Audit de qualité et améliorations

18/05/2020

Jerome Sauvage

Todo&co



Table des matières

[Introduction 2](#_Toc40954482)

[- 1 – Audit du code initial backend / frontend 2](#_Toc40954483)

[1.1 Rapport CodeClimate : 2](#_Toc40954484)

[1.2 Rapport PHPMetrics et PHPCS 3](#_Toc40954485)

[1.3 Version de Symfony et PHP 5](#_Toc40954486)

[1.4 La sécurité : 6](#_Toc40954487)

[1.5 Mesure de performance BlackFire du code initial 6](#_Toc40954488)

[1.5 Mesure de performance BlackFire du code initial (suite) 7](#_Toc40954489)

[1.6 Principes SOLID 9](#_Toc40954490)

[1.7 Frontend 10](#_Toc40954491)

[1.8 Conclusion 11](#_Toc40954492)

[1.8.1 Backend 11](#_Toc40954493)

[1.8.2 Frontend 11](#_Toc40954494)

[- 2 – Améliorations du code initial backend / frontend 12](#_Toc40954495)

[2.1 Migration de php et symfony 12](#_Toc40954496)

[2.2 Amélioration de la sécurité 13](#_Toc40954497)

[2.3 Amélioration de la qualité 18](#_Toc40954498)

[2.4 Amélioration de la maintenabilité et des performances 21](#_Toc40954499)

[- 3 – Améliorations du frontend 25](#_Toc40954500)

[- 4 – Tests unitaires et fonctionnels 26](#_Toc40954501)

## Introduction

Rappel du projet, ToDo & Co est une startup dont le cœur de métier est une application de gestion des tâches quotidiennes.

L’entreprise vient tout juste d’être crée, et l’application a dû être développée rapidement pour permettre de présenter à de potentiels investisseurs que le concept est viable (Minimum Viable Product ou MVP) avec le Framework Symfony dans sa version 3.1.

La mission qui m’a été confier et de développer de nouvelles fonctionnalités, et d’analyser la qualité du code ainsi que ses performances.

Ce document explique l’audit réalisé ainsi que les points d’amélioration à réaliser.

## - 1 – Audit du code initial backend / frontend

### 1.1 Rapport CodeClimate :

Une image contenant capture d’écran

Description générée automatiquement

Des analyses de qualité de codes ont été effectuées avec CodeClimate.

Ces analyses n’ont pas révélé de problèmes particuliers sur le code de l’application.

Il y a des remontées sur le code du framework Symfony, elles ont été ignorées, ainsi qu’une maintenabilité classée D qui doit être améliorée.

Les PSR1 et PSR2 sont respectés mais la PHPDoc est absente, elle est importante pour une meilleure compréhension et maintenance du code.

De plus elle servira aux futurs développeurs qui pourraient reprendre l’application pour la faire évoluer.

### 1.2 Rapport PHPMetrics et PHPCS

Une image contenant capture d’écran

Description générée automatiquement

Le rapport montre un taux de complexité des classes de 6.07 qui doit être abaissé au maximum.

PHPCS scan le code de chaque classe est donne un rapport très détaillé sur les éventuelles erreurs de code et corrections qui doivent être apportés.

Le rapport de phpcs sur les contrôleurs rapporte que l’Indentation, ouverture et fermeture de parenthèses, annotations peuvent être corrigé grâce à l’outil intégré PHPCBF. Certaines corrections doivent être faite à la main. Voir figure 1, figure 2, figure 3.

