

## ToDo & Co

Audit de qualité du code & performance



## Table des matières

Context de l'audit.....	2
Audit de qualité du code.....	3
Codacy & CodeClimate.....	3
Issues.....	3
Conclusion.....	4
Audit de performance de l'application.....	4
Blackfire.....	5
Application initiale.....	5
Application finale.....	5
Conclusion.....	6
Actions menées.....	7
Mises à jour.....	7
Mise à niveau des dépendances et de la version majeur de Symfony.....	7
Frontend.....	8
Correction des anomalies.....	8
Une tâche doit être rattaché à un utilisateur.....	8
Ajout / édition user : choix d'un rôle admin / user.....	9
Frontend.....	9
Nouvelles fonctionnalités.....	9
Gestion des autorisations par rapport aux rôles.....	9
Test unitaires et fonctionnels.....	10
Amélioration du code.....	10

## Context de l'audit

L'audit est réalisée en local avec la configuration suivante :

- Symfony Local Server Web (version: v4.28.1)
- PHP 7.4.8
- MySQL 5.7
- Symfony 5.4 (version « support à long terme ») en Mode : Production

# Audit de qualité du code

Nous allons présenter dans cette partie un comparatif entre l'application initiale (au début de reprise du projet) et l'application améliorée (application finale).

## Codacy & CodeClimate

Les analyses Codacy et CodeClimate n'ont pas révélés de problèmes majeurs sur le code de l'application. Néanmoins, des erreurs de sécurités ont été relevés sur la version initiale de l'application.

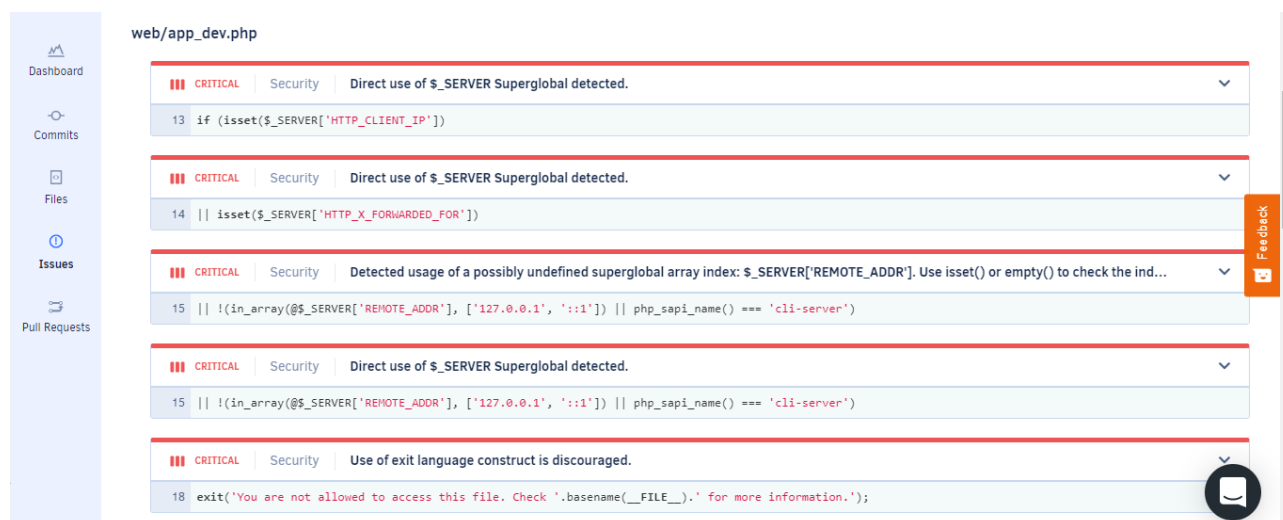
Parmi les plus problématiques, nous retrouvons l'utilisation de variable superglobale sans filtre php particulier (\$\_SERVER) ainsi que l'affichage de code HTML via la méthode php « echo », ce qui est déconseillé.

L'analyse des deux projets (initial et final) sont disponibles en suivant les liens suivants : [projet initial](#) / [projet final](#).

Une analyse **Codacy** du projet initial indique plusieurs choses :

## Issues

Plus d'une dizaine d'issues **Security: Superglobal** concernant les variables superglobale sans filtre php particulier :



The screenshot shows the Codacy interface for the file `web/app_dev.php`. On the left is a sidebar with navigation links: Dashboard, Commits, Files, Issues (selected), and Pull Requests. The main area displays a list of five critical security issues. Each issue is highlighted with a red border and includes a severity indicator (three red bars), the category 'Security', and a description. The first four issues are 'Direct use of \$\_SERVER Superglobal detected.' with corresponding code snippets. The fifth issue is 'Use of exit language construct is discouraged.' with a code snippet. A 'Feedback' button is visible on the right side of the issues list.

