

ToDo & Co

Audit de qualité du code & performance



Table des matières

I. Contexte de l'audit.....	2
II. Audit de qualité du code.....	3
1) Codacy.....	3
1.1) Issues.....	3
2) CodeClimate.....	4
2.1) Issues.....	5
3) Conclusion.....	6
III. Audit de performance de l'application.....	6
1) Blackfire.....	7
1.1) Application initiale.....	7
1.2) Application finale.....	7
2) Conclusion.....	8
IV. Actions menées.....	9
1) Mises à jour.....	9
1.1) Mise à niveau des dépendances et de la version majeur de Symfony..	9
1.2) Frontend.....	10
2) Correction des anomalies.....	10
2.1) Une tâche doit être rattaché à un utilisateur.....	10
2.2) Ajout / édition user : choix d'un rôle admin / user.....	11
2.3) Frontend.....	11
3) Nouvelles fonctionnalités.....	11
3.1) Gestion des autorisations par rapport aux rôles.....	11
3.2) Test unitaires et fonctionnels.....	12
3.3) Amélioration du code.....	12

I. Contexte 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

II. 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).

1) Codacy

L'analyse Codacy 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](#).

1.1) Issues

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

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

The screenshot displays the Codacy web interface for a file named 'web/app_dev.php'. On the left, a sidebar contains navigation links: Dashboard, Commits, Files, Issues (selected), and Pull Requests. The main area shows a list of five critical security issues, each with a red header bar and a dropdown arrow. The issues are:

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

On the right side of the interface, there is a vertical orange 'Feedback' button and a circular icon at the bottom right.

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

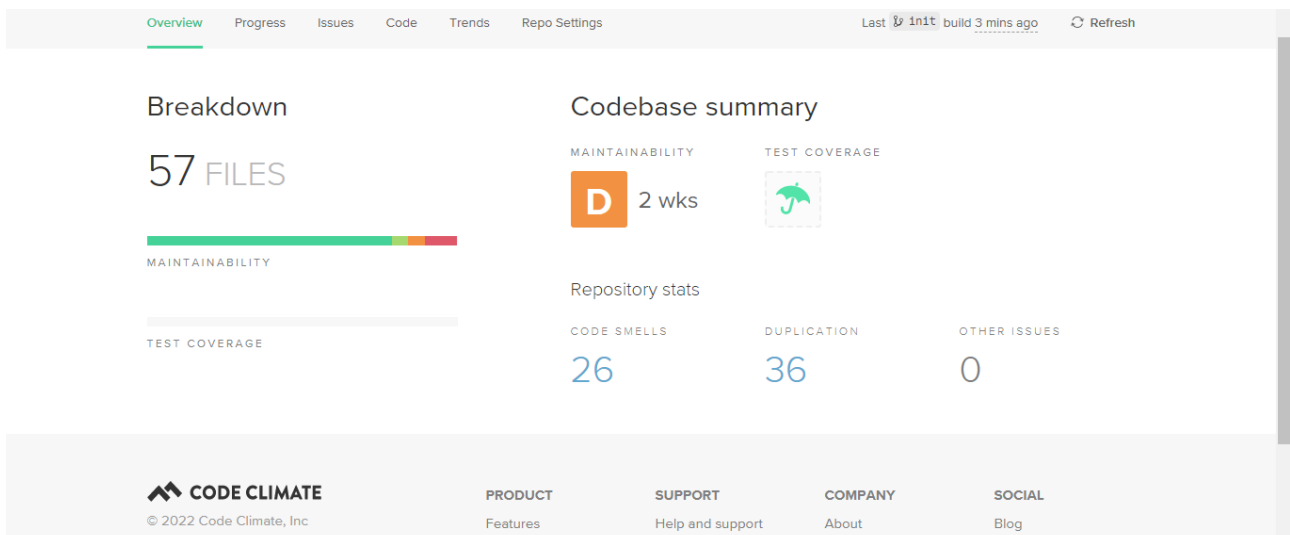
ToDo & Co



2) CodeClimate

L'analyse **CodeClimate** analyse la maintenabilité du projet et a révéls des problèmes majeurs sur le code de l'application. Comme **26** « Code smells » relevé et **36** « Duplication ». Suite à cette analyse **CodeClimate** nous attribut la note de D, sachant que la meilleur note est A.