Une image contenant assis, table

Description générée automatiquement

Figure

Une image contenant assis, noir, table, ordinateur

Description générée automatiquement

Figure

Une image contenant assis, table, ordinateur

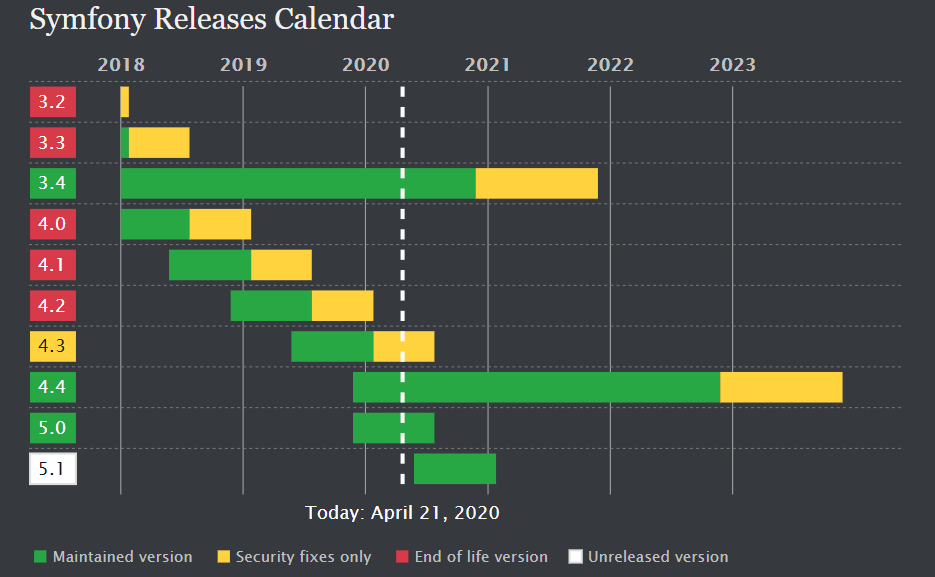
Description générée automatiquement

Figure

### 1.3 Version de Symfony et PHP

La version de Symfony initial de l’application est la 3.1.

Cette version a été mis en ligne en mai 2016 n’est plus maintenu depuis 2017 et n’apparaît plus dans la roadmap.



Passer sur la version 4.4 (LTS = Long Term Support) de Symfony permet de recevoir des mises à jour jusqu’en Novembre 2020 et des correctifs de sécurité jusqu’en Novembre 2022.

Seconde optimisation : Upgrader la version de PHP

La version de PHP utilisé est la 5.5.9, passer à la version 7.2 de PHP augmentera les performances au niveau de l’utilisation des ressources et offrira la possibilité d’utilisé les dernières fonctionnalités de PHP comme le typage des variables qui rendra le code plus stable.

Comparaison des performances de Symfony 3.1 avec PHP 5 et la version 4.4 de Symfony avec PHP 7.3 sur le site php benchmarks :

Une image contenant capture d’écran, différent, groupe, nombreux

Description générée automatiquement

Une image contenant capture d’écran

Description générée automatiquement

1.4 La sécurité :

Le niveau de sécurité initial est insuffisant, il permet d’accéder à certaines pages sans être authentifié.

La page qui liste les utilisateurs /users et qui permet leur modification /users/{id}/edit est accessible sans authentification. Un correctif doit être apporté pour que cette page soit accessible seulement à un utilisateur ayant les droits ROLE\_ADMIN.

Des améliorations sont à apportés au formulaire d’authentification, il n’est pas protégé contre la faille CSRF.

Source : <https://symfony.com/doc/current/security/form_login_setup.html>

### 1.5 Mesure de performance BlackFire du code initial

#### 1.5.1 Mesure de la page Home

Une image contenant moniteur, écran, assis, ordinateur

Description générée automatiquement

#### 1.5.2 Mesure de la page Login

Une image contenant capture d’écran

Description générée automatiquement

#### 1.5.3 Mesure de la page Logout

Une image contenant moniteur, écran, grand, horloge

Description générée automatiquement

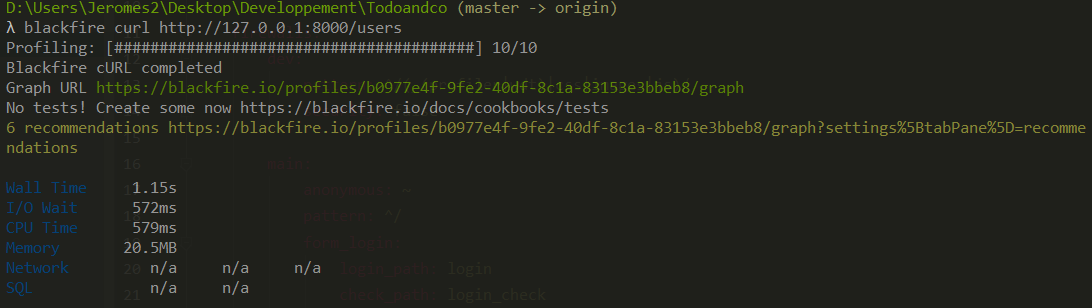
### 1.5 Mesure de performance BlackFire du code initial (suite)