Line	Issue Description	Code Snippet
13	Direct use of \$_SERVER Superglobal detected.	<code>if (isset(\$_SERVER['HTTP_CLIENT_IP']))</code>
14	Direct use of \$_SERVER Superglobal detected.	<code>   isset(\$_SERVER['HTTP_X_FORWARDED_FOR'])</code>
15	Detected usage of a possibly undefined superglobal array index: \$_SERVER['REMOTE_ADDR']. Use isset() or empty() to check the ind...	<code>   !(in_array(@\$_SERVER['REMOTE_ADDR'], ['127.0.0.1', '::1'])    php_sapi_name() === 'cli-server')</code>
15	Direct use of \$_SERVER Superglobal detected.	<code>   !(in_array(@\$_SERVER['REMOTE_ADDR'], ['127.0.0.1', '::1'])    php_sapi_name() === 'cli-server')</code>
18	Use of exit language construct is discouraged.	<code>exit('You are not allowed to access this file. Check '.basename(__FILE__).' for more information.');</code>

76 issues **Consistent Return** concernant le framework frontend Bootstrap importé :

## ToDo & Co



## Conclusion

L'audit de qualité de code des deux projets (initial et amélioré) nous a permis d'observer que la montée de version a eu des bénéfices en terme de qualité de code mais aussi de sécurité.

Ce n'est pas pour autant que le projet initial n'était pas sécurisé, mais la version de Symfony 3.1 n'était plus supportée et des failles de sécurité commençaient à apparaître.

## Audit de performance de l'application

Nous allons présenter dans cette partie un comparatif **de performance** entre l'application initiale (au début de reprise du projet) et l'application améliorée (application finale).

Un audit de performance a été réalisé à l'aide de **Blackfire**.

Le test a été réalisé sur quatre pages représentatives de l'application :

- La page de login : **/login**
- La page d'accueil : **/**
- La page des tâches à réaliser : **/tasks**
- La liste des utilisateurs : **/users**

Blackfire

Application initiale

Page	Temps de chargement (ms)	Mémoire consommée (mb)
/login	366	15,1
/	433	18,2
/tasks	503	18,3
/users	467	18,2

200 GET https://localhost:8000/users

Application initiale

Created less than a minute ago by Maxime Doutreluingne

0 rq

467 ms

18.2 MB

0 µs / 0 rq

0 µs / 0 rq

Compare

200 GET https://localhost:8000/tasks

Application initiale

Created 1 minute ago by Maxime Doutreluingne

0 rq

503 ms

18.3 MB

0 µs / 0 rq

0 µs / 0 rq

Compare

200 GET https://localhost:8000/

Application initiale

Created 1 minute ago by Maxime Doutreluingne

0 rq

433 ms

18.2 MB

0 µs / 0 rq

0 µs / 0 rq

Compare

200 GET https://localhost:8000/login

Application initiale

Created 2 minutes ago by Maxime Doutreluingne

0 rq

356 ms

15.1 MB

0 µs / 0 rq

0 µs / 0 rq

Compare

Application finale

Page	Temps de chargement (ms)	Mémoire consommée (mb)
/login	200	14
/	246	17
/tasks	248	17,3
/users	242	17,3

## ToDo & Co

<b>200 GET https://localhost:8000/users</b> <i>Application finale</i> Created less than a minute ago by Maxime Doutreluingne 🔍 0 rq ⌚ 242 ms 📄 17.3 MB ⚡ 0 µs / 0 rq 🗑 0 µs / 0 rq	Compare 🔗 🗑
<b>200 GET https://localhost:8000/tasks</b> <i>Application finale</i> Created less than a minute ago by Maxime Doutreluingne 🔍 0 rq ⌚ 248 ms 📄 17.3 MB ⚡ 0 µs / 0 rq 🗑 0 µs / 0 rq	Compare 🔗 🗑
<b>200 GET https://localhost:8000/</b> <i>Application finale</i> Created 3 minutes ago by Maxime Doutreluingne 🔍 0 rq ⌚ 246 ms 📄 17 MB ⚡ 0 µs / 0 rq 🗑 0 µs / 0 rq	Compare 🔗 🗑
<b>200 GET https://localhost:8000/login</b> <i>Application finale</i> Created 3 minutes ago by Maxime Doutreluingne 🔍 0 rq ⌚ 200 ms 📄 14 MB ⚡ 0 µs / 0 rq 🗑 0 µs / 0 rq	Compare 🔗 🗑

On constate une diminution du temps de chargement et de la mémoire consommée. Suite à une analyse des graph **Blackfire**, je constate que ce ralentissement est principalement dû à l'utilisation de l'autoload de Composer

Function calls	% Excl. ▼	% Incl.	Calls
Composer\Autoload\includeFile			235
Composer\Autoload\ClassLoader::findFileWithExtension			515
Composer\Autoload\includeFile@1			156
file_exists			524

[La documentation de Composer](#) offre une solution à ce problème via la génération d'une "Class Map".