Parmi les plus problématiques, nous retrouvons l'intégration de la librairie frontend Bootstrap sur des problèmes de nombre de ligne dépassé pour chaque méthode et de multiple duplications, ce qui est déconseillé.



2.1) Issues

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

Comme spécifié ci-dessus, l'analyse a relevé 26 « **Code smells** », ce qui signifie qu'il y a **26** problèmes sur les mauvaises pratiques de conception logicielle qui conduisent à l'apparition de défauts. Comme par exemple le dépassement de nombre de ligne autorisé dans une méthode, trop de condition dans une méthode, ...

The screenshot displays three issues from a CodeClimate analysis of the file `web/js/bootstrap.js`. Each issue is a 'Function has too many lines' warning, indicating that a function exceeds the 25-line limit.

- Issue 1:** The `show` function in `Tooltip.prototype` has 54 lines of code. The snippet shows lines 1429 to 1433:


```
1429 Tooltip.prototype.show = function () {
1430   var e = $.Event('show.bs.' + this.type)
1431
1432   if (this.hasContent() && this.enabled) {
1433     this.$element.trigger(e)
```
- Issue 2:** The `slide` function in `Carousel.prototype` has 44 lines of code. The snippet shows lines 416 to 420:


```
416 Carousel.prototype.slide = function (type, next) {
417   var $active = this.$element.find('.item.active')
418   var $next = next || this.getItemForDirection(type, $active)
419   var isCycling = this.interval
420   var direction = type == 'next' ? 'left' : 'right'
```
- Issue 3:** The `show` function (likely in `Modal`) has 39 lines of code.

Each issue includes a 'Found in' message and an estimated time to fix (e.g., 'About 2 hrs to fix').

L'autre problème qui a été relevé par l'analyse est comme je vous l'ai spécifié plus haut c'est la « **Duplication** ». L'analyse a trouvé **36** duplications. Les duplications nuisent à la qualité du code mais aussi de l'application au niveau des performances. CodeClimate nous indique de « refactoring » c'est à dire réorganiser son code pour qu'il soit plus **maintenable**.

ToDo & Co

Similar blocks of code found in 2 locations. Consider refactoring.

[OPEN](#)

```
1859 function Plugin(option) {
1860   return this.each(function () {
1861     var $this = $(this)
1862     var data = $this.data('bs.popover')
1863     var options = typeof option == 'object' && option
```

Found in [web/js/bootstrap.js](#) and 1 other location - About 5 hrs to fix

Similar blocks of code found in 2 locations. Consider refactoring.

[OPEN](#)

```
1750 function Plugin(option) {
1751   return this.each(function () {
1752     var $this = $(this)
1753     var data = $this.data('bs.tooltip')
1754     var options = typeof option == 'object' && option
```

Found in [web/js/bootstrap.js](#) and 1 other location - About 5 hrs to fix

Similar blocks of code found in 2 locations. Consider refactoring.

[OPEN](#)

SEVERITY

☐ Major

☐ Minor

CATEGORY

☐ Complexity

☒ Duplication

STATUS

☒ Open

☒ Confirmed

☐ Invalid

☐ Wontfix

SOURCE

☒ Code Climate

[Explore 3rd-party plugins](#)

LANGUAGE

☐ JavaScript

☐ PHP

3) Conclusion

L'audit de qualité de code et de la maintenabilité des deux projets (initial et final) 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.

III. 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**

1) Blackfire

1.1) 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

<p>200 GET https://localhost:8000/users</p> <p><i>Application initiale</i></p> <p>Created less than a minute ago by Maxime Doutreluingne</p> <p>🔍 ⚠️ ⚡ 0 rq ⌚ 467 ms 📄 18.2 MB 🌐 0 µs / 0 rq 🗑️ 0 µs / 0 rq</p> <p>Compare < 🗑️</p>
<p>200 GET https://localhost:8000/tasks</p> <p><i>Application initiale</i></p> <p>Created 1 minute ago by Maxime Doutreluingne</p> <p>🔍 ⚠️ ⚡ 0 rq ⌚ 503 ms 📄 18.3 MB 🌐 0 µs / 0 rq 🗑️ 0 µs / 0 rq</p> <p>Compare < 🗑️</p>
<p>200 GET https://localhost:8000/</p> <p><i>Application initiale</i></p> <p>Created 1 minute ago by Maxime Doutreluingne</p> <p>🔍 ⚠️ ⚡ 0 rq ⌚ 433 ms 📄 18.2 MB 🌐 0 µs / 0 rq 🗑️ 0 µs / 0 rq</p> <p>Compare < 🗑️</p>
<p>200 GET https://localhost:8000/login</p> <p><i>Application initiale</i></p> <p>Created 2 minutes ago by Maxime Doutreluingne</p> <p>🔍 ⚠️ ⚡ 0 rq ⌚ 356 ms 📄 15.1 MB 🌐 0 µs / 0 rq 🗑️ 0 µs / 0 rq</p> <p>Compare < 🗑️</p>

