b>	<b>3</b>
<b>AUDIT DE QUALITE DE CODE .....</b>	<b>4</b>
<b>1. L'ANALYSE DU PROJET INITIAL PAR SENSIOLABSINSIGHT .....</b>	<b>4</b>
<b>2. L'ANALYSE DU PROJET ACTUEL PAR SENSIOLABSINSIGHT .....</b>	<b>5</b>
<b>3. L'ANALYSE DU PROJET ACTUEL PAR CODACY .....</b>	<b>7</b>
<b>4.SOLUTIONS FOURNIES .....</b>	<b>7</b>
ERREURS CRITIQUES : .....	7
ERREURS MAJEURES : .....	8
ERREURS MINEURES : .....	9
ERREURS INFOS : .....	10
<b>5.MODIFICATION DU CODE PHP .....</b>	<b>11</b>
<b>6.CONSEILS SUR CODACY NON RESOLUS.....</b>	<b>11</b>
UNUSED CODE – CODE INUTILISE .....	11
SECURITY .....	11
CODE STYLE – STYLE DE CODE.....	11
<b>AUDIT DE PERFORMANCE DE L'API .....</b>	<b>13</b>
<b>1. PROFILAGE DE PERFORMANCE BLACKFIRE .....</b>	<b>13</b>
<b>2.COMPRENDRE L'INTERFACE BLACKFIRE .....</b>	<b>14</b>
<b>3.PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL .....</b>	<b>16</b>
<b>4. PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL AVEC AMELIORATION .....</b>	<b>17</b>
4.1.1. MIS EN PLACE LE CACHE DOCTRINE ORM .....	17
4.1.2. MISE EN PLACE LE CACHE DOCTRINE ORM : COMPARAISON .....	18
4.2.1. ACTIVATION DE LA CACHE D'OPCODE PHP (ZEND OPCACHE).....	19
4.2.2. ACTIVATION DE LA CACHE D'OPCODE PHP : COMPARAISON .....	20

---

# 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

The screenshot displays the SensioLabsInsight web application. The header includes the logo and navigation links: Dashboard, Documentation, Pricing, My account, and Blog. The main content area shows the analysis results for the project 'tuyetrinhvt / mytodolist'. A message at the top states: 'Using the Free plan, you are limited to one analysis. Please upgrade to remove this limit.' The analysis was performed 4 minutes ago by 'tuyetrinhvt' and took a few seconds. The project is identified as 'tuyetrinhvt / mytodolist #1'. A progress indicator shows '30/100' with a red 'X' icon. The analysis results are summarized as: 1 Critical, 3 Major, 5 Minor, and 5 Info. A badge indicates '3.5 days to get the Platinum Medal'. Under 'Critical security alerts', it lists: 'Projects must not depend on dependencies with known security issues'. A button 'Upgrade to get the Full Report' is present. The right sidebar contains 'Details' (Source: git@github.com:tuyetrinhvt/myt..., Project type: Symfony 3/4 web project, Default branch: master) and 'Stats' (Lines of code: 749). At the bottom of the sidebar, it shows 'SensioLabsInsight No medal yet 2018-02-19'.

**SensioLabsInsight** Dashboard Documentation Pricing My account Blog

**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**

30/100

1 Critical 3 Major 5 Minor 5 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...

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'. The main header shows the project name 'tuyettrinhvt / mytodolist #1' and a status 'Analyzed 3 minutes ago by tuyettrinhvt, duration: a few seconds'. A sidebar on the left contains a '3.5 days' badge, 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 main content area features a yellow banner prompting an upgrade to a paid plan for full report access. Below this, a section titled 'Projects must not depend on dependencies with known security issues' lists four detected security issues in the 'symfony/symfony' package (version 3.1.6.0):  
 1) CVE-2017-16653: CSRF protection does not use different tokens for HTTP and HTTPS  
 2) CVE-2017-16654: Intl bundle readers breaking out of paths  
 3) CVE-2017-16790: Ensure that submitted data are uploaded files  
 4) CVE-2017-16652: Open redirect vulnerability on security handlers  
 The time to fix is estimated at about 1 day. Action buttons include '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


  
43/100


**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.

  
43/100

Analyzed 2 minutes ago by tuyettrinhvo, duration: a few seconds  
 **tuyettrinhvo / mytodolist #1**