Cela convertit les namespaces PSR-4 en ClassMap ce qui évite de faire appel au filesystem pour vérifier l'existence des classes.

Cette optimisation est faite via la commande : **composer dump-autoload --optimize**.

## Conclusion

Suite à cette optimisation, on constate une amélioration globale du temps de chargement.

En comparant les profils avec l'application initiale/finale, on constate un gain important de performance dans la rapidité d'affichage des pages sans consommation supplémentaire de mémoire significative.

## Actions menées

Suite à l'état des lieux, les actions suivantes ont été menées :

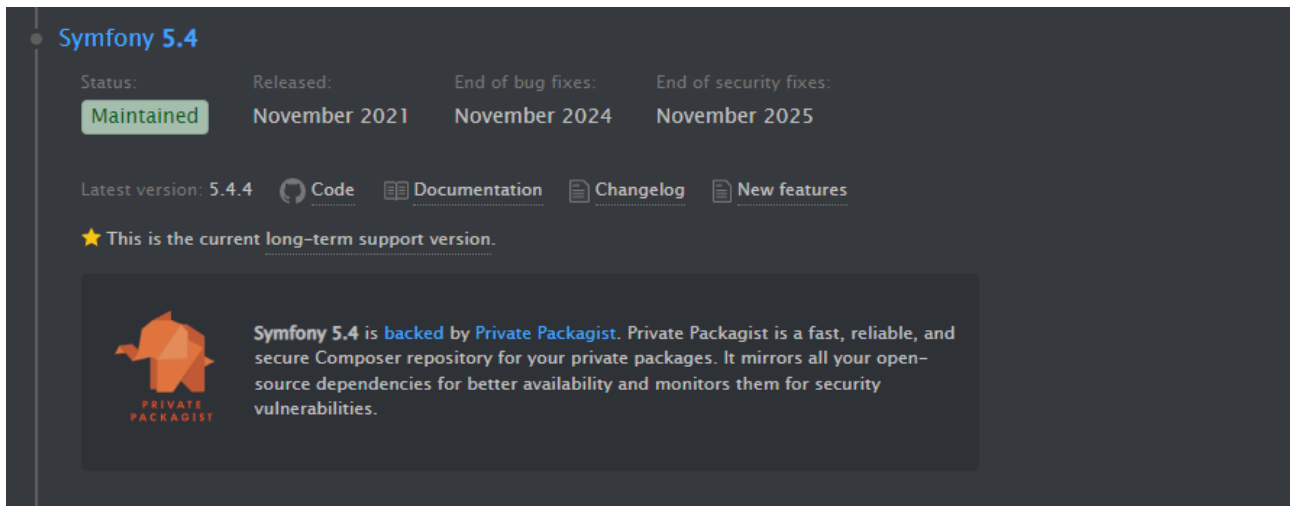
- Montée de version majeur et de ses dépendances pour l'application.
- Correction des différents points relevés dans l'état des lieux initial de l'application afin de réduire sa dette technique et de pérenniser ses futures évolutions.
- Correction des d'anomalies et implémentation des nouvelles fonctionnalités demandées dans le cahier des charges.
- Écriture d'une série de tests unitaires et fonctionnels afin de garantir le bon fonctionnement de l'application.
- Mise en place d'un environnement de développement facilitant l'analyse de la qualité du code et de la performance de l'application.

## Mises à jour

### Mise à niveau des dépendances et de la version majeur de **Symfony**

Nous avons donc décidé d'effectuer une montée de version majeure de Symfony pour passer de la version 3.1 à 5.4, une des dernières versions LTS (Long Time Support).

- Les dépendances ont été supprimées et ré-installées via **Symfony flex**.
- Suppression de la librairie **symfony/swiftmailer-bundle** qui n'est pas utilisée dans le projet.



Nous n'avons pas voulu passer sur Symfony 6 car cette version n'est pas LTS et peut donc contenir des bugs. Indirectement, nous avons du aussi réaliser une montée de version de PHP pour passer de la version **5.4** à la version **7.2**.

### Frontend

- Modification de l'intégration de la librairie Bootstrap dans l'application via le CDN.

## Correction des anomalies

### Une tâche doit être rattaché à un utilisateur

Initialement, lors de la création d'une tâche, cette dernière n'était pas rattachée à un utilisateur.

Nous avons donc modifié le projet pour introduire une relation ManyToOne entre les tâches et les utilisateurs. Pour réaliser cela il a fallu modifier l'entité Task pour introduire cette nouvelle relation. De cette manière, chaque tâche est relié à un utilisateur.

Les tâches déjà créés sont, elles, rattachées à un utilisateur anonyme qui existe en base de données. Des vérifications sont appliquées dans le code pour lier l'utilisateur anonyme avec les tâches qui n'ont pas d'utilisateur.