1.2) 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

200 GET https://localhost:8000/users <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 🔍 🗑
200 GET https://localhost:8000/tasks <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 🔍 🗑
200 GET https://localhost:8000/ <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 🔍 🗑
200 GET https://localhost:8000/login <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**.

2) 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.

IV. Actions menées

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

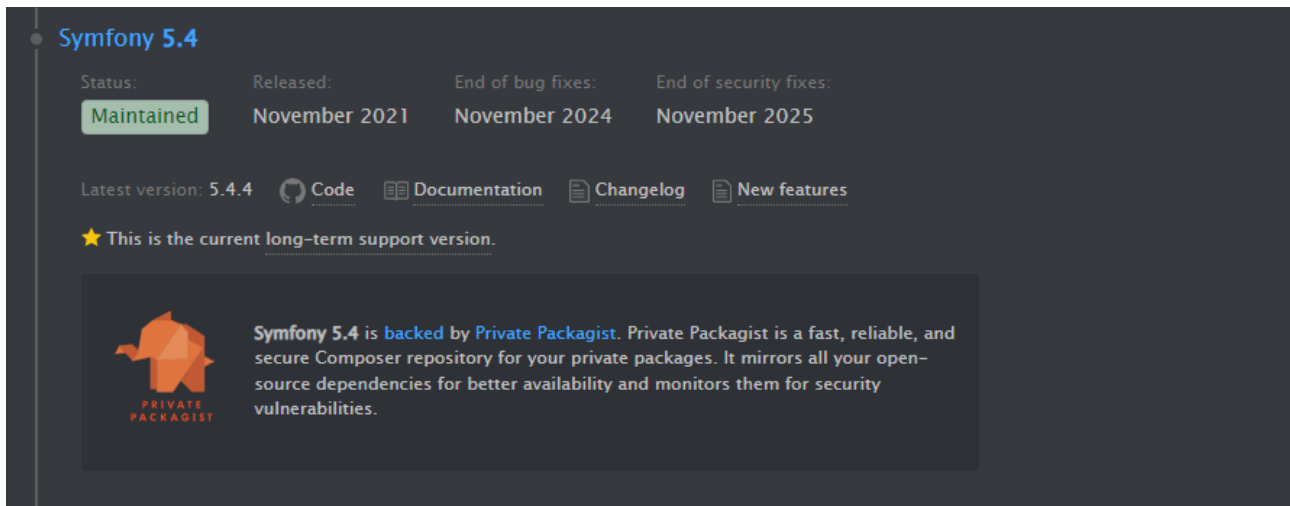
- Montée de version majeur et de ses dépendances pour l'application.
- Correction des différents points relevés dans l'état des lieux initial de l'application afin de réduire sa dette technique et de pérenniser ses futures évolutions.
- Correction des d'anomalies et implémentation des nouvelles fonctionnalités 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.

1) Mises à jour

1.1) 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**.

1.2) Frontend

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

2) Correction des anomalies

2.1) 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.

2.2) 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.

2.3) 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 »

3) Nouvelles fonctionnalités

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

3.1) 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.

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).

3.2) 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		96.88%	186 / 192		91.53%	54 / 59		76.92%	10 / 13
■ Controller		97.10%	67 / 69		84.62%	11 / 13		75.00%	3 / 4
■ Entity		100.00%	48 / 48		100.00%	28 / 28		100.00%	2 / 2
■ EventSubscriber		100.00%	11 / 11		100.00%	4 / 4		100.00%	1 / 1
■ Form		100.00%	25 / 25		100.00%	2 / 2		100.00%	2 / 2
■ Repository		100.00%	11 / 11		100.00%	3 / 3		100.00%	2 / 2
■ Security		85.71%	24 / 28		66.67%	6 / 9		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.

3.3) 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 profiter de l'autowiring permis par Symfony depuis la version 4 et qui permet une injection des bundles simplifiée 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.