#### 1.5.4 Mesure de la page Tasks

Une image contenant capture d’écran

Description générée automatiquement

#### 1.5.5 Mesure BlackFire de la page Users



#### 

#### 1.5.6 Synthèse et conclusion des mesures

|  |  |  |  |
| --- | --- | --- | --- |
|  | Mesure avant modifications | | Mesure après modifications |
| Routes | Temps s ou ms | Mémoire mb | Voir la partie 2 Améliorations du code initial backend / frontend page 24 |
| / | 1.28 s | 14.1 mb |
| /login | 660 ms | 14.4 mb |
| /logout | 582 ms | 13.8 mb |
| /tasks | 686 ms | 14.1 mb |
| /users | 1.15 s | 20.5 mb |
| Environnement de developpement | Windows | |  |

Grace aux données récoltées par BlackFire on peut comprendre pourquoi une page mettra 1.28 s pour s’afficher et consommera 14.1 mb de mémoire.

Blackfire nous donne les Temps d’accès, l’exécution processeur, l’occupation en mémoire, temps de la requête sql ainsi nous pouvons voir ce qui peut être amélioré pour rendre l’application plus réactive.

L’article « Vitesse de chargement : pourquoi et comment l’améliorer ? » nous informe que « La vitesse de chargement d’un site est la première impression donnée au visiteur. A l’heure où tout doit être instantané, même sur mobile, l’utilisateur n’attend plus. Si votre site met trop longtemps à charger, il est susceptible de quitter la page pour retourner à sa recherche, ce qui va augmenter votre taux de rebond\*. Des études ont montré que 47% des utilisateurs s’attendent à ce qu’une page se charge en moins de 2 secondes et que 40% des internautes ferment une page si elle met plus de 3 secondes à s’afficher. Diminuer le temps de chargement va permettre de retenir l’utilisateur plus longtemps, d’augmenter le nombre de conversions (achats, inscriptions) et de donner une image positive à votre marque. Ainsi, le visiteur sera plus enclin à revenir sur vos pages. »

Source : [Genius Interactive](https://www.genious-interactive.com/vitesse-chargement-pourquoi-comment-ameliorer/?cn-reloaded=1)

## 

### 1.6 Principes SOLID

Un des principes SOLID dit qu’une classe, une fonction ou une méthode doit avoir une et une seule responsabilité. Le code sera implémenté en respectant certains des principes.

|  |  |  |
| --- | --- | --- |
|  | Principes | Définitions |
| S | Responsabilité unique (Single responsibility principle) | Une classe, une fonction ou une méthode doit avoir une et une seule responsabilité |
| O | Ouvert/fermé (Open/closed principle) | Une entité applicative (class, fonction, module ...) doit être ouverte à l'extension, mais fermée à la modification |
| L | Substitution de Liskov (Liskov substitution principle) | Une instance de type T doit pouvoir être remplacée par une instance de type G, tel que G sous-type de T, sans que cela ne modifie la cohérence du programme |
| I | Ségrégation des interfaces (Interface segregation principle) | Préférer plusieurs interfaces spécifiques pour chaque client plutôt qu'une seule interface générale |
| D | Inversion des dépendances (Dependency inversion principle) | Il faut dépendre des abstractions, pas des implémentations |

Source : [Wikipédia](https://fr.wikipedia.org/wiki/SOLID_(informatique))

Dans l’application, le fichier « TaskController » n’implémente pas le principe (SRP) qui explique qu’une classe n’a qu’une seule responsabilité. Le contrôleur de tâches implémente plusieurs responsabilités comme indiqué dans la figure 4.

Une image contenant capture d’écran

Description générée automatiquement

Figure

## 1.7 Frontend

Lors de nos tests de l’interface nous avons constaté un design manquant d’ergonomie et de fluidité.

Une image contenant table

Description générée automatiquement

Figure 5 : page de login

Une image contenant capture d’écran

Description générée automatiquement

Figure 6 : page de création d’un utilisateur

Nous voyons sur la figure 5 ci-dessus le bouton « créer un utilisateur » est situé en haut de page au-dessus de l’image principale de la page et le formulaire de connexion sous l’image.