 10 Info

**6.5 hours** to get the Platinum Medal

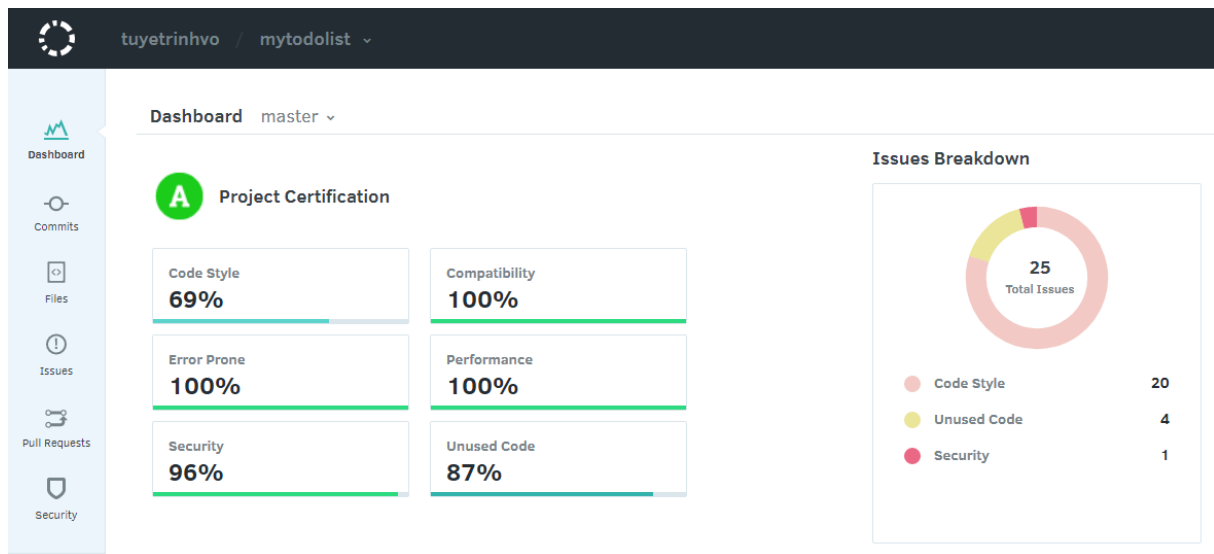
To see this analysis:  
 Upgrade to get the Full Report

**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.CONSEILS SUR CODACY NON RESOLUS

### 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`. Elle 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 DE PERFORMANCE BLACKFIRE

Développé par la société Sensiolabs, le profilage de performance de Blackfire nous permet de rassembler des métriques de performance détaillées à partir de l'exécution de notre code, et visualiser-le dans les « call graphs » interactifs Blackfire.io ; Nous pouvons trouver les « bottlenecks » en un clin d'œil et vérifier l'impact de nos changements en comparant les itérations dans les serveurs de développement, et de production.

Blackfire rassemble des données de profilage sur le temps, le processeur, l'opération d'I/O, la mémoire, les appels réseau, les requêtes HTTP et les requêtes SQL.

Le profilage est fait à la demande. La seule requête indiquant un surcoût est celle qui est profilée, uniquement pour la session de profilage. Aucune autre session ou requête n'est affectée. Nous pouvons utiliser Blackfire en toute sécurité sur les serveurs de production.

Le profilage peut être demandé soit via leur utilitaire CLI, soit avec une extension Chrome intuitive : Blackfire Companion.