Enfin, à la modification d'une tâche, il n'est pas possible de changer l'utilisateur qui a initialement créé la tâche.



## **Ajout / édition user : choix d'un rôle admin / user**

Par défaut, il n'y avait pas de gestion des rôles entre les utilisateurs sur le projet.

Pour corriger cela, nous avons implémenté deux rôles **ROLE\_ADMIN** et **ROLE\_USER** pour gérer les utilisateurs de l'application.

A la création d'un nouvel utilisateur, il est automatiquement demandé de choisir entre le rôle administrateur et le rôle utilisateur. A l'édition d'un utilisateur, il est aussi possible de changer le rôle de ce dernier. Pour rappel, seulement les administrateurs peuvent avoir accès à l'administration des utilisateurs.

## **Frontend**

- Ajout d'un lien sur le "Brand" qui redirige vers la page d'accueil.
- Mise en place des pages « tâches à réaliser » et « tâches terminées »

## **Nouvelles fonctionnalités**

Pour améliorer l'application existante, nous avons aussi mis en place de nouvelles fonctionnalités.

### **Gestion des autorisations par rapport aux rôles**

Nous avons restreints l'accès à la gestion des utilisateurs seulement aux utilisateurs ayant le rôle administrateur.

Tous les autres utilisateurs tentant d'accéder à ce type de page seront redirigés vers la page d'accueil avec un message d'erreur. Les modifications ont été réalisées côté backend mais aussi côté frontend avec la suppression du menu utilisateurs dans la navbar pour les utilisateurs non concernés.

Nous avons aussi permis la suppression des tâches seulement par les utilisateurs qui les ont créées. Un utilisateur tentant de supprimer une tâche qu'il n'a pas créée se verra rediriger vers la liste des tâches avec un message d'erreur. Nous avons également géré le cas pour la modification de la tâche de l'utilisateur pour que seulement l'utilisateur qui a créé la tâche puisse la modifier.

## ToDo & Co

Enfin, nous avons mis en place le système de voters symfony pour pouvoir nous permettre de gérer et de contrôler plus facilement les droits d'accès des utilisateurs à certaines pages de l'application (gestion utilisateur / suppression / modification des tâches).

### Test unitaires et fonctionnels

Nous avons aussi implémenté une série de tests unitaires et fonctionnels, nous permettant d'assurer que l'application fonctionne correctement. Nous utilisons **PHPUnit** pour effectuer les tests unitaires et fonctionnels.

Le rapport de couverture du code nous indique que le taux de couverture est supérieur à 75%. Le rapport de tests complet est [disponible ici](#). Voici un aperçu global du rapport de tests :

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total	<div><div></div></div>	96.88%	186 / 192	<div><div></div></div>	91.53%	54 / 59	<div><div></div></div>	76.92%	10 / 13
Controller	<div><div></div></div>	97.10%	67 / 69	<div><div></div></div>	84.62%	11 / 13	<div><div></div></div>	75.00%	3 / 4
Entity	<div><div></div></div>	100.00%	48 / 48	<div><div></div></div>	100.00%	28 / 28	<div><div></div></div>	100.00%	2 / 2
EventSubscriber	<div><div></div></div>	100.00%	11 / 11	<div><div></div></div>	100.00%	4 / 4	<div><div></div></div>	100.00%	1 / 1
Form	<div><div></div></div>	100.00%	25 / 25	<div><div></div></div>	100.00%	2 / 2	<div><div></div></div>	100.00%	2 / 2
Repository	<div><div></div></div>	100.00%	11 / 11	<div><div></div></div>	100.00%	3 / 3	<div><div></div></div>	100.00%	2 / 2
Security	<div><div></div></div>	85.71%	24 / 28	<div><div></div></div>	66.67%	6 / 9	<div><div></div></div>	0.00%	0 / 2
Kernel.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0

Nous avons testé principalement les entités et les contrôleurs, où sont situées en général toutes les fonctionnalités critiques de l'application.

Enfin, nous avons mis en place des fixtures pour générer des données de test pour l'application.

### Amélioration du code

Suite à la montée de version de Symfony et de PHP, le code et la performance de ce dernier s'est amélioré.

Grâce à la montée de version, nous avons d'ailleurs pu profité de l'autowiring permis par Symfony depuis la version 4 et qui permet une injection des bundles simplifié et performante.

Nous avons aussi amélioré l'authentification en supprimant les méthodes "vides" loginCheck() et logoutCheck() par deux méthodes login() et logout()

## ToDo & Co

(toujours vide mais qui renvoie une exception).

Enfin, dans l'optique d'améliorer un peu plus l'application, nous avons aussi rajouté une documentation au niveau du code en suivant les réglementations et les standards de la PHPDoc.