Sur la figure 6 le formulaire a des proportions trop importantes.

Un test sur la responsive nous montre qu’il faut l’améliorer.

Une mise en forme plus adapté au standard actuel est nécessaire. Un design plus poussé rendra l’application plus attractive et intuitif à utiliser et offrira une meilleure expérience.

## 1.8 Conclusion

|  |  |
| --- | --- |
| Points d’améliorations | Backend |
| Symfony | Passage à la version 4.4 |
| Php | Passage à la version 7.3 |
| Sécurité | Voir la documentation de la sécurité |
| SOLID | Mise en place des principes SOLID |
| PSR | Respect des règles PSR |
| Maintenabilité | Rendre le code plus facilement maintenable |
| Documentation | Mise en place de la documentation |

L’audit de l’application ToDoList nous montre qu’il y a des améliorations à réaliser au niveau backend et frontend. Je propose l’amélioration des points suivant :

### 1.8.1 Backend

### 1.8.2 Frontend

|  |  |
| --- | --- |
|  | Frontend |
| Points d’améliorations |  |
| Design | Amélioration des interfaces |
| Responsive | Amélioration de la responsive |
| Contenu | Ajout d’une page terminée et d’une page d’erreur |

## - 2 – Améliorations du code initial backend / frontend

### 2.1 Migration de php et symfony

Mise à jour de l’environnement de développement et migration vers la version de Symfony 4.4 LTS (long time service).

|  |  |
| --- | --- |
|  | Points d’améliorations traités |
| Symfony | Passage à la version 4.4 |
| Php | Passage à la version 7.3 |

Une image contenant capture d’écran, ordinateur

Description générée automatiquement

Figure Capture d’écran de l’environnement de développement et la version de Symfony de l’application

### 2.2 Amélioration de la sécurité

|  |  |
| --- | --- |
|  | Point d’amélioration traité |
| Sécurité | Voir la documentation de la sécurité |

Cet audit de qualité et d’améliorations est joint avec une documentation sur le fonctionnement et l’implémentation de la sécurité dans une application Symfony que vous trouverez dans le dossier : Todoandco\docs\audit\Sécurité\Documentation\_securité\_de\_symfony.pdf

Les améliorations apportées dans l’application :

* Modification du SecurityController :

Suppression des méthodes sans retour « Login\_Check » et « Logout » qui indique

seulement la route pour la vérification du login et celle de la déconnexion. Les routes ont

été ajoutées dans le fichier « LoginFormAuthenticator.php » pour la connexion (/login)

security.yaml pour la déconnexion (/logout).

* Modification du formulaire d’authentification Login\_Form:

Comme mentionné dans l’audit de code initiale le formulaire ne respecte pas les bonnes

pratiques de Symfony.

Modifications apportées :

Vérification du CSRF Token au moment de l’authentification : permet de protéger les

données envoyées par un utilisateur.



Figure Capture du champs caché « \_csrf\_token »

Afin que le processus fonctionne il a fallu également ajouter la classe

« LoginFormAuthenticator.php » qui étend de la classe

« AbstarctFormLoginAuthenticator.php » qui permet de vérifier toutes les données reçues

comme le nom d’utilisateur, le mot de passe crypté.

* Correction de l’anomalie à l’ajout d’une tâche :

Ajout d’une relation entre l’entitée Task et User.

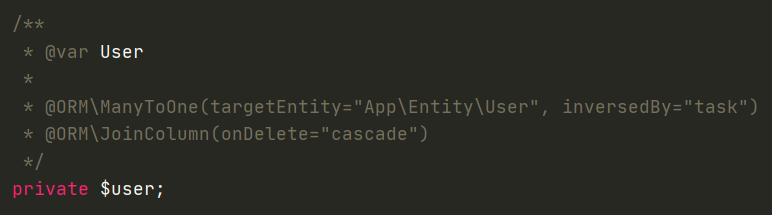


Figure Capture de la relation ManyToOne dans le fichier « src/entity/Task »

Une image contenant alimentation

Description générée automatiquement

Figure Capture de la relation OneToMany dans le fichier « src/entity/User »

Maintenant lors de l’ajout d’une tâche celle-ci est rattaché à l’utilisateur connecté (donc son auteur).