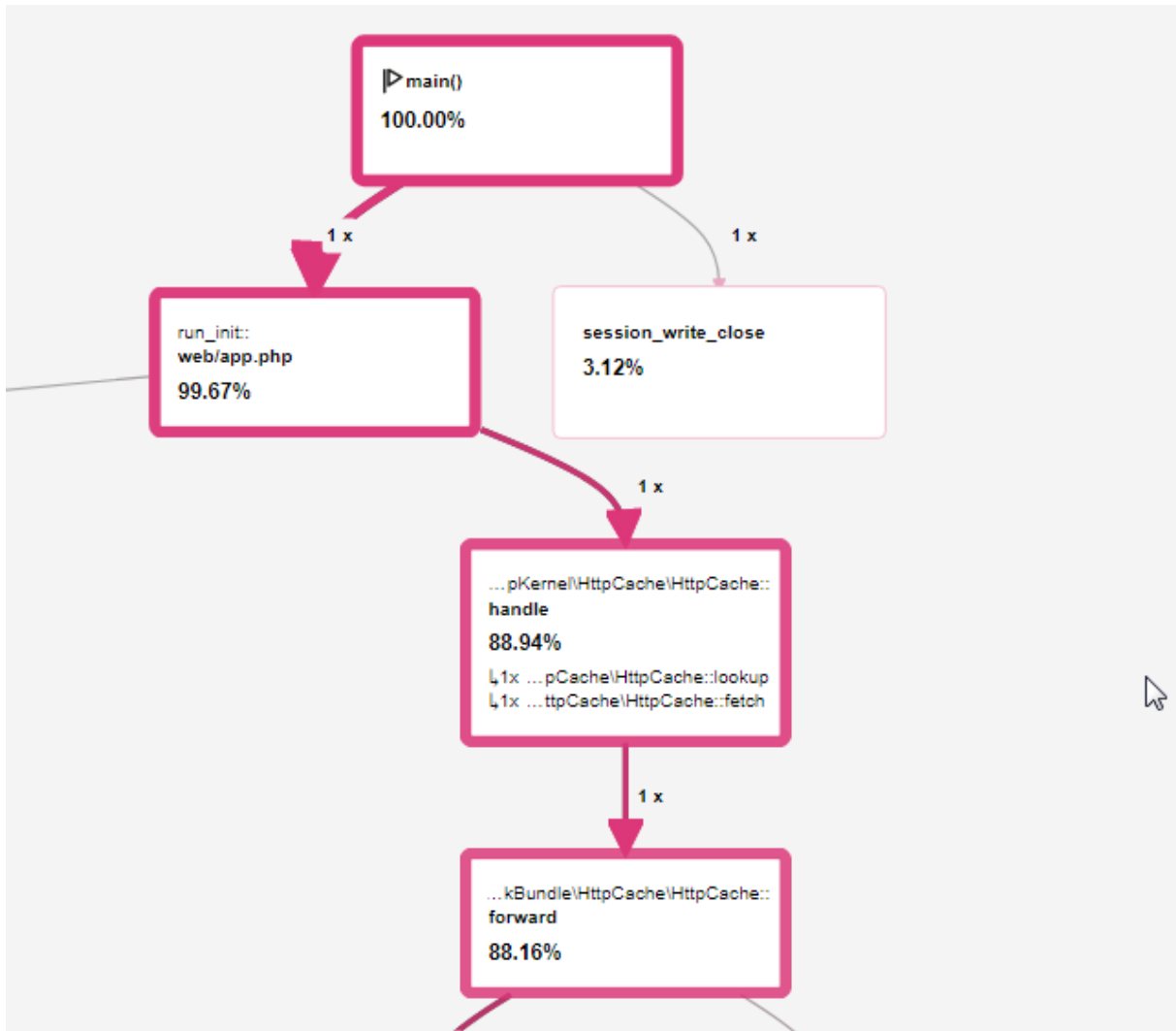
Nous pouvons vérifier l'impact de nos modifications en comparant deux profils. Les différences sont affichées directement via leurs call-graphs.

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.

## 2.COMPRENDRE L'INTERFACE BLACKFIRE

### Call graph

Blackfire affiche les noeuds du graphique d'appel en utilisant un code de couleur où le rouge indique des noeuds de chemin chaud. Plus cette couleur rouge est intense, plus ce nœud a d'importance dans le chemin chaud.



Dans certaines applications, les chemins chauds sont parfaitement valides car ils effectuent une tâche intensive. Cependant, pour la plupart des applications, un chemin d'accès rapide est synonyme d'un « bottleneck » des performances. C'est pourquoi nous devrions toujours commencer par analyser les chemins chauds et vérifier si ces parties de l'application doivent être aussi actives.

Après avoir analysé le diagramme du « call graph », passons en revue les données de la liste des fonctions / méthodes.

### Blackfire nous donne deux sortes de mesures de temps :

**1) Le temps exclusif (Colonne %Excl.)** : combien de temps faut-il pour exécuter une fonction / méthode sans tenir compte du temps passé sur d'autres fonctions / méthodes appelées par lui.

Il nous indique quels sont les nœuds les plus consommés par eux-mêmes.

**2) Le temps inclusif, (Colonne %Incl.)** : le temps total passé à exécuter une fonction / méthode, y compris tout appel externe.

Il nous permet de déterminer le chemin critique de l'application.

Au lieu de se concentrer sur les valeurs absolues du temps exclusif / inclusif, considérons leurs valeurs comme un pourcentage du temps d'exécution total. De cette façon, nous pouvons facilement trouver des fonctions / méthodes qui consomment une quantité disproportionnée de ressources.

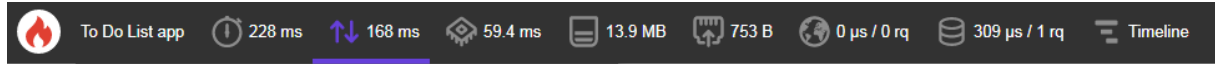
Function / Metric	% Excl. ▼	% Incl.	Calls
Composer\Autoload\includeFile			308
file_exists			562
Composer\Autoload\includeFile@1			162
Composer\Autoload\includeFile@2	13.9 ms	3.61%	
Composer\Autoload\includeFile@2	24.9 ms	556.46%	
Container3rf663x\appProdProjectContainer::get()	% are relative to the total cost of the selected dimension		
...onent\EventDispatcher\EventDispatcher::sortListeners			5

Les données de la liste Fonction / méthode incluent également une colonne appelée **Calls**.

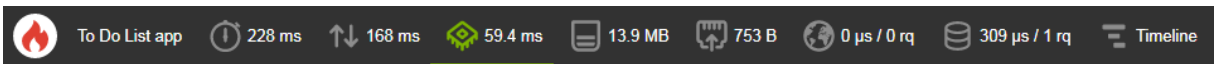
Cette colonne correspond au nombre de fois qu'une fonction/méthode particulière a été appelée. Cela peut être utile pour vérifier qu'il n'y ait pas de ligne avec une valeur disparate, ce qui pourrait être le signe de la présence d'un code défectueux.

## Le temps est en fait composé de deux parties :

1) Le temps d'I/O est l'heure à laquelle le CPU a attendu les opérations d'entrée / sortie (Input / Output) (réseau, disque, ...)



2) Le temps CPU est la durée pendant laquelle le CPU a été utilisée pour le traitement des instructions.



Pour plus information : <https://blackfire.io/docs/reference-guide/analyzing-call-graphs>

## 3.PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL

Ci-dessous le tableau récapitulatif des requêtes testés sur l'environnement de production :

URI	Time	I/O Wait	CPU Time	Memory	Network	SQL Queries
/login	129 ms	97.3 ms	34.5 ms	7.17 MB	0 B	0 μs / 0 rq
/	222 ms	181 ms	41.3 ms	10.8 MB	753 B	334 μs / 1 rq
/tasks/create	231 ms	171 ms	59.8 ms	13.7 MB	753 B	329 μs / 1 rq
/tasks/3/edit	233 ms	171 ms	62.1 ms	13.8 MB	1.42 kB	838 μs / 2 rq
/tasks	190 ms	141 ms	48.9 ms	11.1 MB	3.47 kB	1.82 ms / 3 rq
/users/create	228 ms	168 ms	59.4 ms	13.9 MB	753 B	309 μs / 1 rq
/users/3/edit	239 ms	174 ms	64.3 ms	13.9 MB	1.32 kB	689 μs / 2 rq
/users	177 ms	134 ms	42.9 ms	10.8 MB	2.08 kB	646 μs / 2 rq



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, certaines améliorations pourraient être menées afin d'augmenter la qualité de l'application.

## 4. PROFILAGE VIA BLACKFIRE DU PROJET ACTUEL AVEC AMELIORATION

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

### 4.1.1. 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.

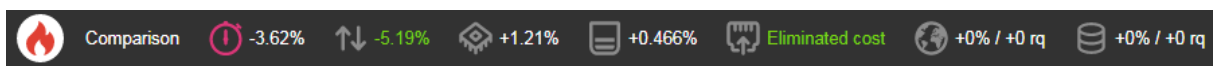
URI	Time	I/O Wait	CPU Time	Memory	Network	SQL Queries
<b>/login</b>	124 ms	92.3 ms	31.9 ms	7.2 MB	0 B	0 µs / 0 rq
<b>/</b>	172 ms	132 ms	40 ms	10.7 MB	753 B	310 µs / 1 rq
<b>/tasks/create</b>	217 ms	164 ms	53.1 ms	13.6 MB	753 B	303 µs / 1 rq
<b>/tasks/3/edit</b>	223 ms	167 ms	55.8 ms	13.7 MB	1.42 kB	725 µs / 2 rq
<b>/tasks</b>	172 ms	129 ms	43.4 ms	10.9 MB	3.47 kB	1.1 ms / 3 rq