Une image contenant capture d’écran, boule, afficher, joueur

Description générée automatiquement

Figure Capture de la partie de la méthode qui rattache l’utilisateur à une tâche

Avec cette relation entre l’utilisateur et ses tâches celui-ci peut visualiser, modifier, supprimer et marquer uniquement ses tâches. Une vérification est faite avant chaque action, le système vérifie que la tâche appartient bien à l’utilisateur connecté.



Figure Capture de la partie de la méthode qui vérifie si l’utilisateur connecté est bien l’auteur de la tâche

- Type « Hinting » :

La version initiale de PHP (5) ne permettait pas de typer les retours de méthodes ou des

paramètres. Le typage a été ajouté suite au passage de la version 7 de PHP.

- « Validation password » :

Ajout de la validation sur le mot de passe utilisateur car aucun été paramétré initialement.

- La commande « CreateAdminCommand » dans le dossier (App/Command/) pour faciliter l’ajout d’un administrateur à l’application en ligne de commande.

Une image contenant capture d’écran

Description générée automatiquement

Figure Capture d’écran de la commande de création d’un administrateur

- Optimisation des paramètres de PHP :

Les recommandations de Symfony sont de modifier quelques paramètres du fichier PHP.ini pour

optimiser le chargement.

Réglage du RealPath et de l’OP Cache

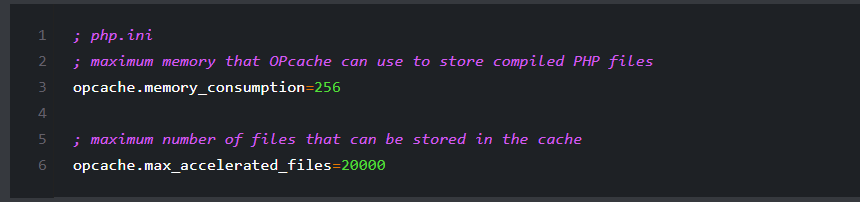


Figure Capture de la configuration de php.ini de la documentation de Symfony

Source : [Documentation Symfony opcache](https://symfony.com/doc/4.4/performance.html#performance-install-apcu-polyfill)

- Mise en place du cache avec REDIS :

Afin d’améliorer les performances de l’application, la mise en cache des données est primordiale.

Le cache REDIS à été mis en place pour optimiser les requêtes Doctrine et obtenir un affichage rapide de la liste des tâches. Plusieurs caches sont gérés par Redis, le « metadataCache », le « queryCache » et le « resultCache ». Paramétrage du « metadataCache » et « queryCache » se situant fichier App/Config/doctrine.yaml.

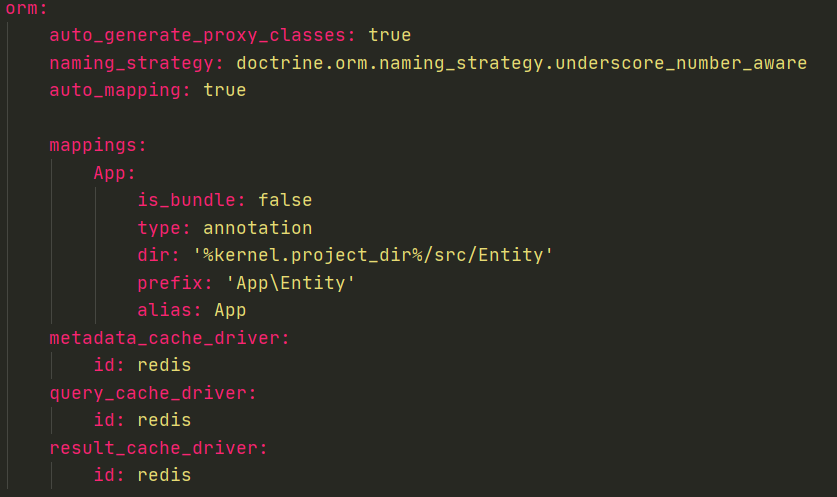


Figure Capture de la configuration de redis pour Doctrine

Création du fichier App/Config/redis.yaml ou l’on indique l’adresse du serveur Redis.

Une image contenant texte, assis, signe, écran

Description générée automatiquement

Figure Capture d’écran de la configuration de Redis

Pour la mise en cache des données récupérer avec Doctrine il l’indiqué dans une requête (DQL ou avec le QueryBuilder).

Une image contenant noir, écran, moniteur, téléphone

Description générée automatiquement

Figure Capture du requête DQL pour la mise en cache dans le fichier TaskRepository

### 2.3 Amélioration de la qualité

|  |  |
| --- | --- |
|  | Point d’amélioration traité |
| SOLID | Mise en place des principes SOLID |
| PSR | Respect des règles PSR |

Les classes TaskController et UserController comme nous l’avons vu dans la page 9, de ce document s’occupait de toutes les tâches. Maintenant, les contrôleurs s’occupent uniquement d’un rôle (create, read, delete, update), c’est le principe de single responsability principle, la logique métier des méthodes traitant les données de formulaires a été déplacer dans des Handler (App/Handler).

Les méthodes s’occupant d’afficher la liste des tâches ou des utilisateurs on dorénavant leur propre contrôleur.

Les contrôleurs étendaient de la classe Controller (classe dépréciée sur les nouvelles versions de Symfony elle et remplacer par la classe AbstractControler).

La classe Controller a donc été supprimer, mais pour de meilleure performance je n’ai pas étendu la classe vers sa remplaçante mais plutôt choisi d’injecter les services vraiment utiles aux contrôleurs.

Ajout de la PHPDOC pour une meilleure lisibilité du code.

Source : [Documentation de Symfony](https://symfony.com/doc/4.4/controller.html#the-base-controller-classes-services)

Une image contenant capture d’écran

Description générée automatiquement

Figure TaskController avant modification

Une image contenant capture d’écran, assis, moniteur, téléphone

Description générée automatiquement

Figure TackCreateController après modification

Une image contenant capture d’écran, assis, noir

Description générée automatiquement

Figure TaskCreateHandler en lien avec le TaskCreateController

Vérification de la qualité du et du respect des PSR avec PHP\_CS\_FIXER

Une image contenant texte, table, assis, noir

Description générée automatiquement

Figure Capture d’écran utilisation de PHP\_CS\_FIXER

[Source : PHP\_CS\_FIXER](https://github.com/FriendsOfPHP/PHP-CS-Fixer)

### 2.4 Amélioration de la maintenabilité et des performances

|  |  |
| --- | --- |
|  | Point d’amélioration traité |
| Maintenabilité | Rendre le code plus facilement maintenable |
| Documentation | Mise en place de la documentation |

#### 2.4.1 Analyse CodeClimate

Avec les différentes modifications apportées au projet le nombre de fichiers passe de 56 à 102, ajout de nouveaux contrôleurs, listerner, handler, repository ainsi qu’un code maintenable plus facilement. Le code venant de librairies externes et celle de Symfony a été ignoré lors des mesures.

Une image contenant capture d’écran

Description générée automatiquement

Figure Capture d’écran nouvelle mesure CoceClimate

#### 2.4.2 Analyse PHPMetrics

Le nouveau rapport de PHPMetrics montre une très bonne maintenabilité des classes (représenté par les cercles vert), un résultat de complexité des classes passant de 6.07 à 2 grâce au respect du principe SOLID : une classe une responsabilité. *Deux erreurs apparaissent elles demandent de passer les classes entity et repository en classe abstraite (ce qui est impossible pour le bon fonctionnement de l’application, je pense donc à un bug de PHPMetrics)*

Une image contenant capture d’écran

Description générée automatiquement

Figure Capture d’écran nouvelle mesure PHPMetrics partie 1

Une image contenant capture d’écran

Description générée automatiquement

Figure Capture d’écran nouvelle mesure PHPMetrics partie 2

#### 2.4.3 Analyse Blackfire

La nouvelle analyse montre un léger gain au niveau du temps d’accès des différentes routes. Le passage à PHP 7 et Symfony 4.4 a eu son effet. L’application gagne en fluidité, la navigation est plus agréable surtout lors de l’affichage des tâches grâce à l’implémentation du cache REDIS, et des optimisations du fichier PHP.INI préconisé par Symfony.