<b>/users/create</b>	222 ms	163 ms	58.4 ms	13.8 MB	753 B	309 µs / 1 rq
<b>/users/3/edit</b>	220 ms	160 ms	60.1 ms	13.9 MB	1.32 kB	660 µs / 2 rq
<b>/users</b>	170 ms	130 ms	40.2 ms	10.8 MB	2.32 kB	643 µs / 2 rq

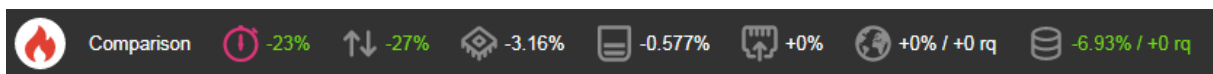
#### 4.1.2. Mise en place le cache Doctrine orm : Comparaison

La comparaison de test **avant** et **après** la mise en cache Doctrine :

URI : /login



URI : /



URI : /tasks/create



URI : /tasks/3/edit



URI : /tasks



URI : /users/create



URI : /users/3/edit



URI : /users/



Nous économisons 23% de temps d'exécution de la page d'accueil, et quelques pourcentages des autres pages.

#### 4.2.1. Activation de la cache d'opcode PHP (Zend OPcache)

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).

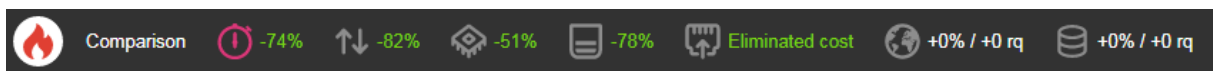
URI	Time	I/O Wait	CPU Time	Memory	Network	SQL Queries
/login	32.6 ms	17 ms	15.6 ms	1.59 MB	0 B	0 µs / 0 rq
/	42 ms	22.7 ms	19.4 ms	2.29 MB	753 B	307 µs / 1 rq
/tasks/create	57.9 ms	29.2 ms	28.8 ms	3.08 MB	753 B	293 µs / 1 rq
/tasks/3/edit	60.1 ms	30.6 ms	29.5 ms	3.17 MB	1.42 kB	604 µs / 2 rq
/tasks	48.4 ms	26.3 ms	22 ms	2.39 MB	3.47 kB	820 µs / 3 rq

<b>/users/create</b>	63.9 ms	31.4 ms	32.6 ms	3.3 MB	753 B	310 µs / 1 rq
<b>/users/3/edit</b>	63.7 ms	30.1 ms	33.7 ms	3.33 MB	1.32 kB	596 µs / 2 rq
<b>/users</b>	44.1 ms	24.3 ms	19.8 ms	2.31 MB	2.32 kB	546 µs / 2 rq

#### 4.2.2. Activation de la cache d'opcode PHP : Comparaison

La comparaison de test **avant** et **après** l'activation de l'extension Zend OPcache :

URI : /login



URI : /



URI : /tasks/create



URI : /tasks/3/edit



URI : /tasks



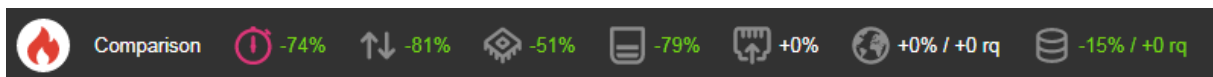
URI : /users/create



URI : /users/3/edit



URI : /users/



Nous pouvons constater que grâce à l'activation de la cache d'opcode PHP sur Wamp64, l'exécution de l'application est plus rapide, et il requiert moins de mémoire.

Pour finir, il serait une très bonne pratique, que d'intégrer les tests de performances aux tests unitaires, comme décrit dans la documentation de blackfire.

Pour en savoir plus sur la gestion de la performance, consulter le <https://blackfire.io/docs/24-days/index>, écrit par Fabien Potencier.