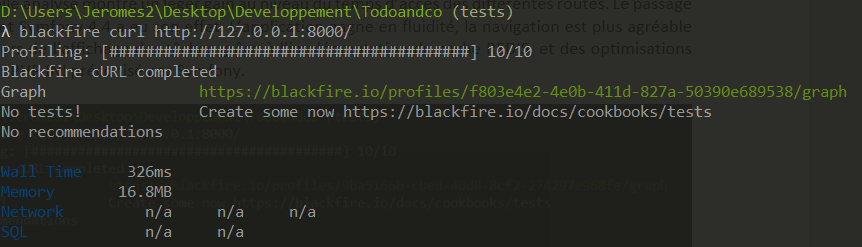


Figure Mesure de la route /

Une image contenant capture d’écran, moniteur, écran, assis

Description générée automatiquement

Figure Mesure de la route /login

Une image contenant capture d’écran, moniteur, écran, assis

Description générée automatiquement

Figure Mesure de la route /logout

Une image contenant capture d’écran, moniteur, écran, noir

Description générée automatiquement

Figure Mesure de la route /tasks

Une image contenant capture d’écran, moniteur, noir, assis

Description générée automatiquement

Figure Mesure de la route /users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Mesure avant modifications | | Mesure après modifications | |
| Routes | Temps s ou ms | Mémoire mb | Temps s ou ms | Mémoire mb |
| / | 1.28 s | 14.1 mb | 326 ms | 16.8 mb |
| /login | 660 ms | 14.4 mb | 336 ms | 17.6 mb |
| /logout | 582 ms | 13.8 mb | 304 ms | 16.8 mb |
| /tasks | 686 ms | 14.1 mb | 442 ms | 20.6 mb |
| /users | 1.15 s | 20.5 mb | 416 ms | 20.6 mb |
| Environnement de développement | Windows | | Windows | |

## - 3 – Améliorations du frontend

Les points suivants ont été améliorés :

* L’application est basée sur la version 3 de bootstrap, je l’ai remplacé par la version 4 via CDN qui améliore le chargement et qui apporte les dernières nouveautés du langage CSS offrant un design plus moderne avec des fonctionnalités plus poussée.
* Suppression du code inutile dans le fichier Shop-homepage.css.
* Lorsqu’un administrateur est connecté deux liens apparaissent dans la barre de navigation : gérer les utilisateurs (si l’utilisateur à le rôle « admin ») et créer un utilisateur.
* Réorganisation de la page d’accueil. Alignement du titre, de l’image et des boutons.
* Implémentation du nouveau logo d’Openclassrooms
* Réorganisation de l’affichage d’une tâche. Les boutons sont maintenant dans le cadre.
* Ajout d’une bannière obligatoire pour mentionner aux visiteurs que l’application utilise des cookies.
* Amélioration de l’ergonomie avec des titres de page dynamique.
* Ajout de la page tâche terminé :
* Le bouton « consulter les tâches terminées » été bien présent mais la page n’existe pas. Ajout d’une information : la date à laquelle la tâche à été marquer comme terminée.
* Ajout d’une page d’erreur.
* Réorganisation générale de l’application pour la rendre responsive.

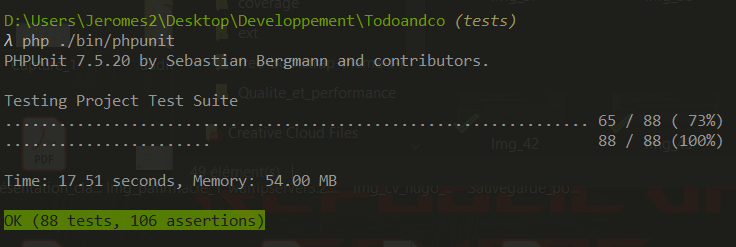
## - 4 – Tests unitaires et fonctionnels

Réalisation des tests unitaires et fonctionnels avec PHP Unit.

La couverture des tests unitaires et fonctionnels avec PHPUnit couvre 100% du code.

Les tests suivants ont été réalisés :

* Réponses et redirection de chaque méthode des contrôleurs.



* Scénario d’ajout, de modification et suppression d’une tâche.
* Scénario d’ajout, de modification et suppression d’un utilisateur.